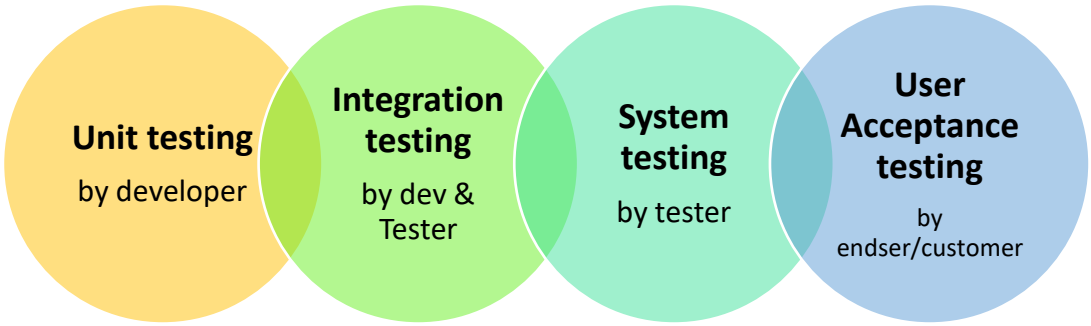
Test Levels

A thorough understanding and definition of the various test levels will identify missing areas and prevent overlap and repetition. Understanding whether we want overlaps and removing the gaps will make the test levels more complementary thus leading to more effective and efficient testing.



Test level are characterized by the following attributes:

1. Specific objectives
2. Test basis, referenced to drive test cases
3. Test objects (what is being tested)
4. Typical defects and failure
5. Specific approaches and responsibilities

Here for every test level suitable test environment required.

1. **Component/Unit Testing:**

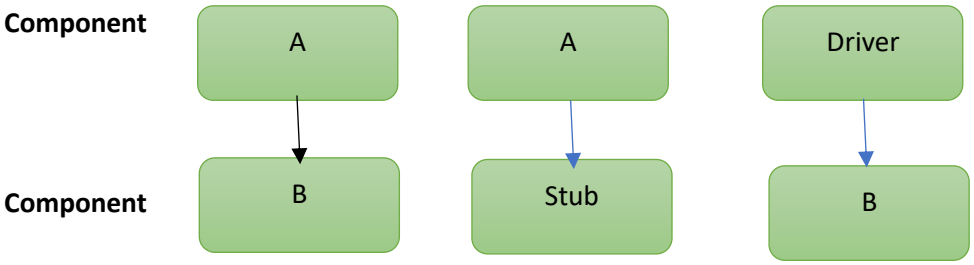
   Component testing, also known as unit, module and program testing, searches for defects in, and verifies the functioning of software, that are separately testable. Component testing may be done in isolation from the rest of the system.
   Objectives of component testing:
   a. Reducing risk
   b. Verify the functional and non-functional behaviour of the components are as designed or specified.
   c. Building confidence in the component's quality
   d. Finding defects in the component
   e. Preventing defects from the escaping to higher test levels.

Component testing may be done in the isolation from the rest of the system depending on the context of the development life cycle and the system.

Most of the time stubs and drivers used to replace the missing module and simulate the interface b/w the s/w components in a simple manner. may require mock objects, service virtualization, harnesses, stubs, and drivers.

Stubs: A stub is called from the s/w component to be tested.

Derivers: call a component to be tested.



On approach In component testing, used in Extreme Programming, is to prepare and automate test cases before coding. This is called a test-first approach or **test-driven development (TDD).** This approach is highly iterative and is based on cycle of developing test cases, then building and integrating small pieces of code, and executing the component tests until the test pass.

Test Basis: example of work product that can be used as a test basis for component testing including:

- Detailed design
- Code
- Data model
- Component specifications

Test Objects:

Test objects for component testing:

- Components, units or modules
- Code and data structure
- Classes
- Database

Defects and Failures:

- Incorrect functionality (not as per describe in specification)
- Data flow problems
- Incorrect coding and logic

2. Integration Testing:

Integration testing focuses on interactions between components or systems. Integration testing should be differentiated from other integration activities. Integration testing often carried out by integrator, but preferably by a specific integration tester or test team.

There may be more than one level of integration and it may be carried out on test objects of varying size.

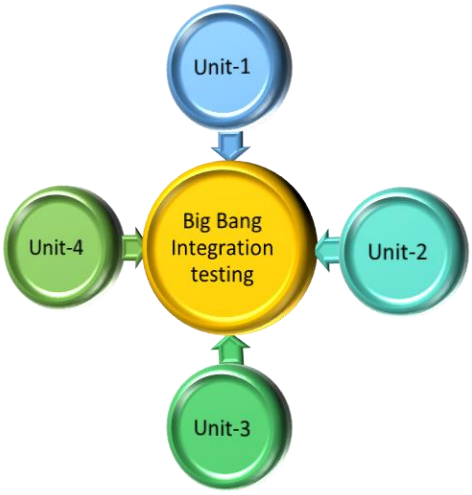
- ⇨ Component integration testing focuses on the interactions and interfaces between integrated components. Component integration testing is performed after component testing, and is generally automated.
- ⇨ System integration testing focuses on the interactions and interfaces between systems, packages, and microservices. System integration testing can also cover interactions with, and interfaces provided by, external organizations.

Objectives of component testing

- Reducing risk
- Verifying whether the functional and non-functional behaviours of the interfaces are as designed and specified.
- Building confidence in the quality of the interfaces
- Finding defects (which may be in the interfaces themselves or within the components or systems)
- Preventing defects from escaping to higher test levels

The greater the scope of integration, the more difficult it become to isolate failure to a specific interface, which may lead to increase the risk. This leads to varying approach of integration.

One extreme is that, all components or systems are integrated simultaneously, which everything is tested as whole system. This is called **"big-bang" integration testing**.



Big-Bang Integration testing have an advantage that everything will finished before you started testing. The major disadvantage is , it is time consuming, difficult to find the cause of failure.

Incremental Testing, all programs will integrate one by one and a test carried out after each step of integration.

The incremental approach has a feature that the defects are found early in a smaller assembly when its easy to find the cause but, the major disadvantage is that its time consuming since stubs and driver have to be developed.

Within Incremental integration testing a range of possibilities exist, partly depending on the system architecture:

- Top-Down: testing takes place top to bottom, following the control flow or architecture structure, components or systems are substituted by stubs.
- Bottom-up: testing takes place from the bottom of the control flow upwards. Components or systems are substituted by drivers.
- Functional incremental: integration and testing takes place on the basis of the function or functionality, as documented in the functional specification.

The best choice to start integration with those interfaces that are expected to cause most problems. Doing so prevents major defects at the end of the integration test stage. To reduce the risk of late defect discovery , integration should normally be incremental rather than “big-bang”.

3. System testing:

System integration is concerned with the behaviour of the whole system as defined by the scope of a development product. It may include test based on risk and requirements specification, business process, use cases or other high-level descriptions of system behaviour, interaction with the system and system resources.
System testing should investigate both functional and non-functional requirements of system. **non-functional** test cases include performance and reliability.
Functional system testing starts by using the most appropriate specification-based (black-box) techniques for the aspects of the system to be tested. **Structure based system testing** used to test thoroughness of testing elements such as menu dialog structure or web page navigation.

4. Acceptance testing:

When the development organization performed its system test and has corrected all or most defects. The system will be delivered to the user or customer for User acceptance testing.

Acceptance testing is most of the responsibility of the user and the customer , other stakeholders can also be involved as well. The execution of acceptance test required a test environment that is for acceptance testing for e.g production environment.

Objective of acceptance testing:

a. Establishing confidence in the quality of the system/product.
b. Validating that the system is completed and work as expected.
c. Verifying that functional and non-functional behaviour of the system as per specified.

Finding defects is not the main focus in acceptance testing . although acceptance testing is mostly focused on validation type of testing, where we are trying to validate that the system is fit to the purpose.

Acceptance testing may occur at more then just a single level:

a. A Commercial off the Shelf (COTS) software product may be acceptance tested when it is installed or integrated.
b. Acceptance testing of the usability of a component may be done during component testing.
c. Acceptance testing of new functional enhancement may come before system testing.

Common forms of acceptance testing include:

i. User Acceptance Testing: user acceptance test mainly focusses on the functionality thereby validating the fitness-for-use of the system by the business user.
ii. Operational acceptance testing: operational acceptance test validates whether the system meets the requirements for operation, the operational acceptance test may include testing of backup/restore, disaster recovery, maintenance tasks and periodic check of security vulnerabilities.
iii. Contractual and regulatory acceptance testing: contract acceptance testing is performed against a contract's acceptance criteria for producing custom-developed software. Acceptance should be formally defined when the contract is agreed. Compliances acceptance testing / regulation acceptance testing is performed against the regulations which must be adhered to, such as governmental, legal or safety regulations.
iv. Alpha and Beta Testing: **Alpha testing** takes place at developer's site . a cross section of potential users and members of the organization are invited to use the system. It may also be carried out by an independent team.

Beta Testing, it is also known as field testing, send the system to a cross -section of users who install it and use it under real-world working condition.