Ms. Mandakini Ingle

Assistant Professor

Computer Science & Engineering

Subject :- SEPM

Subject Code :- BTCS504

Class :- CS 3 Year

SYLLABUS

UNIT–IV

Software Testing Strategies-Approach: Issues, Validation Testing and Their Criteria, System Testing, Alpha-Beta Testing, Debugging, Testing Conventional Applications, Testing Object Oriented Applications ,Testing Web Applications.

SOFTWARE TESTING

- The aim of testing process is to identifies the defects existing in a software product.
- It is important activity carried out in order to improve the quality of the software.
- To find out the possible errors the testing must be conducted systematically & test cases must be designed using disciplined techniques.
- Critical phase of software quality assurance & represent the review of specifications ,design & coding.
- Lake of testing leads to waste time & money.
- Effective software testing means:

1. -Quality software products
2. Meet user's requirements

SOFTWARE TESTING.....

- It is the process used to identify the correctness,completeness and quality of developed computersoftware.
- It is the process of executing a program/application under positive and negative conditions by manual or automated means.
- It checks for the :-

1. Specification
2. Functionality
3. Performance

SOFTWARE TESTING :OBJECTIVES

- Uncover as many as errors (or bugs) as possible in a given product.
- Demonstrate a given software product matching its requirement specifications.
- Validate the quality of a software testing using the minimum costand efforts.
- Generate high quality test cases, perform effective tests, and issuecorrect and helpful problem reports.
- A successful test is one that uncovers a yet undiscovered error.
- A good test case have a high probability of finding an undiscover error.

TERMS USED IN SOFTWARE TESTING

- **Error :** It is a human action that produces the incorrect result that produces a fault in a software.
- **Bug :** The presence of error at the time of execution of the software.
- **Fault :** State of software caused by an error.
- **Failure :** Deviation of the software from its expected result. It is an event.
- **Software Tester:** Perform testing.
- **Test Plan:** Systematic approach to perform testing
- **Test Cases:** Specific procedure to testing requirement.

Software Testing Principles

1) **Software testing can help in detecting bugs:**
Testing any software or project can help in revealing a few or some defects that may or may not be detected by developers. However, testing of software alone cannot confirm that your developed project or software is error free. Hence, it’s essential to devise test cases and find out as many defects as possible.

Software Testing Principles…..

2) **Error free or Bug-free software is a myth:** Just because when a tester tested an application and didn't detect any defects in that project, doesn't indicate or imply that your software is ready for shipping

3) **Testing must be performed in different ways** and cannot be tested in a similar way for all modules. All testers have their own individuality, likewise the system under test.

Software Testing Principles…..

4) **Normally a defect is clustered** around a set of modules or functionalities. Once they are identified, testing can be focused on the defective areas, and yet continue to find defects in other modules simultaneously.

5) **Testing will not be as effective and efficient** if the same kinds of tests are performed over a long duration.

Software Testing Principles…..

- The requirements of customers should be traceable and identified by all different tests.
- Planning of tests that how tests will be conducted should be done long before the beginning of the test.
- The Pareto principle can be applied to software testing- 80% of all errors identified during testing will likely be traceable to 20% of all program modules.
- Testing should begin “in the small” and progress toward testing “in the large”.
- Exhaustive testing which simply means to test all the possible combinations of data is not possible.
- Testing conducted should be most effective and for this purpose, an independent third party is required.

BENEFITS OF SOFTWARE TESTING

- **Cost-Effective:** It is one of the important advantages of software testing. Testing any IT project on time helps you to save your money for the long term. In case if the bugs caught in the earlier stage of software testing, it costs less to fix.
- **Security:** It is the most vulnerable and sensitive benefit of software testing. People are looking for trusted products. It helps in removing risks and problems earlier.
- **Product quality:** It is an essential requirement of any software product. Testing ensures a quality product is delivered to customers.
- **Customer Satisfaction:** The main aim of any product is to give satisfaction to their customers. UI/UX Testing ensures the best user experience.

Software Testing Issues

- **Testing the Complete Application**

It is not possible to test each combination in both Manual as well as Automation Testing. If you try all these combinations, you will never ship the product

- **Misunderstanding of Company Processes**

There are some myths in testers that they should only go with the company processes even if these processes are not applicable for their current testing scenario. This results in incomplete and inappropriate Application Testing

Software Testing Issues….

➢ Inadequate schedule of testing:

- ✓ Testing is a time-consuming affair. It must be so since it is done to bring out the defects or inadequacies of the system under different conditions and not to show that it works. Testing needs to go hand in hand with development.

Software Testing Issues ……

- **Insufficient testing environment and tools:**

Tools and environments are backbones of proper software testing. However, testing is often carried out in inadequate testing environment. Moreover, some of the environmental components themselves suffer from defects.

Team managers must ensure that actual or close enough hardware and software requirements are met in a testing environment.

Software Testing Issues ……

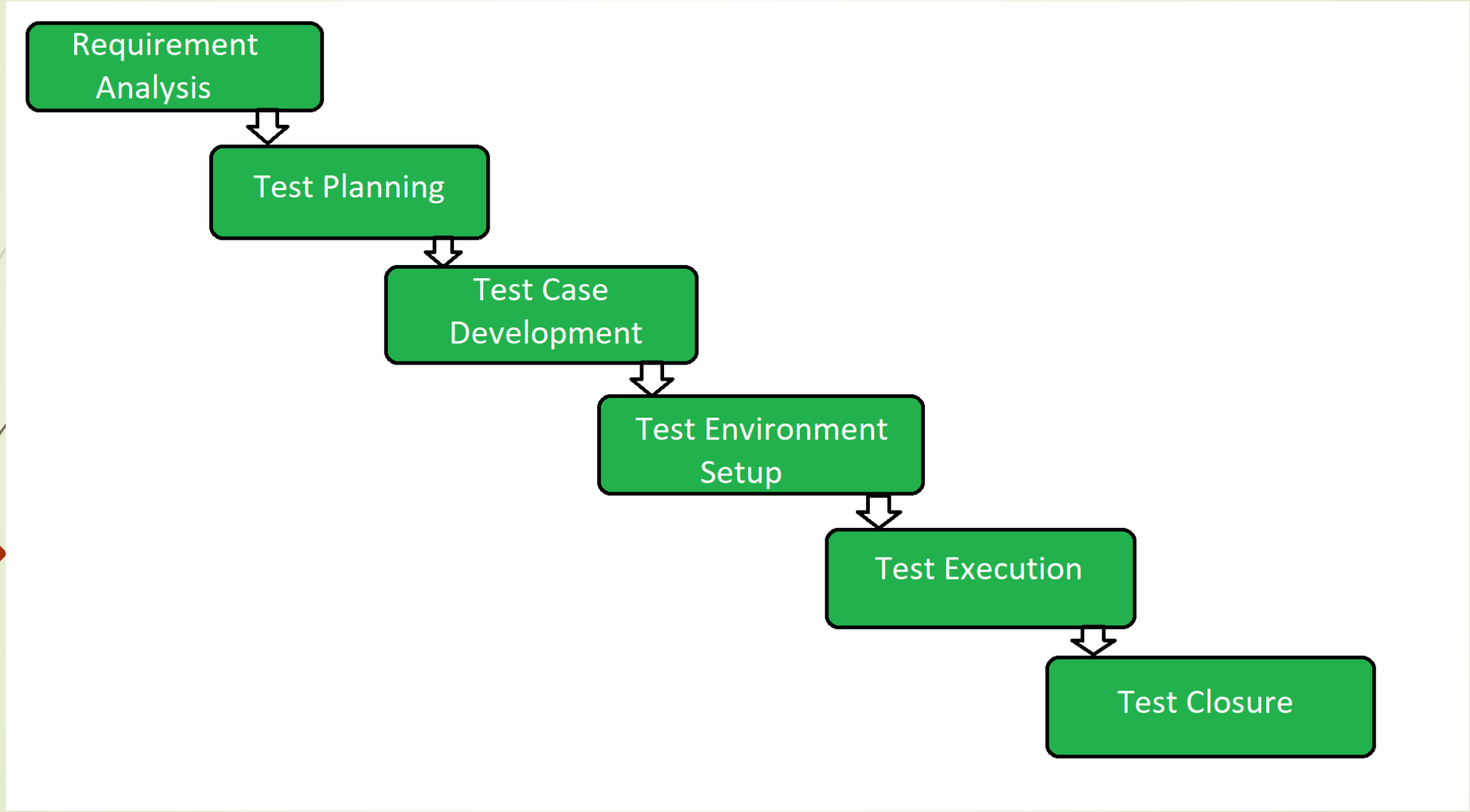
➢ Wrong testing mindset

- ✓ Often the mindset of the software testing team revolves around finding out functionality of the system rather than finding defects in it. This itself prohibits the team from finding out flaws in the software.
- ✓ It is the duty of team lead to instruct the idea that testing is done to find fault with the system or software under different conditions and not to prove that it works

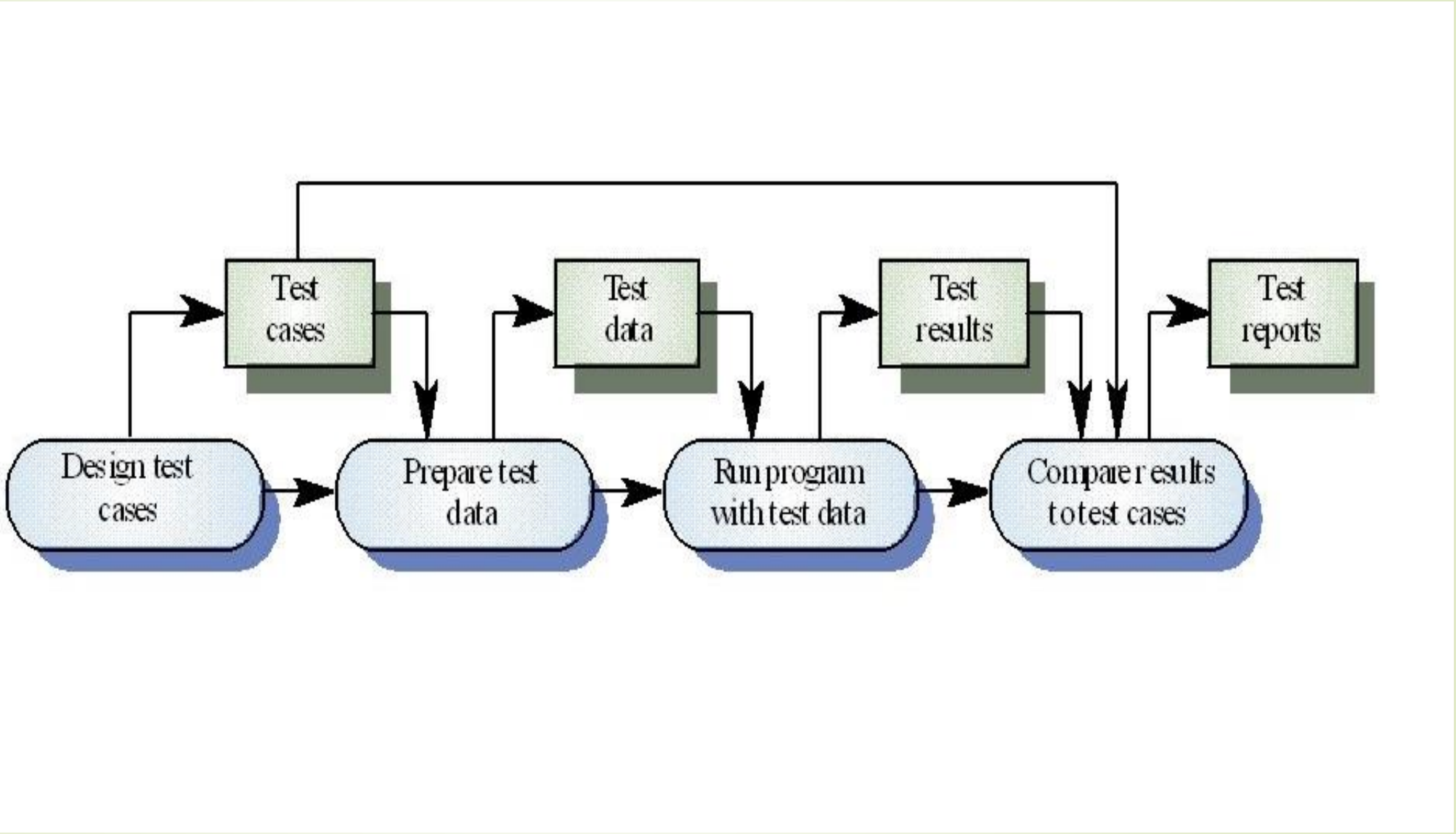
SOFTWARE TESTING PROCESS

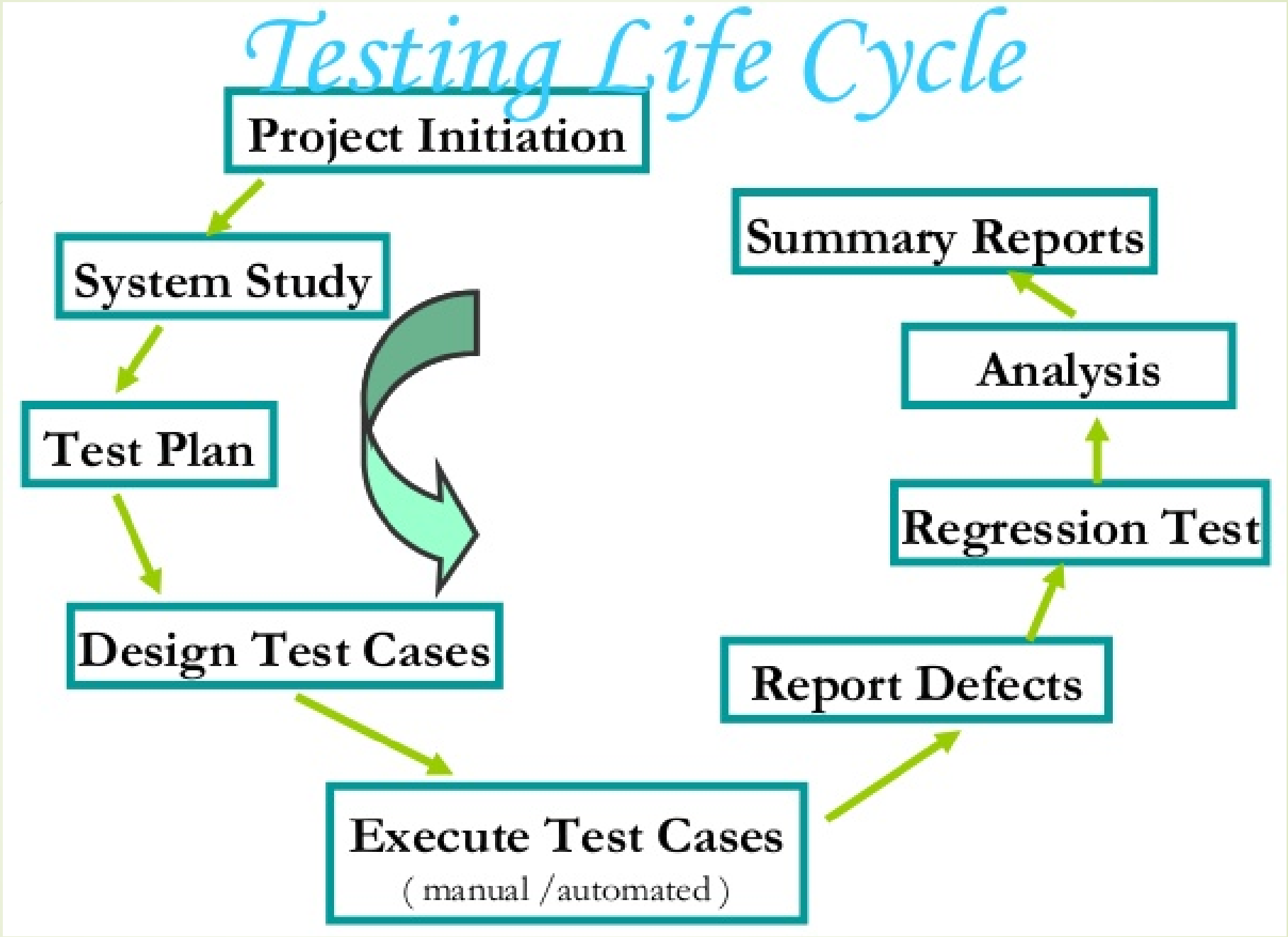
- **Requirement Analysis:** Required testing data is identified.
- **Planning & control:** Test script is prepared
- **Analysis & Design [Test Case]:**Set of tests are analyzed & created that are effective in testing.
- **Implementation & execution of Test cases:** Test cases are executed in order to obtain test result.
- **Evaluating exit criteria & reporting:** All tests cases are performed & errors are uncovered.
- **Test closure Activity:** Standard document for future work.

TESTING PROCESS life cycle



Process of Testing




Testing Life Cycle
Project Initiation
System Study
Test Plan
Design Test Cases
Execute Test Cases
(manual /automated)
Report Defects
Regression Test
Analysis
Summary Reports

LEVELS SOFTWARE TESTING

Software Testing is an activity performed to identify errors so that errors can be removed to obtain a product with greater quality.

To assure and maintain the quality of software and to represents the ultimate review of specification, design, and coding, Software testing is required.

There are different levels of testing :

LEVELS SOFTWARE TESTING

Software Testing is an activity performed to identify errors so that errors can be removed to obtain a product with greater quality.

To assure and maintain the quality of software and to represents the ultimate review of specification, design, and coding, Software testing is required.

There are different levels of testing :

❖ Unit Testing :

In this type of testing, errors are detected individually from every component or unit by individually testing the components or units of software to ensure that if they are fit for use by the developers. It is the smallest testable part of the software.

LEVELS SOFTWARE TESTING .....

❖ Integration Testing :

In this testing, two or more modules which are unit tested are integrated to test i.e. technique interacting components and are then verified if these integrated modules work as per the expectation or not and interface errors are also detected.

❖ System Testing :

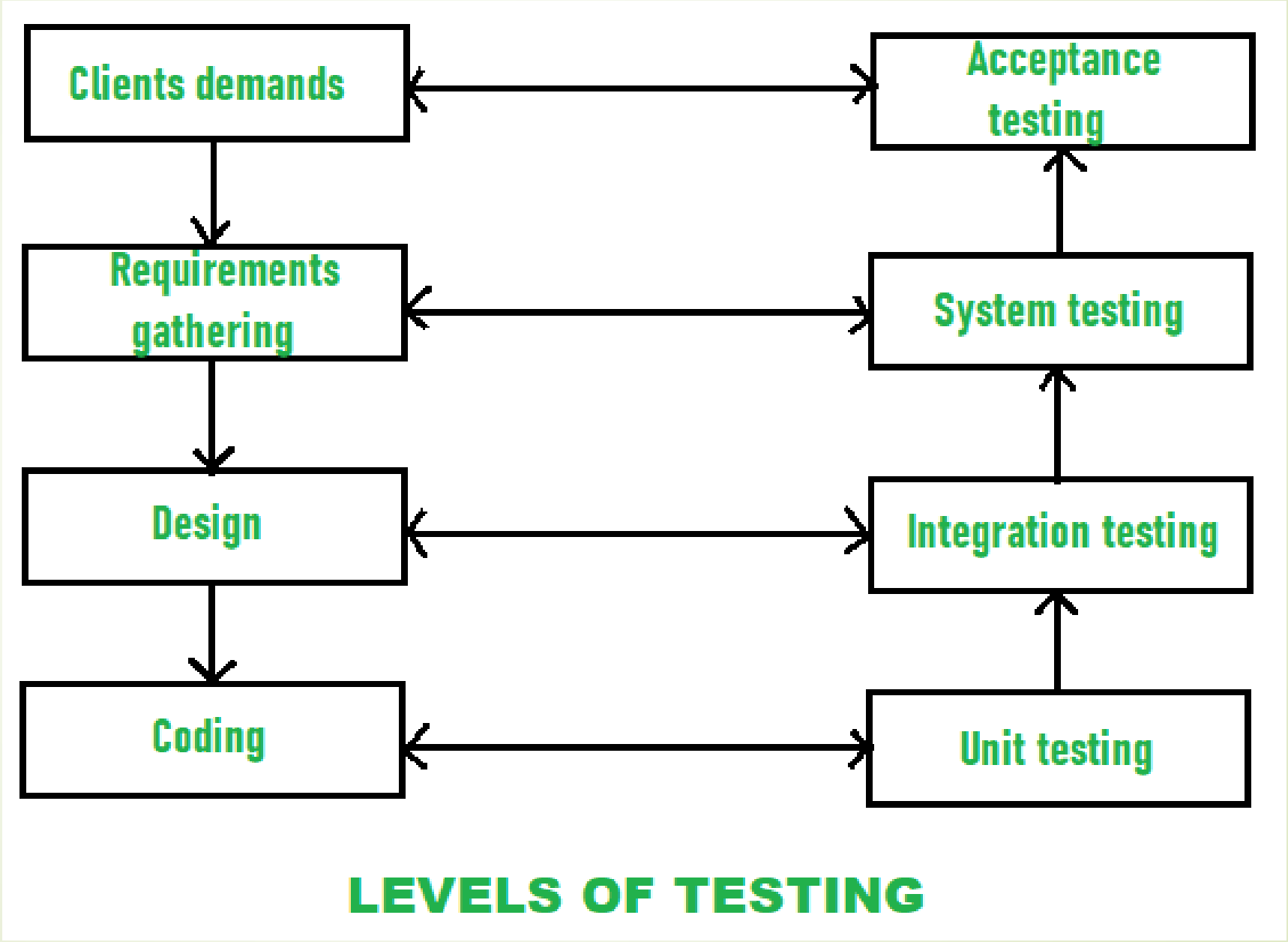
In system testing, complete and integrated Softwares are tested i.e. all the system elements forming the system is tested as a whole to meet the requirements of the system.

LEVELS SOFTWARE TESTING……

- **Acceptance Testing :** It is a kind of testing conducted to ensure whether the requirement of the users are fulfilled prior to its delivery and the software works correctly in the user's working environment.

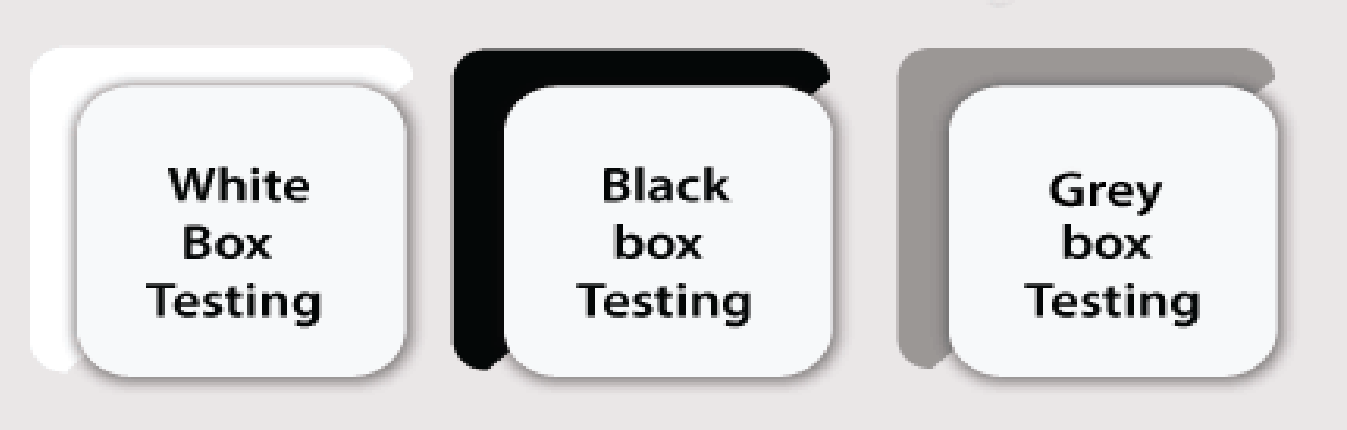
- ✓ These testing can be conducted at various stages of software development.
- ✓ The levels of testing along with the corresponding software development phase is shown by the following diagram –

TESTING LEVELS



Classification of Manual Testing

- In software testing, manual testing can be further classified into three different types of testing, which are as follows



TESTING TECHNIQUES

White Box Testing

- White Box Testing is a testing technique in which software's internal structure, design, and coding are tested to verify input-output flow and improve design, usability, and security.
- In white box testing, code is visible to testers, so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing, and Glass box testing

WHITE BOX TESTING…..

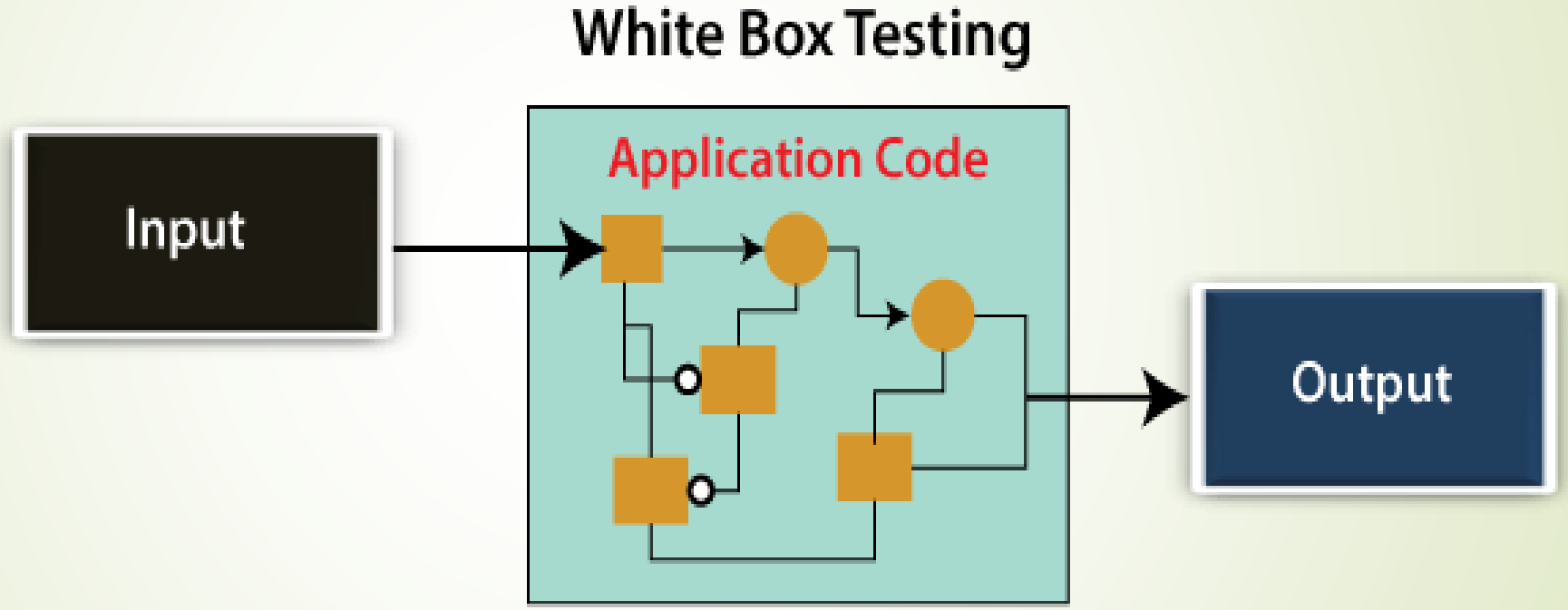
- White-box testing is the detailed investigation of internal logic and structure of the code or close exmination of detail procedure.
- White-box testing is also called glass testing or open-box testing.
- In order to perform white-box testing on an application, a tester needs to know the internal workings of the code.
- The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.
- Examining all independent path,logical paths, loops & internal data structure

White Box Testing Working

Working process of white box testing:

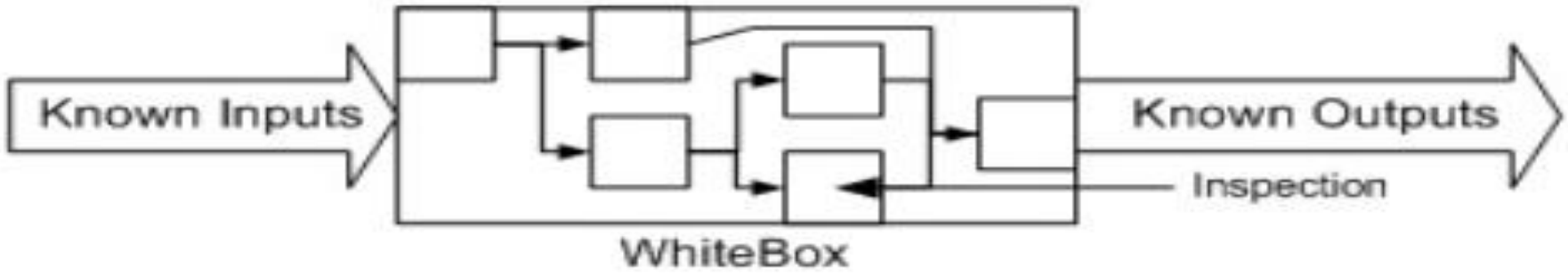
- Input: Requirements, Functional specifications, design documents, source code.
- Processing: Performing risk analysis for guiding through the entire process.
- Proper test planning: Designing test cases so as to cover entire code. Execute rinse-repeat until error-free software is reached. Also, the results are communicated.

White Box Testing



WHITE BOX TESTING

- Focus: Thoroughness (Coverage). Every statement in the component is executed at least once.
- White box testing is also known as structural testing or code-based testing.
- The major objective of white box testing is to focus in internal program structure, and discover all internal program errors.



White Box Testing Types

White box testing refers to a variety of testing methods that are used to assess the usability of an application, a piece of code, or a specific software package. The following is a list −

•**Unit testing** − Unit testing is frequently the first type of application testing performed. As each unit or block of code is developed, it is subjected to unit testing. The programmer is primarily responsible for unit testing. As a software developer, you write a few lines of code, a single function, or an object, then test it to ensure it works before moving on to the next step. Early in the software development lifecycle, unit testing helps in the detection of the majority of issues. Bugs discovered at this stage are less expensive and easier to fix.

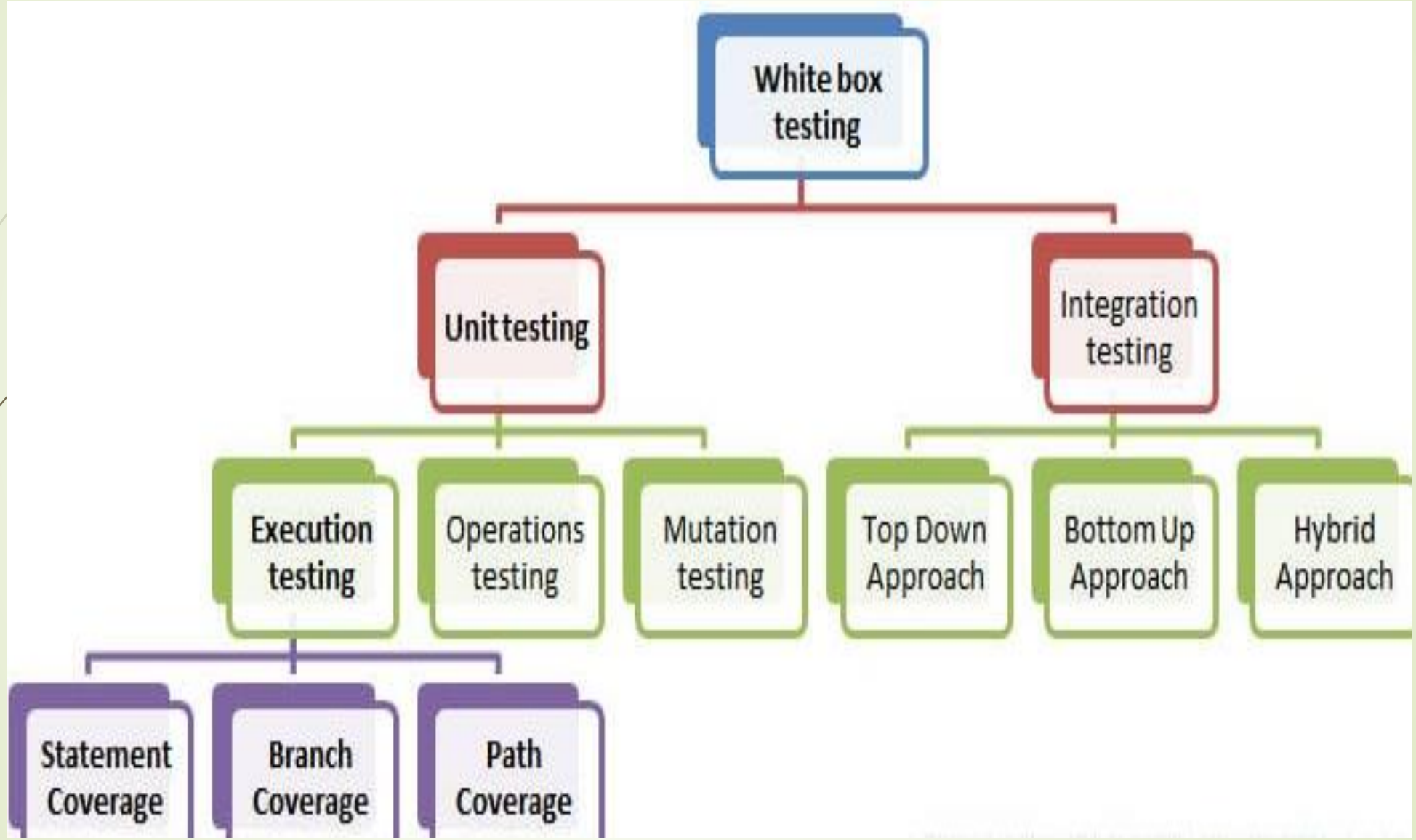
White Box Testing Types…..

•**Testing for Memory Leaks** − Memory leaks are one of the most common reasons for slow-running apps. When you have a slow-running software application, you need a QA professional who is skilled in detecting memory leaks. Apart from the aforementioned, both black box and white box testing include a few forms of testing. Below is a list of them.

•**White Box Penetration Testing** − In this type of testing, the tester/developer has access to the entire source code of the program, as well as extensive network information, IP addresses involved, and all server information. The goal is to attack the code from several aspects in order to expose security flaws.

•**White Box Mutation Testing** − White box mutation testing is frequently used to determine the optimum coding strategies for growing a software solution.

WHITE BOX TESTING TECHNIQUES:



WHITE BOX TESTING TECHNIQUES:

White box testing techniques includes:

1. **Statement Coverage -** This technique is aimed at exercising all programming statements with minimal tests.
2. **Branch/Decision Coverage -** This technique is running a series of tests to ensure that all branches are tested at least once.
3. **Path Coverage -** This technique corresponds to testing all possible paths which means that each statement and branch is covered

WHITE BOX TESTING TECHNIQUES:

Statement Coverage

Statement Coverage

Executing a percentage of statements

What is a statement?

— See the shaded rectangles

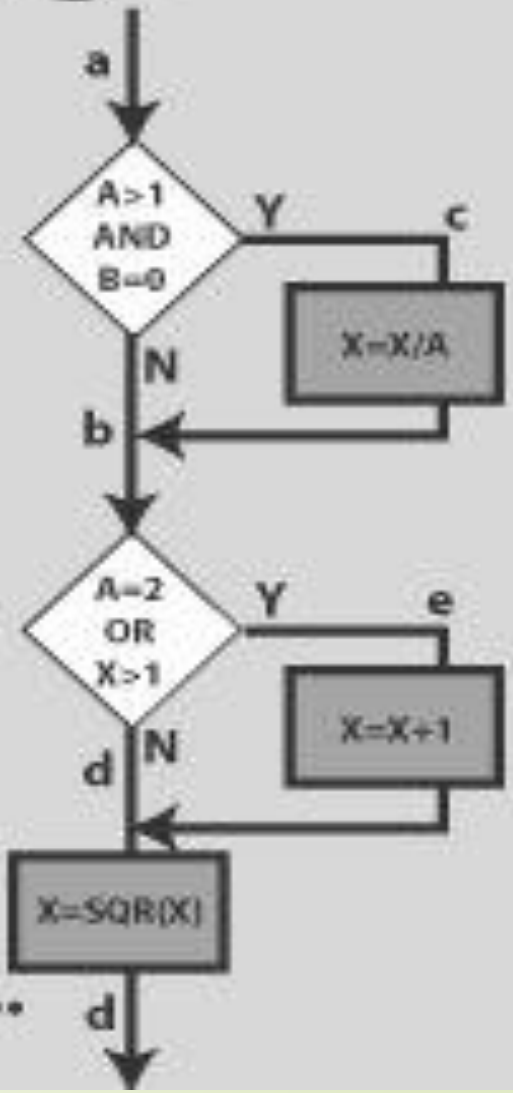
All statements covered for path

— aced'

This is considered the weakest form of coverage

— What is untested in this example?

...Statement, branch, or condition coverage...

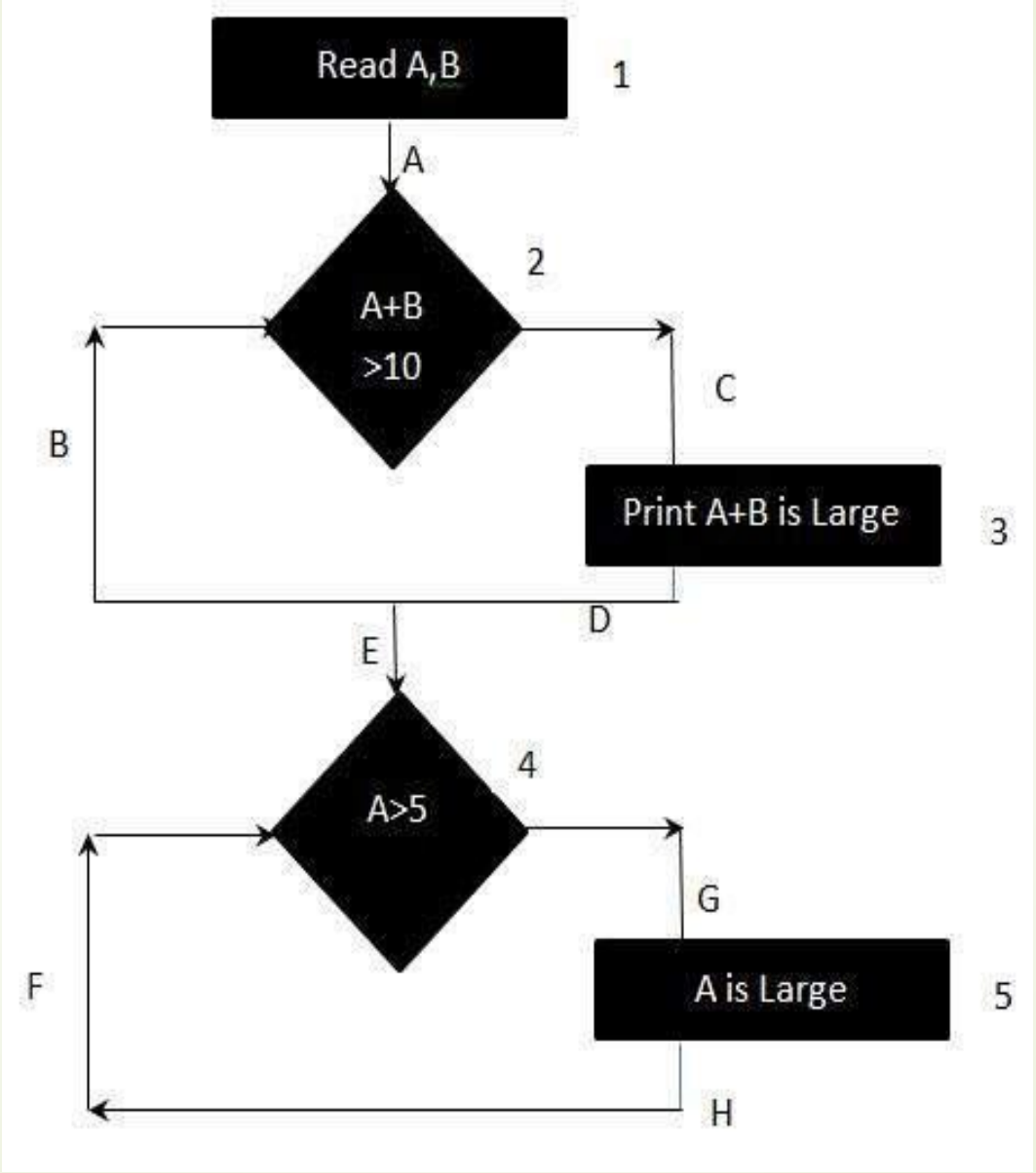


WHITE BOX TESTING TECHNIQUES:

2) Decision/branch coverage: **Decision/branch coverage** is said to test that each branch/output of a decisions is tested, i.e. all statements in both false/true branches will be executed.

But is it not the same? In Statement coverage I need to execute all statements so it can be only done by running all possible ways.

WHITE BOX TESTING TECHNIQUES:



Decision/branch coverage

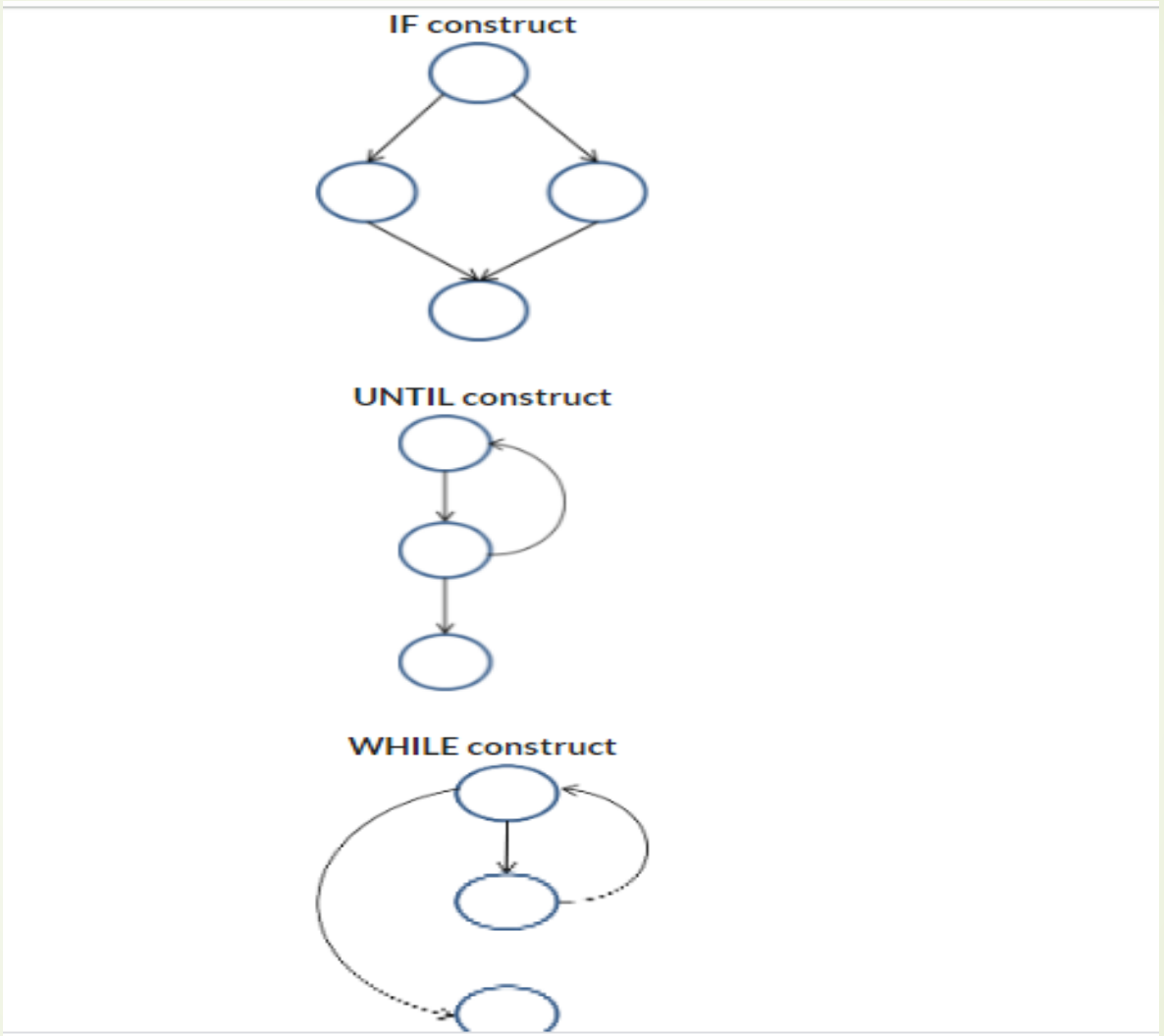
WHITE BOX TESTING TECHNIQUES:

3. Path Coverage

Path coverage refers to designing test cases such that all linearly independent paths in the program are executed at least once.

A linearly independent path can be defined in terms of what's called a control flow graph of an application.

WHITE BOX TESTING TECHNIQUES:



Path coverage

WHITE BOX TESTING:Advantages

1. As the tester has knowledge of the source code, it becomes very easy to find out which type of data can help in testing the application effectively.
2. It helps in optimizing the code.
3. Extra lines of code can be removed which can bring in hidden defects.
4. Due to the tester's knowledge about the code, maximum coverage is attained during test scenario writing.
5. All possible code pathways can be tested including error handling, resource dependencies, & additional internal code logic/flow.
6. Enables tester to find programming errors quickly.
7. Good quality of coding work and coding standards.

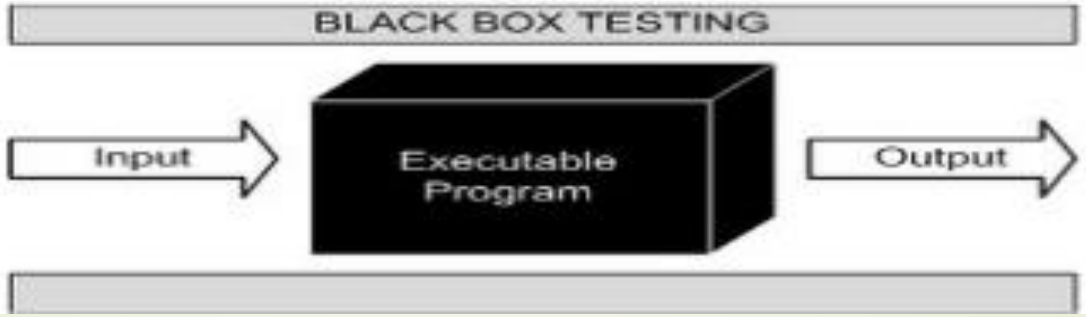
WHITE BOX TESTING:Disdvantages

1. Sometimes it is impossible to look into every Look and corner to find out hidden errors that may create problems, as many paths will go untested.
2. It is difficult to maintain white-box testing, as it requires specialized tools like code analyzers and debugging tools.
3. Knowledge of code & internal structure is a prerequisite, a skilled tester is needed to carry out this type of testing, which increase the cost.
4. Very expensive technique.
5. Requires knowledge of target system, testing tools and coding language.
6. Requires specialized tools such as source code analyzer, debugger, and fault injectors.

BLACK BOX TESTING:

Black Box Testing

- *Black-box testing*, also called *behavioral testing*, focuses on the functional requirements of the software.
- That is, black-box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program.
- Black-box testing is not an alternative to white-box techniques. Rather, it is a complementary approach that is likely to uncover a different class of errors than white-box methods.



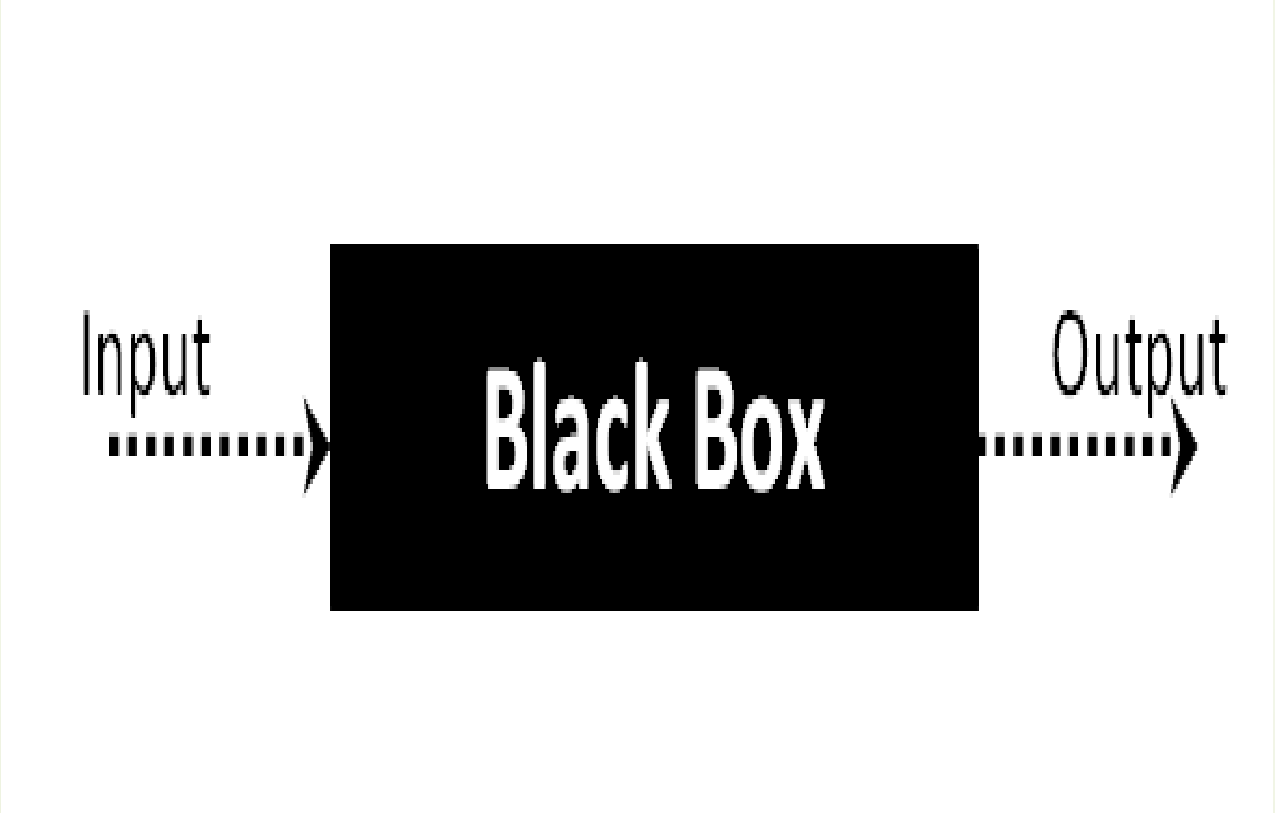
BLACK BOX TESTING:

- Black-box testing is a method of software testing that examines the functionality of an application based on the specifications.
- It is also known as Specifications based testing.
- Independent Testing Team usually performs this type of testing during the software testing lifecycle.
- This method of test can be applied to each and every level of software testing such as unit, integration, system and acceptance testing.

Black Box Testing

- ***Black box*** testing is a type of software testing in which the functionality of the software is not known. The testing is done without the internal knowledge of the products.
- Black Box Testing mainly focuses on input and output of software applications, and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing.

Working of Testing



BLACK BOX TESTING:

Black box testing uncovers following types of errors:

- Incorrect or missing functions
- Interface errors
- Errors in data structures
- Performance errors
- Initialization or termination errors

Types of Black Box Testing

[1] 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, and it is not concerned about the source code of the application.

Types of Black Box Testing …..

Functional Testing

- Every functionality of the system is tested by providing appropriate input, verifying the output and comparing the actual results with the expected results.
- This testing involves checking of User Interface, APIs, Database, security, client/ server applications and functionality of the Application Under Test.
- The testing can be done either manually or using automation

Types of Black Box Testing …..

Functional Testing

Examples of Functional Testing are:

- Unit Testing
- Smoke Testing
- Sanity Testing
- Integration Testing `
- User Acceptance Testing

Types of Black Box Testing …..

[2] Non-Functional Testing

- *Non-functional* testing is a type of testing to check non-functional aspects (performance, usability, reliability, etc.) of a software application.
- It is explicitly designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing.
- It verifies whether the behavior of the system is as per the requirement or not.

Difference between Functional and Non-fuctional Testing

Parameters	Functional	Non-Functional
Execution	It is performed before non-functional testing.	It is performed after the functional testing.
Focus area	It is based on customer's requirements.	It focusses on customer's expectation.
Requirement	It is easy to define functional requirements.	It is difficult to define the requirements for non-functional testing.
Usage	Helps to validate the behavior of the application.	Helps to validate the performance of the application.
Objective	Carried out to validate software actions.	It is done to validate the performance of the software.

Difference between Functional and Non-fuctional Testing

Parameters	Functional	Non-Functional
Manual testing	Functional testing is easy to execute by manual testing.	It’s very hard to perform non-functional testing manually.
Functionality	It describes what the product does.	It describes how the product works.
Example Test Case	Check login functionality.	The dashboard should load in 2 seconds.
Testing Types	Examples of Functional Testing Types Unit testing Smoke testing User Acceptance Integration Testing System Testing	Examples of Non-functional Testing Types Performance Testing Volume Testing Scalability Usability Testing Load Testing Portability Testing

Types of Black Box Testing …..

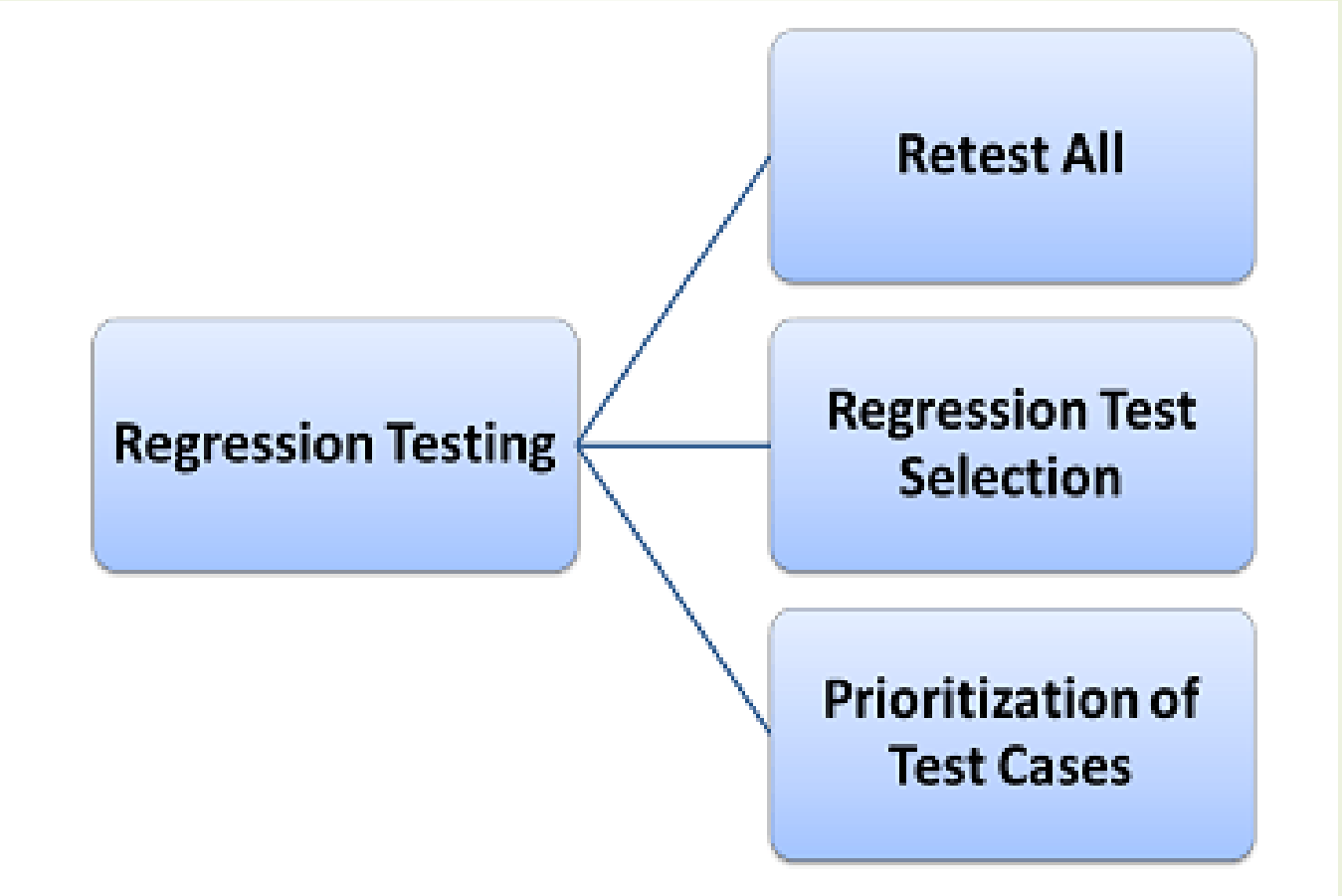
[3] Regression Testing

- Each time a new module is added as part of integration testing, the software changes. New data flow paths are established, new I/O may occur, and new control logic is invoked.
- These changes may cause problems with functions that previously worked flawlessly.
- In the context of an integration test strategy, regression testing is the re-execution of some subset of tests that have already been conducted to ensure that changes have not propagated unintended side effects.
- Regression testing is the activity that helps to ensure that changes (due to testing or for other reasons) do not introduce unintended behavior or additional errors.

Regression Testing....

- The regression test suite (the subset of tests to be executed) contains three different classes of test cases: □
  - A representative sample of tests that will exercise all software functions.
  - Additional tests that focus on software functions that are likely to be affected by the change. □
  - Tests that focus on the software components that have been changed.
- As integration testing proceeds, the number of regression tests can grow quite large.□
- Therefore, the regression test suite should be designed to include only those tests that address one or more classes of errors in each of the major program functions.
- □ It is impractical and inefficient to re-execute every test for every program function once a change has occurred.

Regression Testing....



BLACK BOX TESTING TECHNIQUES :

There are different techniques involved in Black Box testing.

1. **Equivalence partitioning**
2. **Boundary Value Analysis**
3. **Cause-Effect Graphing.**
4. **Error-Guessing.**

Advantage's of Black Box Testing

- More effective on larger units of code than glass box testing
- Tester needs no knowledge of implementation, including specific programming language.
- Tester and programmer are independent of each other Tests are done from a users point of view.
- Will help to expose any ambiguities or inconsistencies in the specification.
- Test cases can be designed as soon as the specifications are complete

Disadvantage's of Black Box Testing

- Only a small number of possible inputs can actually be tested, to test every possible input stream would take nearly foreve.
- Without clear and concise specifications, test cases are hard to design.
- There may be unnecessary repetition of test inputs if the tester is not informed of test cases the programmer has already tried

Unit Testing

- In Unit testing the individual components are tested independently to ensure their quality.
- The focus is to uncover the errors in design and implementation.
- It is concerned with functional correctness of the stand alone modules.
- The main aim is to isolate each unit of the system to identify, analyze and fix the defects.

Unit Testing...

- Unit testing focuses verification effort on the smallest unit of software design—the software component or module.
- Using the component-level design description as a guide, important control paths are tested to uncover errors within the boundary of the module.
- The relative complexity of tests and uncovered errors is limited by the constrained scope established for unit testing.
- The unit test is white-box oriented, and the step can be conducted in parallel for multiple components.

Unit Testing Considerations

- The module interface is tested to ensure that information properly flows into and out of the program unit under test.
- The local data structure is examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithms execution.
- Boundary conditions are tested to ensure that the module operates properly at boundaries established to limit or restrict processing.
- All independent paths (basis paths) through the control structure are exercised to ensure that all statements in a module have been executed at least once.

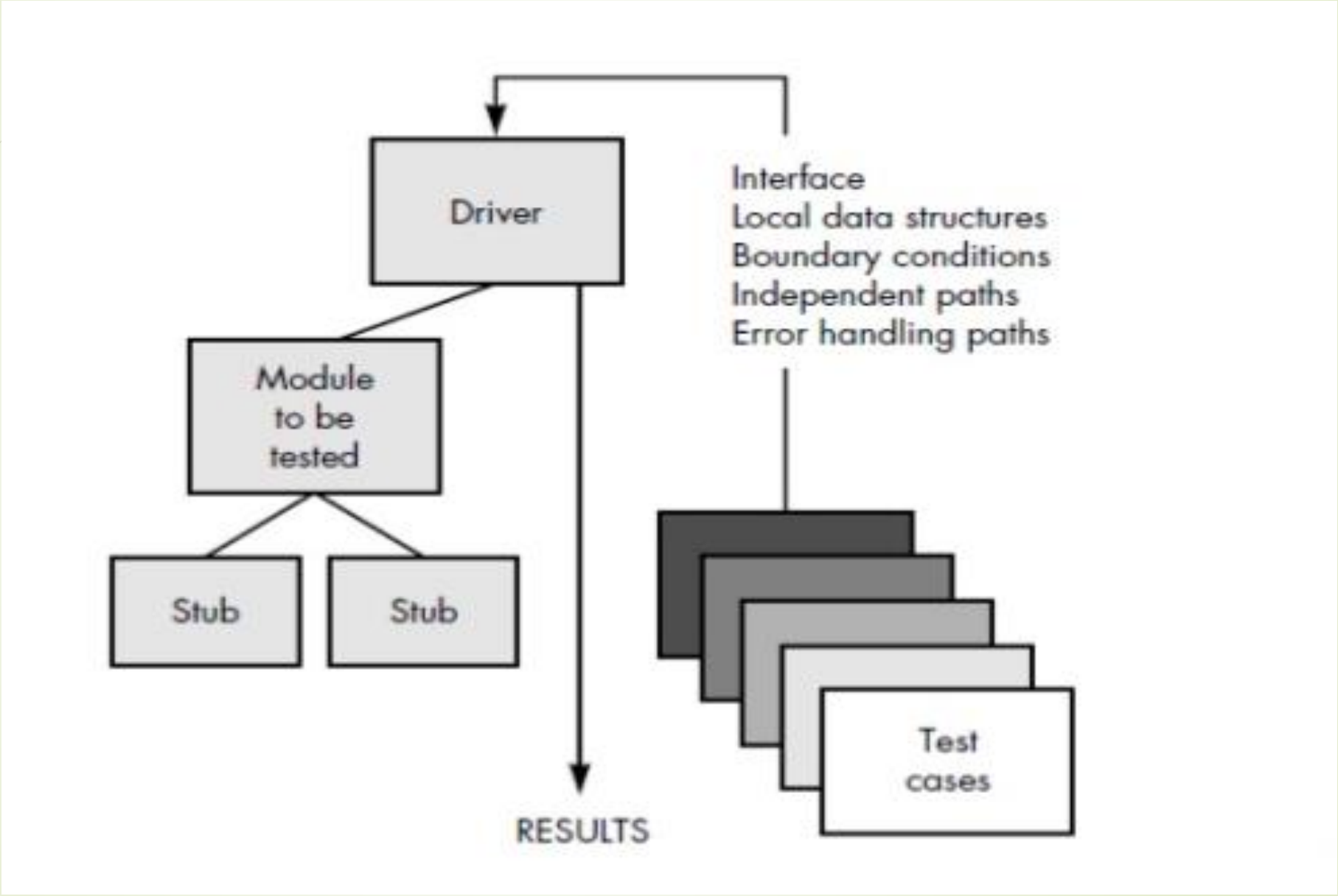
Unit Testing Consideration

- And finally, all error handling paths are tested.
- Following are some types of errors commonly found during unit testing
  - misunderstood or incorrect
  - arithmetic precedence
  - mixed mode operations,
  - incorrect initialization,
  - precision inaccuracy,

Unit Testing Precedence

- Once the source level code has been developed, reviewed, and verified for correspondence to component level design, unit test case design begins.
- A component is not a stand-alone program, driver and/or stub software must be developed for each unit test.
- In most applications a driver is nothing more than a "main program" that accepts test case data, passes such data to the component (to be tested), and prints relevant results.
- Stubs serve to replace modules that are subordinate (called by) the component to be tested. A stub or "dummy subprogram" uses the subordinate modules interface, may do minimal data manipulation, prints verification of entry, and returns control to the module undergoing testing.
- Drivers and stubs represent overhead. That is, both are software that must be written (formal design is not commonly applied) but that is not delivered with

Unit Testing



Unit Testing Techniques

- **Black Box Testing** - Using which the user interface, input and output are tested.
- **White Box Testing** - used to test each one of those functions behaviour is tested.
- **Gray Box Testing** - Used to execute tests, risks and assessment methods.

Unit Testing Advantages

- Reduces Defects in the Newly developed features or reduces bugs when changing the existing functionality.
- Reduces Cost of Testing as defects are captured in very early phase.
- Improves design and allows better refactoring of code.
- Unit Tests, when integrated with build gives the quality of the build as well.

Unit Testing Disadvantages

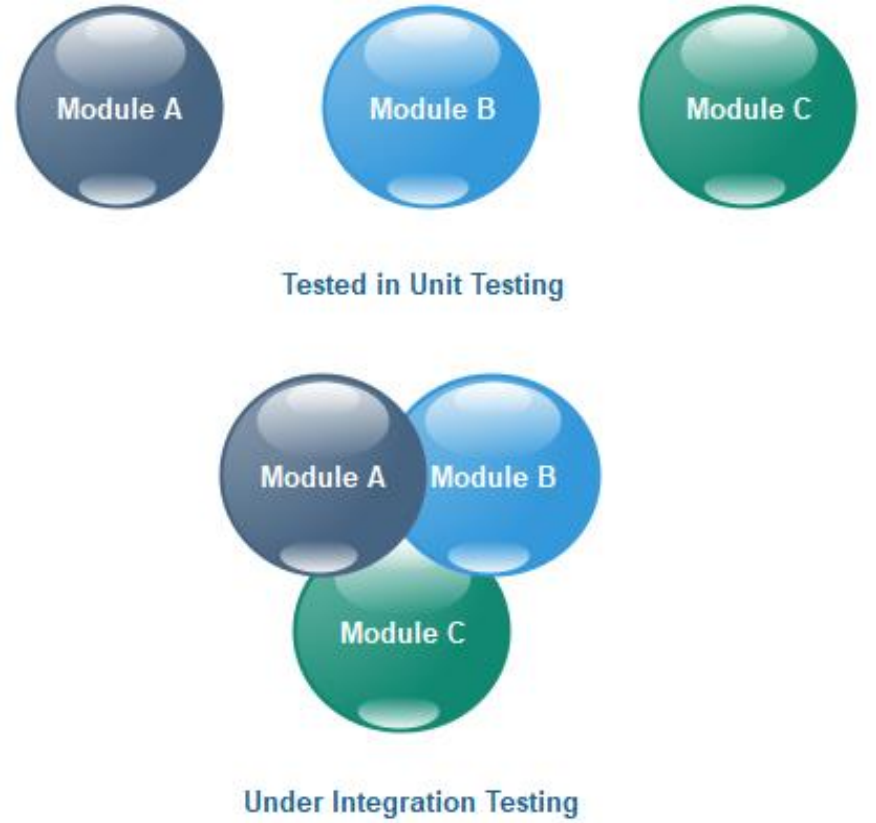
- Unit testing can't be expected to catch every error in a program.
- It is not possible to evaluate all execution paths even in the most trivial programs
- Unit testing by its very nature focuses on a unit of code. Hence it can't catch integration errors or broad system level errors.

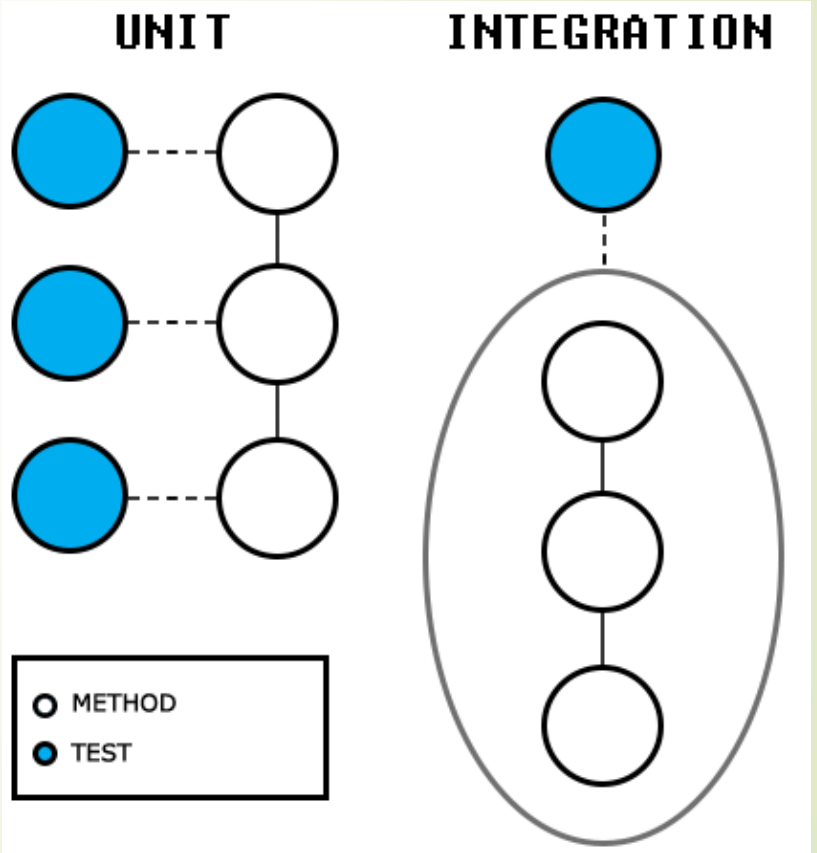
Integration Testing

- Unit that otherwise seem to be working fine individually, starts causing problems when integrated together.
- Data can be lost across an interface; one module can have an inadvertent, adverse affect on another; sub- functions, when combined, may not produce the desired major function; individually acceptable imprecision may be magnified to unacceptable levels; global data structures can present problems.
- Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing.

Integration Testing

- A group of dependent components are tested together and ensure their quality of their integration unit.
- The objective is to take unit tested components and build a program structure that has bssn dictated by software design.





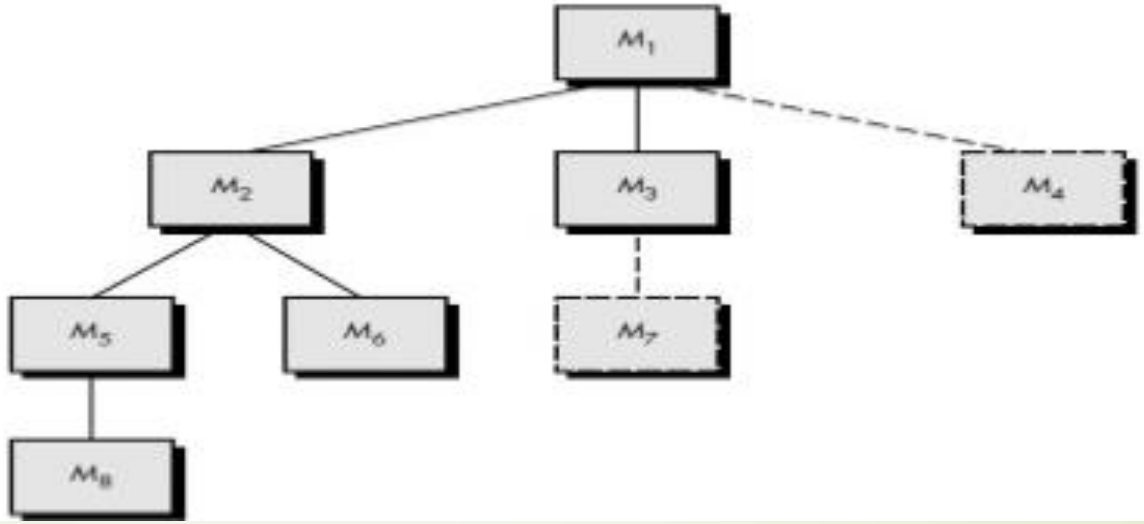
Integration Testing

- **Non-Incremental integration** or "big bang" approach:
  - Combines all the components at the same time and then tests the program on the whole. This approach gives a number of errors that are difficult to isolate and trace as the whole program now is very vast. The correction of such errors seems to go into infinite loop.
- **Incremental integration:**
  - constructed and tested in small increments, where errors are easier to isolate and correct; interfaces are more likely to be tested completely; and a systematic test approach may be applied.
  - Top-down Integration
  - Bottom-up Integration

Integration Testing

Top-down Integration

- *Top-down integration testing* is an incremental approach to construction of program structure.
- Modules are integrated by moving downward through the control hierarchy, beginning with the main control module (main program).
- Modules subordinate (and ultimately subordinate) to the main control module are incorporated into the structure in either a depth-first or breadth-first manner.



Integration Testing……

Depth-first integration & Breadth-first integration

- *depth-first integration* would integrate all components on a major control path of the structure.
- Selection of a major path is somewhat arbitrary and depends on application-specific characteristics.
- For example, selecting the left-hand path, components M1, M2 , M5 would be integrated first. Next, M8 or (if necessary for proper functioning of M2) M6 would be integrated.
- Then, the central and right-hand control paths are built.
- *Breadth-first integration* incorporates all components directly subordinate at each level, moving across the structure horizontally.
- components M2, M3, and M4 would be integrated first. The next control level, M5, M6, and so on, follows.

Integration Testing……

Depth-first integration & Breadth-first integration

- *depth-first integration* would integrate all components on a major control path of the structure.
- Selection of a major path is somewhat arbitrary and depends on application-specific characteristics.
- For example, selecting the left-hand path, components M1, M2 , M5 would be integrated first. Next, M8 or (if necessary for proper functioning of M2) M6 would be integrated.
- Then, the central and right-hand control paths are built.
- *Breadth-first integration* incorporates all components directly subordinate at each level, moving across the structure horizontally.
- components M2, M3, and M4 would be integrated first. The next control level, M5, M6, and so on, follows.

Integration Testing.....

❖ The integration process is performed in a series of five steps:

1. The main control module is used as a test driver and stubs are substituted for all components directly subordinate to the main control module.

2. Depending on the integration approach selected (i.e., depth or breadth first), subordinate stubs are replaced one at a time with actual components.

3. Tests are conducted as each component is integrated.

4. On completion of each set of tests, another stub is replaced with the real component.

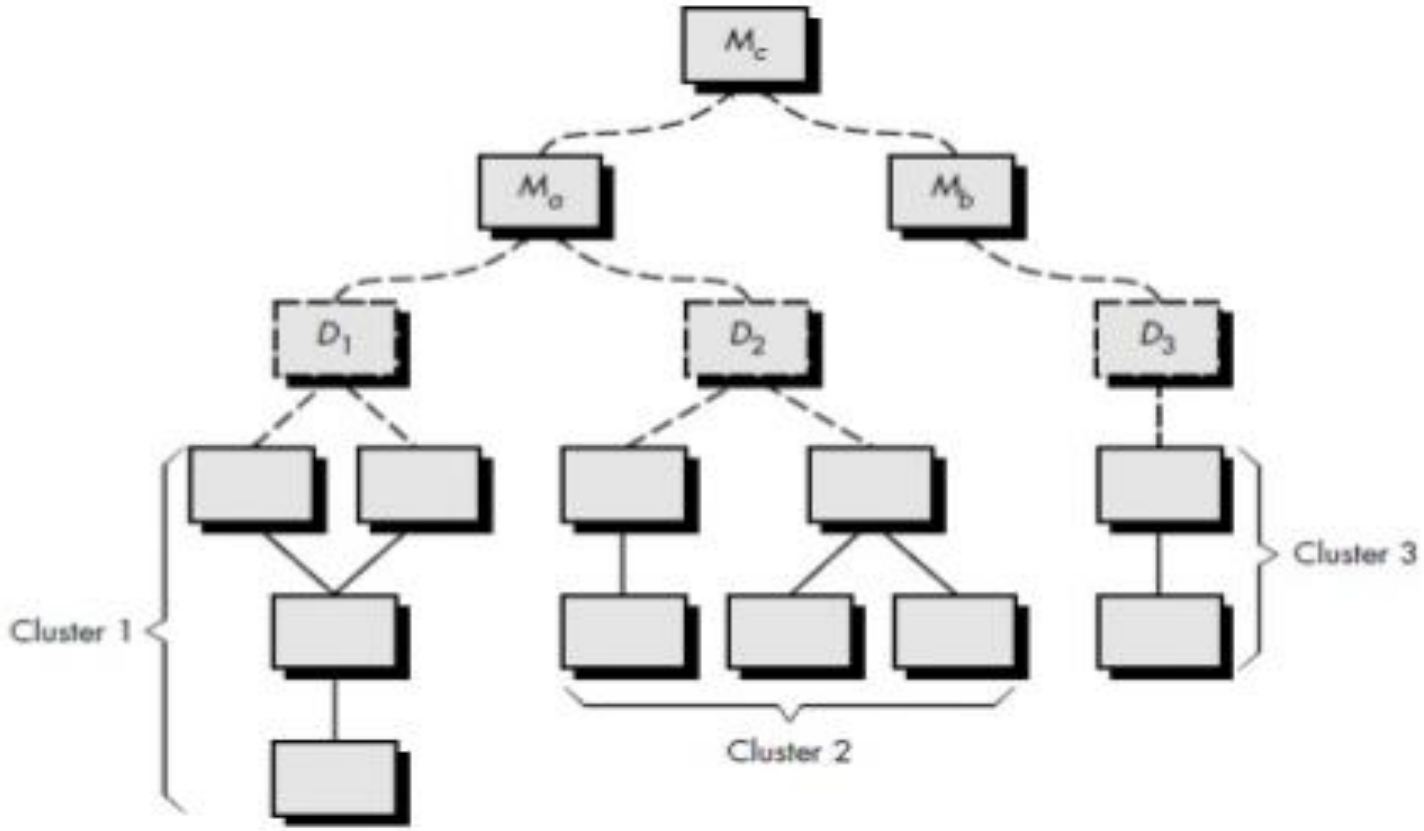
5. Regression testing may be conducted to ensure that new errors have not been introduced. The process continues from step 2 until the entire program structure is built.

Integration Testing.....

Bottom-up Integration

- *Bottom-up integration testing*, as its name implies, begins construction and testing with *atomic modules* i.e., components at the lowest levels in the program structure.
- Because components are integrated from the bottom up, processing required for components subordinate to a given level is always available and the need for stubs is eliminated.
- A bottom-up integration strategy may be implemented with the following steps:
  - **1.** Low-level components are combined into clusters (sometimes called *builds*) that perform a specific software subfunction.
  - **2.** A driver (a control program for testing) is written to coordinate test case input and output.
  - **3.** The cluster is tested.
  - **4.** Drivers are removed and clusters are combined moving upward in the program structure.

Integration Testing.....



- Components are combined to form clusters 1, 2, and 3. Each of the clusters is tested using a driver (shown as a dashed block). Components in clusters 1 and 2 are subordinate to Ma.
- Drivers D1 and D2 are removed and the clusters are interfaced directly to Ma.
- Similarly, driver D3 for cluster 3 is removed prior to integration with module Mb. Both Ma and Mb will ultimately be integrated with component Mc, and so forth.

[2] Smoke Testing

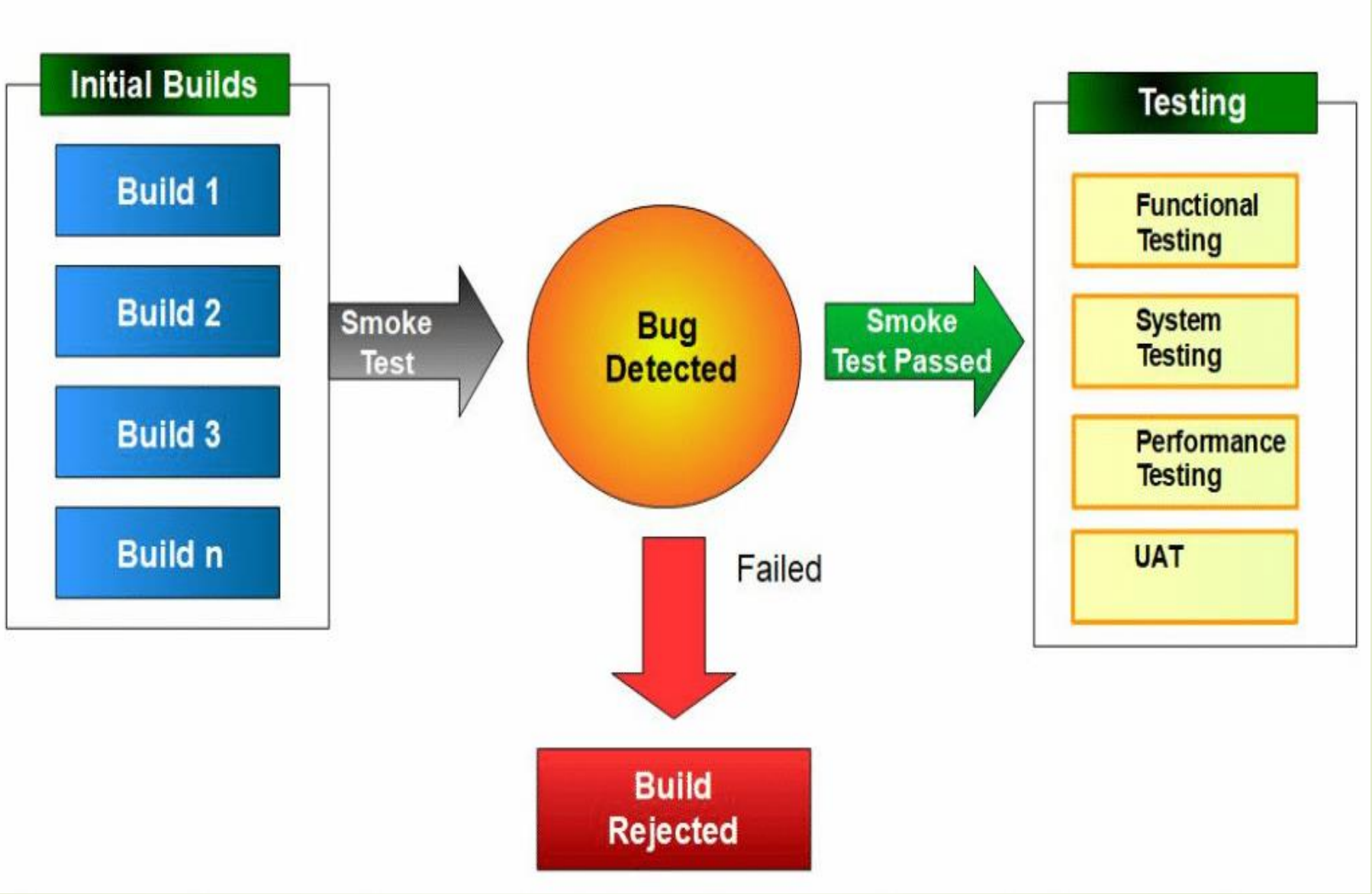
- Smoke testing is an integration testing approach that is commonly used when "shrinkwrapped" software products are being developed. It is designed as a pacing mechanism for time-critical projects, allowing the software team to assess its project on a frequent basis.
- the smoke testing approach encompasses the following activities:

1. Software components that have been translated into code are integrated into a "build." A build includes all data files, libraries, reusable modules, and engineered components that are required to implement one or more product functions.
2. A series of tests is designed to expose errors that will keep the build from properly performing its function. The intent should be to uncover "show stopper" errors that have the highest likelihood of throwing the software project behind schedule.
3. The build is integrated with other builds and the entire product (in its current form) is smoke tested daily. The integration approach may be top down or bottom up.

Smoke Testing....

- Smoke testing provides a number of benefits when it is applied on complex, timecritical software engineering projects:
  - *Integration risk is minimized*. Because smoke tests are conducted daily, incompatibilities and other show-stopper errors are uncovered early, thereby reducing the likelihood of serious schedule impact when errors are uncovered.
  - *The quality of the end-product is improved*. Because the approach is construction (integration) oriented, smoke testing is likely to uncover both functional errors and architectural and component-level design defects. If these defects are corrected early, better product quality will result.
  - *Error diagnosis and correction are simplified*. Like all integration testing approaches, errors uncovered during smoke testing are likely to be associated with "new software increments"—that is, the software that has just been added to the build(s) is a probable cause of a newly discovered error.
  - *Progress is easier to assess*. With each passing day, more of the software has been integrated and more has been demonstrated to work. This improves team morale and gives managers a good indication that progress is being made.

Smoke Testing....



Validation Testing

- After integration testing, one may start with validation testing.
- validation succeeds when software functions in a manner that can be reasonably expected by the customer.
- Software validation is achieved through a series of black-box tests that demonstrate conformity with requirements.
- This is to ensure that all functional requirements are satisfied, all behavioral characteristics are achieved, all performance requirements are attained, documentation is correct, and other requirements are met

Validation Testing……

- The process of evaluating software during the development process or at the end of the development process to determine whether it satisfies specified business requirements.
- Validation Testing ensures that the product meets the client's needs.
- It can also be defined as to demonstrate that the product fulfills its intended use when deployed on appropriate environment.

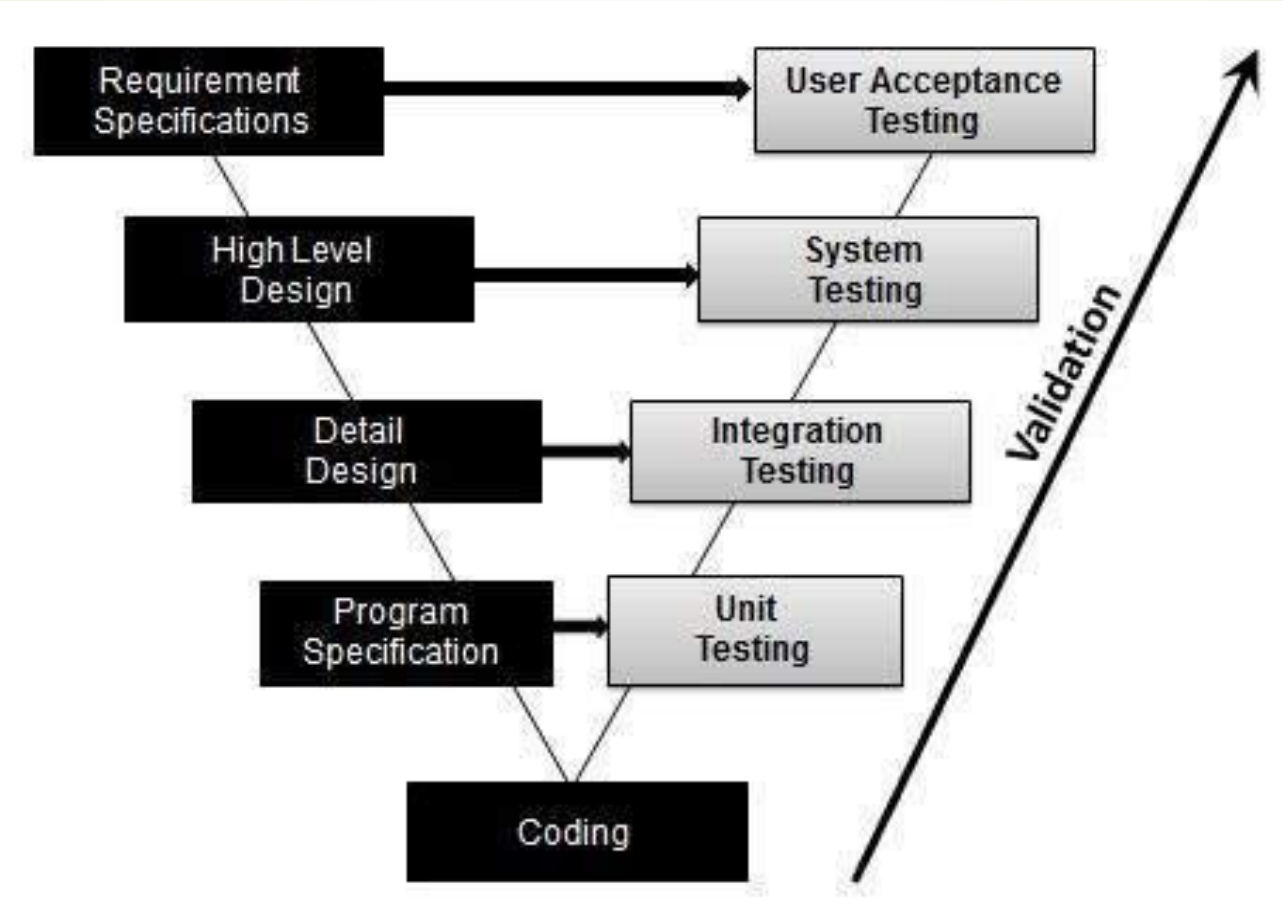
Validation Testing……

- Validation testing is also known as dynamic testing, where we are ensuring that "we have developed the product right."
- And it also checks that the software meets the business needs of the client.
- It is the process of checking the validation of product i.e., it checks what we are developing is the right product.
- Validation testing can be best demonstrated using ***V-Model***. The Software/product under test is evaluated during this type of testing.

Validation Testing....

- It is virtually impossible for a software developer to fore see how the customer will really use a program.
- When custom software is built for one customer, a series of acceptance tests are conducted to enable the customer to validate all requirements.
- Conducted by the end-user rather than software engineers, an acceptance test can range from an informal "test drive" to a planned and systematically executed series of tests.

Validation Testing



Validation Testing

- Activities involved in validation:
  - ✓ Black box testing
  - ✓ White box testing
  - ✓ Unit testing
  - ✓ Integration testing

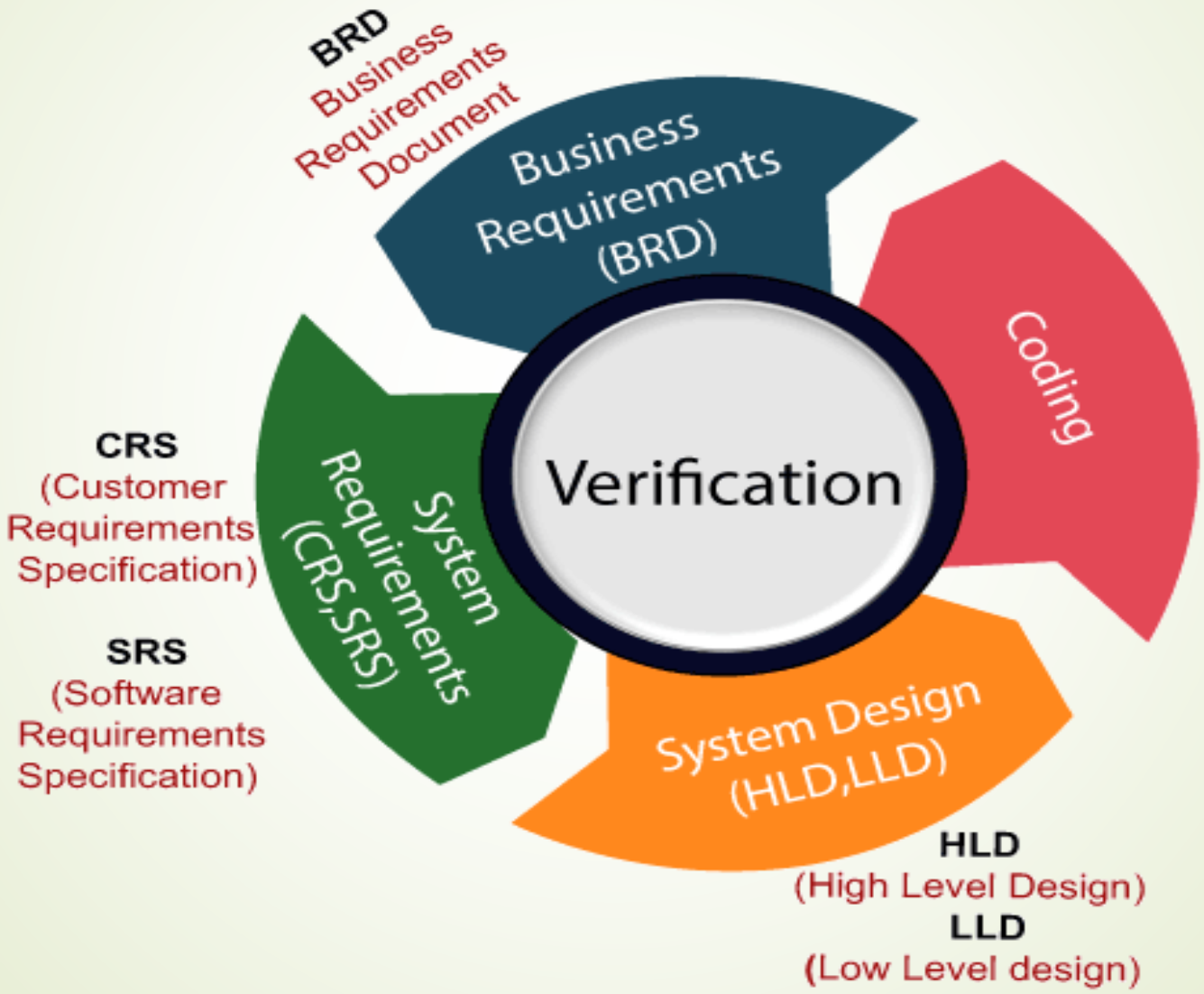
Verification Testing

- Verification is the process of evaluating work-products of a development phase to determine whether they meet the specified requirements.
- Verification ensures that the product is built according to the requirements and design specifications.
- It also answers to the question, Are we building the product, right?

Validation Testing

- Verification testing includes different activities such as business requirements, system requirements, design review, and code walkthrough while developing a product.
- It is also known as static testing, where we are ensuring that "**we are developing the right product or not**".

Verification Testing



Difference between Verification and Validation Testing

- Verification and Validation is the process of investigating that a software system satisfies specifications and standards, and it fulfills the required purpose. **Barry Boehm** described verification and validation as the following:
- *Verification: Are we building the product, right?*
- *Validation: Are we building the right product?*

Validation and Verification Testing

S. No.	Verification	Validation
1	It consists of checking of documents/files and is performed by human.	It consists of execution of program and is performed by computer.
2	**It comes before validation.**	**It comes after verification**
3	Quality assurance team does verification	Validation is executed on software code with the help of testing team.
4	The goal of verification is application and software architecture and specification	The goal of validation is an actual product.
5	It can find the bugs in the early stage of the development	It can only find the bugs that could not be found by the verification process

Validation and Verification Testing

S. No.	Verification	Validation
6	Verification is the static testing.	Validation is the dynamic testing.
7	It does *not* include the execution of the code	Validation is the dynamic testing.
8	QA team does verification and make sure that the software is as per the requirement in the SRS document.	With the involvement of testing team validation is executed on software code.
9	Methods used in verification are reviews, walkthroughs, inspections and desk-checking.	Methods used in validation are Black Box Testing, White Box Testing and non-functional testing.

Validation Testing:- Alpha & Beta Testing

- The alpha test is conducted at the developers site by a customer.
- The software is used in a natural setting with the developer "looking over the shoulder" of the user and recording errors and usage problems.
- Alpha tests are conducted in a controlled environment.
- The beta test is conducted at one or more customer sites by the end-user of the software.
- Unlike alpha testing, the developer is generally not present. Therefore, the beta test is a "live" application of the software in an environment that cannot be controlled by the developer.
- The customer records all problems that are encountered during beta testing and reports these to the developer at regular intervals.
- As a result of problems reported during beta tests, software engineers make modifications and then prepare for release of the software product to the entire customer base.

Alpha Testing

- Alpha Testing is a type of acceptance testing; performed to identify all possible issues and bugs before releasing the final product to the end users.
- Alpha testing is carried out by the testers who are internal employees of the organization.
- The main goal is to identify the tasks that a typical user might perform and test them.
- The focus of alpha testing is to simulate real users by using a black box and white box techniques.

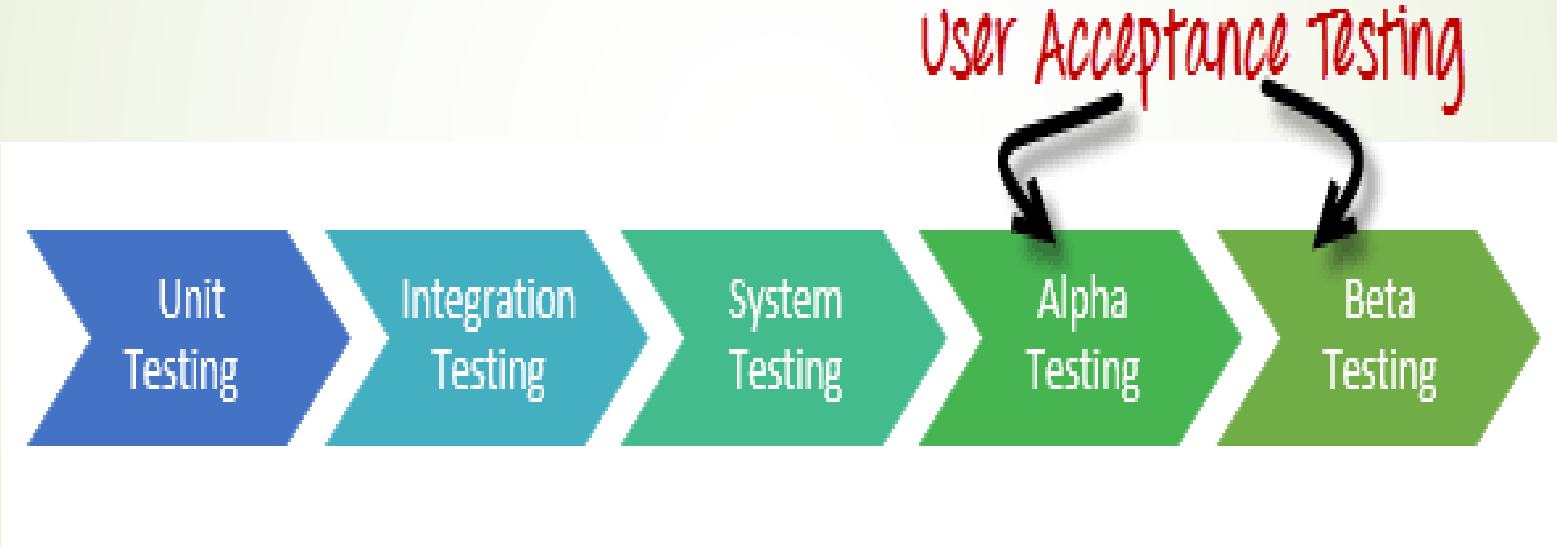
Beta Testing

- Beta Testing is a type of acceptance testing; it is the final test before shipping a product to the customers.
- Beta testing of a product is implemented by "real users “of the software application in a "real environment.”
- Direct feedback from customers is a major advantage of Beta Testing. This testing helps to test products in customer’s environment.

Beta Testing…

- Beta version of the software is released to a limited number of end-users of the product to obtain feedback on the product quality.
- Beta testing reduces product failure risks and provides increased quality of the product through customer validation.

Alpha Beta Testing



Difference between Alpha and Beta Testing

S. No.	Alpha	Beta
1	Alpha testing involves both the white box and black box testing.	Beta testing commonly uses black-box testing.
2	Alpha testing is performed by testers who are usually internal employees of the organization.	Beta testing is performed by clients who are not part of the organization.
3	Alpha testing is performed at the developer's site.	Beta testing is performed at the end-user of the product.
4	Reliability and security testing are not checked in alpha testing.	Reliability, security and robustness are checked during beta testing.
5	Alpha testing ensures the quality of the product before forwarding to beta testing.	Beta testing also concentrates on the quality of the product but collects users input on the product and ensures that the product is ready for real time users.

Difference between Alpha and Beta Testing…

S. No.	Alpha	Beta
1	Alpha testing requires a testing environment or a lab.	Beta testing doesn't require a testing environment or lab.
2	Alpha testing may require a long execution cycle.	Beta testing requires only a few weeks of execution.
3	Developers can immediately address the critical issues or fixes in alpha testing.	Most of the issues or feedback collected from the beta testing will be implemented in future versions of the product.
4	Multiple test cycles are organized in alpha testing.	Only one or two test cycles are there in beta testing.

System Testing

- software is incorporated with other system elements e.g., hardware, people, information, and a series of system integration and validation tests are conducted.
- System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system.

1. **Recovery Testing**
2. **Security Testing**
3. **Stress Testing**
4. **Performance Testing**

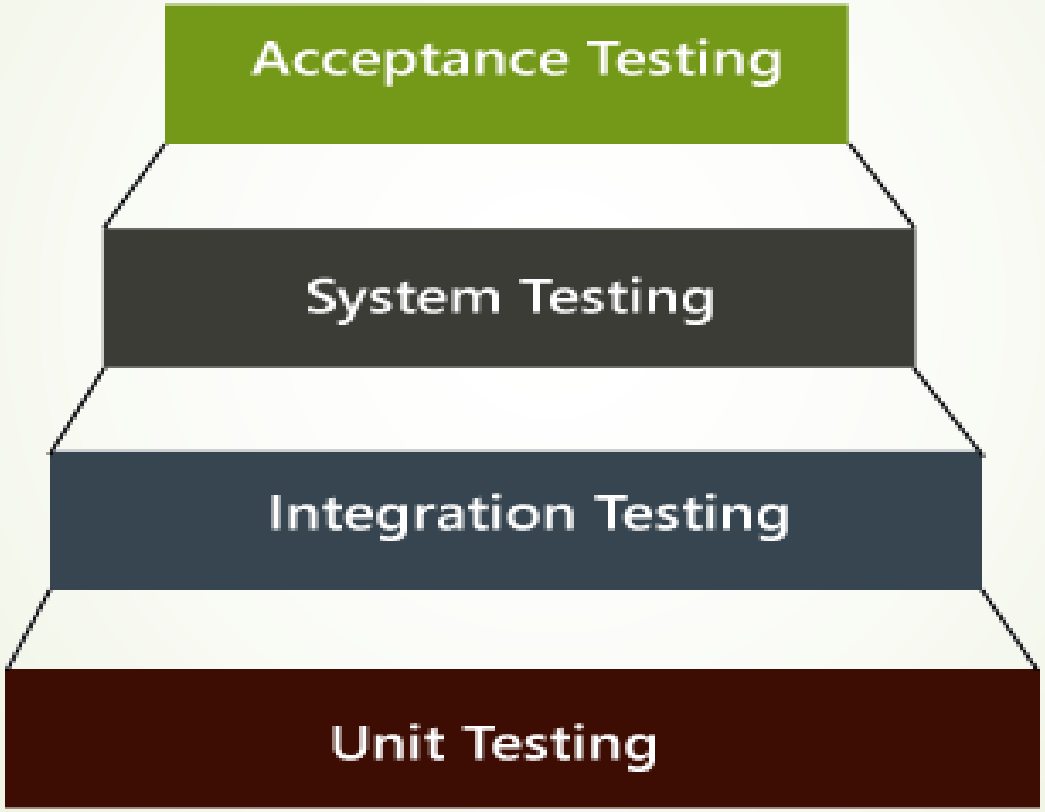
System Testing …

- System Testing is a level of testing that validates the complete and fully integrated software product.
- The purpose of a system test is to evaluate the end-to-end system specifications.
- The goal of integration testing is to detect any irregularity between the units that are integrated together.
- System testing detects defects within both the integrated units and the whole system.

System Testing …..

- The result of system testing is the observed behavior of a component or a system when it is tested.
- System testing tests the design and behavior of the system and the expectations of the customer.
- It is performed to test the system beyond the bounds mentioned in the software requirements specification (SRS).
- System Testing is basically performed by a testing team that is independent of the development team that helps to test the quality of the system impartial.

System Testing



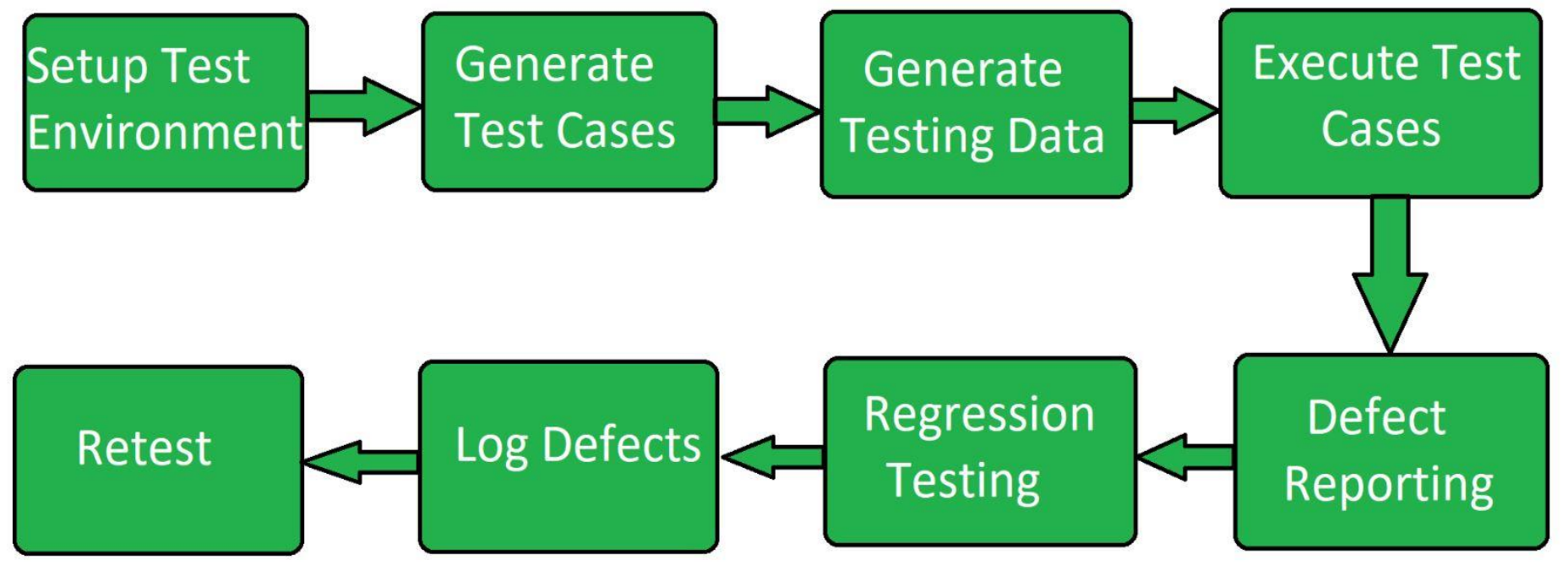
System Testing Process

- Test Environment Setup: Create testing environment for the better-quality testing.
- Create Test Case: Generate test case for the testing process.
- Create Test Data: Generate the data that is to be tested.
- Execute Test Case: After the generation of the test case and the test data, test cases are executed.

System Testing ……

- **Defect Reporting:** Defects in the system are detected.
- **Regression Testing:** It is carried out to test the side effects of the testing process.
- **Log Defects:** Defects are fixed in this step.
- **Retest:** If the test is not successful then again test is performed.

Process of System Testing Life Cycle



Types of System Testing

1. Recovery Testing

- Many computer based systems must recover from faults and resume processing within a pre-specified time.
- Recovery testing is a system test that forces the software to fail in a variety of ways and verifies that recovery is properly performed.
- If recovery is:
  - Automatic recovery,
  - reinitialization,
  - Checkpointing
  - data recovery, and
  - restart are evaluated for correctness.

Types of System Testing ….

2. Security Testing

- Security testing attempts to verify that protection mechanisms built into a system will, in fact, protect it from improper penetration.
- During security testing, the tester plays the role(s) of the individual who desires to penetrate the system.
- Anything goes! The tester may attempt to acquire passwords through external clerical means; may attack the system with custom software designed to breakdown any defenses that have been constructed; may overwhelm the system, thereby denying service to others; may purposely cause system errors, hoping to penetrate during recovery; may browse through insecure data, hoping to find the key to system entry.

Types of System Testing ….

3. Stress Testing

- Stress testing executes a system in a manner that demands resources in abnormal quantity, frequency, or volume.

(1) special tests may be designed that generate ten interrupts per second, when one or two is the average rate

(2) input data rates may be increased by an order of magnitude to determine how input functions will respond,

(3) test cases that require maximum memory or other resources are executed, (4) test cases that may cause thrashing in a virtual operating system are designed,

(5) test cases that may cause excessive hunting for disk-resident data are created. Essentially, the tester attempts to break the program.

variation of stress testing is a technique called sensitivity testing.

Types of System Testing ….

4. Performance Testing

- For real-time and embedded systems, software that provides required function but does not conform to performance requirements is unacceptable Performance testing is designed to test the run-time performance of software within the context of an integrated system.
- Performance testing occurs throughout all steps in the testing process.
- Performance tests are often coupled with stress testing and usually require both hardware and software instrumentation

Types of System Testing…

5. Load Testing: Load Testing is a type of software Testing which is carried out to determine the behavior of a system or software product under extreme load.

6. Scalability Testing: Scalability Testing is a type of software testing which is carried out to check the performance of a software application or system in terms of its capability to scale up or scale down the number of user request load.

Debugging

- To launch an application into the market, it is very necessary to cross-check it multiple times to deliver an error-free product.
- In the context of software engineering, debugging is the process of fixing a bug in the software. In other words, it refers to identifying, analyzing, and removing errors.
- Software programs undergo heavy testing, updating, troubleshooting, and maintenance during the development process.

Debugging……..

- Debugging is a developer activity and effective debugging is very important before testing begins to increase the quality of the system.
- Debugging is a developer activity and effective debugging is very important before testing begins to increase the quality of the system.
- Debugging will not give confidence that the system meets its requirements completely, but testing gives confidence.

Why do we need Debugging

- The process of debugging begins as soon as the code of the software is written. Then, it continues in successive stages as code is combined with other units of programming to form a software product. *Debugging has many benefits such as:*

✓ It **reports** an **error condition immediately**. This allows earlier detection of an error and makes the process of software development stress-free and unproblematic.

✓ It also provides **maximum useful information** of data structures and allows easy interpretation.

Why do we need Debugging ….

- ✓ Debugging assists the developer in **reducing useless** and **distracting information**.
- ✓ Through debugging the developer can avoid **complex one-use testing code** to save time and energy in software development.

Steps involved in Debugging

Steps involved in Debugging

1) Identify the Error: A bad identification of an error can lead, to wasted developing time. It is usual that production errors reported by users are hard to interpret and sometimes the information we receive is misleading. It is import to identify the actual error.
2) Find the Error Location: After identifying the error correctly, you need to go through the code to find the exact spot where the error is located. In this stage, you need to focus on finding the error instead of understanding it.

Steps involved in Debugging…..

3) Analyze the Error: In the third step, you need to use a bottom-up approach from the error location and analyze the code. This helps you in understanding the error.

4) Prove the Analysis: Once you are done analyzing the original bug, you need to find a few more errors that may appear on the application. This step is about writing automated tests for these areas with the help of a test framework.

Steps involved in Debugging…….

5) Cover Lateral Damage: In this stage, you need to create or gather all the unit tests for the code where you are going to make changes. Now, if you run these unit tests, they all should pass.

6) Fix & Validate: The final stage is the fix all the errors and run all the test scripts to check if they all pass.

Debugging Tools

- Debugging tool is a computer program that is used to test and debug other programs. A lot of public domain software like *gdb* and *dbx* are available for debugging. They offer console-based command-line interfaces. Examples of automated debugging tools include *code-based tracers*, *profilers*, *interpreters*, etc. Some of the widely used debuggers are:

✓ *Radare2*

✓ *WinDbg*

✓ *Valgrind*

Testing Conventional Applications

- These techniques provide systematic guidance for designing tests that
- ✓ (1) exercise the internal logic and interfaces of every software component and
- ✓ (2) exercise the input and output domains of the program to uncover errors in program function, behavior, and performance

Types of System Testing…

- *Load Testing:* Load Testing is a type of software Testing which is carried out to determine the behavior of a system or software product under extreme load.
- *Scalability Testing:* Scalability Testing is a type of software testing which is carried out to check the performance of a software application or system in terms of its capability to scale up or scale down the number of user request load.

Testing Conventional Applications

What is it?

- ✓ Once source code has been generated, software must be tested to uncover (and correct) as many errors as possible before delivery to your customer.
- ✓ Your goal is to design a series of test cases that have a high likelihood of finding errors—but how?
- ✓ That's where software testing techniques enter the picture.

Testing Conventional Applications

- **Who does it?**
- During early stages of testing, a software engineer performs all tests.
- However, as the testing process progresses, testing specialists may become involved

Testing Conventional Applications

What are the steps?

- ✓ For conventional applications, software is tested from two different perspectives:
- ✓ (1) internal program logic is exercised using “white box” test-case design techniques and
- ✓ (2) software requirements are exercised using “black box” test-case design techniques.
- ✓ Use cases assist in the design of tests to uncover errors at the software validation level.
- ✓ In every case, the intent is to find the maximum number of errors with the minimum amount of effort and time.

Testing Conventional Applications

- **What is the work product?**

✓ A set of test cases designed to exercise both internal logic, interfaces, component collaborations, and external requirements is designed and documented, expected results are defined, and actual results are recorded.

Testing Conventional Applications

- **How do you ensure that you've done it right?**

✓ When you begin testing, change your point of view.

✓ Try hard to "break" the software! Design test cases in a disciplined fashion and review the test cases you do create for thoroughness.

✓ In addition, you can evaluate test coverage and track error detection activities.

Testing Object Oriented Applications

What is it?

- ✓ The architecture of object-oriented (OO) software results in a series of layered subsystems that collaborating classes.
- ✓ Each of these system elements (subsystems and classes) performs functions that help to achieve system requirements.
- ✓ It is necessary to test an OO system at a variety of different levels in an effort to uncover errors that may occur as classes collaborate with one another and subsystems communicate across architectural layers

Testing Object Oriented Applications

- **Who does it?**

✓ Object-oriented testing is performed by software engineers and testing specialists.

- **Why is it important?**

✓ You must execute the program before it gets to the customer with the specific intent of removing all errors, so that the customer will not experience the frustration associated with a poor-quality product.

✓ In order to find the highest possible number of errors, tests must be conducted systematically, and test cases must be designed using disciplined techniques.

Testing Object Oriented Applications

What are the steps?

- ✓ OO testing is strategically analogous to the testing of conventional systems, but it is intentionally different.
- ✓ Because the OO analysis and design models are similar in structure and content to the resultant OO program, “testing” is initiated with the review of these models.
- ✓ Once code has been generated, OO testing begins “in the small” with class testing.
- ✓ A series of tests are designed that exercise class operations and examine whether errors exist as one class collaborates with other classes

Testing Object Oriented Applications

- **What is the work product?**
  - ✓ A set of test cases, designed to exercise classes, them
  - ✓ collaborations, and behaviors is designed and documented; expected results are defined, and actual results are recorded.
- **How do you ensure that you've done it right?**
  - ✓ When you begin testing, change your point of view.
  - ✓ Try hard to "break" the software! Design test cases in a disciplined fashion, and review the tests cases you do create for thoroughness

ANY QUERY