CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Object Oriented Software Engineering

LECTURE 13

Outline

Software Testing strategies and Tactics:

A Strategic approach for Software Testing,

Software Testing Strategies:

Unit Testing,

Integration Testing,

Validation Testing,

System Testing,

Testing Strategy for Object-Oriented Software-

Unit testing in OO Context

Integration testing in OO Context

CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Today's Agenda:

Software Testing Strategies and Tactics :

- A strategic approach for Software Testing
- Software Testing Strategies:
    - Unit Testing
    - Integration Testing
    - Validation Testing
    - System Testing

Introduction to Testing:

- Software testing is the process of evaluating a software application to identify if it meets specified requirements and to identify any defects.
- Testing is a set of activities which are decided in advance i.e. before the start of development and organized systematically.
- All the strategies give a testing template.

Software Testing Strategies :

- Software is tested to uncover errors introduced during design and construction. Testing often accounts for more project effort than other s/e activity. Hence it has to be done carefully using a testing strategy. The strategy is developed by the project manager, software engineers and testing specialists.
- Testing is the process of execution of a program with the intention of finding errors involves 40% of total project cost. Testing Strategy provides a road map that describes the steps to be conducted as part of testing. It should incorporate test planning, test case design, test execution and resultant data collection and execution.

CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Software Testing Strategies :

- Validation refers to a different set of activities that ensures that the software is traceable to the Customer requirements.

- V&V encompasses a wide array of Software Quality Assurance

A strategic Approach for Software testing

- Testing is a set of activities that can be planned in advance and conducted systematically. Testing strategy should have the following characteristics:
- -- usage of Formal Technical reviews(FTR)
- -- Begins at component level and covers entire system
- -- Different techniques at different points
- -- conducted by developer and test group
- --should include debugging

A strategic Approach for Software testing

- Software testing is one element of **verification and validation.**
- Verification refers to the set of activities that ensure that software correctly implements a specific function. (Ex: Are we building the product right?)
- Validation refers to the set of activities that ensure that the software built is traceable to customer requirements. (Ex: Are we building the right product ?)

CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Testing Strategy

- Testing can be done by software developer and independent testing group. Testing and debugging are different activities.
- Debugging follows testing Low level tests verifies small code segments. High level tests validate major system functions against customer requirements

CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Who Tests the Software?

developer

Understands the system
but, will test "gently"
and, is driven by "*delivery*"

independent tester

Must learn about the system,
but, will attempt to **break** it
and, is driven by *quality*

Characteristics of Testable Software

Operable – The better it works (i.e., better quality), the easier it is to test

• Observable – Incorrect output is easily identified; internal errors are automatically detected

• Controllable – The states and variables of the software can be controlled directly by the tester

• Decomposable – The software is built from independent modules that can be tested independently

• Simple – The program should exhibit functional, structural, and code simplicity

• Stable – Changes to the software during testing are infrequent and do not invalidate existing tests

• Understandable – The architectural design is well understood; documentation is available and organized

Test Characteristics

- A good test has a high probability of finding an error – The tester must understand the software and how it might fail
- A good test is not redundant – Testing time is limited; one test should not serve the same purpose as another test
- A good test should be “best of breed” – Tests that have the highest likelihood of uncovering a whole class of errors should be used
- A good test should be neither too simple nor too complex – Each test should be executed separately; combining a series of tests could cause side effects and mask certain errors

A strategy for software testing

- Integrates the design of software test cases into a well-planned series of steps that result in successful development of the software.
- The strategy provides a road map that describes the steps to be taken, when, and how much effort, time, and resources will be required.
- The strategy incorporates test planning, test case design, test execution, and test result collection and evaluation.
- The strategy provides guidance for the practitioner and a set of milestones for the manager.
- Because of time pressures, progress must be measurable and problems must surface as early as possible

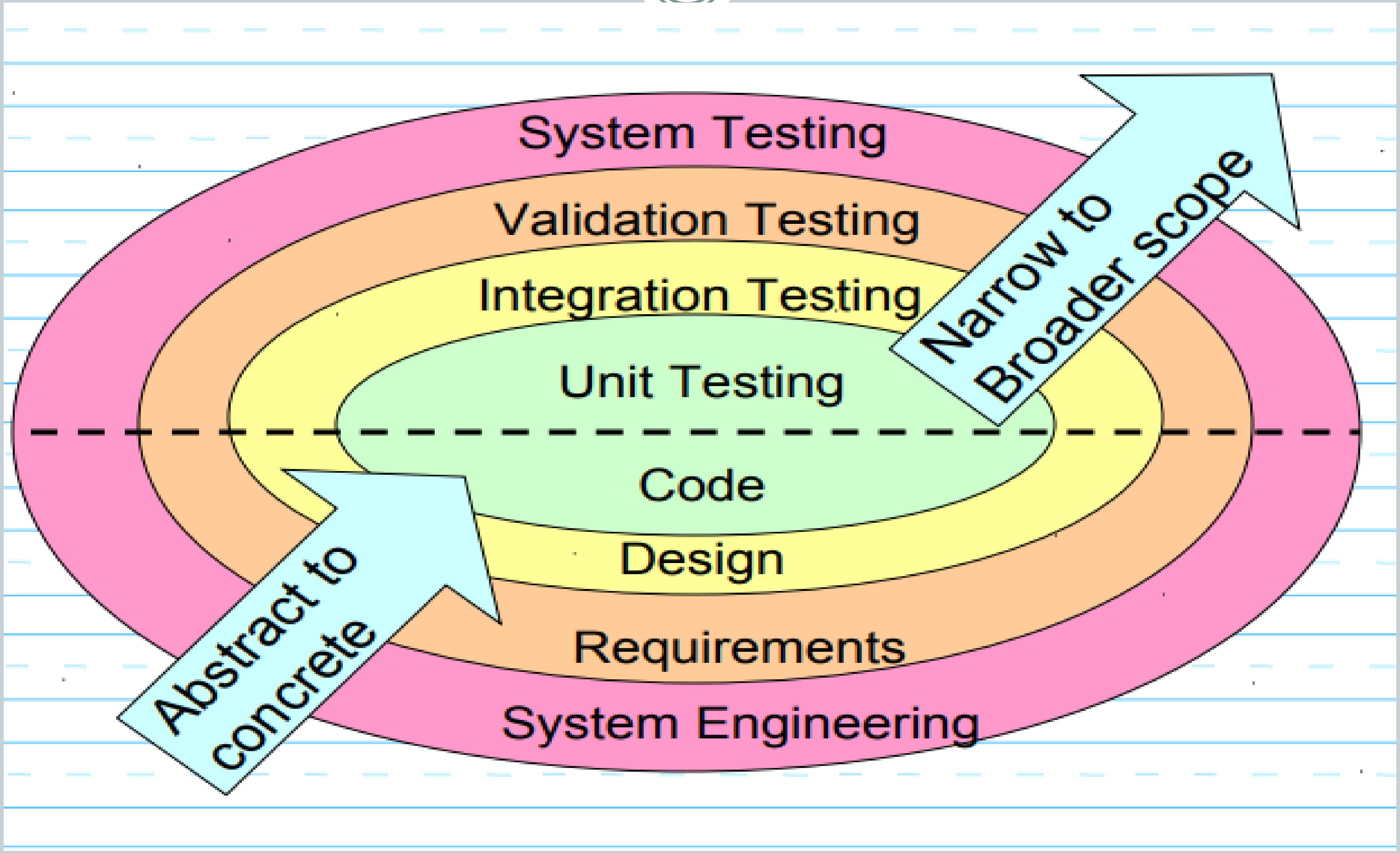
A Strategic Approach to Testing

General Characteristics of Strategic Testing

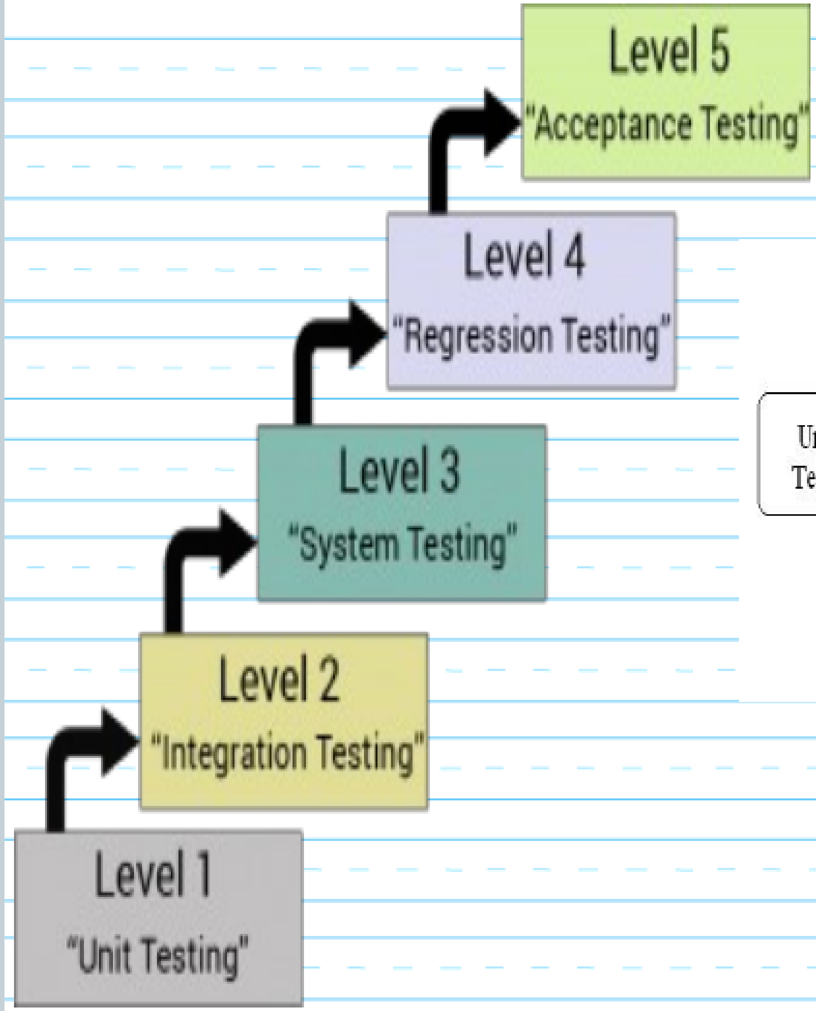
- To perform effective testing, a software team should conduct effective formal technical reviews.
- Testing begins at the component level and work outward toward the integration of the entire computer-based system.
- Different testing techniques are appropriate at different points in time.
- Testing is conducted by the developer of the software and (for large projects) by an independent test group.
- Testing and debugging are different activities, but debugging must be accommodated in any testing strategy

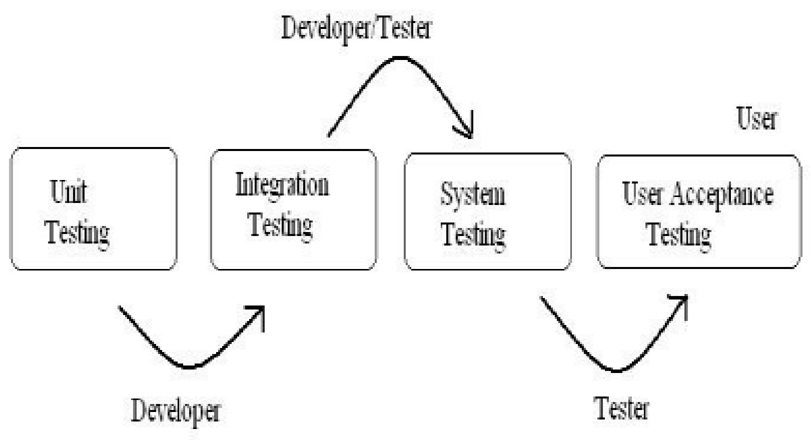
CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Spiral Representation of Testing for Conventional Software



Levels of Testing





CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Test Strategies for Conventional Software:

- Testing Strategies for Conventional Software can be viewed as a spiral consisting of four levels of testing:

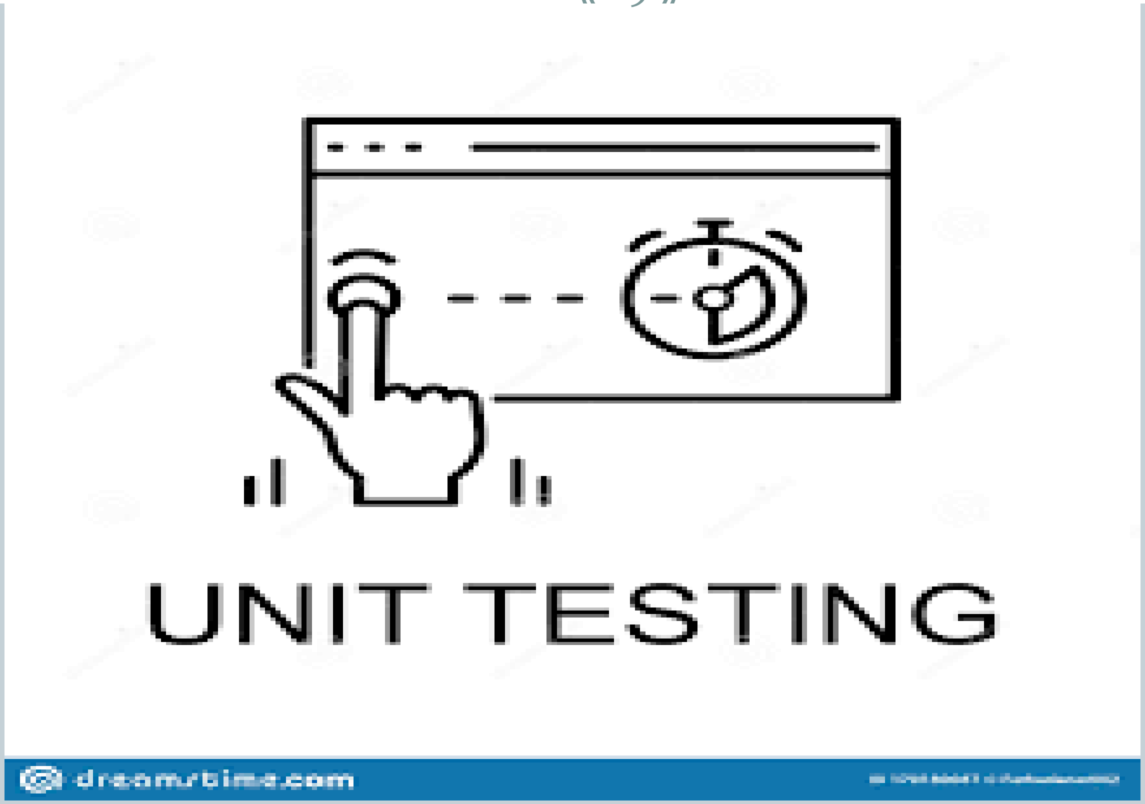
1) Unit Testing

2) Integration Testing

3)Validation Testing and

4) System Testing

I. Unit Testing

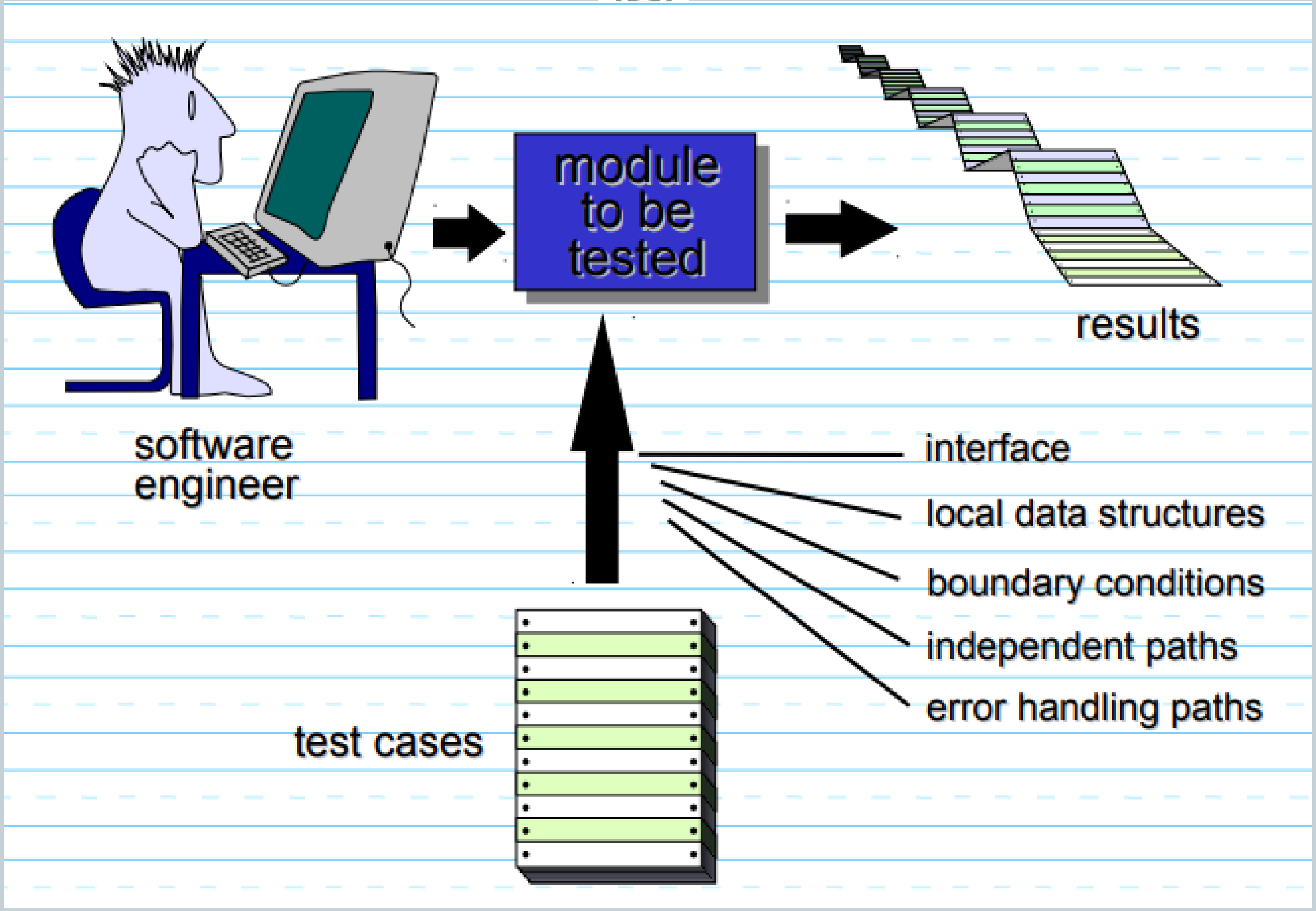


CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Unit Testing

- Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation.
- Unit testing is often automated but it can also be done manually.
  - • Algorithms and logic
  - • Data structures (global and local)
  - • Interfaces
  - • Independent paths
  - • Boundary conditions
  - • Error handling

Unit Testing



Unit Testing

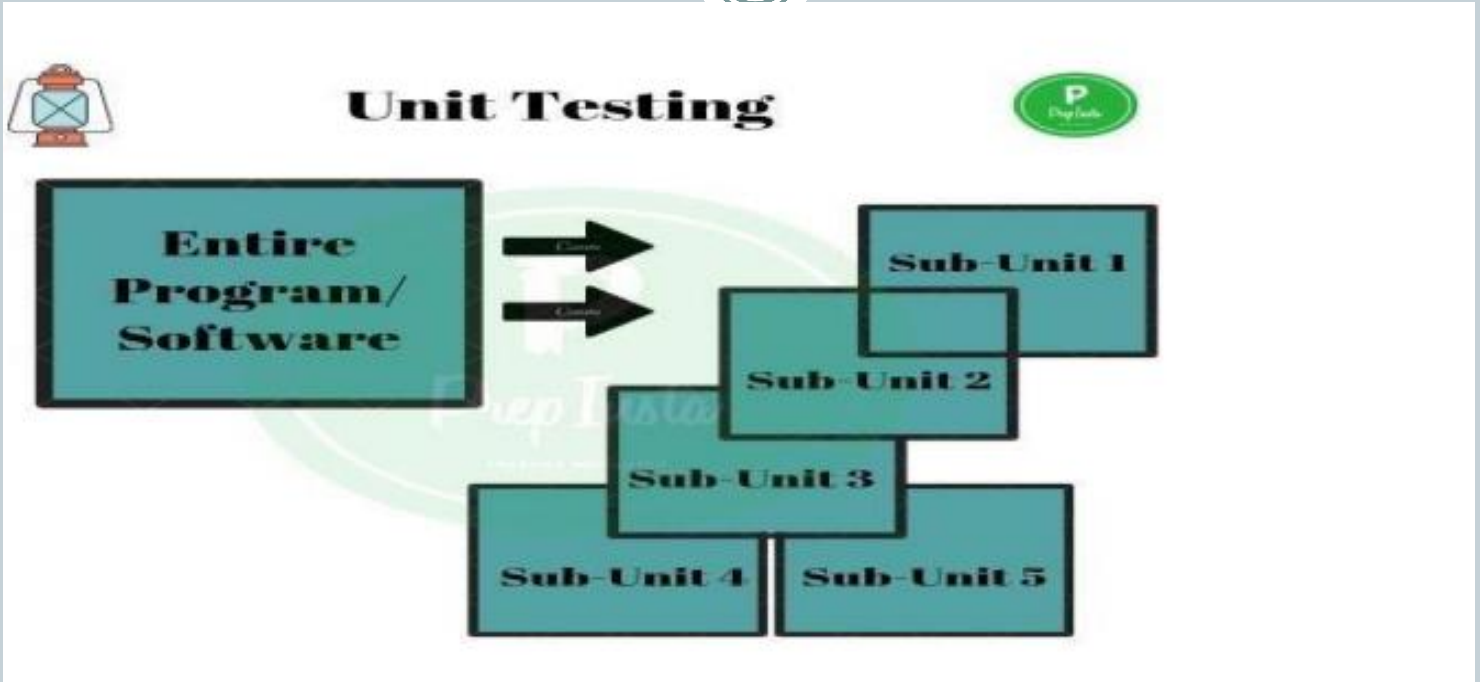


Fig: Unit Testing

Boundary testing also should be done as s/w usually fails at its boundaries.
Unit tests can be designed before coding begins or after source code is generated.

Unit Testing

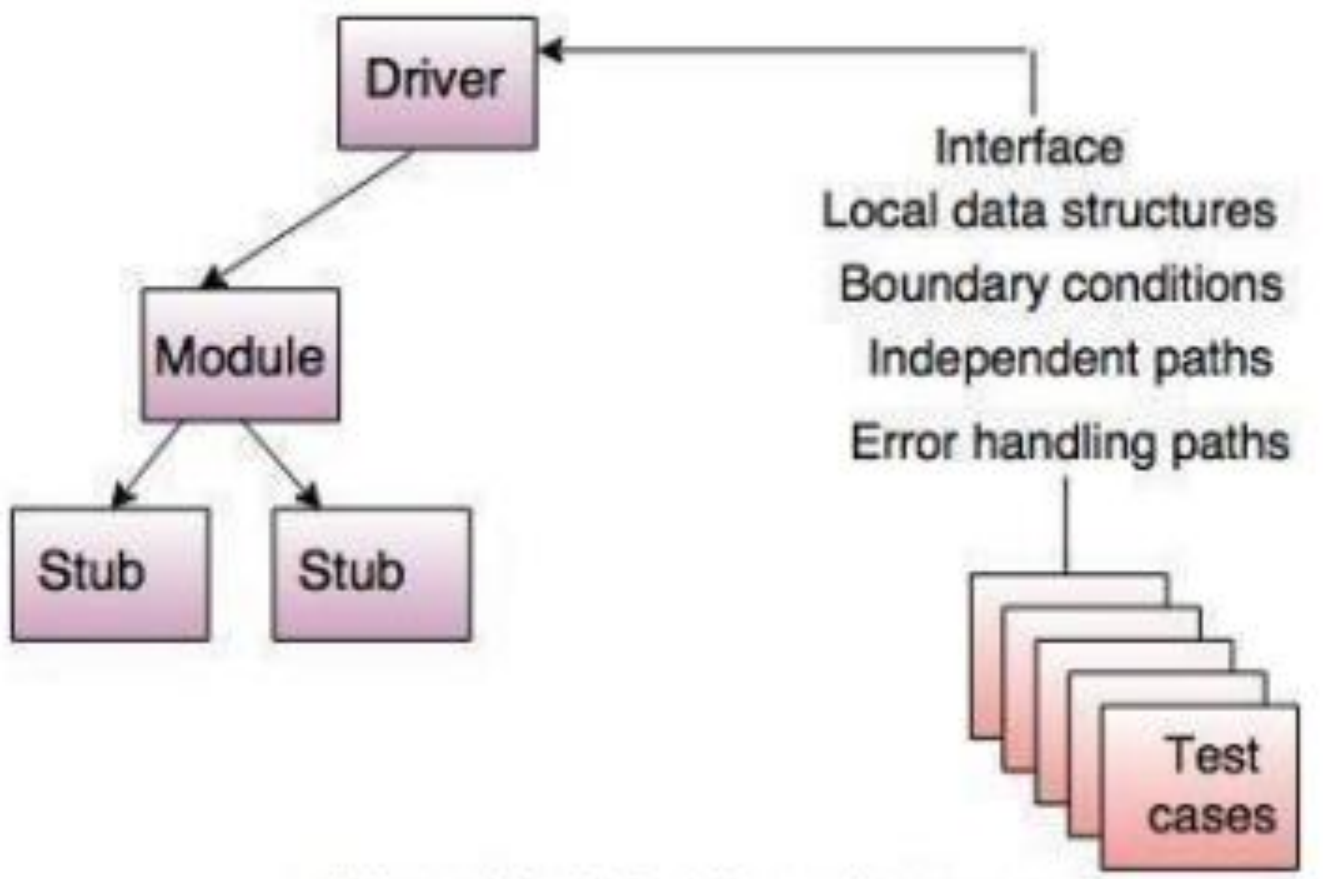
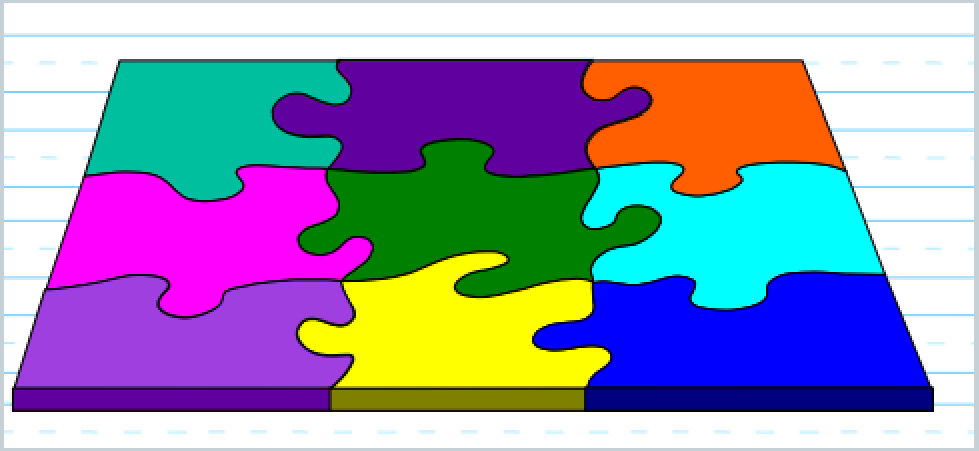


Fig. - Unit test environment

II. Integration Testing



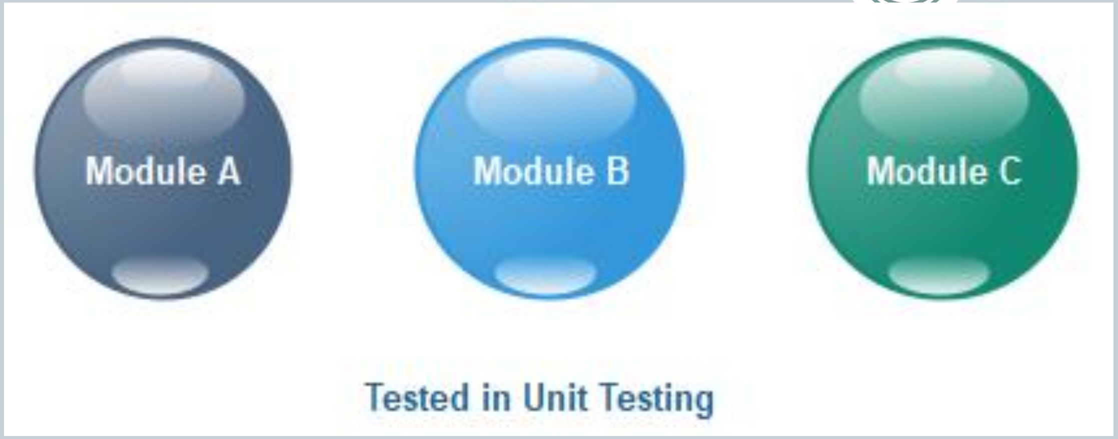
Integration testing:

- Integration testing is the second level of the software testing process comes after unit testing. In this testing, units or individual components of the software are tested in a group. The focus of the integration testing level is to expose defects at the time of interaction between integrated components or units.
- Unit testing uses modules for testing purpose, and these modules are combined and tested in integration testing. The Software is developed with a number of software modules that are coded by different coders or programmers. The goal of integration testing is to check the correctness of communication among all the modules.

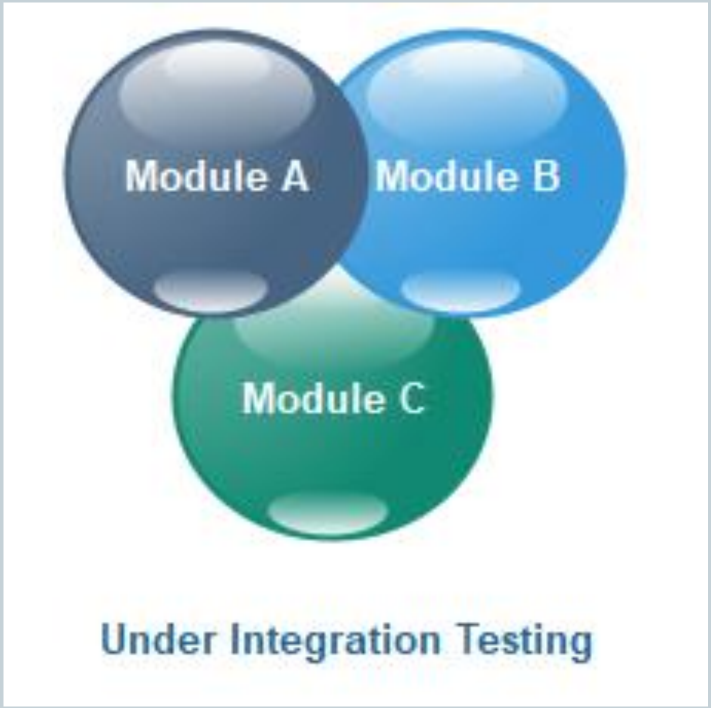
Integration testing:

- Integration Testing focuses on checking data communication amongst these modules. Hence it is also termed as **'I & T'** (Integration and Testing), **'String Testing'** and sometimes **'Thread Testing'**.

Integration testing:



Once all the components or modules are working independently, then we need to check the data flow between the dependent modules is known as integration testing.



Integration testing:

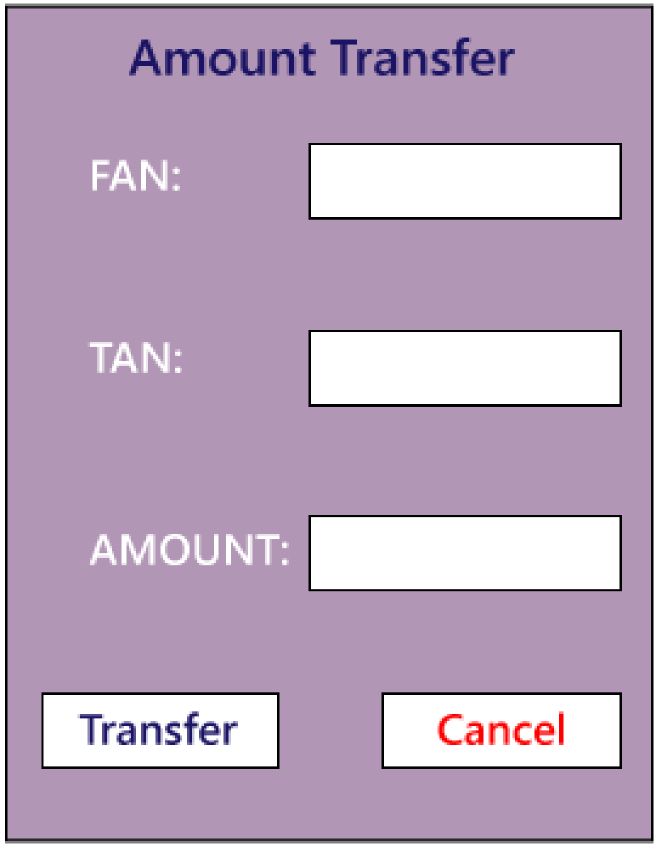
- In this the focus is on design and construction of the software architecture. It addresses the issues associated with problems of verification and program construction by testing inputs and outputs. Though modules function independently problems may arise because of interfacing. This technique uncovers errors associated with interfacing. We can use top-down integration wherein modules are integrated by moving downward through the control hierarchy, beginning with the main control module. The other strategy is bottom –up which begins construction and testing with atomic modules which are combined into clusters as we move up the hierarchy. A combined approach called Sandwich strategy can be used i.e., top- down for higher level modules and bottom-up for lower level modules.

Example of Integration testing:

•First, we will login as a user **P** to amount transfer and send Rs200 amount, the confirmation message should be displayed on the screen as **amount transfer successfully**. Now logout as P and login as user **Q** and go to amount balance page and check for a balance in that account = Present balance + Received Balance. Therefore, the integration test is successful.

•Also, we check if the amount of balance has reduced by Rs200 in P user account.

•Click on the transaction, in P and Q, the message should be displayed regarding the data and time of the amount transfer.



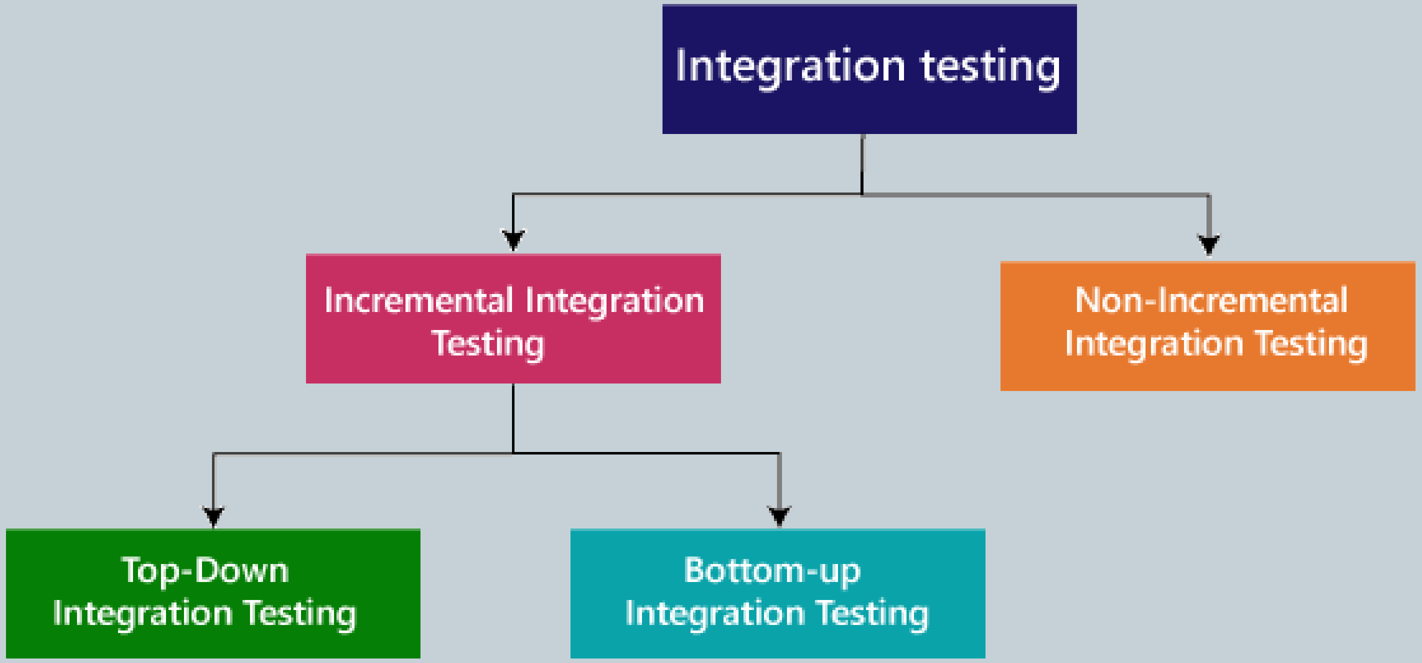
Reason Behind Integration Testing

- Although all modules of software application already tested in unit testing, errors still exist due to the following reasons:
- Each module is designed by individual software developer whose programming logic may differ from developers of other modules so; integration testing becomes essential to determine the working of software modules.
- To check the interaction of software modules with the database whether it is an erroneous or not.
- Requirements can be changed or enhanced at the time of module development. These new requirements may not be tested at the level of unit testing hence integration testing becomes mandatory.
- Incompatibility between modules of software could create errors.
- To test hardware's compatibility with software.
- If exception handling is inadequate between modules, it can create bugs.

CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Types of Integration Testing

- **Incremental integration testing**
- **Non-incremental integration testing**


Integration testing
Incremental Integration Testing
Non-Incremental Integration Testing
Top-Down Integration Testing
Bottom-up Integration Testing

Types of Integration Testing

Incremental integration testing

A. Top-down Integration

B. Bottom-Up Integration

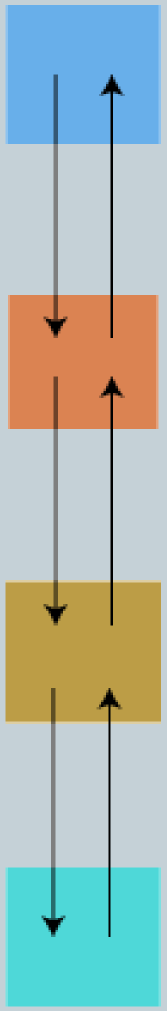
C. Hybrid Integration

Non-Incremental integration testing

A. Big-Bang Integration

Incremental Approach /Incremental integration testing

- In the Incremental Approach, modules are added in ascending order one by one or according to need. The selected modules must be logically related. Generally, two or more than two modules are added and tested to determine the correctness of functions. The process continues until the successful testing of all the modules.



For example: Suppose we have a Flipkart application, we will perform incremental integration testing, and the flow of the application would like this:

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

CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Incremental integration

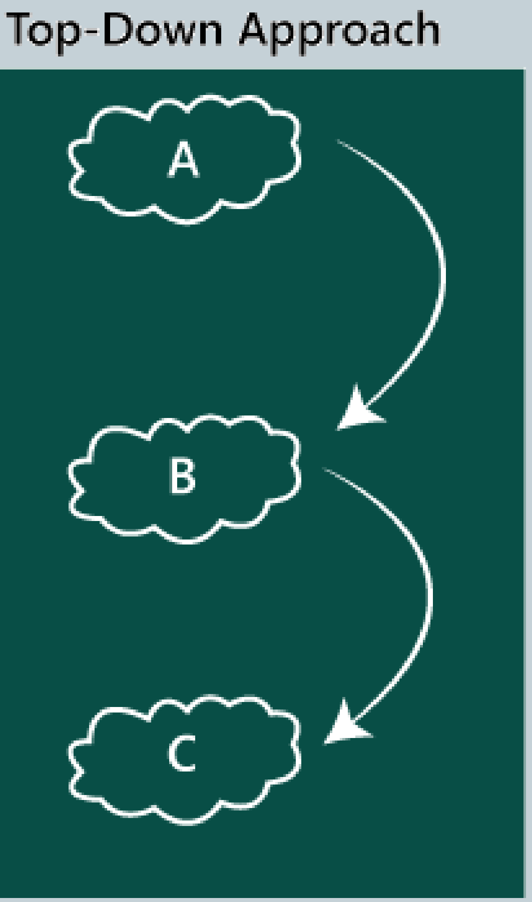
Incremental integration testing is carried out by further methods:

- Top-Down approach
- Bottom-Up approach

CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

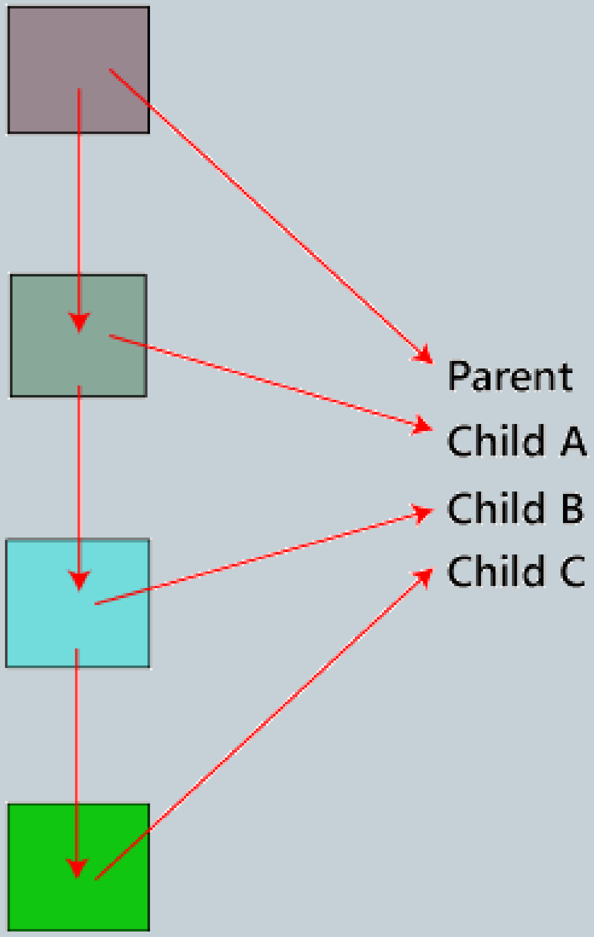
A. Top-Down Approach

- The top-down testing strategy deals with the process in which higher level modules are tested with lower level modules until the successful completion of testing of all the modules. Major design flaws can be detected and fixed early because critical modules tested first. In this type of method, we will add the modules incrementally or one by one and check the data flow in the same order.



Top-Down Approach

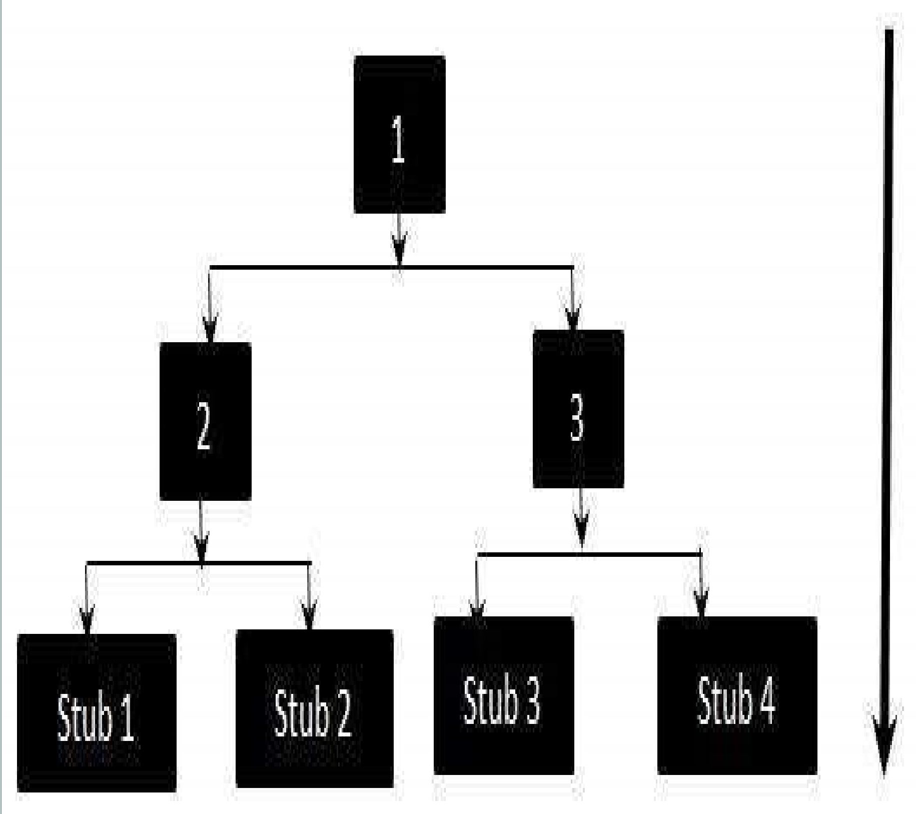
- In the top-down approach, we will be ensuring that the module we are adding is the **child of the previous one like Child C is a child of Child B** and so on as we can see in the below image:



Top-down integration

- Top-down integration testing is an integration testing technique used in order to simulate the behaviour of the lower-level modules that are not yet integrated. Stubs are the modules that act as temporary replacement for a called module and give the same output as that of the actual product.
- The replacement for the 'called' modules is known as '**Stubs**' and is also used when the software needs to interact with an external system.

Stub - Flow Diagram:



The given diagrams clearly states that Modules 1, 2 and 3 are available for integration, whereas, below modules are still under development that cannot be integrated at this point of time. Hence, Stubs are used to test the modules. The order of Integration will be:

1,2
1,3
2,Stub 1
2,Stub 2
3,Stub 3
3,Stub 4

Stubs and Drivers

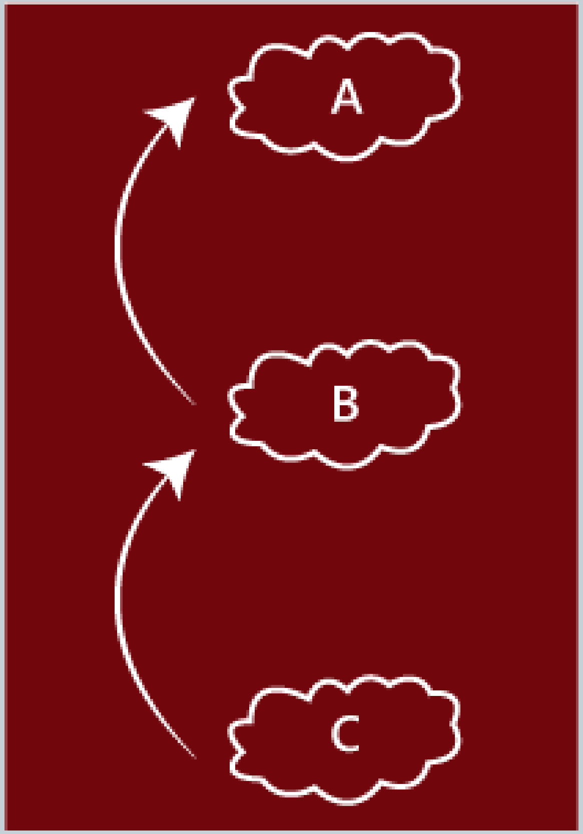
- **Stubs and Drivers** are the dummy programs in Integration testing used to facilitate the software testing activity. These programs act as a substitutes for the missing models in the testing. They do not implement the entire programming logic of the software module but they simulate data communication with the calling module while testing.
- **Stub**: Is called by the Module under Test.
- **Driver**: Calls the Module to be tested.

CHITKARA UNIVERSITY HIMACHAL PRADESH - NAAC Accredited

B. Bottom-Up Method

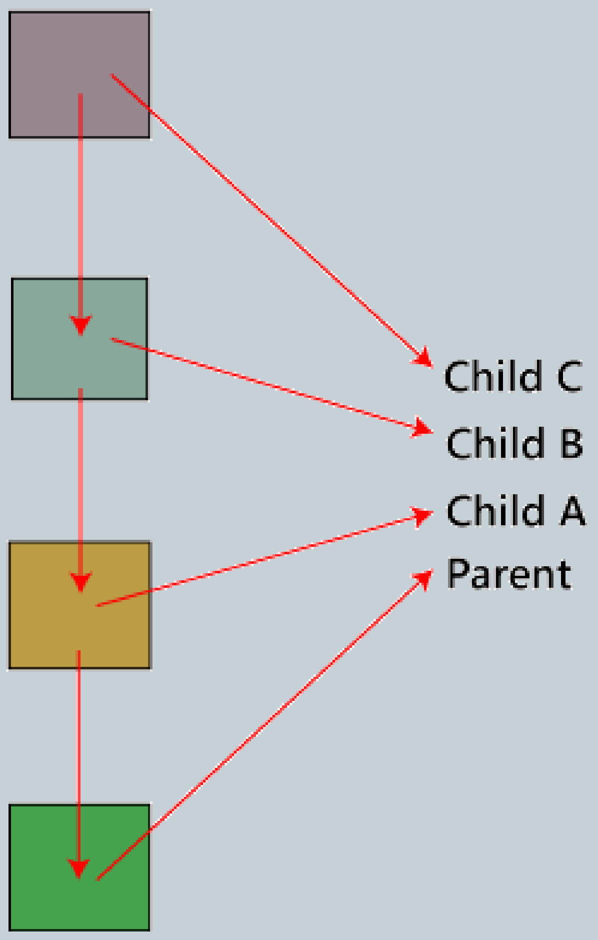
- The bottom to up testing strategy deals with the process in which lower level modules are tested with higher level modules until the successful completion of testing of all the modules. Top level critical modules are tested at last, so it may cause a defect. Or we can say that we will be adding the modules from **bottom to the top** and check the data flow in the same order.

Bottom-up Appoarch



Bottom-Up Method

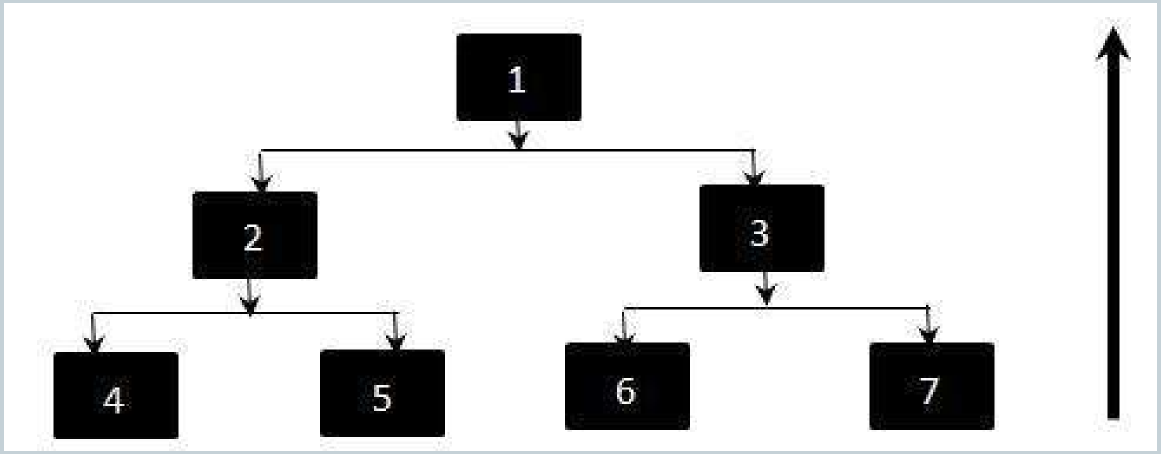
- In the bottom-up method, we will ensure that the modules we are adding **are the parent of the previous one** as we can see in the below image:



Bottom-up integration

- Bottom-up testing is a specific type of integration testing that tests the lowest components of a code base first. More generally, it refers to a middle phase in software testing that involves taking integrated code units and testing them together, before testing an entire system or code base.

Bottom-up integration



The order of Integration by Bottom-down approach will be:

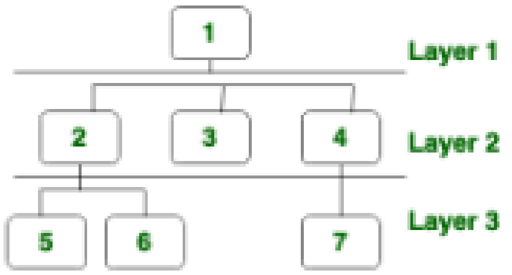
4,2
5,2
6,3
7,3
2,1
3,1

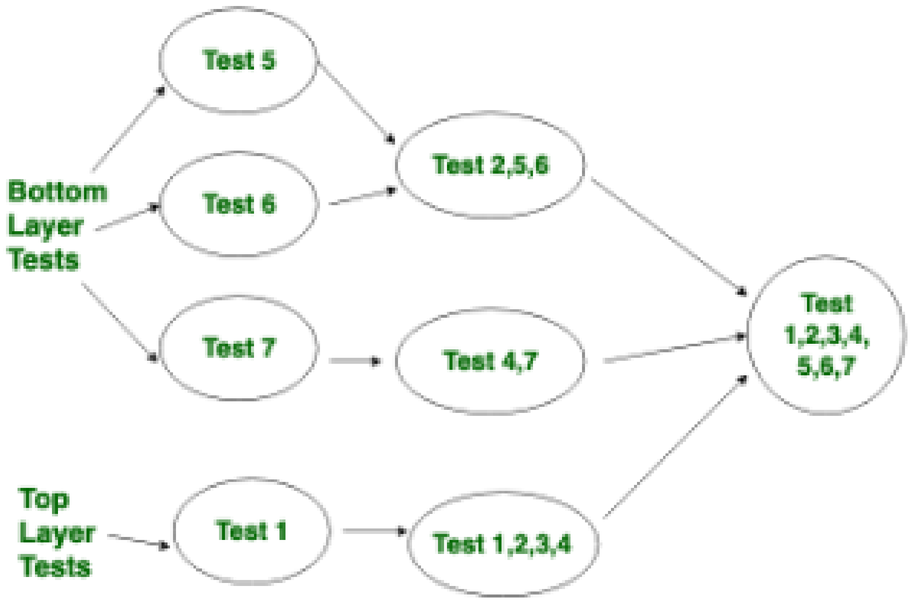
C. Sandwich/ Hybrid Integration

- **Sandwich Testing** is the combination of the bottom-up approach and top-down approach, so it uses the advantage of both the bottom-up approach and top-down approach. Initially, it uses the stubs and drivers where stubs simulate the behavior of a missing component. It is also known as **Hybrid Integration Testing.**

Example: Sandwich testing

For example:

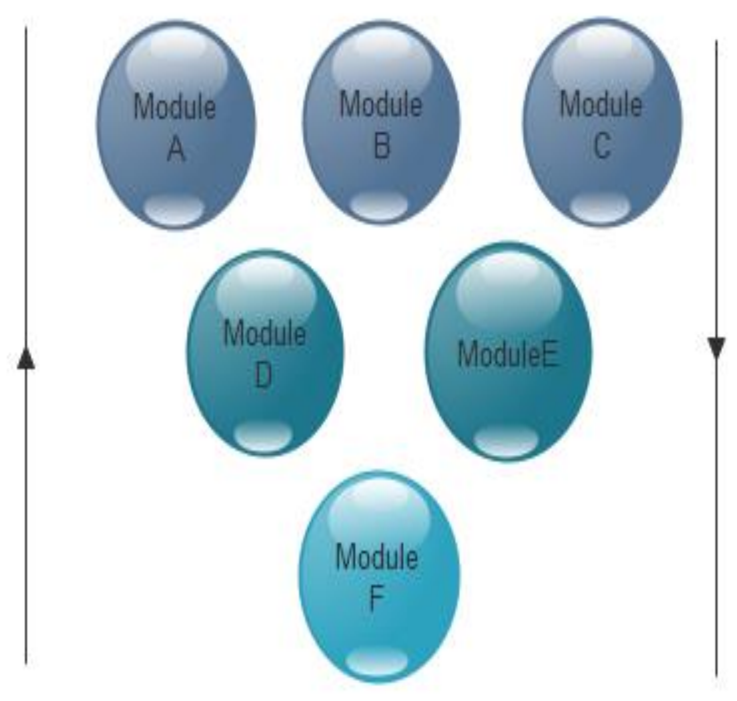




CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Hybrid Testing

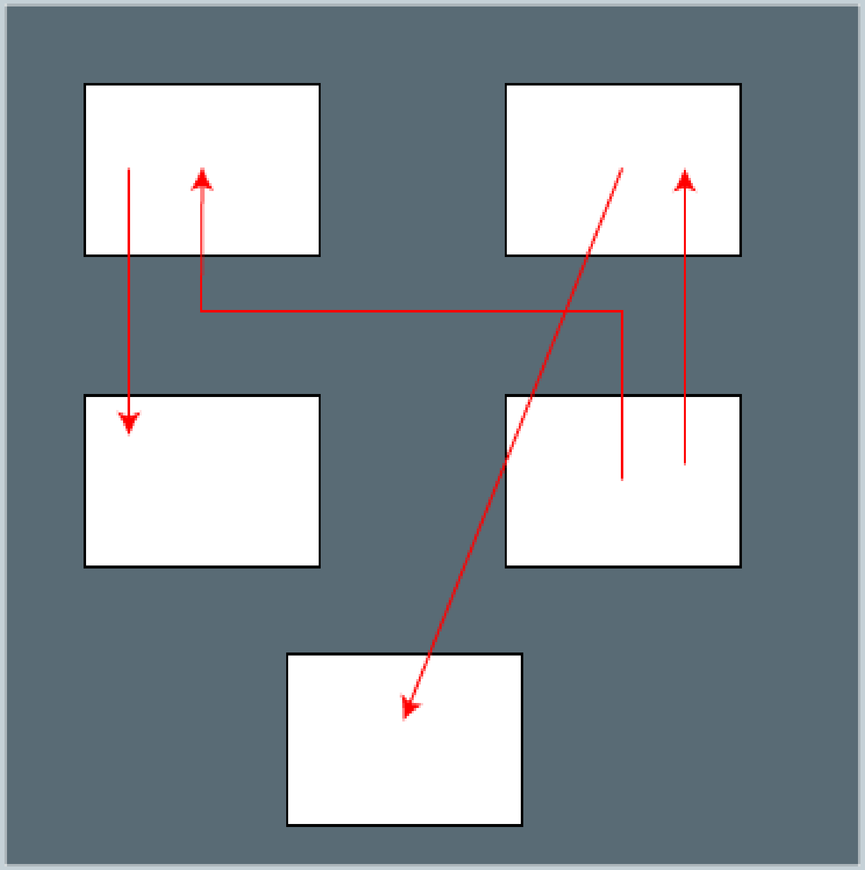
- In this approach, both **Top-Down** and **Bottom-Up** approaches are combined for testing. In this process, top-level modules are tested with lower level modules and lower level modules tested with high-level modules simultaneously. There is less possibility of occurrence of defect because each module interface is tested.



Hybrid Method

Non- incremental integration testing

- We will go for this method, when the data flow is very complex and when it is difficult to find who is a parent and who is a child. And in such case, we will create the data in any module bang on all other existing modules and check if the data is present. Hence, it is also known as the **Big bang method**.

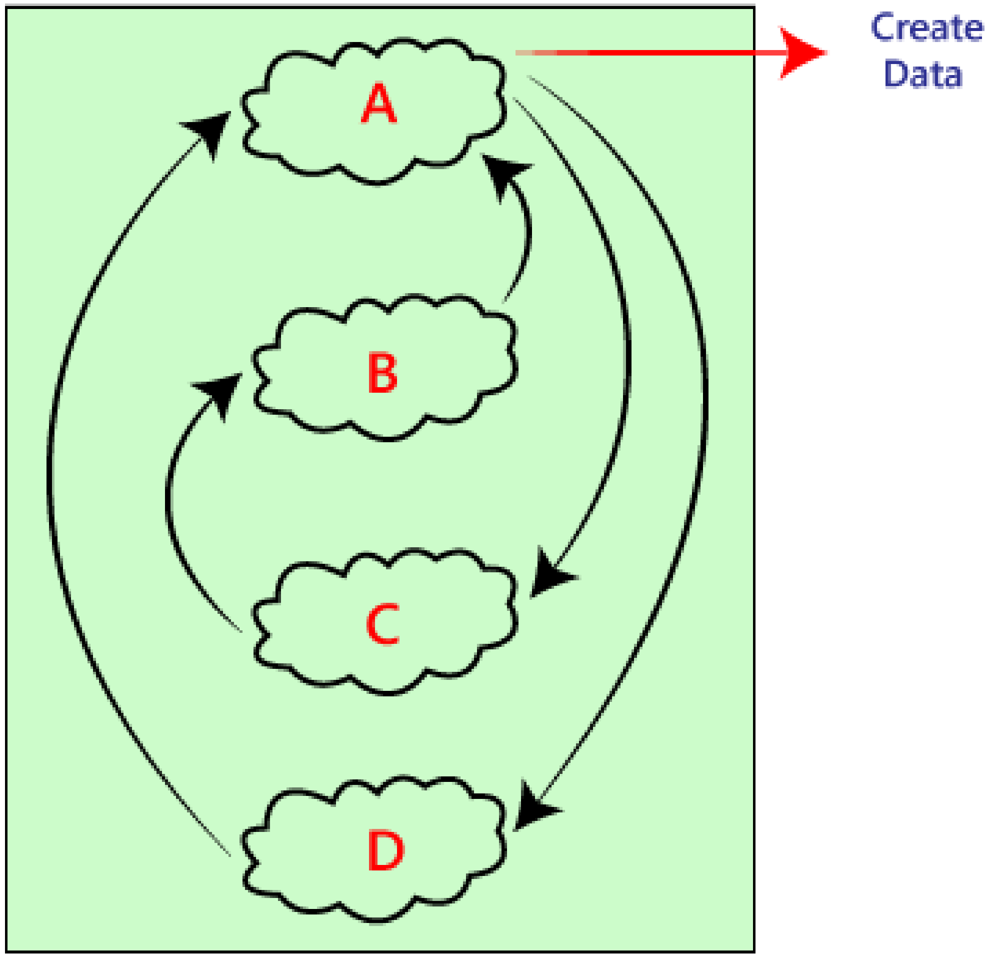


CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Big Bang Method

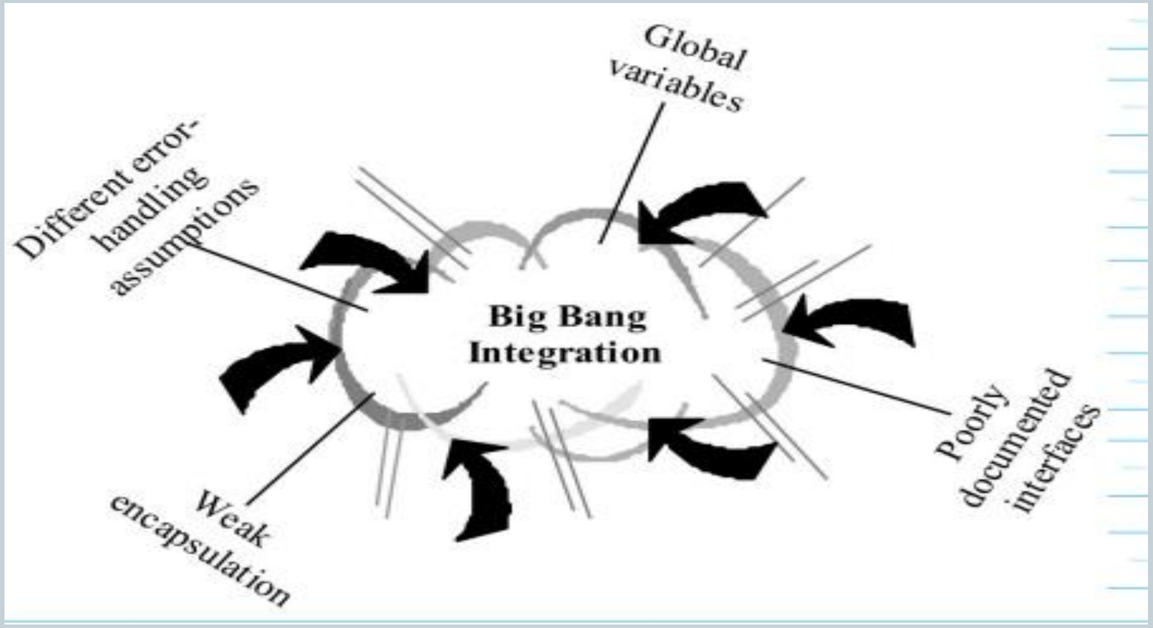
- In this approach, testing is done via integration of all modules at once. It is convenient for small software systems, if used for large software systems identification of defects is difficult.
- Since this testing can be done after completion of all modules due to that testing team has less time for execution of this process so that internally linked interfaces and high-risk critical modules can be missed easily.

Big Bang Method



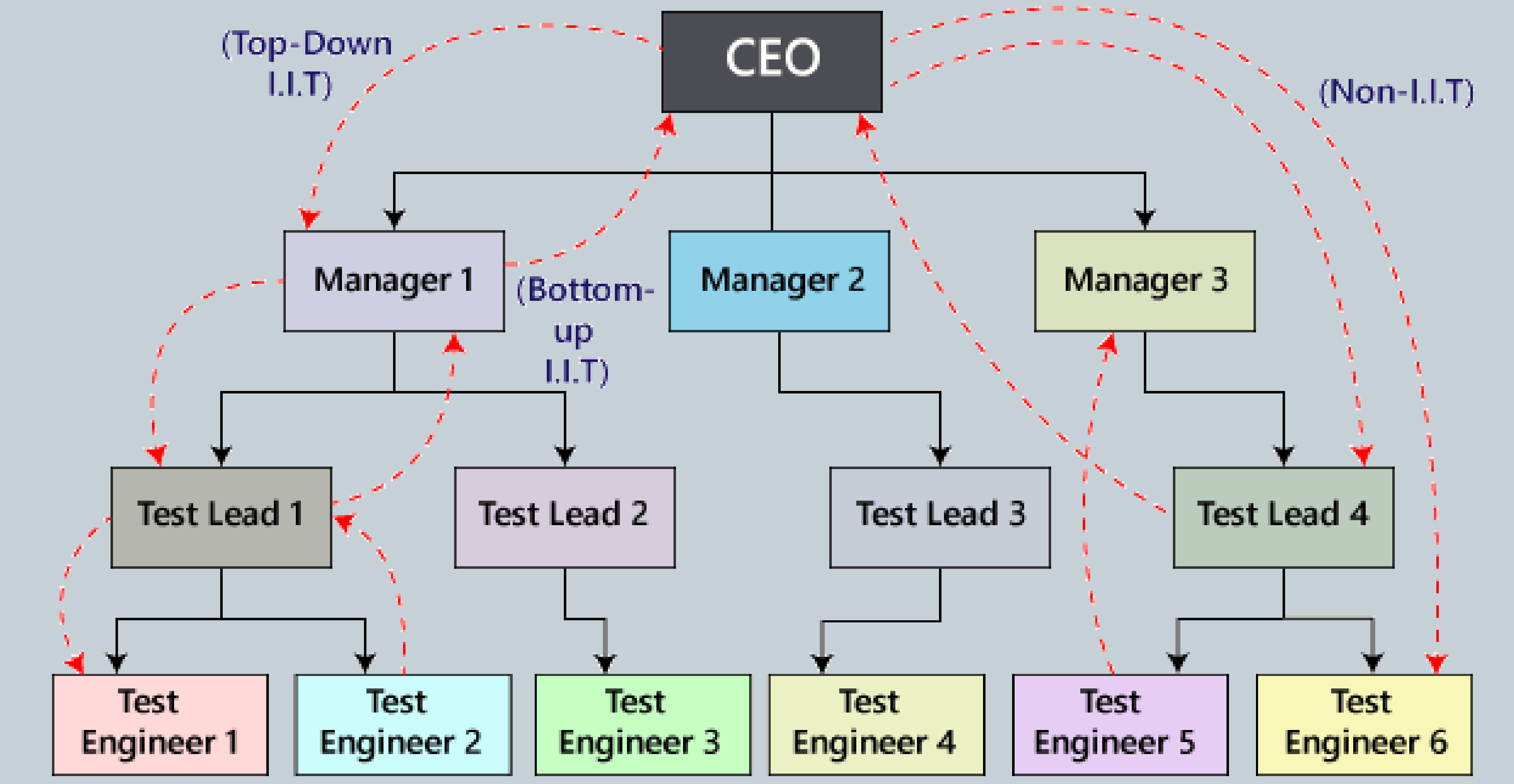
Big-Bang Integration

- Big Bang Integration Testing is an integration testing strategy wherein all units are linked at once, resulting in a complete system. When this type of testing strategy is adopted, it is difficult to isolate any errors found, because attention is not paid to verifying the interfaces across individual units.

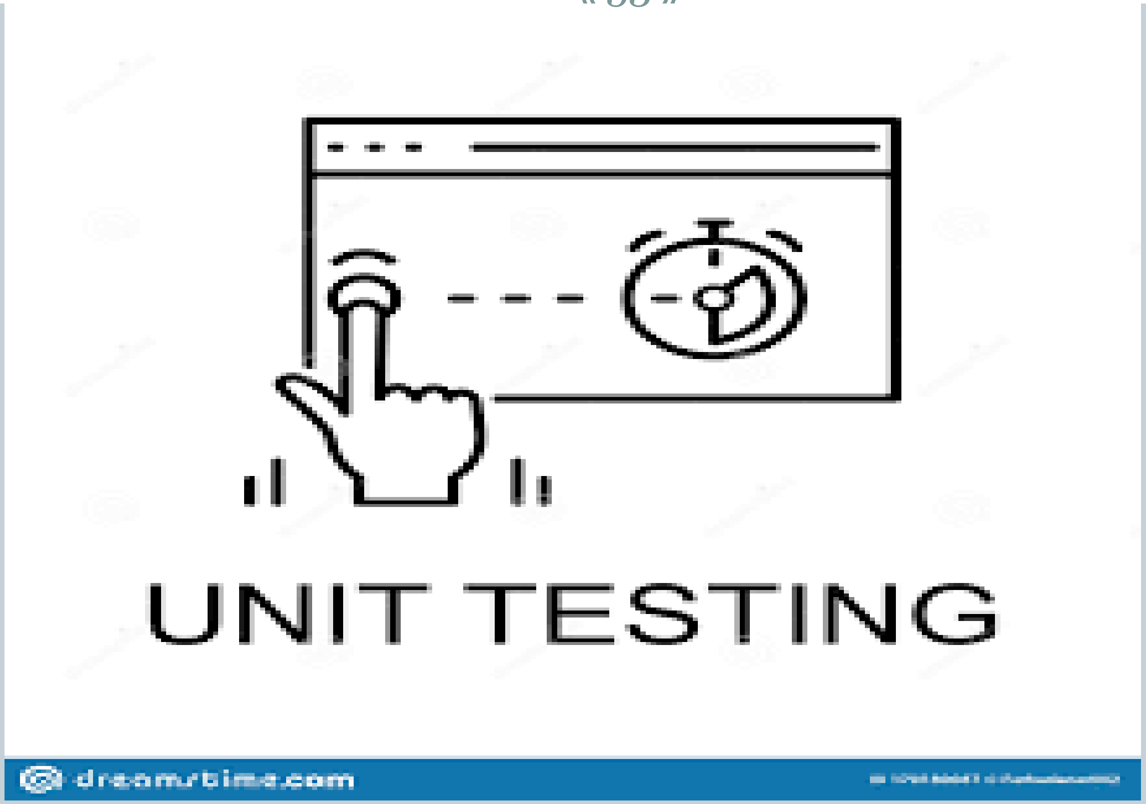

Global variables
Different error-handling assumptions
Big Bang Integration
Poorly documented interfaces
Weak encapsulation

Non-incremental integrating testing or big bang method: Example

- In the below example, the development team develops the application and sends it to the CEO of the testing team. Then the CEO will log in to the application and generate the username and password and send a mail to the manager. After that, the CEO will tell them to start testing the application.
- Then the manager manages the username and the password and produces a username and password and sends it to the **test leads**. And the **test leads** will send it to the **test engineers** for further testing purposes. This order from the CEO to the test engineer is **top-down incremental integrating testing.**
- In the same way, when the test engineers are done with testing, they send a report to the **test leads**, who then submit a report to the **manager**, and the manager will send a report to the **CEO**. This process is known as **Bottom-up incremental integration testing** as we can see in the below image:


CEO
Manager 1
Manager 2
Manager 3
Test Lead 1
Test Lead 2
Test Lead 3
Test Lead 4
Test Engineer 1
Test Engineer 2
Test Engineer 3
Test Engineer 4
Test Engineer 5
Test Engineer 6
(Top-Down I.I.T)
(Bottom-up I.I.T)
(Non-I.I.T)

III. Validation testing



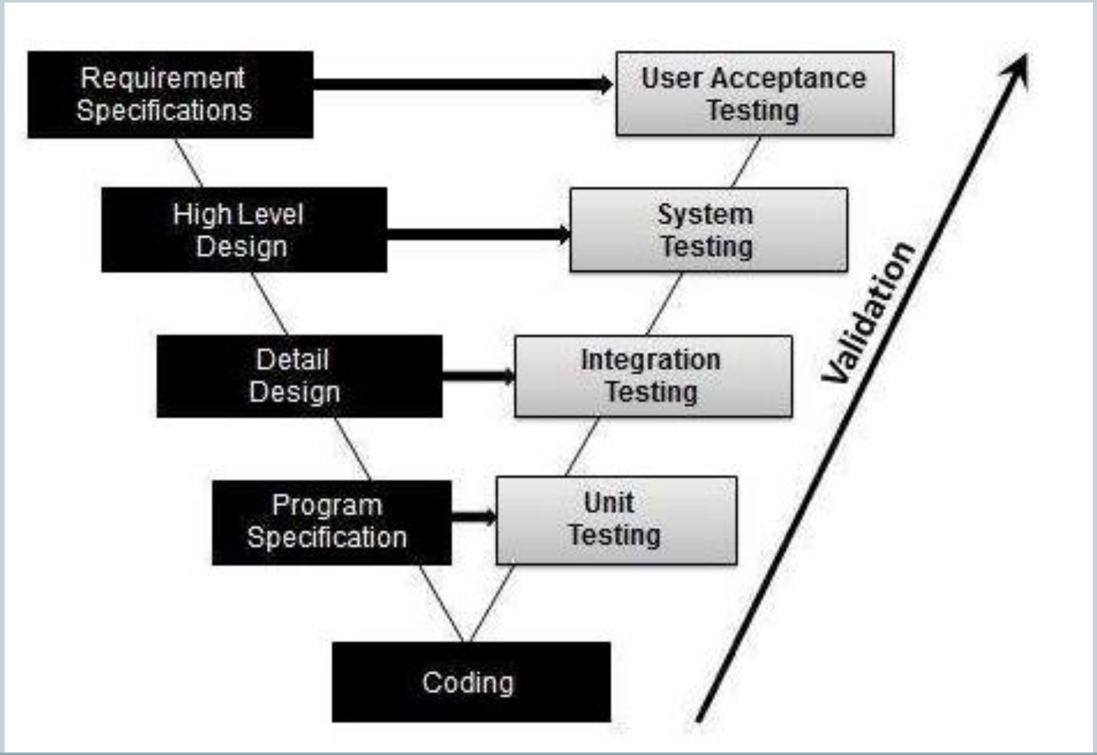
Validation Testing

- 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 actually 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.

CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Validation Testing - Workflow:

- Validation testing can be best demonstrated using V-Model. The Software/product under test is evaluated during this type of testing.



Validation Testing

- The definition of validation testing in software engineering is in place to determine if the existing system complies with the system requirements and performs the dedicated functions for which it is designed along with meeting the goals and needs of the organization. This mode of testing is extremely important especially if you want to be ***one of the best software testers***. The software verification and validation testing is the process after the validation testing stage is secondary to verification testing.

Why is Validation Testing Important?

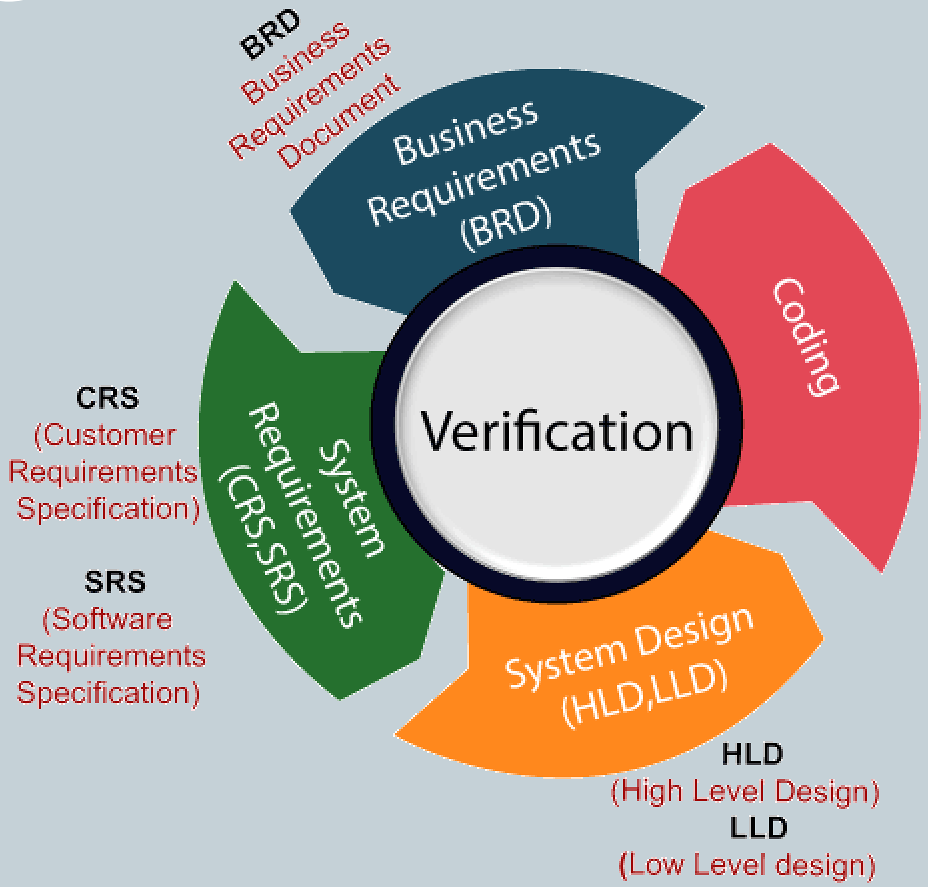
- *To ensure customer satisfaction*
- *To be confident about the product*
- *To fulfill the client's requirement until the optimum capacity*
- *Software acceptance from the end-user*
- Whenever any particular software is tested then the main motive is to check the quality against the defects being found. The developers fix the bugs and the software is rechecked to make sure that absolutely no bugs are left out in that. This not only shoots the product's quality but also its user acceptance

What are the Phases of the Validation Testing Process?

- **Define Requirements**– Mapping out the plan for the requirement gathering process. This will not only include planning out the entire process beforehand but also mapping out the exact requirements that are needed.
- **Team Selection**– To get a qualified, knowledgeable, and skilled team onboard. The team selection process involves selecting the individuals as per their past capacity and technical tuning so that they can easily attune themselves to the nature of the bug.
- **Maintaining Documentation**– Any form of testing requires voluminous user specification documentation along with several release cases, test cases, and manuals that have to be jolted down so that they are no confusion even in case of an exit of any core member of the team.
- **Validation Report**– The software is evaluated as per user specifications and a proper validation report is submitted in order to cross-check the evaluations along with getting an estimated date and round-off for the bug removal and for the system to start functioning properly.
- **Incorporation of changes**–Incorporate the changes that have been validated in the last stage.

Verification and Validation Testing

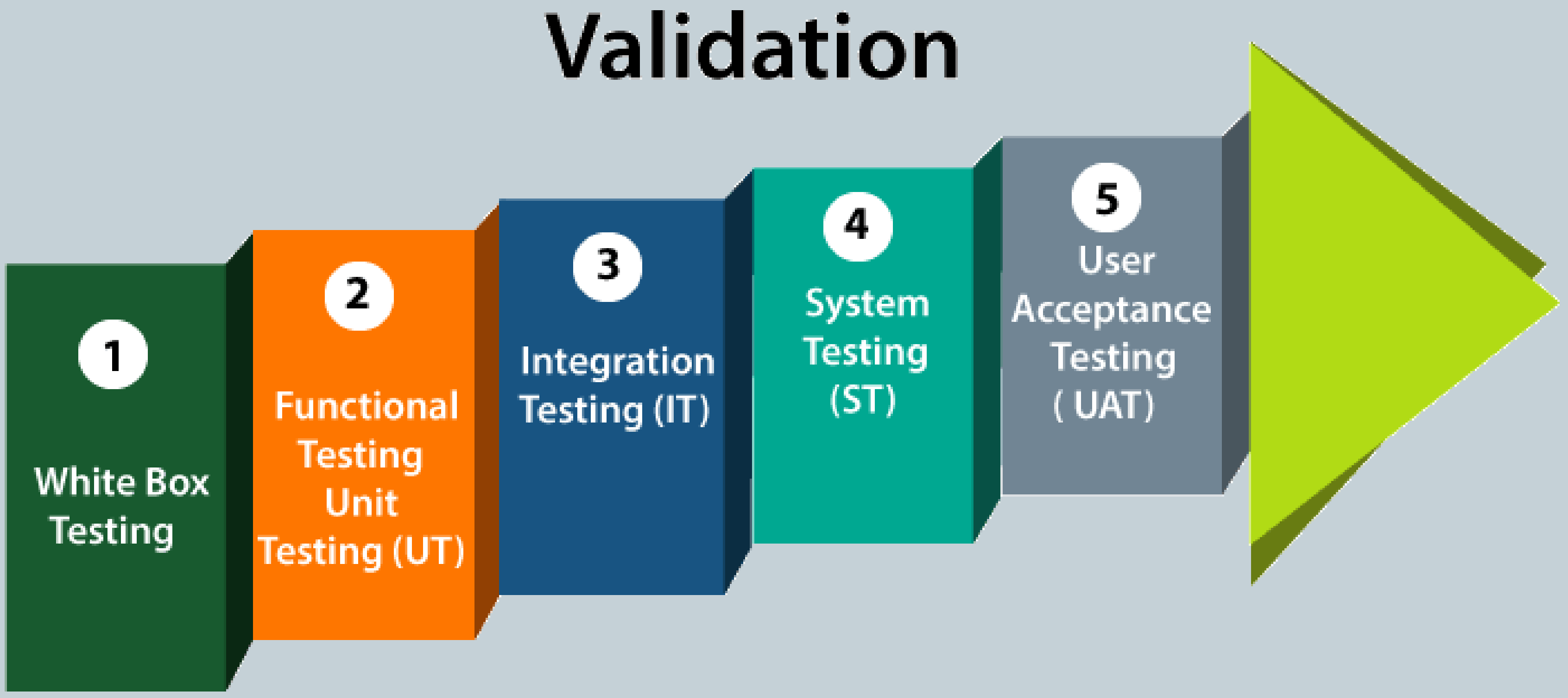
- **Verification testing**
- Verification testing includes different activities such as business requirements, system requirements, design review, and code walkthrough while developing a product.
- It is also known as static testing, where we are ensuring that "**we are developing the right product or not**". And it also checks that the developed application fulfilling all the requirements given by the client.


BRD
Business Requirements Document
Business Requirements (BRD)
Coding
Verification
CRS
(Customer Requirements Specification)
SRS
(Software Requirements Specification)
System Requirements (CRS,SRS)
System Design (HLD,LLD)
HLD
(High Level Design)
LLD
(Low Level design)

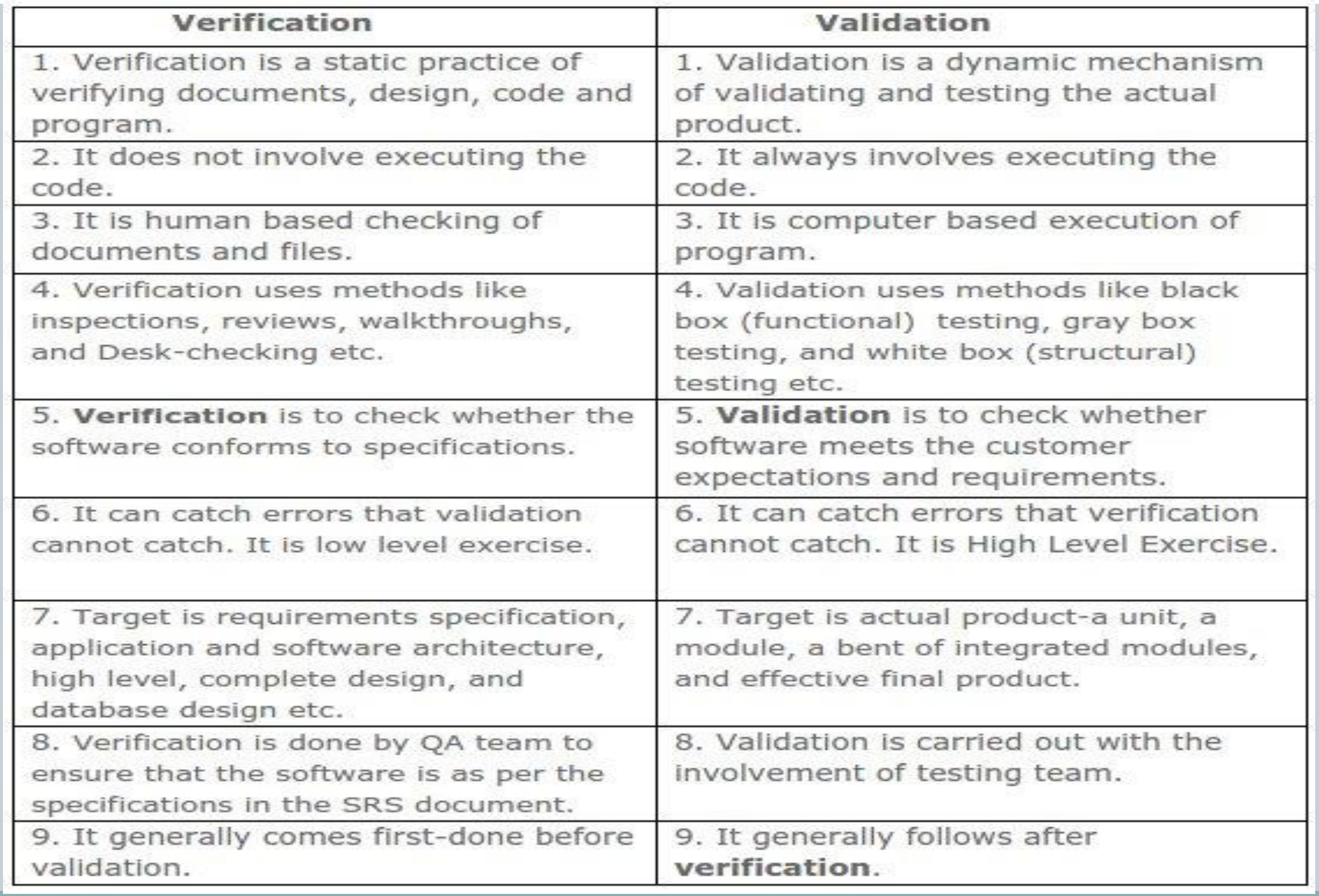
Verification and Validation Testing

- **Validation testing**
- Validation testing is testing where tester performed functional and non-functional testing. Here **functional testing** includes Unit Testing (UT), Integration Testing (IT) and System Testing (ST), and **non-functional** testing includes User acceptance testing (UAT).
- 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.

Verification and Validation Testing



Verification vs Validation



Verification	Validation
1. Verification is a static practice of verifying documents, design, code and program.	1. Validation is a dynamic mechanism of validating and testing the actual product.
2. It does not involve executing the code.	2. It always involves executing the code.
3. It is human based checking of documents and files.	3. It is computer based execution of program.
4. Verification uses methods like inspections, reviews, walkthroughs, and Desk-checking etc.	4. Validation uses methods like black box (functional) testing, gray box testing, and white box (structural) testing etc.
5. **Verification** is to check whether the software conforms to specifications.	5. **Validation** is to check whether software meets the customer expectations and requirements.
6. It can catch errors that validation cannot catch. It is low level exercise.	6. It can catch errors that verification cannot catch. It is High Level Exercise.
7. Target is requirements specification, application and software architecture, high level, complete design, and database design etc.	7. Target is actual product-a unit, a module, a bent of integrated modules, and effective final product.
8. Verification is done by QA team to ensure that the software is as per the specifications in the SRS document.	8. Validation is carried out with the involvement of testing team.
9. It generally comes first-done before validation.	9. It generally follows after **verification**.

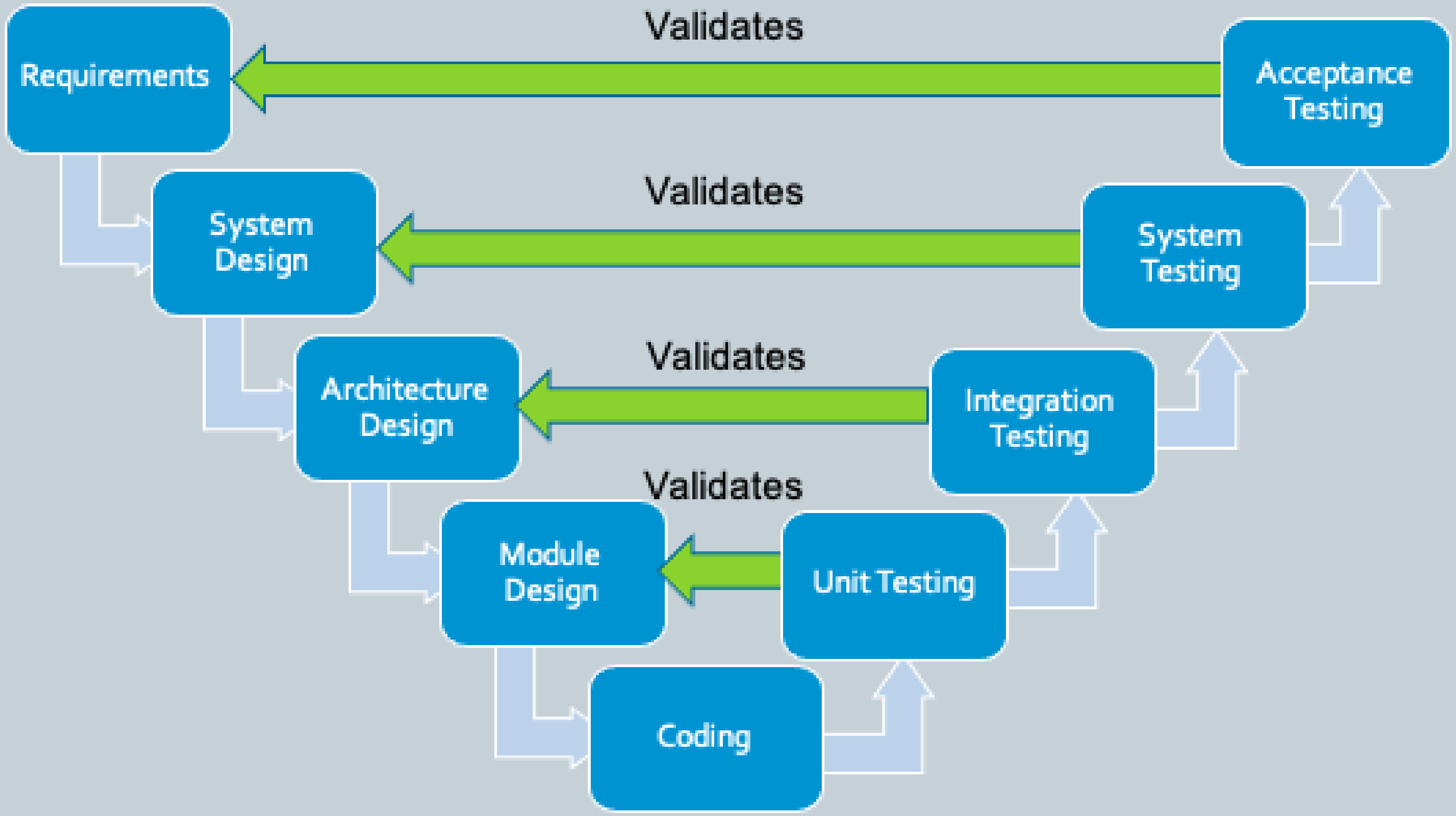
What are the Types of Validation Testing?

- Validation testing types a V-shaped testing pattern, which includes its variations and all the activities that it consists of are:
- **Unit Testing**
- **Integration testing**
- **System testing**
- **User acceptance testing**

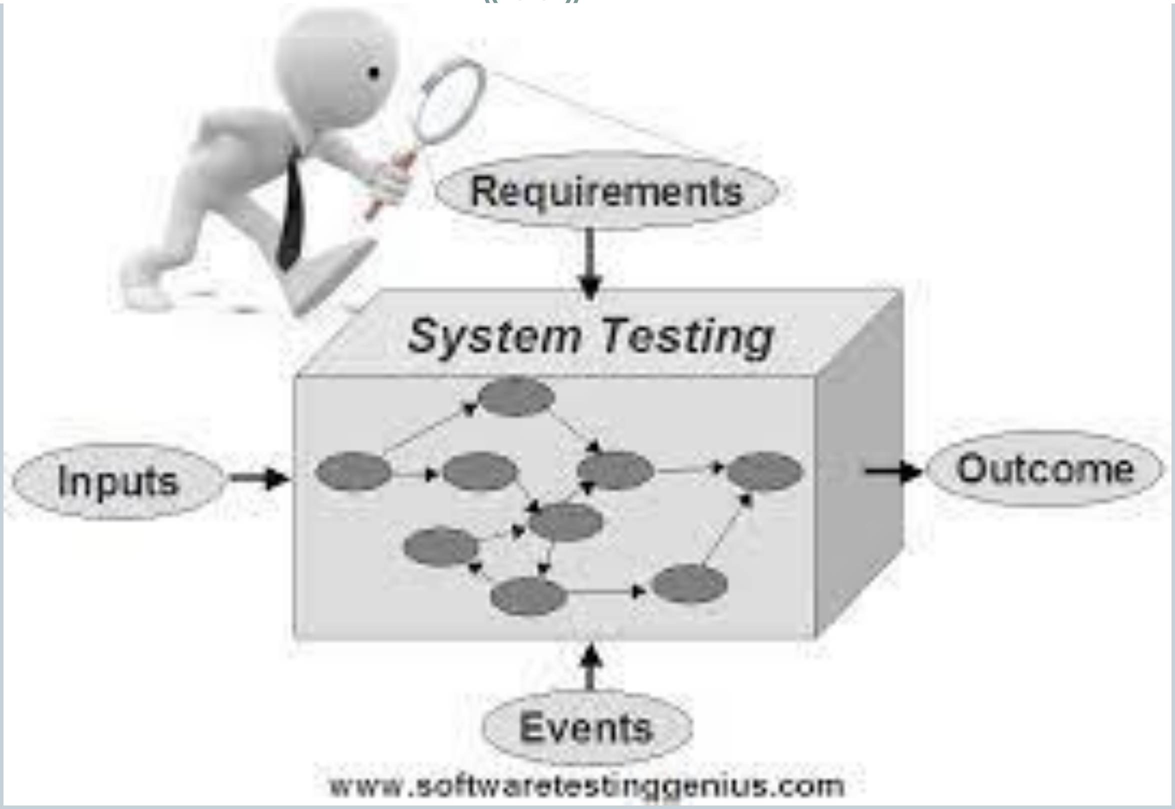
CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Difference Between Verification and Validation

- <u>**Example:**</u>
- **Client Requirement:**
- The proposed injection should not weigh above 2 cms.
- **Verification Test:**
- Check if the injection is the injection that does not weigh above 2 cms by using checklist, review, and design.
- **Validation Test:**
- Check if the injection does not weigh above 2 cms by using manual or automation testing.
- You have to check each and every possible scenario pertaining to the injection weight by using any suitable method of testing (functional and non-functional methods).
- Check for measurements less than 2 cm and above 2 cms.


Validates
Requirements
Acceptance Testing
Validates
System Design
System Testing
Validates
Architecture Design
Integration Testing
Validates
Module Design
Unit Testing
Coding

IV. System Testing



System Testing

- System Testing includes testing of a fully integrated software system. Generally, a computer system is made with the integration of software (any software is only a single element of a computer system). The software is developed in units and then interfaced with other software and hardware to create a complete computer system. In other words, a computer system consists of a group of software to perform the various tasks, but only software cannot perform the task; for that software must be interfaced with compatible hardware. System testing is a series of different type of tests with the purpose to exercise and examine the full working of an integrated software computer system against requirements.

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 falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic.

Introduction

- There are mainly two widely used methods for software testing, one is **White box testing** which uses internal coding to design test cases and another is black box testing

which uses GUI or user perspective to develop test cases.

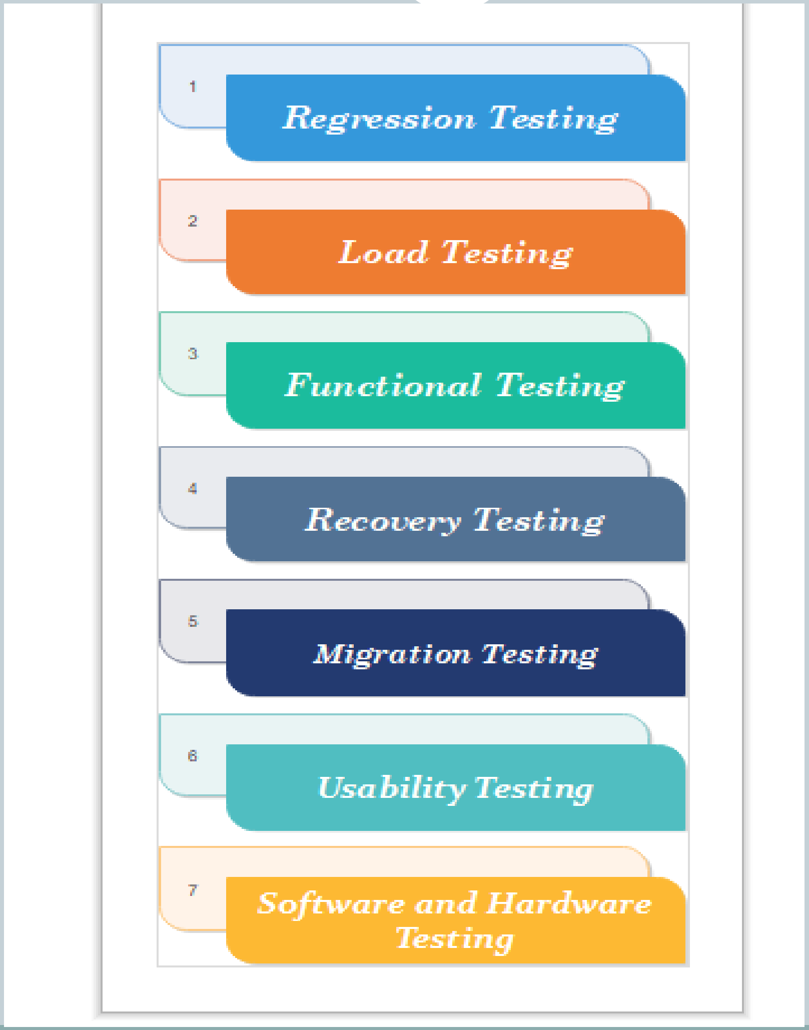
- White box testing
- Black box testing
- **System testing falls under Black box testing** as it includes testing of the external working of the software. Testing follows user's perspective to identify minor defects.

CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

System Testing includes the following steps.

- Verification of input functions of the application to test whether it is producing the expected output or not.
- Testing of integrated software by including external peripherals to check the interaction of various components with each other.
- Testing of the whole system for End to End testing.
- Behavior testing of the application via a user's experience

Types of System Testing



CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Types of System Testing

- With system testing, a QA team gauges if an application meets all of its technical, business and functional requirements. To accomplish this, the QA team might utilize various types of software testing techniques that determine the overall test coverage for an application and help catch critical defects that hamper an application's core functionalities before release.

CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Types of System Testing

- Regression Testing
- Regression testing is performed under system testing to confirm and identify that if there's any defect in the system due to modification in any other part of the system. It makes sure, any changes done during the development process have not introduced a new defect and also gives assurance; old defects will not exist on the addition of new software over the time.

CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Types of System Testing

- Load Testing
- Load testing is performed under system testing to clarify whether the system can work under real-time loads or not.
- Functional Testing
- Functional testing of a system is performed to find if there's any missing function in the system. Tester makes a list of vital functions that should be in the system and can be added during functional testing and should improve quality of the system.

CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Types of System Testing

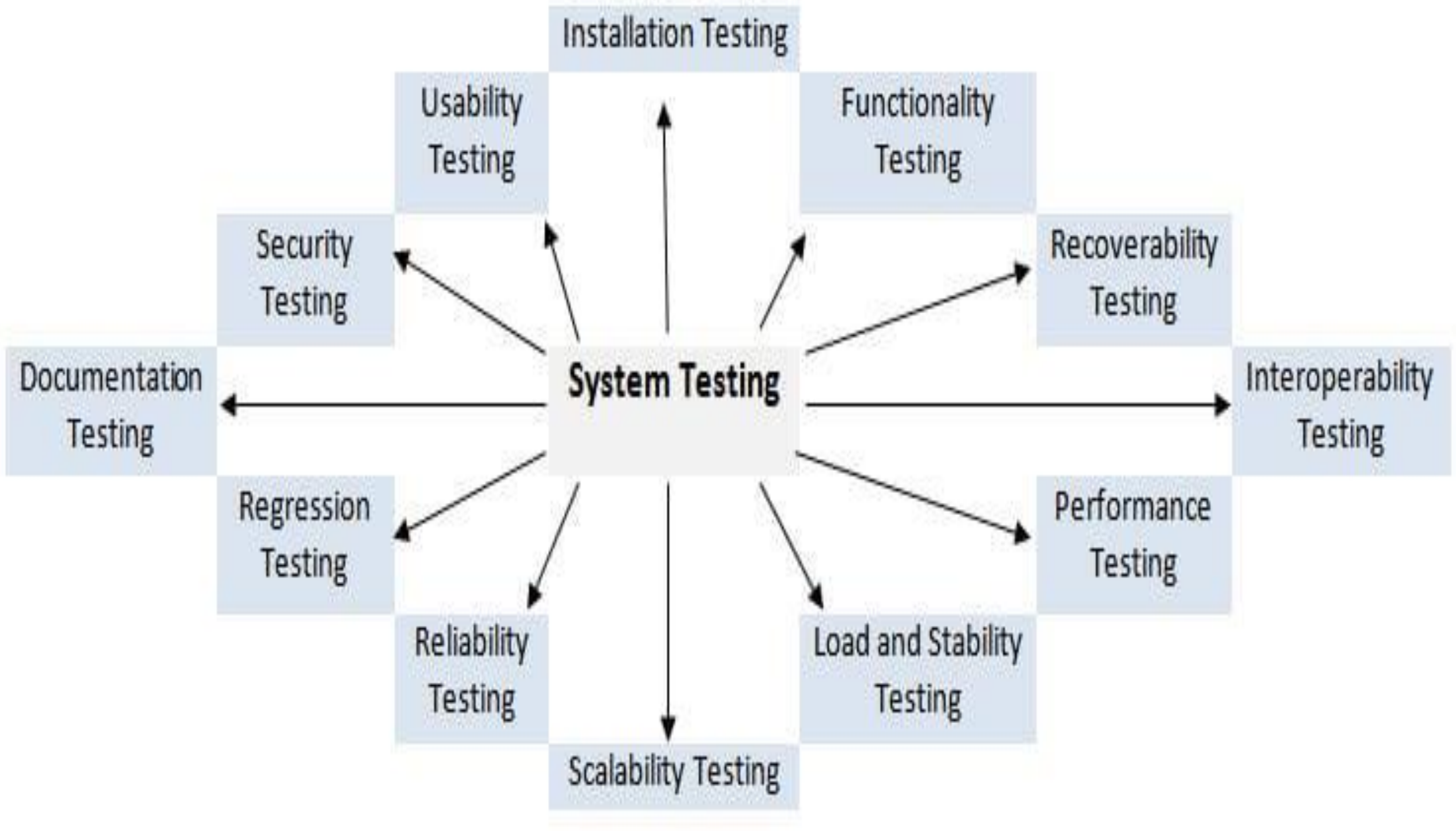
- Recovery Testing
- Recovery testing of a system is performed under system testing to confirm reliability, trustworthiness, accountability of the system and all are lying on recouping skills of the system. It should be able to recover from all the possible system crashes successfully.

Types of System Testing

- Migration Testing
- Migration testing is performed to ensure that if the system needs to be modified in new infrastructure so it should be modified without any issue.
- Usability Testing
- The purpose of this testing to make sure that the system is well familiar with the user and it meets its objective for what it supposed to do.

CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Types Of System Testing



Types Of System Testing

- ST is called a superset of all types of testing as all the major types of testing are covered in it. Although a focus on types of testing may vary on the basis of product, organization processes, timeline, and requirements.
- **Functionality Testing:** To make sure that functionality of the product is working as per the requirements defined, within the capabilities of the system.
- **Recoverability Testing:** To make sure how well the system recovers from various input errors and other failure situations.
- **Interoperability Testing:** To make sure whether the system can operate well with third-party products or not.

Types Of System Testing

- **Performance Testing:** To make sure the system's performance under the various condition, in terms of performance characteristics.
- **Scalability Testing:** To make sure the system's scaling abilities in various terms like user scaling, geographic scaling, and resource scaling.
- **Reliability Testing:** To make sure the system can be operated for a longer duration without developing failures.
- **Regression Testing:** To make sure the system's stability as it passes through an integration of different subsystems and maintenance tasks.

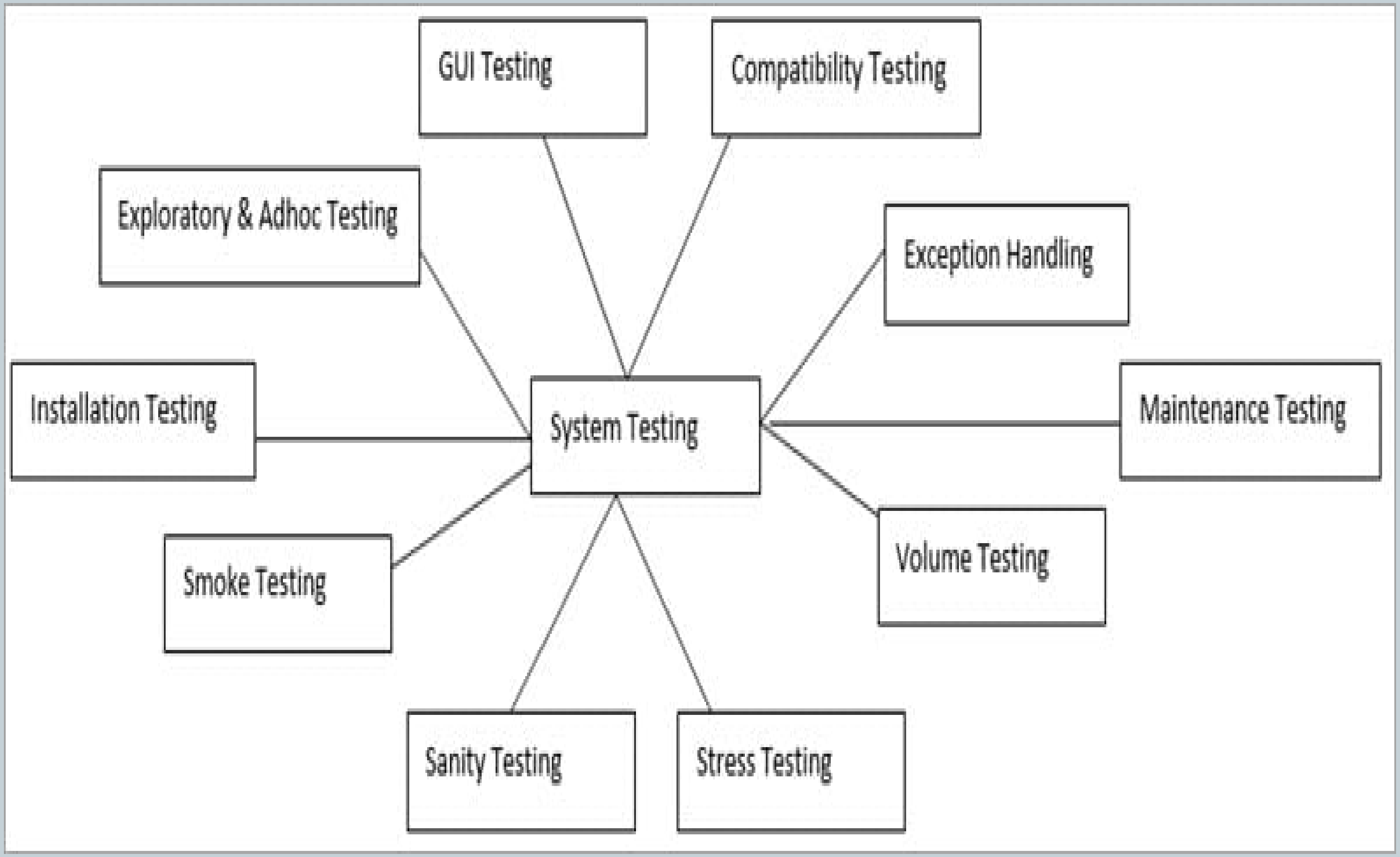
CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Types Of System Testing

- **Documentation Testing:** To make sure that the system's user guide and other help topics documents are correct and usable.
- **Security Testing:** To make sure that the system does not allow unauthorized access to data and resources.
- **Usability Testing:** To make sure that the system is easy to use, learn and operate.

CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

More System Testing Types



More System Testing Types

- **#1) Graphical User Interface Testing (GUI):**
- GUI testing is done to verify if the GUI of a system works as expected or not. GUI is basically what is visible to a user while he uses the application. GUI testing involves testing buttons, icons, checkboxes, List box, Textbox, menus, toolbars, dialog boxes, etc.
- **#2) Compatibility Testing:**
- Compatibility testing is done to ensure that the developed product is compatible with different browsers, Hardware Platforms, Operating System and databases as per the requirement document.

More System Testing Types

- **#3) Exception Handling:**
- Exception Handling Testing is performed to verify that even if an unexpected error occurs in the product, it should show the correct error message and does not let the application stop. It handles the exception in a way that the error is shown meanwhile the product recovers and allows the system to process the incorrect transaction.

More System Testing Types

- **#4) Volume Testing:**
- Volume Testing is a type of non-functional testing wherein testing is done using a huge amount of data. **For Example,** the Volume of data is increased in the database to verify the system performance.
- **#5) Stress Testing:**
- Stress Testing is done by increasing the number of users (at the same time) on an application to an extent that the application breaks down. This is done to verify the point at which the application will break down.

More System Testing Types

- **#6) Sanity Testing:**
- Sanity Testing is performed when the build is released with a change in the code or functionality or if any bug has been fixed. It verifies that the changes done have not affected the code and no other issue has occurred because of that and the system works as previously.
- If in case any issue occurs, then the build is not accepted for further testing.
- Basically, thorough testing is not done for the build in order to save time & cost as it rejects the build for an issue found. Sanity testing is done for the change done or for the fixed issue and not for the complete system.

More System Testing Types

- **#7) Smoke Testing:**
- Smoke Testing is a testing that is performed on the build to verify if the build is further testable or not. It verifies that the build is stable to test and all the critical functionalities are working fine. Smoke testing is done for the complete system i.e. end to end testing is done.
- **#8) Exploratory Testing:**
- Exploratory Testing as the name itself suggests it is all about exploring the application. No scripted testing is performed in exploratory testing. Test cases are written along with the testing. It focuses more on execution than planning.
- Tester has the freedom to test on his own using his intuition, experience, and intellect. A tester can choose any feature to test first i.e. randomly he can pick the feature to test, unlike the other techniques where the structural way is used to perform testing.

More System Testing Types

- **#9) Adhoc Testing:**
- Adhoc Testing is informal testing where no documentation or planning is done to test the application. Tester tests the application without any test cases. The aim of a tester is to break the application. The tester uses his experience, guess and intuition to find the critical issues in the application.

More System Testing Types

- **#10) Installation Testing:**
- Installation Testing is to verify if the software gets installed without any issues.
- This is the most important part of testing as the installation of the software is the very first interaction between the user and the product. The type of installation testing depends on various factors like operating system, Platform, distribution of software, etc.
- **Test cases which can be included if an installation is done via internet:**
- Bad network speed and broken connection.
- Firewall and security-related.
- Size and approximate time are taken.
- Concurrent installation/downloads.
- Insufficient memory
- Insufficient Space
- Aborted installation

More System Testing Types

- **#11) Maintenance Testing:**
- Once the product goes live, the issue can occur in a live environment or some enhancement might be required in the product.
- The product needs maintenance once it goes live and that is taken care by the maintenance team. The testing done for any issues or enhancement or migration to the hardware falls under maintenance testing.

Software and Hardware Testing

- This testing of the system intends to check hardware and software compatibility. The hardware configuration must be compatible with the software to run it without any issue. Compatibility provides flexibility by providing interactions between hardware and software.

CHITKARA UNIVERSITY
HIMACHAL PRADESH - NAAC Accredited

Why is System Testing Important?

- System Testing gives hundred percent assurance of system performance as it covers end to end function of the system.
- It includes testing of System software architecture and business requirements.
- It helps in mitigating live issues and bugs even after production.
- System testing uses both existing system and a new system to feed same data in both and then compare the differences in functionalities of added and existing functions so, the user can understand benefits of new added functions of the system.

Principles of System Testing

- **Function testing:** does the integrated system perform as promised by the requirements specification?
- **Performance testing:** are the non-functional requirements met?
- **Acceptance testing:** is the system what the customer expects?
- **Installation testing:** does the system run at the customer site(s)?

Importance of system testing

Main benefits of system testing:

- Improved product quality.
- Error reduction
- Cost savings
- Security
- Customer satisfaction
- Easier code modification
- Software performance

Phases of System Testing

- Typically, system testing goes through the following stages:
- **Test environment.** In this initial stage, a test server is set up for creating a testing environment, which enables a tester to run a set of predefined test cases and test scripts.
- **Test case.** This stage generates the test case for the testing process.
- **Test data.** At this stage, the data to be tested is generated.
- **Test case execution.** Once the test case and test data are generated, test cases are executed.
- **Defect logging.** All the identified defects are fixed at this stage.

Phases of System Testing

- **Retest.** A test is repeated if it's unsuccessful.
- **Reporting of defects.** This is the stage where defects in the system are identified.
- **Regression testing.** This is done to see if any problems were introduced into the development process by the previous stages.

Software Engineering-Object Oriented Testing Strategies

- The classical strategy for testing computer software begins with “testing in the small” and works outward toward “testing in the large.” Stated in the jargon of software testing , we begin with unit testing, then progress toward integration testing, and culminate with validation and system testing. In conventional applications, unit testing focuses on the smallest compilable program unit—the subprogram (e.g., module, subroutine, procedure, component). Once each of these units has been tested individually, it is integrated into a program structure while a series of regression tests are run to uncover errors due to interfacing between the modules and side effects caused by the addition of new units. Finally, the system as a whole is tested to ensure that errors in requirements are uncovered.

Unit Testing in the OO Context

- When object-oriented software is considered, the concept of the unit changes. Encapsulation drives the definition of classes and objects. This means that each class and each instance of a class (object) packages attributes (data) and the operations (also known as methods or services) that manipulate these data. Rather than testing an individual module, the smallest testable unit is the encapsulated class or object. Because a class can contain a number of different operations and a particular operation may exist as part of a number of different classes, the meaning of unit testing changes dramatically.

Unit Testing in the OO Context

- We can no longer test a single operation in isolation (the conventional view of unit testing) but rather as part of a class. To illustrate, consider a class hierarchy in which an operation X is defined for the superclass and is inherited by a number of subclasses. Each subclass uses operation X, but it is applied within the context of the private attributes and operations that have been defined for the subclass. Because the context in which operation X is used varies in subtle ways, it is necessary to test operation X in the context of each of the subclasses. This means that testing operation X in a vacuum (the traditional unit testing approach) is ineffective in the object-oriented context.

Unit Testing in the OO Context

- Class testing for OO software is the equivalent of unit testing for conventional software. Unlike unit testing of conventional software, which tends to focus on the algorithmic detail of a module and the data that flow across the module interface, class testing for OO software is driven by the operations encapsulated by the class and the state behavior of the class.

Integration Testing in the OO Context

- Because object-oriented software does not have a hierarchical control structure, conventional top-down and bottom-up integration strategies have little meaning. In addition, integrating operations one at a time into a class (the conventional incremental integration approach) is often impossible because of the “direct and indirect interactions of the components that make up the class”.

Integration Testing in the OO Context

- There are two different strategies for integration testing of OO systems . The first, thread-based testing, integrates the set of classes required to respond to one input or event for the system. Each thread is integrated and tested individually. Regression testing is applied to ensure that no side effects occur. The second integration approach, use-based testing, begins the construction of the system by testing those classes (called independent classes) that use very few (if any) of server classes. After the independent classes are tested, the next layer of classes, called dependent classes, that use the independent classes are tested. This sequence of testing layers of dependent lasses continues until the entire system is constructed. Unlike conventional integration, the use of drivers and stubs as replacement operations is to be avoided, when possible.

Validation Testing in the OO Context

- At the validation or system level, the details of class connections disappear. Like conventional validation, the validation of OO software focuses on user-visible actions and user-recognizable output from the system. To assist in the derivation of validation tests, the tester should draw upon the use-cases that are part of the analysis model. The use-case provides a scenario that has a high likelihood of uncovered errors in user interaction requirements.

THANK YOU