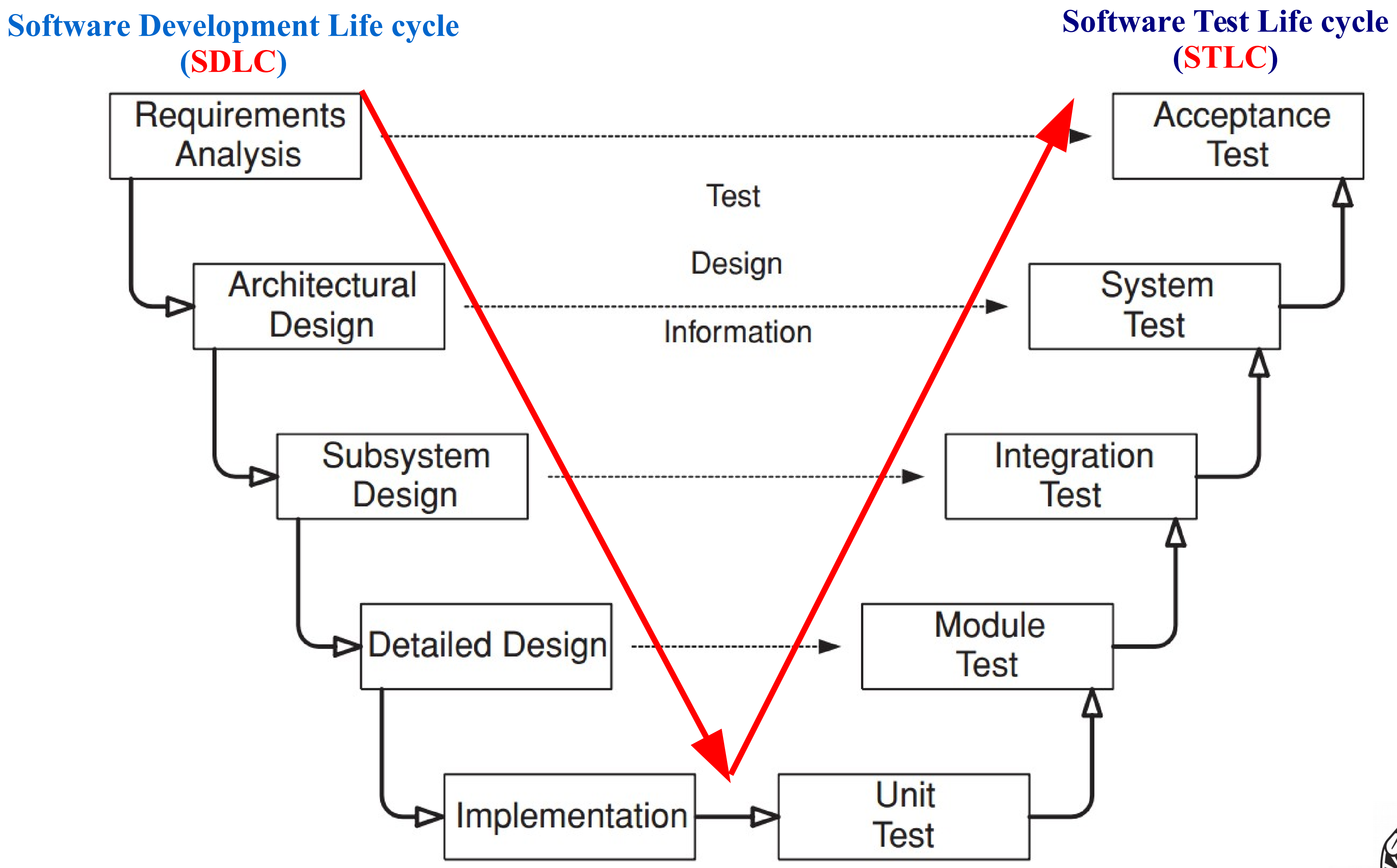
Topics for this Lecture

- Testing Granularity Levels
- The Concept of Integration Testing
- System Integration Techniques: Incremental, Top-down, Bottom-up, and Big-bang

V Model of Testing



Software development activities and testing levels- the "V Model"

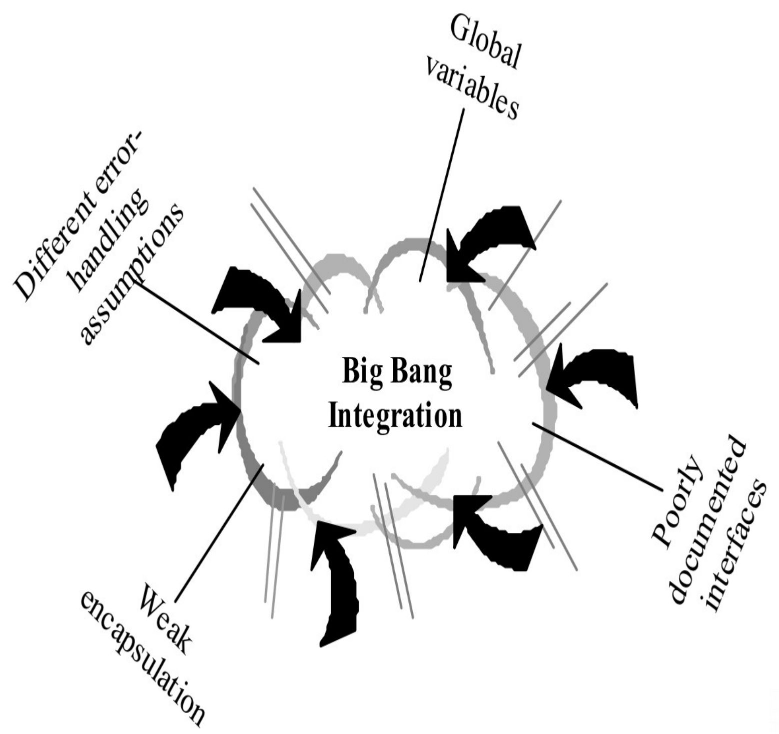
Integration testing

- **Integration testing** is the combined execution of two or more modules, packages, components, or subsystems that have been created by multiple programmers or programming teams.
- This kind of testing typically starts as soon as there are two classes to test and continues until the entire system is complete.
- **Integration versus Unit Testing**
  - Integration testing focuses mainly on the interfaces and flow of data between the modules/components.
  - Unit testing tests small pieces of code, typically individual module/function, alone.
- **Challenges**:
  - How to test a partial system where not all parts exist?
    - Creating dummy programs called **Stubs** and **Drivers**.
      - **Stub**: Is called by the module under Test.
      - **Driver**: Calls the module to be tested.

Oregon State University

Big Bang Integration Test

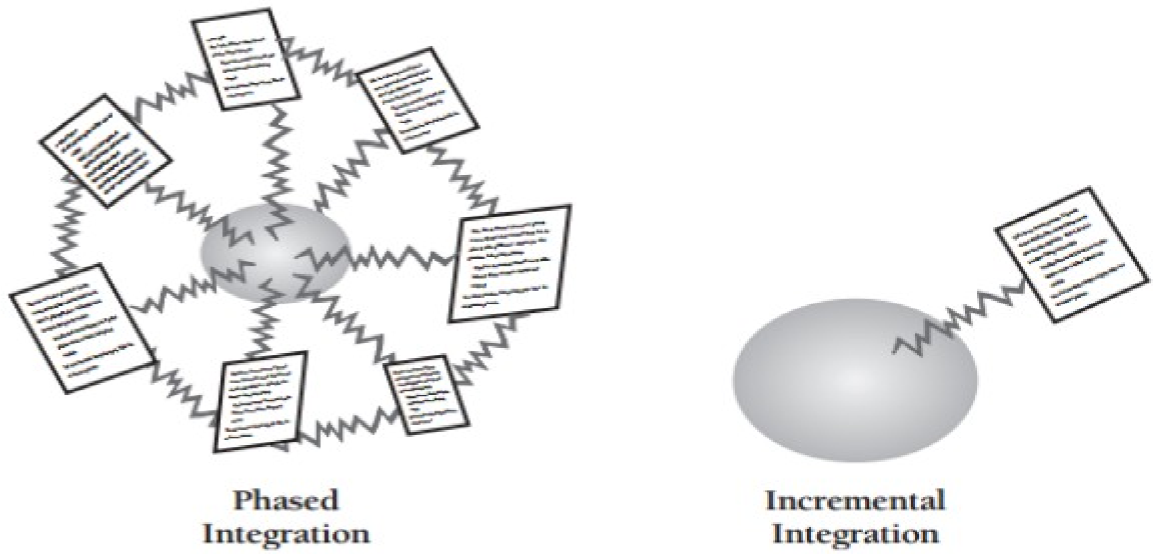
- First, design, code, test, and debug each model. This step is called "unit development."
- Next, combine all those modules to construct the entire system.
- Then, test and debug the whole system.
- **Phased integration** can't begin until late in the project, after all the classes have been developer-tested.
- For small programs phased integration might be the best approach.


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

Oregon State University

Incremental integration

- **Incremental integration**
- Develop a small, functional part of the system, which it will serve as a skeleton system
- Design, code, test, debug a small new piece
- Integrate this piece with the skeleton
- Test/debug it before adding any other pieces



Oregon State University

Benefits of incremental

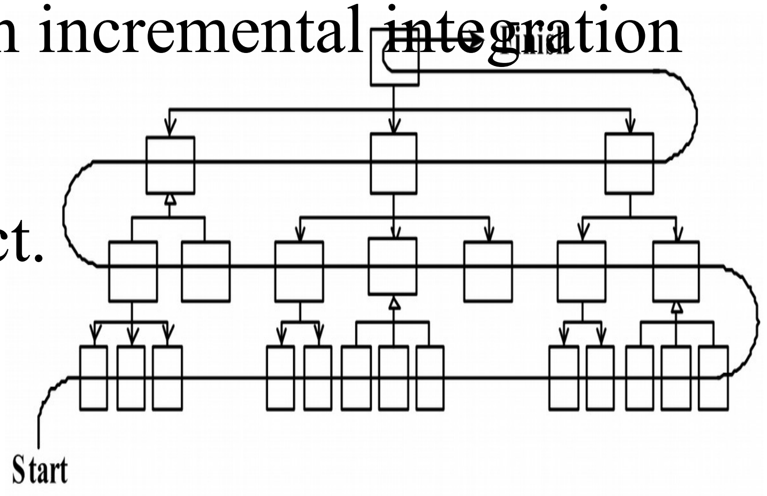
- **Advantages**:
  - Errors are easy to locate and fix
  - System is always in a (relatively) working state
  - You get improved progress monitoring
- **Drawbacks**:
  - With incremental integration, you have to plan more carefully
  - May need to create "stub" and “driver”versions of some features that have not yet been integrated

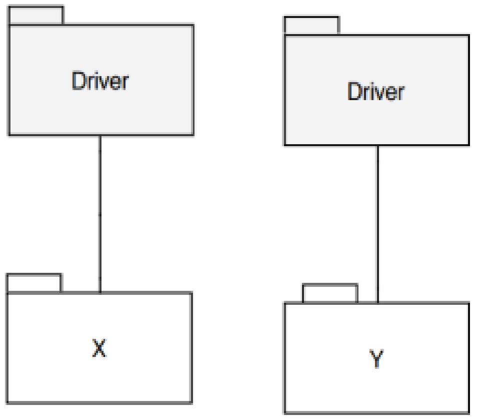
Incremental integration

- Incremental Approach is carried out by two different Methods:
  - Bottom Up Integration
  - Top Down Integration

Bottom-up integration

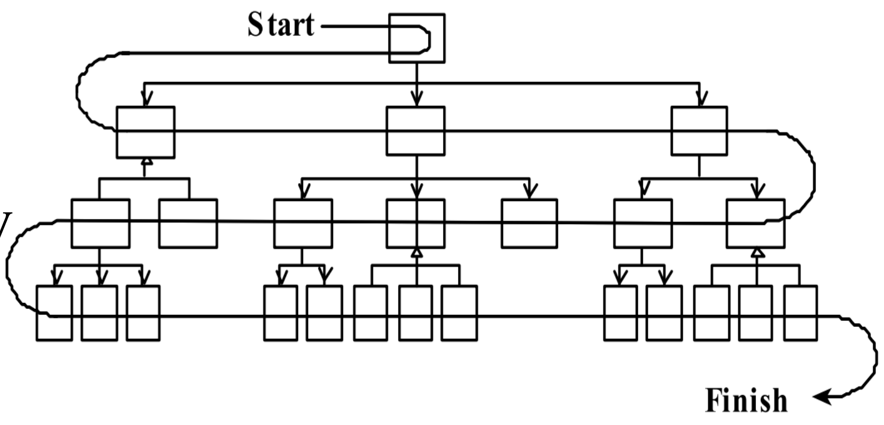
- You write and integrate the classes at the bottom of the hierarchy first.
- Adding the low-level classes one at a time rather than all at once is what makes bottom-up integration an incremental integration strategy
- Integration can start early in the project.
- The main problem with *bottom-up integration* is that it leaves integration of the major, high-level system interfaces until last.



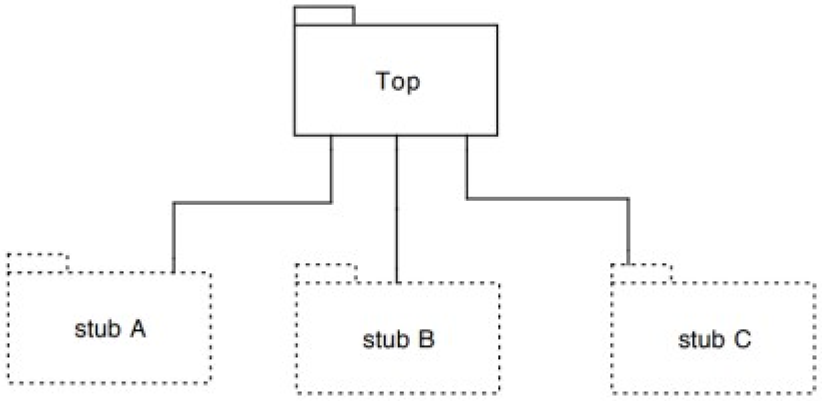


Top-down integration

- **Top down Integration Approach**: The module at the top of the hierarchy is written and integrated first



- Must write (lots of) **stub** lower modules to interact with.
- Then, as classes are integrated from the top down, stub classes are replaced with real ones
- You can complete a partially working system early in the project.



References:

Code Complete: A Practical Handbook of Software Construction, Second Edition 2nd Edition by Steve McConnell
Ammann, Paul, and Jeff Offutt. Introduction to software testing. Cambridge University Press, 2008.
Patton, R. (2005). Software Testing (2nd ed.).