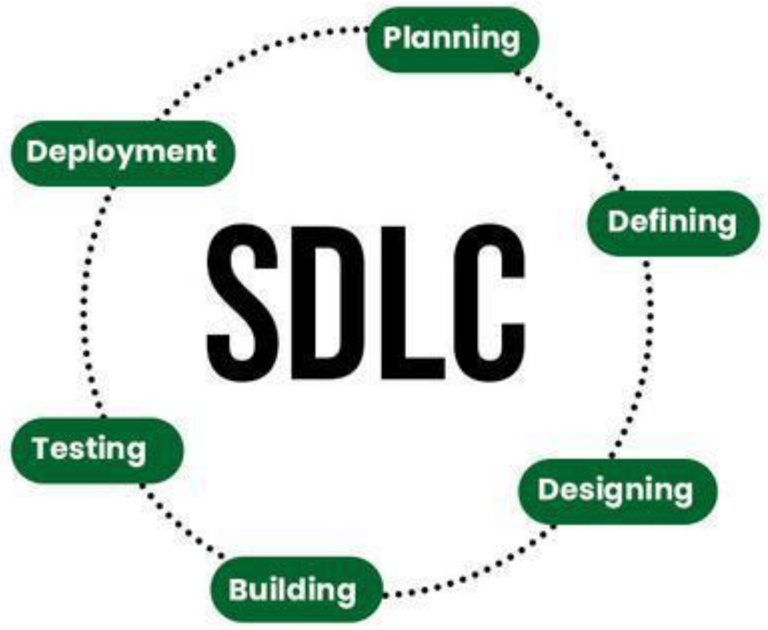
# Product Testing and Integration

Lecture-9

## Software Development Life Cycle (SDLC)

**Software development life cycle (SDLC) is a structured process that is used to design, develop, and test good-quality software.** SDLC, or software development life cycle, is a methodology that defines the entire procedure of software development step-by-step.

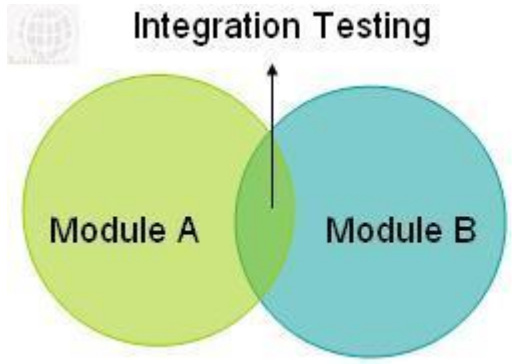


## Software Development Life Cycle (SDLC)

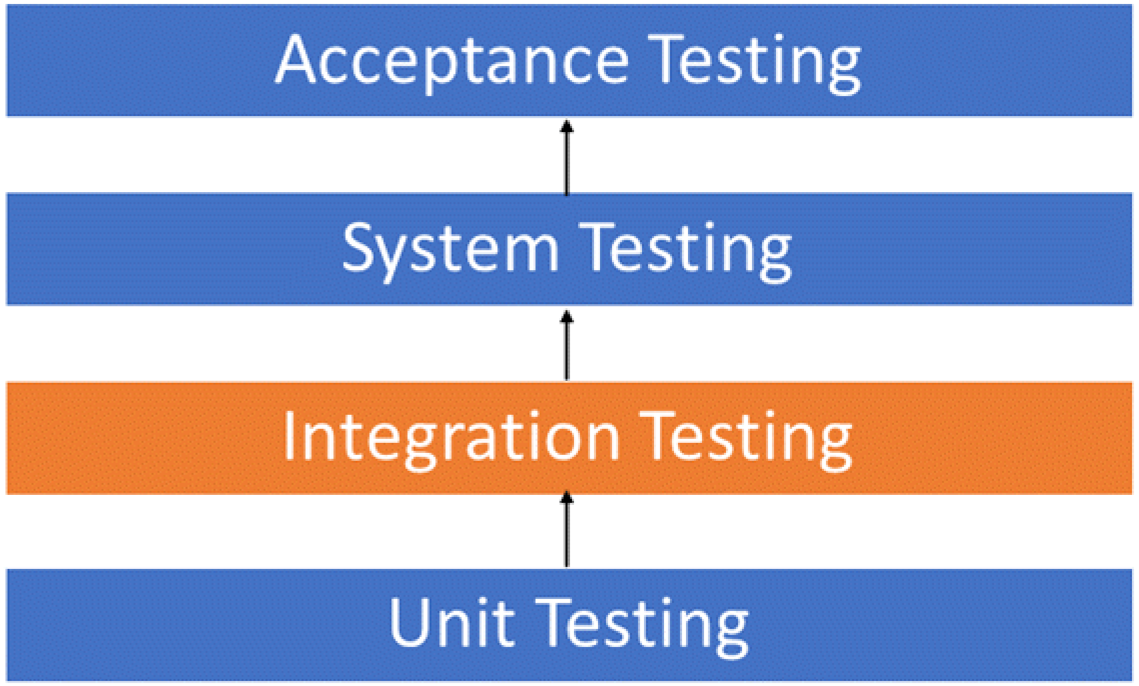
- Stage-1: Planning and Requirement Analysis
- Stage-2: Defining Requirements
- Stage-3: Designing Architecture
- Stage-4: Developing Product
- Stage-5: Product Testing and Integration
- Stage-6: Deployment and Maintenance of Products

## What is Integration Testing?

- **Integration Testing** is defined as a type of testing where software modules are integrated logically and tested as a group. A typical software project consists of multiple software modules, coded by different programmers. The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated.
- 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'**.



## Why do Integration Testing?



## Why do Integration Testing?

- A Module, in general, is designed by an individual software developer whose understanding and programming logic may differ from other programmers. Integration Testing becomes necessary to verify the software modules work in unity
- At the time of module development, there are wide chances of change in requirements by the clients. These new requirements may not be unit tested and hence system integration Testing becomes necessary.
- Interfaces of the software modules with the database could be erroneous
- External Hardware interfaces, if any, could be erroneous
- Inadequate exception handling could cause issues.

# Integration Testing Example

- After each module was assigned to a developer, the developers began coding the functionality on their individual machines. They deployed their respective modules on their own machines to see what worked and what didn’t, as they went about developing the module.
- After they completed the development, the developers tested their individual functionalities as part of their **unit testing** and found some defects. They fixed these **defects**. At this point they felt their modules were complete.
- The QA Manager suggested that integration testing should be performed to confirm that all the modules work together.
- When they deployed all of their code in a common machine, they found that the application did not work as expected since the individual modules did not work well together. There were bugs like – after logging in, the user’s shopping cart did not show items they had added earlier, the bill amount did not include shipping cost etc.
- In this way, Integration Testing helps us identify, fix issues and ensure that the application as a whole, works as expected.

## Integration Testing Example

Let us understand Integration Testing with example. Let us assume that you work for an IT organization which has been asked to develop an online shopping website for Camp World, a company that sells camping gear.

After requirements gathering, analysis and design was complete, one developer was assigned to develop each of the modules below.

- User registration and Authentication/Login
- Product Catalogue
- Shopping Cart
- Billing
- Payment gateway integration
- Shipping and Package Tracking

## Types of Integration Testing

- Software Engineering defines variety of strategies to execute Integration testing.
- Big Bang Approach :
- Incremental Approach: which is further divided into the following
  - Top Down Approach
  - Bottom Up Approach
  - Sandwich Approach – Combination of Top Down and Bottom Up

# Big Bang Testing

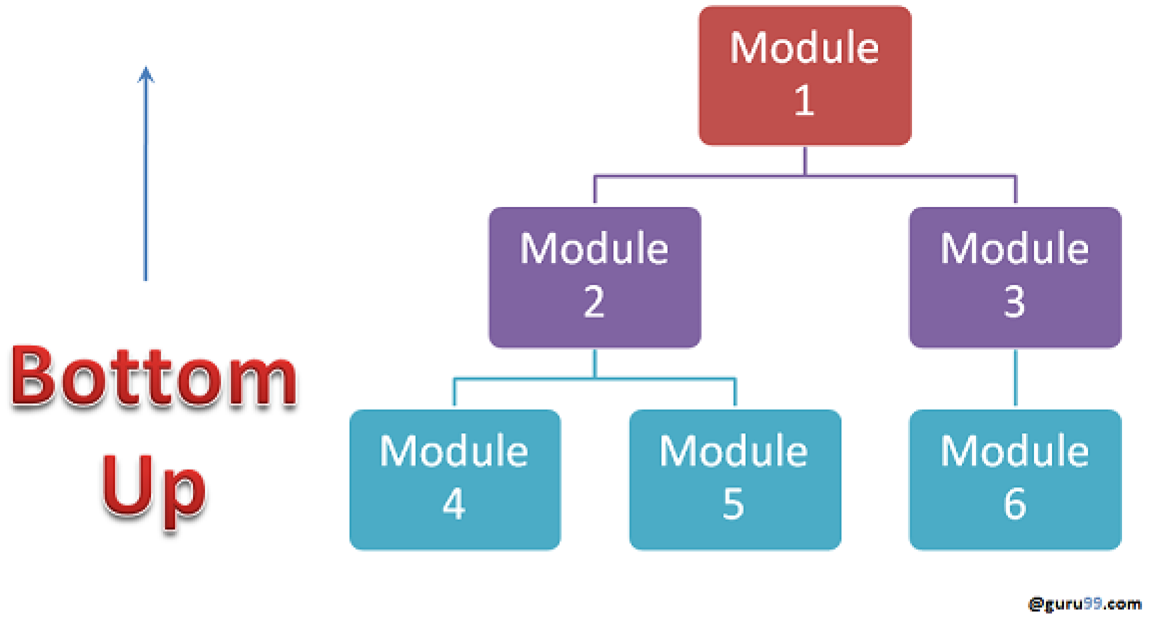
- **Big Bang Testing** is an Integration testing approach in which all the components or modules are integrated together at once and then tested as a unit. This combined set of components is considered as an entity while testing. If all of the components in the unit are not completed, the integration process will not execute.
- **Advantages:**
- Convenient for small systems.
- **Disadvantages:**
- Fault Localization is difficult.
- Given the sheer number of interfaces that need to be tested in this approach, some interfaces link to be tested could be missed easily.
- Since the Integration testing can commence only after "all" the modules are designed, the testing team will have less time for execution in the testing phase.
- Since all modules are tested at once, high-risk critical modules are not isolated and tested on priority. Peripheral modules which deal with user interfaces are also not isolated and tested on priority.

## Incremental Testing

- In the **Incremental Testing** approach, testing is done by integrating two or more modules that are logically related to each other and then tested for proper functioning of the application. Then the other related modules are integrated incrementally and the process continues until all the logically related modules are integrated and tested successfully.
- Incremental Approach, in turn, is carried out by two different Methods:
- Bottom Up
- Top Down

## Bottom-up Integration Testing

- **Bottom-up Integration Testing** is a strategy in which the lower level modules are tested first. These tested modules are then further used to facilitate the testing of higher level modules. The process continues until all modules at top level are tested. Once the lower level modules are tested and integrated, then the next level of modules are formed.

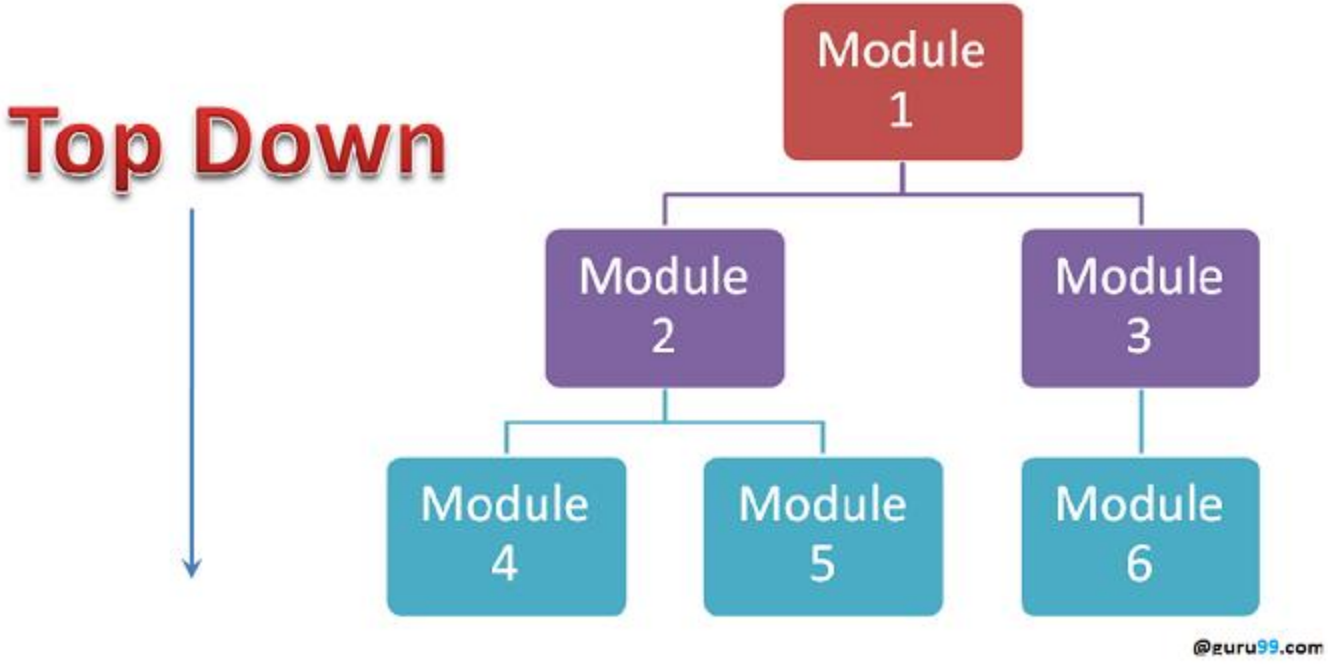


## Bottom-up Integration Testing

- **Advantages:**
- Fault localization is easier.
- No time is wasted waiting for all modules to be developed unlike Big-bang approach
- **Disadvantages:**
- Critical modules (at the top level of software architecture) which control the flow of application are tested last and may be prone to defects.
- An early prototype is not possible

# Top-down Integration Testing

- **Top Down Integration Testing** is a method in which integration testing takes place from top to bottom following the control flow of software system. The higher level modules are tested first and then lower level modules are tested and integrated in order to check the software functionality. Stubs are used for testing if some modules are not ready.

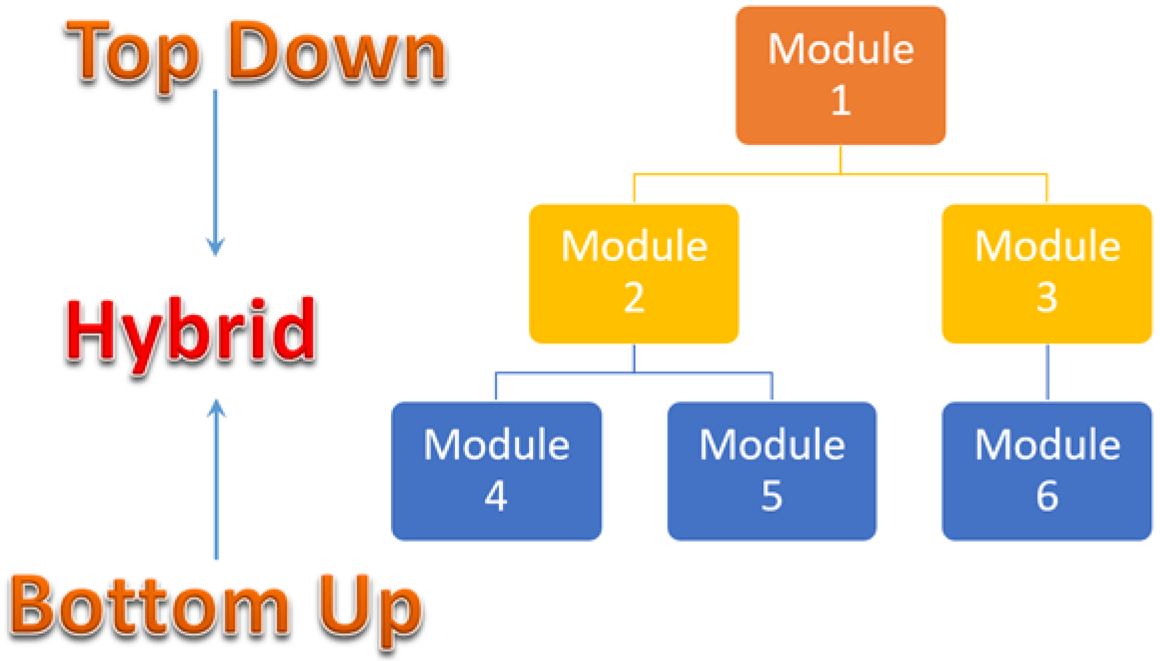


## Top-down Integration Testing

- **Advantages:**
- Fault Localization is easier.
- Possibility to obtain an early prototype.
- Critical Modules are tested on priority; major design flaws could be found and fixed first.
- **Disadvantages:**
- Needs many Stubs.
- Modules at a lower level are tested inadequately.

# Sandwich Testing

- **Sandwich Testing** is a strategy in which top level modules are tested with lower level modules at the same time lower modules are integrated with top modules and tested as a system. It is a combination of Top-down and Bottom-up approaches therefore it is called **Hybrid Integration Testing**. It makes use of both stubs as well as drivers.



# Integration testing tools

- **Popular programming-language specific**
- Some tools are designed to test software coded in a specific coding language, for example, Pytest for Python, H2 Database for Java, or jQuery for JavaScript and PHP.
- **Multiple-language tools**
- These tools are designed to be able to perform tests on programs coded in various languages. Some examples are VectorCast for C and C++, Selenium for Java and Dockerfile, and Cucumber for Java, JavaScript, Ruby, and Kotlin.