ISTQB Ch. 2 – Part A

SOFTWARE DEVELOPMENT LIFECYCLE

WHERE DOES TESTING - LIES IN SDLC

Impact of SDLC in Testing

Various types and Levels of Testing

What is SDLC ?

A Framework that describes the activities performed at each stage of a software development project.

Software Development Models

- The development process adopted for a project will depend on the project aims and goals.

- There are numerous development life cycles that have been developed in order to achieve different required objectives.

Software Testing Models

- The life cycle model that is adopted for a project will have a big impact on the testing that is carried out.
- Testing does not exist in isolation.
- **Testing activities are highly related to software development activities.**
- It will define what, where, and when of our planned testing, influence regression testing, and largely determine which test techniques to use.
- The way testing is organized must fit the development life cycle or it will fail to deliver its benefit.

Waterfall Model

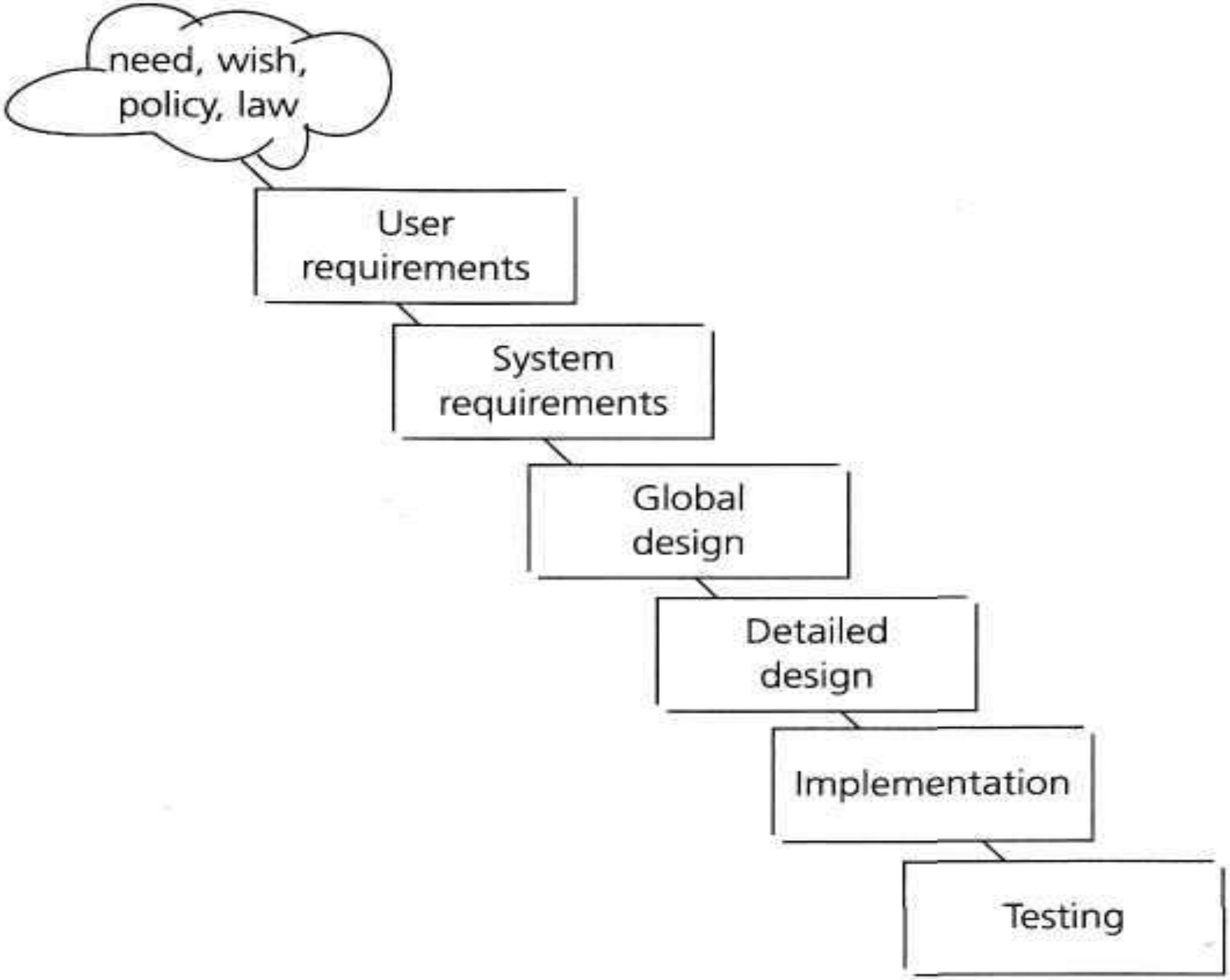


FIGURE 2.1 Waterfall model

Features of a Waterfall Model

- A waterfall model is easy to follow.
- It can be implemented for any size project.
- Every stage has to be done separately at the right time so you cannot jump stages.
- Documentation is produced at every stage of a waterfall model allowing people to understand what has been done.
- Testing is done only at the end.

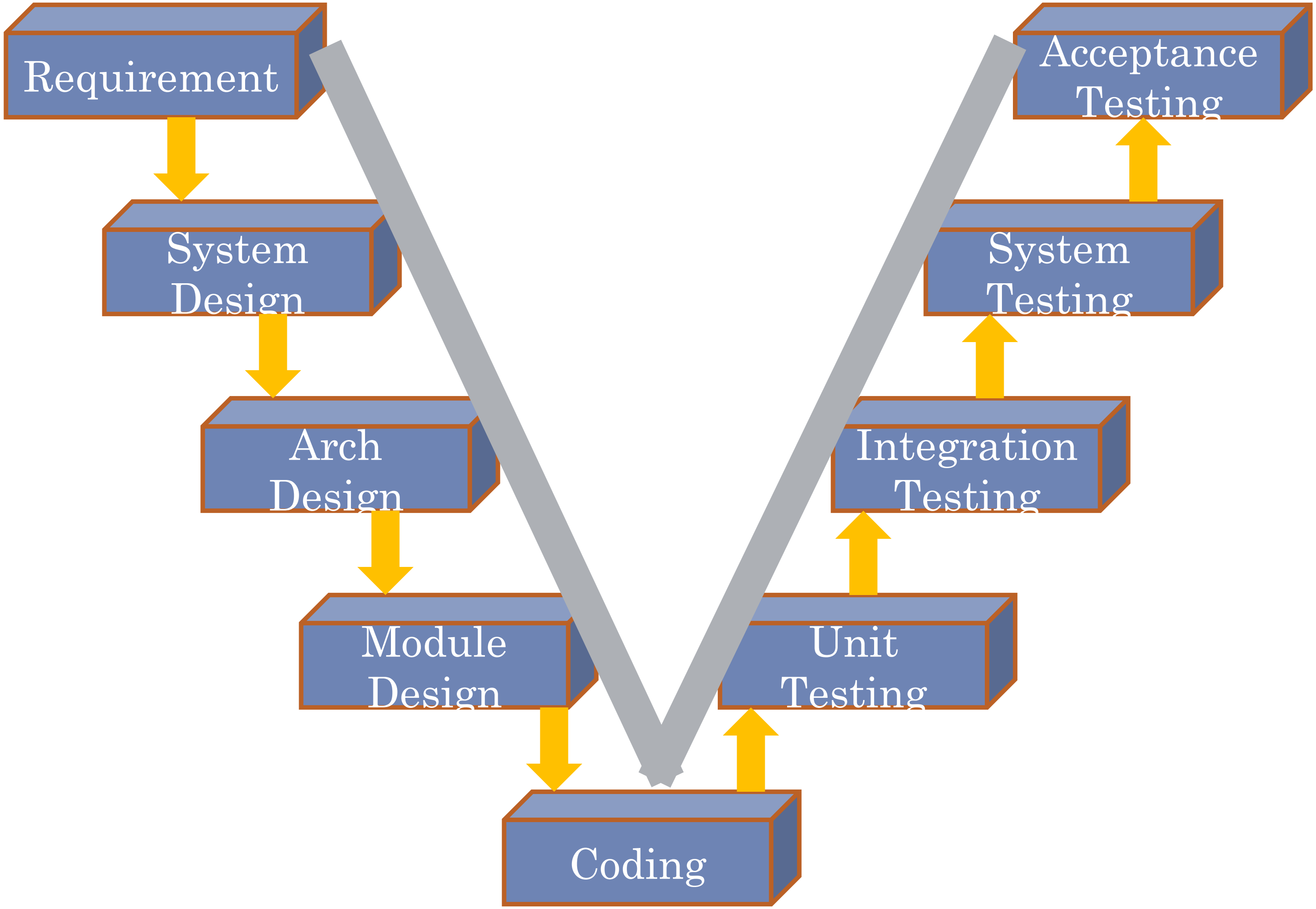
Advantages of a Waterfall Model

- Simple to follow.
- Requirements will be set and these wouldn't be changed.(Advantage for Developer).
- As everything is documented a new team member can easily understand what's to be done.
- Implementers have to follow the design accurately.

Disadvantages of a Waterfall Model

- If requirements change the Waterfall model may not work. (Disadvantage for Customer).
- Many believe it is impossible at one stage to make the projects perfect.
- Difficult to estimate time and cost for each stage of the development process.

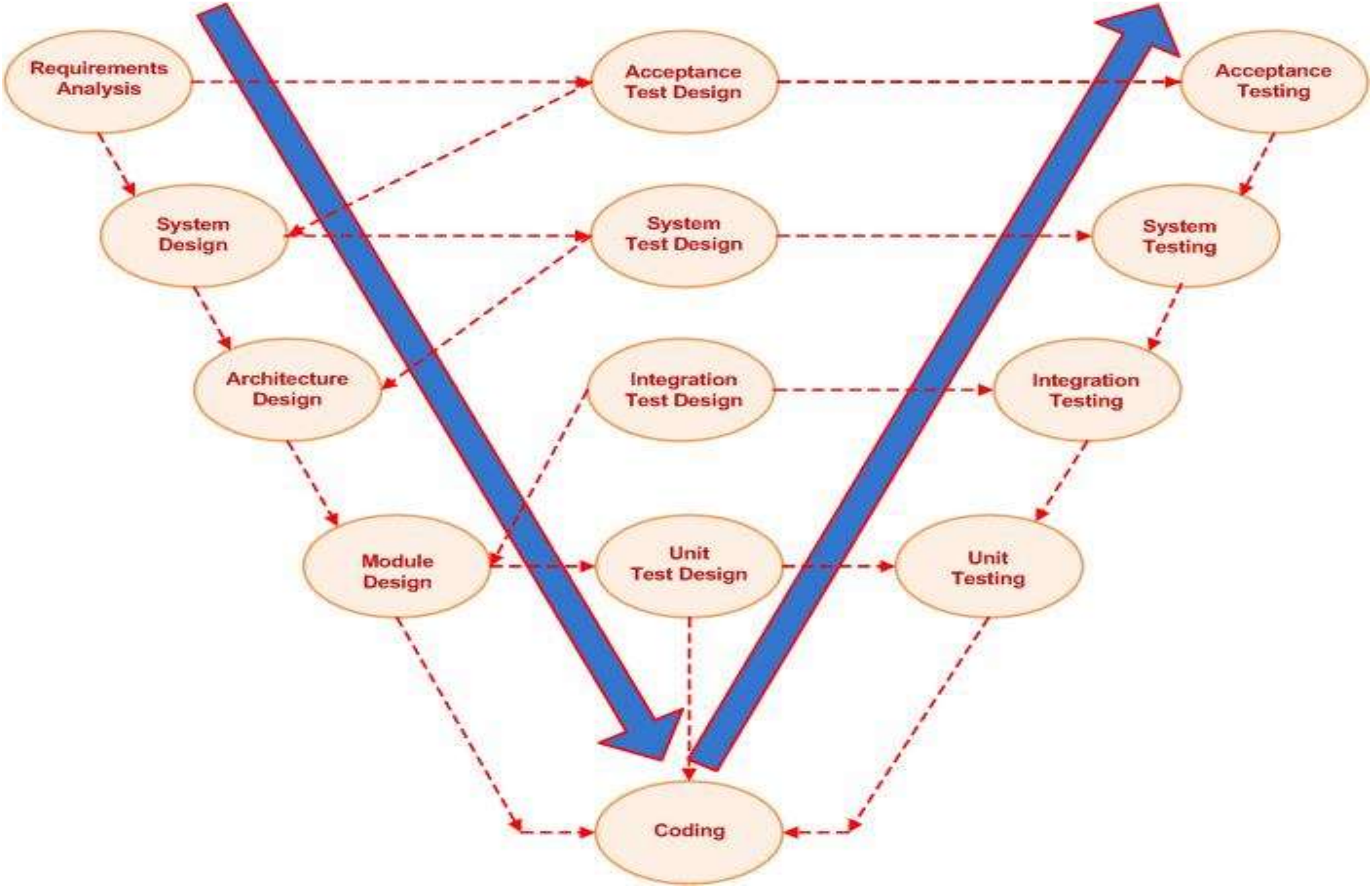
V- MODEL


Requirement
System Design
Arch Design
Module Design
Coding
Unit Testing
Integration Testing
System Testing
Acceptance Testing

The V-Shaped Model

- Testing should not be considered late in project
- **Development and testing go hand in hand**
- When you are developing software, you should perform static testing
  - ✓ Plan for testing
  - ✓ Design test cases
  - ✓ Create test data
- When you finish development, perform dynamic testing
  - ✓ Perform testing at various levels
    - Unit testing
    - Integration testing
    - System Testing
    - Acceptance Testing`

Steps in the V-Shaped Model



Quality is guaranteed at each project stage.

Verification and Validation

Verification	Validation
Static	Dynamic
Are we developing product right ?	Are we developing right product ?
Involves checking of documents	Involves execution of code
We get presence and location of defect	We get only presence of defect, not location
Examples – Walkthrough, inspection, technical review	Examples – unit testing, integration testing, system testing, acceptance testing

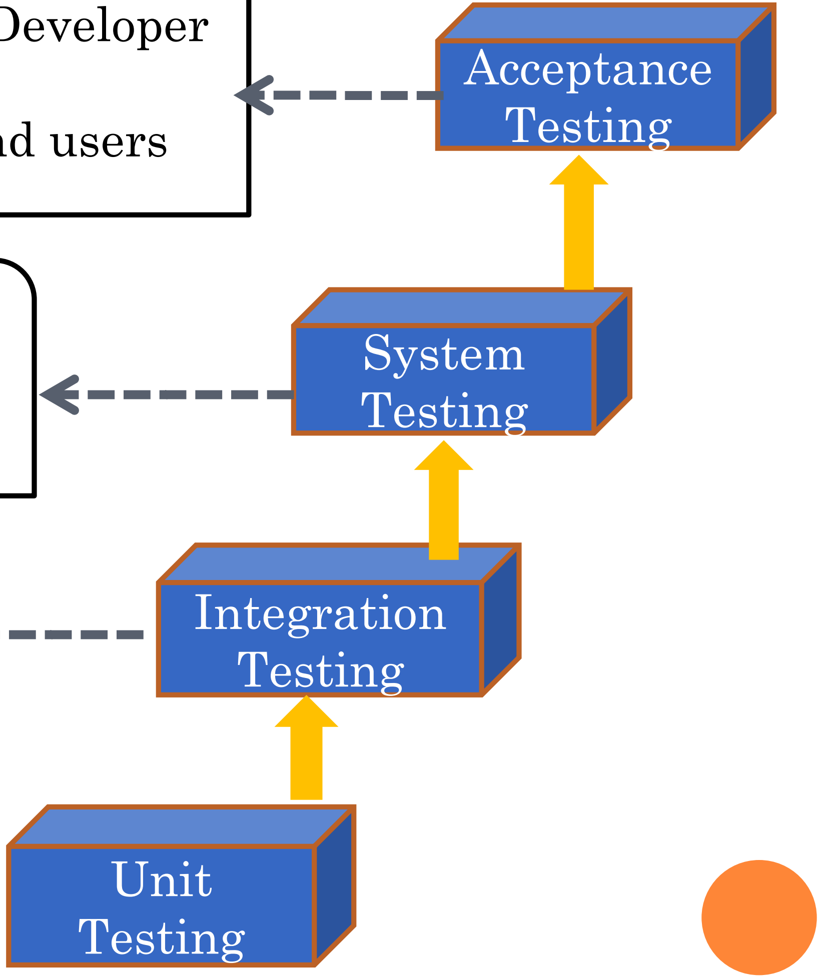
Levels of Testing

Acceptance Testing :
->Alpha Testing:-It is conducted at the Developer sight by end users
->Beta Testing:-It is conducted at the end users site

System Testing :
-> will compare the system specifications against the actual system

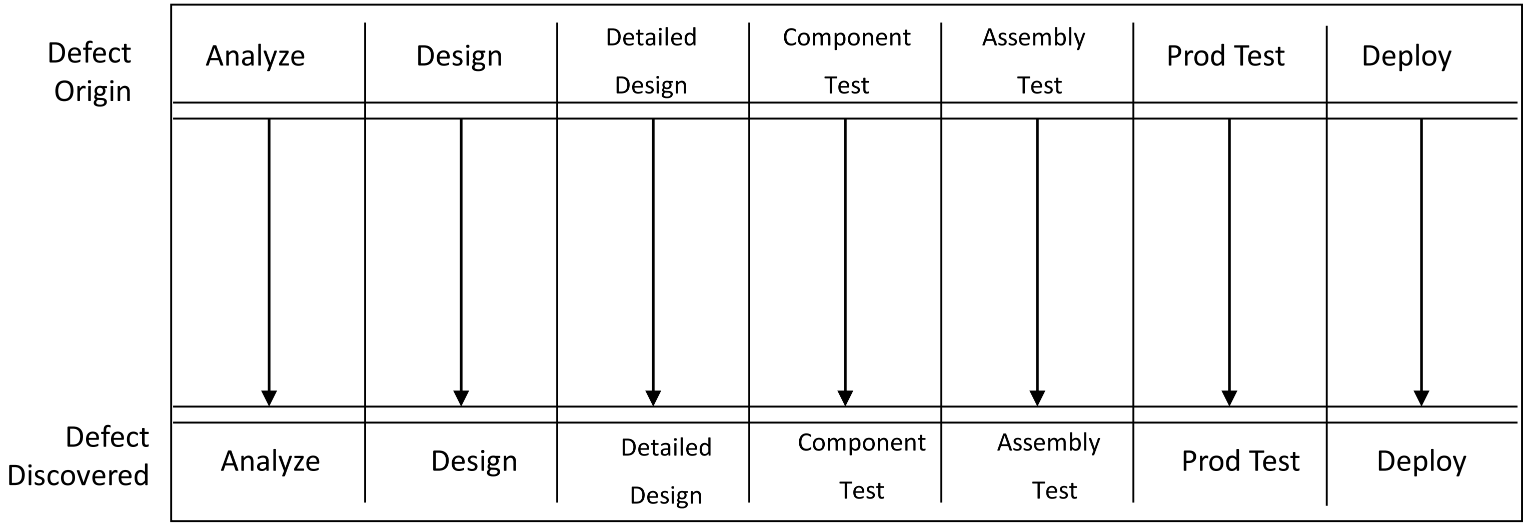
Integration Testing :
-> After combining all the Modules

Unit Testing :
-> White Box Testing
-> Black Box Testing



Stage Containment

- This term is used to identify problems existing in the product being developed before proceeding to the following stage.
  - ✓ More errors than defects.
  - ✓ Cost and effort for fixing problems is minimized.

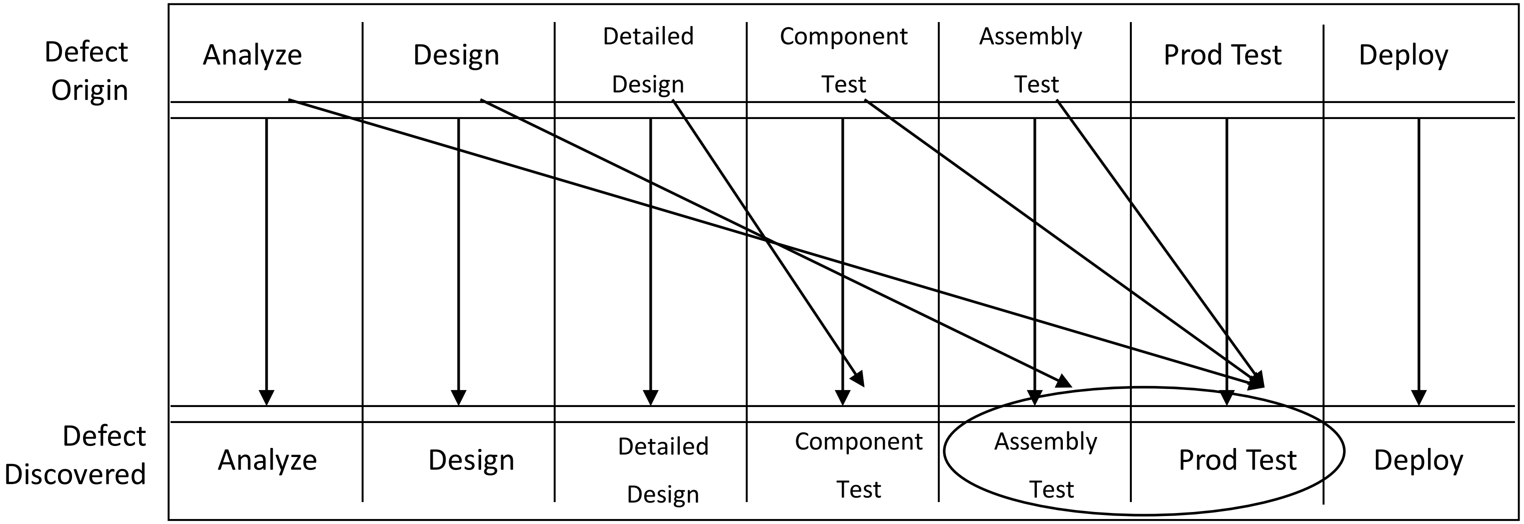


Defect Origin	Analyze	Design	Detailed Design	Component Test	Assembly Test	Prod Test	Deploy
Defect Discovered	Analyze	Design	Detailed Design	Component Test	Assembly Test	Prod Test	Deploy

With Stage Containment

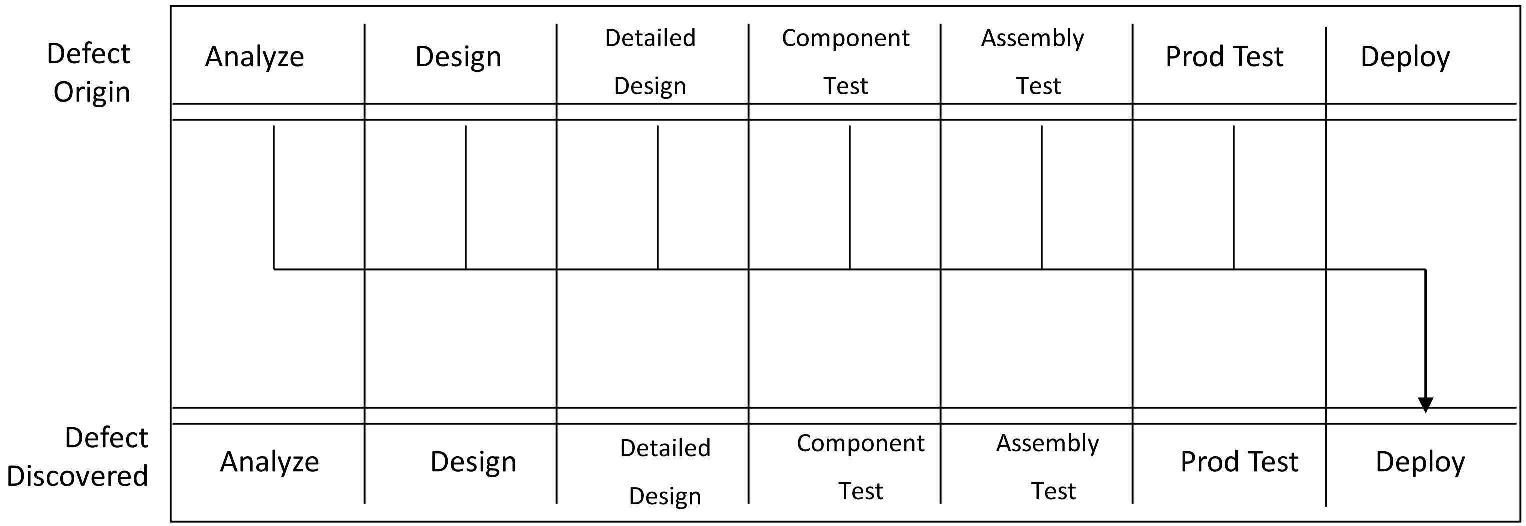
Without Stage Containment

- More defects than errors.
- Fixes become more expensive and difficult.



Without Stage Containment

Without Stage Containment



Defect Origin

Analyze | Design | Detailed Design | Component Test | Assembly Test | Prod Test | Deploy

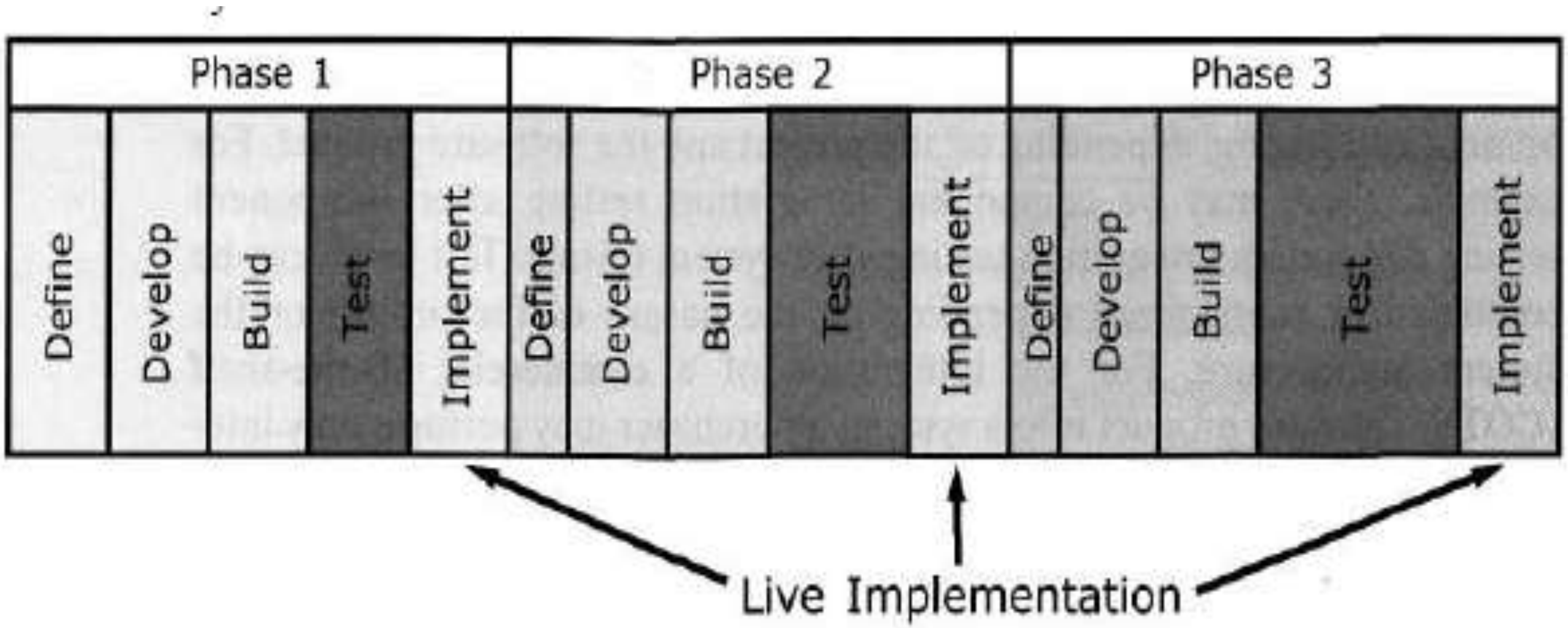
Defect Discovered

Analyze | Design | Detailed Design | Component Test | Assembly Test | Prod Test | Deploy

Worst Case!

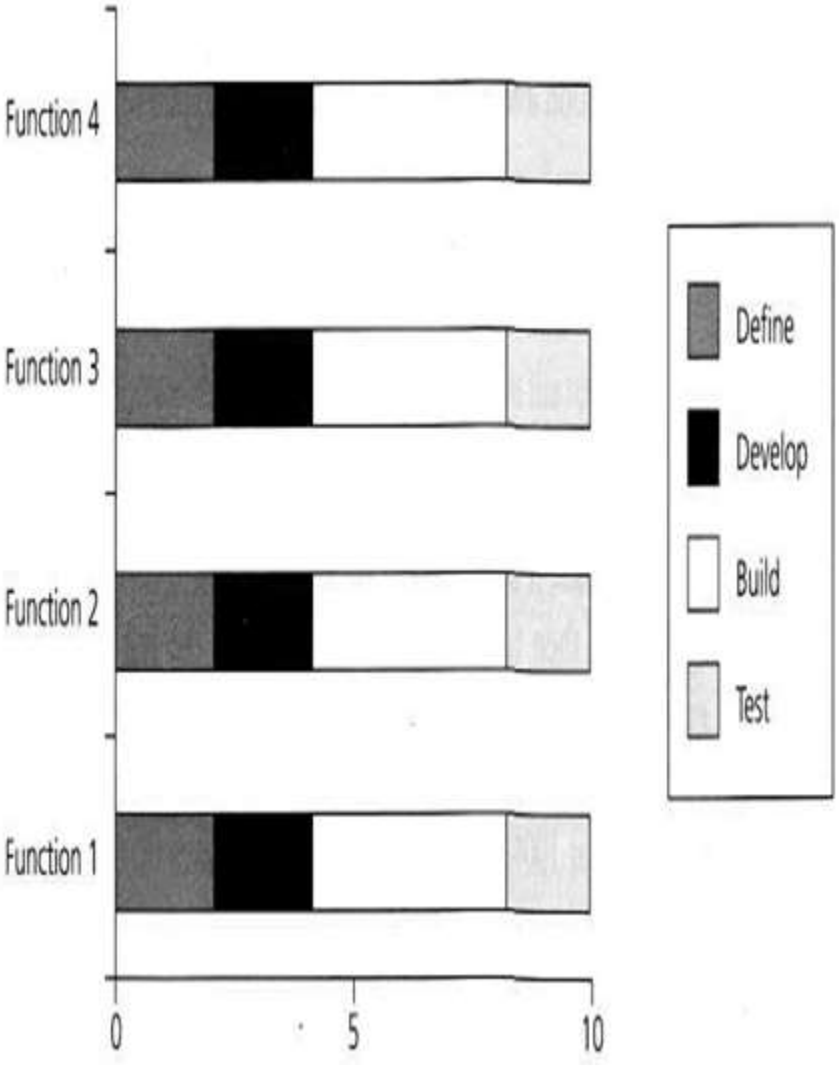
Iterative Life Cycle

- Not all life cycles are sequential.
- There are also iterative or incremental life cycles where, instead of one large development time line from beginning to end, we cycle through a number of smaller self-contained life cycle phases for the same project



Rapid Application Development (RAD)

- Rapid Application Development (RAD) is formally a parallel development of functions and subsequent integration.
- Components/functions are developed in parallel as if they were mini projects,
- The developments are time-boxed, delivered, and then assembled into a working prototype.
- This can very quickly give the customer something to see and use .
- Helps to get feedback regarding the delivery and their requirements.



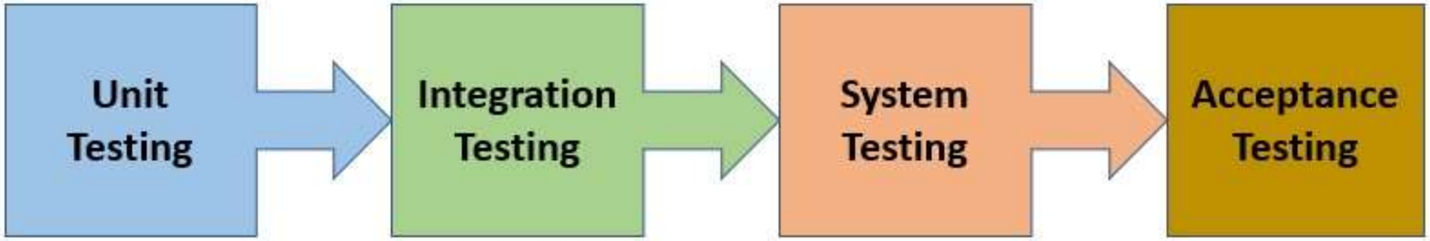
Agile Development

More human friendly than traditional development methodology.

- It promotes the generation of business stories to define the functionality.
- It demands an on-site customer for continual feedback and to define and carry out functional acceptance testing.
- It promotes pair programming and shared code ownership amongst the developers.
- It states that component test scripts shall be written before the code is written and that those tests should be automated.
- It states that integration and testing of the code shall happen several times a day.
- It states that we always implement the simplest solution to meet today's problems.

Validation

- Dynamic testing
- Levels of testing
  - Unit / component testing
  - Integration testing
  - System testing
  - Acceptance testing



Unit Testing

Integration Testing

System Testing

Acceptance Testing

Test Level – Component Testing

- Component testing, also known as unit, module and program testing.
- Searches for defects in, and verifies the functioning of software e.g.
  - modules,
  - programs,
  - objects, classes, etc. that are separately testable.
- Component testing may be done in isolation from the rest of the system.

Integration Testing

- The entire system is viewed as a collection of subsystems (sets of classes) determined during the system and object design

- Goal: Test all interfaces between subsystems and the interaction of subsystems

- The Integration testing strategy determines the order in which the subsystems are selected for testing and integration.

WHY DO WE DO INTEGRATION TESTING?

- Unit tests only test the unit in isolation
- Many failures result from faults in the interaction of subsystems
- Often many off-the-shelf components are used that cannot be unit tested
- Without integration testing the system test will be very time consuming
- Failures that are not discovered in integration testing will be discovered after the system is deployed and can be very expensive.

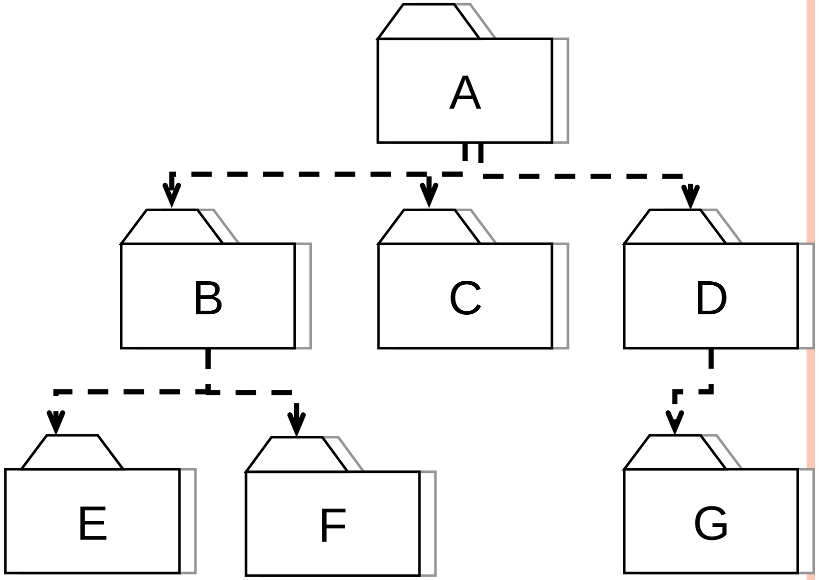
Integration Testing - Methods

- Bottom Up
- Top Down
- Big Bang
- Critical Part First

Bottom-up Testing Strategy

- The subsystems in the lowest layer of the call hierarchy are tested individually
- Then the next subsystems are tested that call the previously tested subsystems
- This is repeated until all subsystems are included
- Drivers are needed.

Bottom-up Integration



A

B

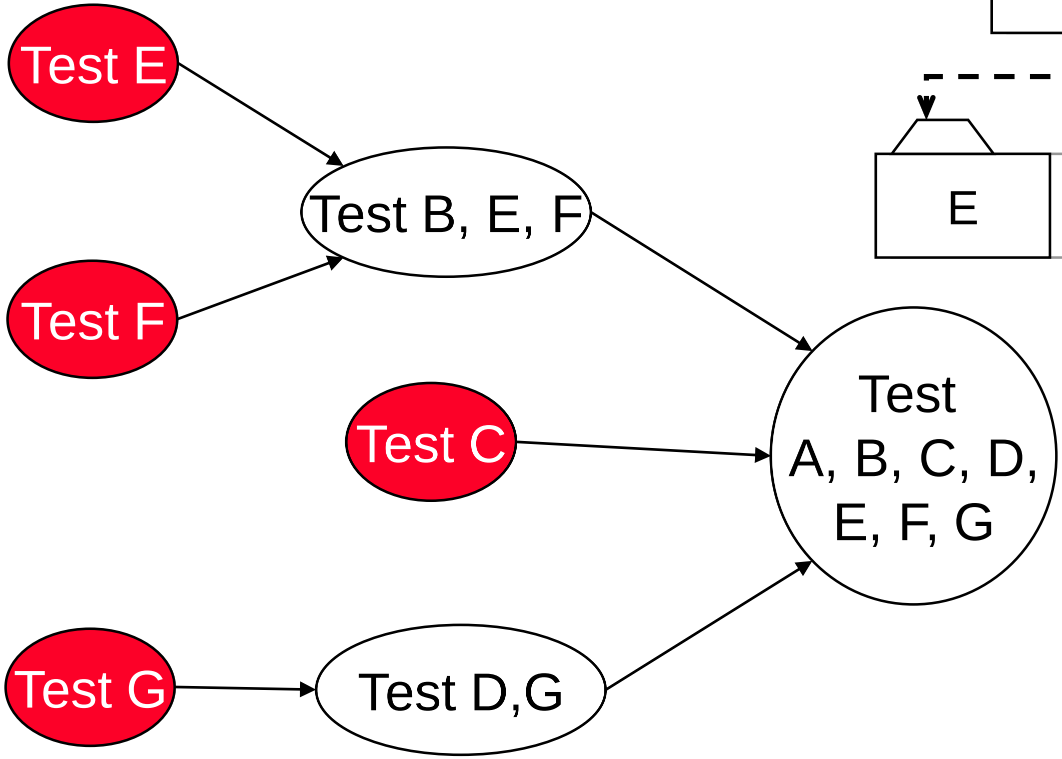
C

D

E

F

G



Test E

Test F

Test B, E, F

Test C

Test
A, B, C, D,
E, F, G

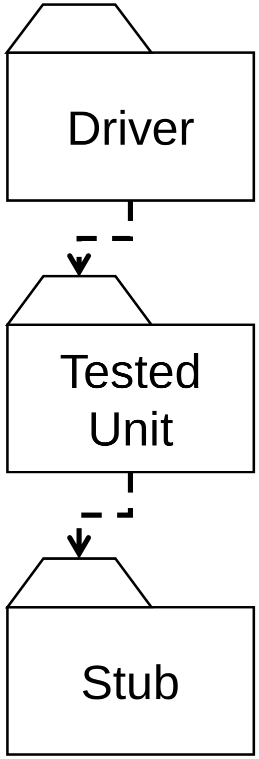
Test G

Test D,G

Stubs and drivers

- Driver:
  - A component, that calls the `TestedUnit`
  - Controls the test cases

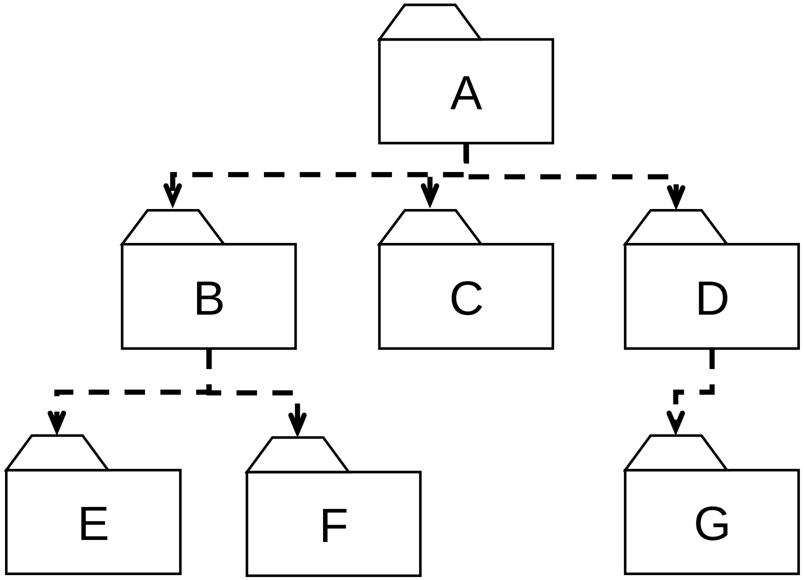
- Stub:
  - A component, the `TestedUnit` depends on
  - Partial implementation
  - Returns fake values.

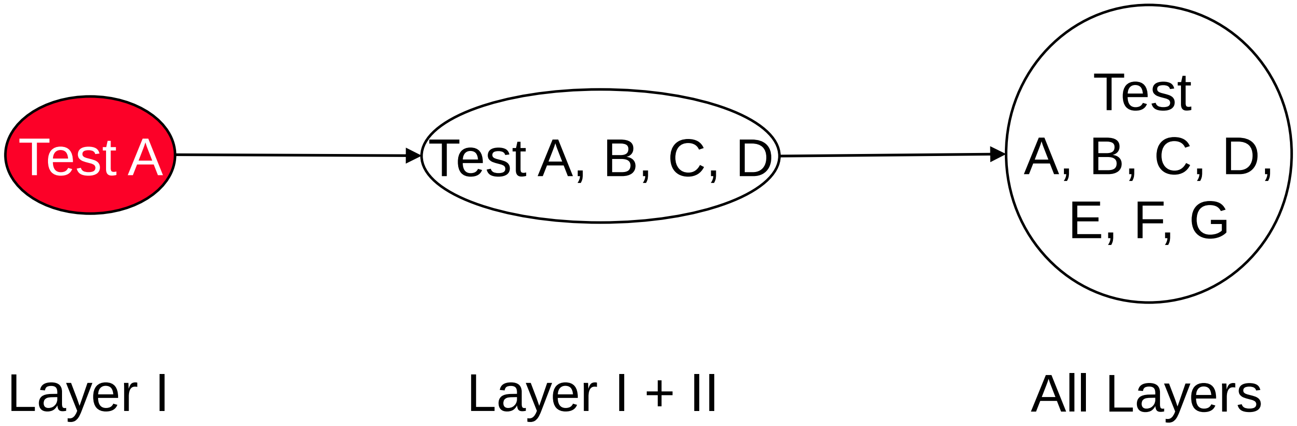


Top-down Testing Strategy

- Test the top layer or the controlling subsystem first
- Then combine all the subsystems and test the resulting collection of subsystems
- Do this until all subsystems are incorporated into the test
- Stubs are needed to do the testing.

Top-down Integration





Test A

Test A, B, C, D

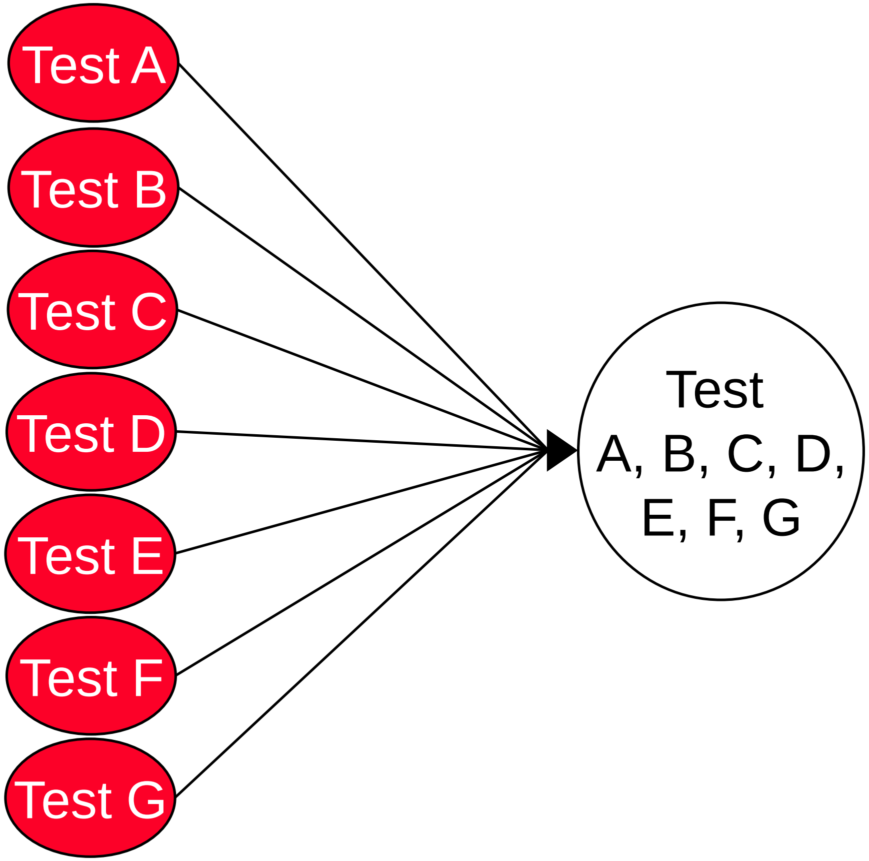
Test
A, B, C, D,
E, F, G

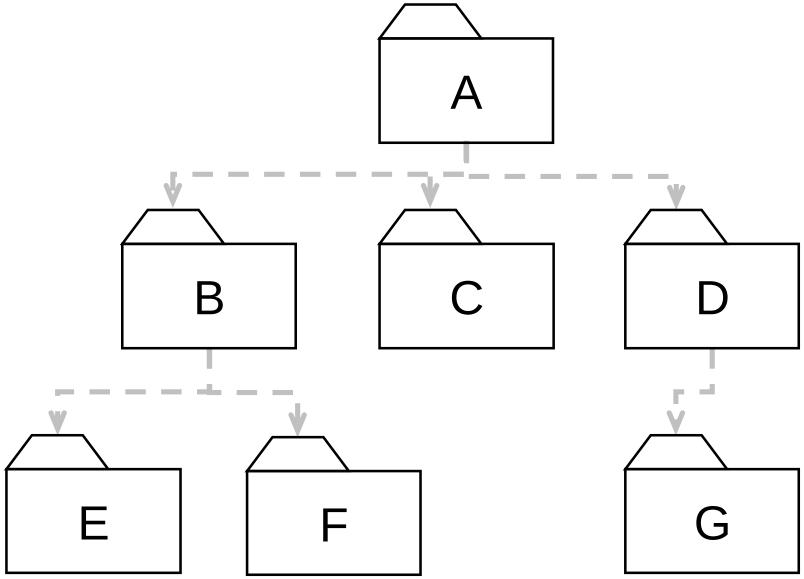
Layer I

Layer I + II

All Layers

Big-Bang Approach





Big Bang Integration

- Advantages
  - Convenient for small systems
- Disadvantages
  - Does not need driver and stubs
  - Integration testing can only begin when all modules are ready
  - Fault localization difficult
  - Easy to miss interface faults

Critical Part First

- The entire application is not tested, neither from Top-Down or Bottom-Up
- A part of system is selected for integration testing
- Normally, the critical modules are selected

System Testing

- It includes testing of entire system, consisting of
  - Hardware
  - OS
  - RDBMS
  - Tools. Etc
- Mostly, it is carried out by
  - Specialist testers
  - Independent test team
  - In some organizations system testing is carried out by a third party team or by business analysts.
- System testing should investigate both functional and non-functional requirements of the system. Typical non-functional tests include performance and reliability.

Acceptance Testing

- Acceptance testing is most often the responsibility of the user or customer, although other stakeholders may be involved as well.
- The execution of the acceptance test requires a test environment that is for most aspects, representative of the production environment.
- The goal of acceptance testing is to establish confidence in the system, part of the system or specific non-functional characteristics, e.g. usability, of the system.
- For a Commercial Off The Shelf (COTS) software product acceptance testing may be the only testing when the product is installed or integrated.

Alpha Testing

- This test takes place at the developer's site.
- A cross-section of potential users and members of the developer's organization are invited to use the system.
- Developers observe the users and note problems.
- Alpha testing may also be carried out by an independent test team.

Beta Testing

- Also called field testing.
- System is sent to a cross-section of users who install it and use it under real-world working conditions.
- The users send records of incidents with the system to the development organization where the defects are repaired.