What are Software Testing Levels?

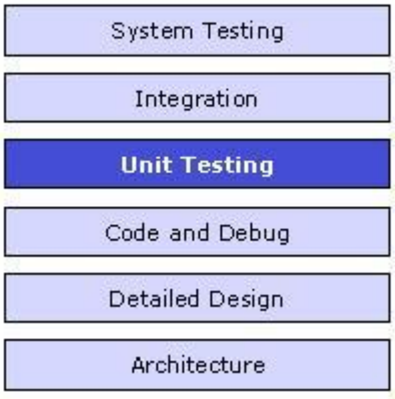
Testing levels are basically to identify missing areas and prevent overlap and repetition between the development life cycle phases. In software development life cycle models there are defined phases like requirement gathering and analysis, design, coding or implementation, testing and deployment. Each phase goes through the testing. Hence there are various levels of testing. The various levels of testing are:

1. Unit testing: It is basically done by the developers to make sure that their code is working fine and meet the user specifications. They test their piece of code which they have written like classes, functions, interfaces and procedures.
2. Component testing: It is also called as module testing. The basic difference between the unit testing and component testing is in unit testing the developers test their piece of code but in component testing the whole component is tested. For example, in a student record application there are two modules one which will save the records of the students and other module is to upload the results of the students. Both the modules are developed separately and when they are tested one by one then we call this as a component or module testing.
3. Integration testing: Integration testing is done when two modules are integrated, in order to test the behavior and functionality of both the modules after integration. Below are few types of integration testing:
    - Big bang integration testing
    - Top down
    - Bottom up
    - Functional incremental
4. Component integration testing: In the example above when both the modules or components are integrated then the testing done is called as Component integration testing. This testing is basically done to ensure that the code should not break after integrating the two modules.
5. System integration testing: System integration testing (SIT) is a testing where testers basically test that in the same environment all the related systems should maintain data integrity and can operate in coordination with other systems.
6. System testing: In system testing the testers basically test the compatibility of the application with the system.
7. Acceptance testing: Acceptance testing are basically done to ensure that the requirements of the specification are met.
8. Alpha testing: Alpha testing is done at the developers site. It is done at the end of the development process
9. Beta testing: Beta testing is done at the customers site. It is done just before the launch of the product.

What is Unit testing?

- A unit is the smallest testable part of an application like functions, classes, procedures, interfaces. Unit testing is a method by which individual units of source code are tested to determine if they are fit for use.
- **Unit tests are basically written and executed by software developers** to make sure that code meets its design and requirements and behaves as expected.
- The goal of unit testing is to segregate each part of the program and test that the individual parts are working correctly.
- This means that for any function or procedure when a set of inputs are given then it should return the proper values. It should handle the failures gracefully during the course of execution when any invalid input is given.

- A unit test provides a written contract that the piece of code must assure. Hence it has several benefits.
- Unit testing is basically done before integration as shown in the image below.



Method Used for unit testing: White Box Testing method is used for executing the unit test.

When Unit testing should be done?

Unit testing should be done before Integration testing.

By whom unit testing should be done?

Unit testing should be done by the developers.

Advantages of Unit testing:

1. Issues are found at early stage. Since unit testing are carried out by developers where they test their individual code before the integration. Hence the issues can be found very early and can be resolved then and there without impacting the other piece of codes.

2. Unit testing helps in maintaining and changing the code. This is possible by making the codes less interdependent so that unit testing can be executed. Hence chances of impact of changes to any other code gets reduced.

3. Since the bugs are found early in unit testing hence it also helps in reducing the cost of bug fixes. Just imagine the cost of bug found during the later stages of development like during system testing or during acceptance testing.

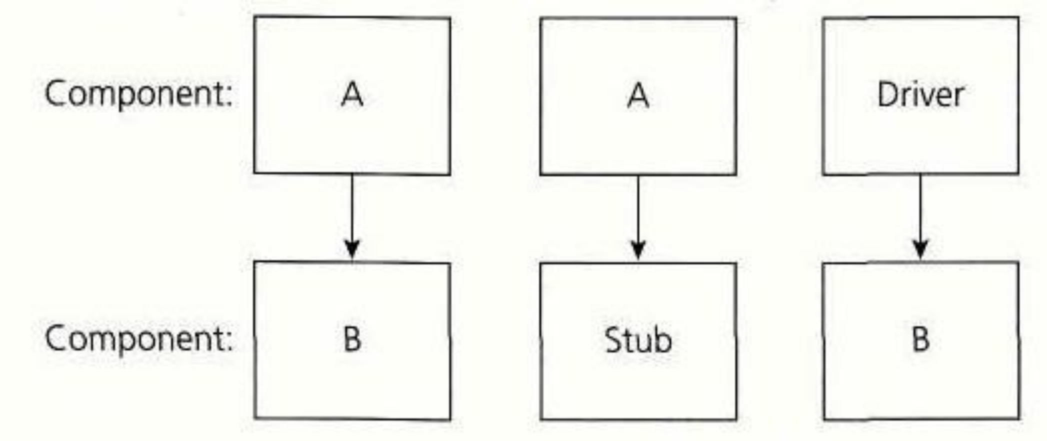
4. Unit testing helps in simplifying the debugging process. If suppose a test fails then only latest changes made in code needs to be debugged.

What is Component testing?

What is Component testing?: Component testing is a method where testing of each component in an application is done separately. Suppose, in an application there are 5 components. Testing of each 5 components separately and efficiently is called as component testing.

- Component testing is also known as module and program testing. It finds the defects in the module and verifies the functioning of software.
- Component testing is done by the tester.
- Component testing may be done in isolation from rest of the system depending on the development life cycle model chosen for that particular application. In such case the missing software is replaced by **Stubs** and **Drivers** and simulate the interface between the software components in a simple manner.
- Let's take an example to understand it in a better way. Suppose there is an application consisting of three modules say, module A, module B and module C. The developer has developed the module B and now wanted to test it. But in order to test the module B completely few of it's functionalities are dependent on module A and few on module C. But the module A and module C has not been developed yet. In that case to test the module B completely we can replace the module A and module C by stub and drivers as required.
- **Stub:** A stub is called from the software component to be tested. As shown in the diagram below 'Stub' is called by 'component A'.
- **Driver:** A driver calls the component to be tested. As shown in the diagram below 'component B' is called by the 'Driver'.

Below is the diagram of the component testing:



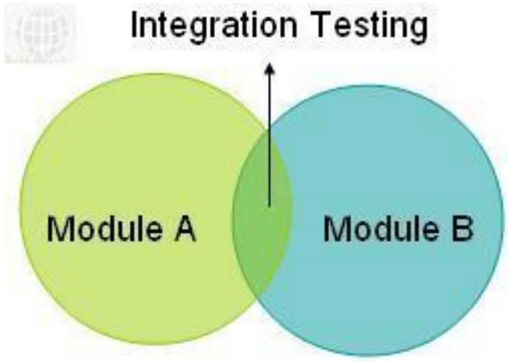
As discussed in the previous article of the 'Unit testing' it is done by the developers where they do the testing of the individual functionality or procedure. After unit testing is executed, component testing comes into the picture. Component testing is done by the testers.

Component testing plays a very important role in finding the bugs. Before we start with the integration testing it's always preferable to do the component testing in order to ensure that each component of an application is working effectively.

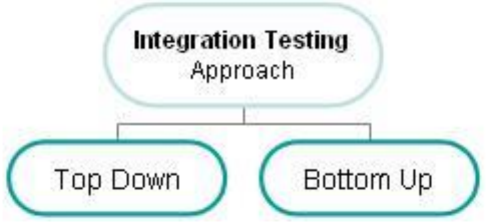
Integration testing is followed by the component testing.

What is Integration testing?

- Integration testing tests integration or interfaces between components, interactions to different parts of the system such as an operating system, file system and hardware or interfaces between systems.
- Also after integrating two different components together we do the integration testing. As displayed in the image below when two different modules 'Module A' and 'Module B' are integrated then the integration testing is done.



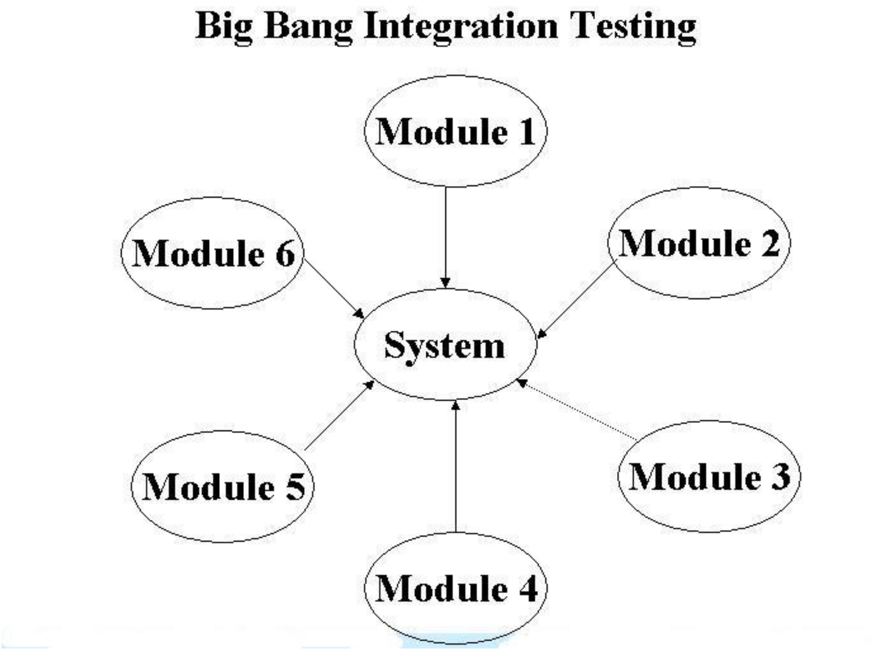
- Integration testing is done by a specific integration tester or test team.
- Integration testing follows two approach known as ‘Top Down’ approach and ‘Bottom Up’ approach as shown in the image below:



Below are the integration testing techniques:

1. Big Bang integration testing:

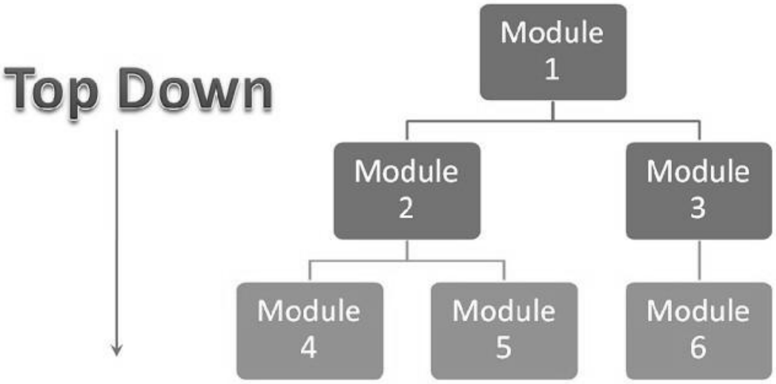
In Big Bang integration testing all components or modules are integrated simultaneously, after which everything is tested as a whole. As per the below image all the modules from ‘Module 1′ to ‘Module 6′ are integrated simultaneously then the testing is carried out.



Advantage: Big Bang testing has the advantage that everything is finished before integration testing starts.

Disadvantage: The major disadvantage is that in general it is time consuming and difficult to trace the cause of failures because of this late integration.

2. Top-down integration testing: Testing takes place from top to bottom, following the control flow or architectural structure (e.g. starting from the GUI or main menu). Components or systems are substituted by stubs. Below is the diagram of 'Top down Approach':



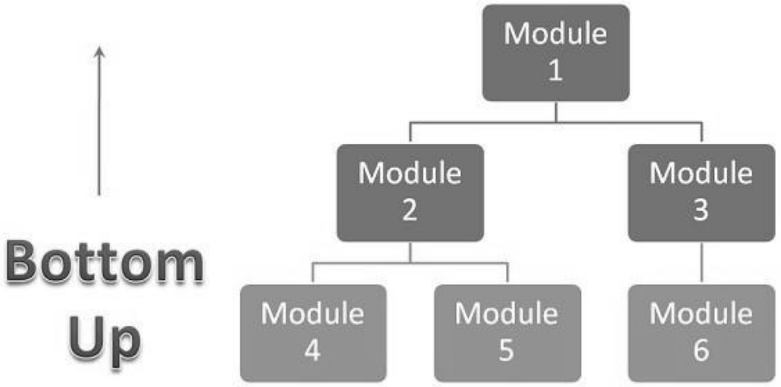
Advantages of Top-Down approach:

- The tested product is very consistent because the integration testing is basically performed in an environment that almost similar to that of reality
- Stubs can be written with lesser time because when compared to the drivers then Stubs are simpler to author.

Disadvantages of Top-Down approach:

- Basic functionality is tested at the end of cycle

3. Bottom-up integration testing: Testing takes place from the bottom of the control flow upwards. Components or systems are substituted by drivers. Below is the image of 'Bottom up approach':



Advantage of Bottom-Up approach:

- In this approach development and testing can be done together so that the product or application will be efficient and as per the customer specifications.

Disadvantages of Bottom-Up approach:

- We can catch the Key interface defects at the end of cycle
- It is required to create the test drivers for modules at all levels except the top control

Incremental testing:

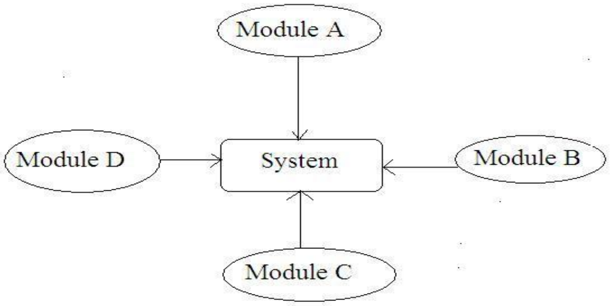
Another extreme is that all programmers are integrated one by one, and a test is carried out after each step. The incremental approach has the advantage that the defects are found early in a smaller assembly when it is relatively easy to detect the cause.
A disadvantage is that it can be time-consuming since stubs and drivers have to be developed and used in the test.
Within incremental integration testing a range of possibilities exist, partly depending on the system architecture.

Functional incremental: Integration and testing takes place on the basis of the functions and functionalities, as documented in the functional specification.

What is Big Bang integration testing?

- In Big Bang integration testing all components or modules are integrated simultaneously, after which everything is tested as a whole.
- In this approach individual modules are not integrated until and unless all the modules are ready.
- In Big Bang integration testing all the modules are integrated without performing any integration testing and then it’s executed to know whether all the integrated modules are working fine or not.
- This approach is generally executed by those developers who follows the ‘Run it and see’ approach.
- Because of integrating everything at one time if any failures occurs then it become very difficult for the programmers to know the root cause of that failure.
- In case any bug arises then the developers has to detach the integrated modules in order to find the actual cause of the bug.

Below is the image of the big bang integration testing:



Suppose a system consists of four modules as displayed in the diagram above. In big bang integration all the four modules ‘Module A, Module B, Module C and Module D’ are integrated simultaneously and then the testing is performed. Hence in this approach no individual integration testing is performed because of which the chances of critical failures increases.

Advantage of Big Bang Integration:

- Big Bang testing has the **advantage** that everything is finished before integration testing starts.

Disadvantages of Big Bang Integration:

- The major **disadvantage** is that in general it is very time consuming
- It is very difficult to trace the cause of failures because of this late integration.
- The chances of having critical failures are more because of integrating all the components together at same time.
- If any bug is found then it is very difficult to detach all the modules in order to find out the root cause of it.
- There is high probability of occurrence of the critical bugs in the production environment

What is Incremental testing in software?

- Another extreme is that all programmers are integrated one by one, and a test is carried out after each step.
- The incremental approach has the advantage that the defects are found early in a smaller assembly when it is relatively easy to detect the cause.
- A disadvantage is that it can be time-consuming since stubs and drivers have to be developed and used in the test.
- Within incremental integration testing a range of possibilities exist, partly depending on the system architecture:
    - **Top down:** Testing takes place from top to bottom, following the control flow or architectural structure (e.g. starting from the GUI or main menu). 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 functions nad functionalities, as documented in the functional specification.

What is Component integration testing?

- It tests the interactions between software components and is done after component testing.
- The software components themselves may be specified at different times by different specification groups, yet the integration of all of the pieces must work together.
- It is important to cover negative cases as well because components might make assumption with respect to the data.

What is System integration testing?

- It tests the interactions between different systems and may be done after system testing.
- It verifies the proper execution of software components and proper interfacing between components within the solution.
- The objective of SIT Testing is to validate that all software module dependencies are functionally correct and that data integrity is maintained between separate modules for the entire solution.
- As testing for dependencies between different components is a primary function of SIT Testing, this area is often most subject to Regression Testing.

What is System testing?

- In system testing the behavior of whole system/product is tested as defined by the scope of the development project or product.
- It may include tests based on risks and/or requirement specifications, business process, use cases, or other high level descriptions of system behavior, interactions with the operating systems, and system resources.
- System testing is most often the final test to verify that the system to be delivered meets the specification and its purpose.
- System testing is carried out by specialists testers or independent testers.
- System testing should investigate both functional and non-functional requirements of the testing.

What is Acceptance testing?

- After the system test has corrected all or most defects, the system will be delivered to the user or customer for acceptance testing.
- Acceptance testing is basically done by the user or customer although other stakeholders may be involved as well.
- The goal of acceptance testing is to establish confidence in the system.
- Acceptance testing is most often focused on a validation type testing.
- Acceptance testing may occur at more than just a single level, for example:

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

- The **types of acceptance testing** are:

  - The **User Acceptance test:** focuses mainly on the functionality thereby validating the fitness-for-use of the system by the business user. The user acceptance test is performed by the users and application managers.
  - The **Operational Acceptance test:** also known as Production acceptance test validates whether the system meets the requirements for operation. In most of the organization the operational acceptance test is performed by the system administration before the system is released. The operational acceptance test may include testing of backup/restore, disaster recovery, maintenance tasks and periodic check of security vulnerabilities.
  - **Contract Acceptance testing**: It is performed against the contract’s acceptance criteria for producing custom developed software. Acceptance should be formally defined when the contract is agreed.

- **Compliance acceptance testing:** It is also known as regulation acceptance testing is performed against the regulations which must be adhered to, such as governmental, legal or safety regulations.

What is Alpha testing?

Alpha testing is one of the most common software testing strategy used in software development. Its specially used by product development organizations.

- This **test takes place at the developer's site**. Developers observe the users and note problems.
- Alpha testing is testing of an application when development is about to complete. Minor design changes can still be made as a result of alpha testing.
- Alpha testing is typically performed by a group that is independent of the design team, but still within the company, e.g. in-house software test engineers, or software QA engineers.
- Alpha testing is final testing before the software is released to the general public. It has two phases:
    - In the **first phase** of alpha testing, the software is tested by in-house developers. They use either debugger software, or hardware-assisted debuggers. The goal is to catch bugs quickly.
    - In the **second phase** of alpha testing, the software is handed over to the software QA staff, for additional testing in an environment that is similar to the intended use.

- Alpha testing is simulated or actual operational testing by potential users/customers or an independent test team at the developers' site. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing, before the software goes to beta testing.

What is Beta testing?

- It is also known as field testing. It takes place at **customer's site**. It sends the system to users who install it and use it under real-world working conditions.
- A beta test is the second phase of software testing in which a sampling of the intended audience tries the product out. (Beta is the second letter of the Greek alphabet.) Originally, the term *alpha test* meant the first phase of testing in a software development process. The first phase includes unit testing, component testing, and system testing. Beta testing can be considered "pre-release testing.

- The goal of beta testing is to place your application in the hands of real users outside of your own engineering team to discover any flaws or issues from the user's perspective that you would not want to have in your final, released version of the application.

Open and closed beta:
Developers release either a **closed beta** or an **open beta**;

- *Closed beta versions are released to a select group of individuals for a user test and are invitation only, while*
- *Open betas are from a larger group to the general public and anyone interested. The testers report any bugs that they find, and sometimes suggest additional features they think should be available in the final version.*

Advantages of beta testing:

- You have the opportunity to get your application into the hands of users prior to releasing it to the general public.
- Users can install, test your application, and send feedback to you during this beta testing period.
- Your beta testers can discover issues with your application that you may have not noticed, such as confusing application flow, and even crashes.
- Using the feedback you get from these users, you can fix problems before it is released to the general public.
- The more issues you fix that solve real user problems, the higher the quality of your application when you release it to the general public.
- Having a higher-quality application when you release to the general public will increase customer satisfaction.
- These users, who are early adopters of your application, will generate excitement about your application.