Topic #1 – Software Integration Testing

Instructor: Jerry Gao, Ph.D., Professor
San Jose State University

Software Testing

Module #6– Software Integration Testing

What Is Software Integration Testing?

Why do we perform Integration Testing ?

Software Integration Strategy

Integration Testing Approaches

Summary

What is software integration testing?

Definition: Testing activities that integrate software components together to form a complete system. To perform a cost-effective software integration, integration test strategy, integration test set are needed.

What are the major testing focuses ?

- ✓Interfaces between modules (or components)
- ✓Integrated functional features
- ✓Interacting protocols and messages
- ✓System architectures

Why do we need Integration Testing ?

✓Unit tests only test the unit in isolation

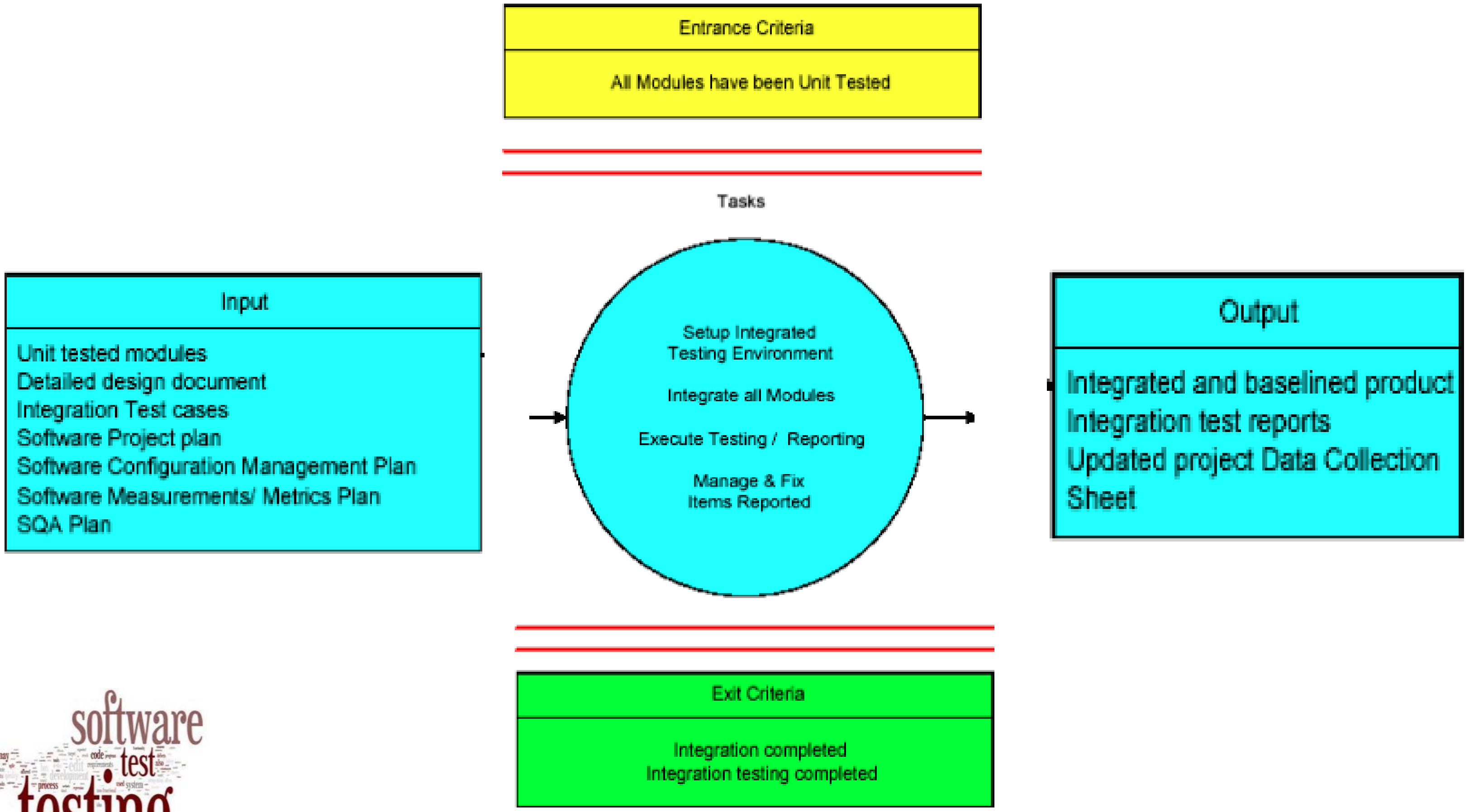
✓Many failures result from faults in the interaction of subsystems

✓Often many Off-the-shelf components are used that cannot be unit tested

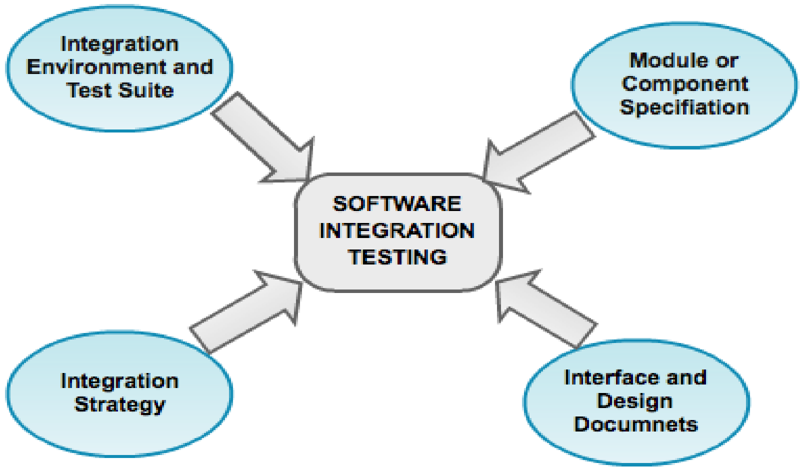
✓Without integration testing the system test will be very time consuming

✓Failures that are not discovered in integration testing will be discovered after the system is deployed and can be very expensive.

Integration Testing Focuses


Entrance Criteria
All Modules have been Unit Tested
Tasks
Input
Unit tested modules
Detailed design document
Integration Test cases
Software Project plan
Software Configuration Management Plan
Software Measurements/ Metrics Plan
SQA Plan
Setup Integrated Testing Environment
Integrate all Modules
Execute Testing / Reporting
Manage & Fix Items Reported
Output
Integrated and baselined product
Integration test reports
Updated project Data Collection Sheet
Exit Criteria
Integration completed
Integration testing completed

What do we need for Integration Testing


Integration Environment and Test Suite
Module or Component Specifiation
SOFTWARE INTEGRATION TESTING
Integration Strategy
Interface and Design Documnets

Who performs Integration Testing ?

✓Developers and Test Engineers

Software Integration Strategy

What is software integration strategy ?

✓Software test strategy provides the basic strategy and guidelines to test engineers to perform software testing activities in a rational way.

Software integration strategy usually refers to
✓ An integration sequence (or order) to integrate different parts (or components) together.
✓A test model is needed to support the definition of software integration test strategies.

Test Models in Integration Testing

- ✓ control flow graph
- ✓object-oriented class diagram
- ✓scenario-based model
- ✓component-based integration model
- ✓architecture-based integration model

Traditional Software Integration Strategy

There are two groups of software integration strategies:

→ Non Incremental software integration
→ Incremental software integration

Non Incremental software integration:

→ Big band integration approach

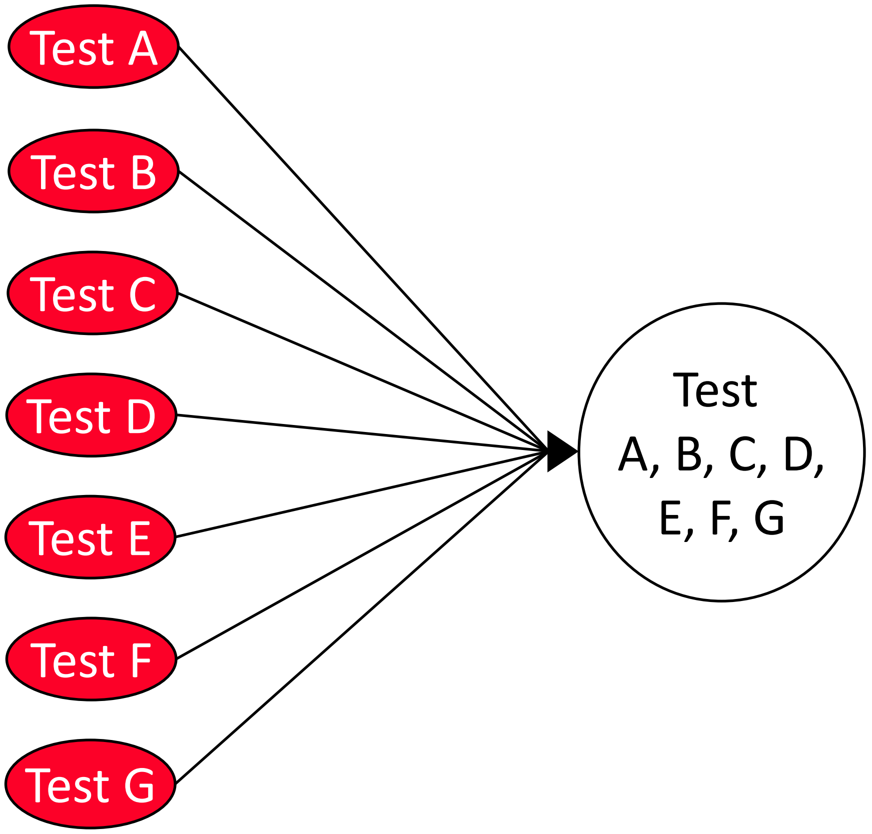
Incremental software integration:

→Top- down software integration
→ Bottom-up software integration
→ Sandwich integration

Non Incremental Software Integration

Big band Integration Approach

✓Combine (or integrate) all parts at once.



MODULE #6– SOFTWARE INTEGRATION TESTING

Non Incremental Software Integration

Big Band Integration Approach

Advantages:

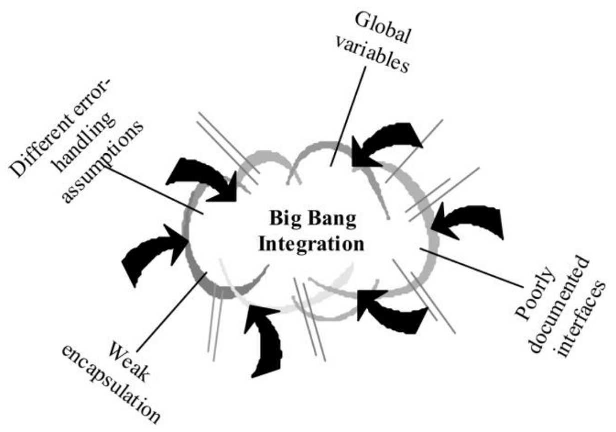
✓simple

Disadvantages:

✓ hard to debugging, not easy to isolate errors

✓not easy to validate test results

✓ impossible to form an integrated system


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

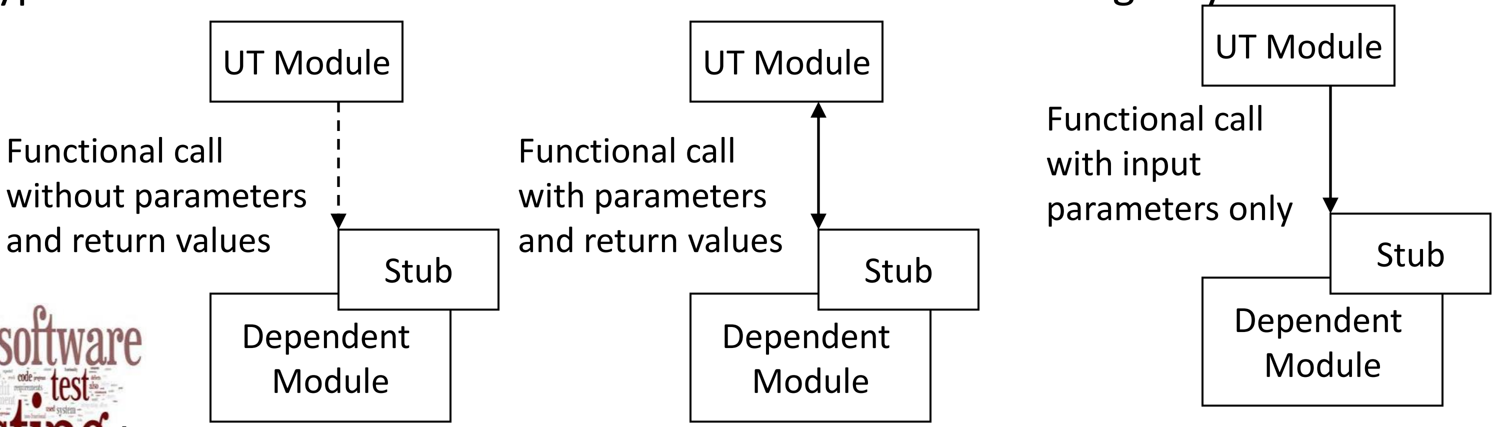
MODULE #6– SOFTWARE INTEGRATION TESTING

Test Stubs and Test Drivers

What are software test stubs?

→Software test stubs are programs which simulate the behaviors of software components (or modules) that are the dependent modules of a under test module.

Typical stubs relates to a under test module in the following ways:

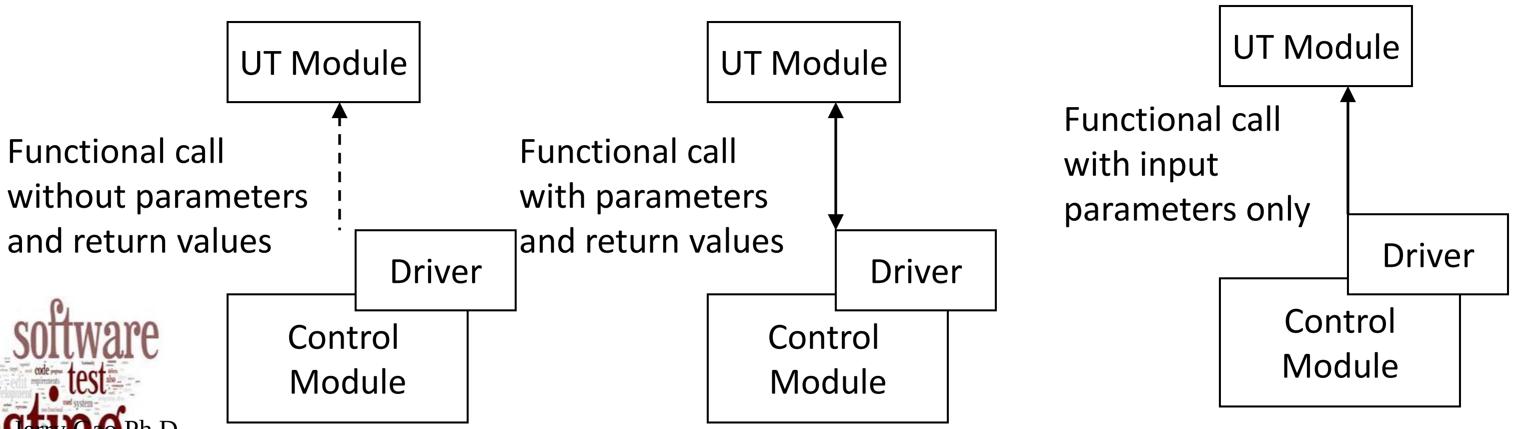

UT Module
Functional call without parameters and return values
Stub
Dependent Module
UT Module
Functional call with parameters and return values
Stub
Dependent Module
UT Module
Functional call with input parameters only
Stub
Dependent Module

Test Stubs and Test Drivers

What are software test drivers?

→Software test drivers are programs which simulate the behaviors of software components (or modules) that are the control modules of a under test module.

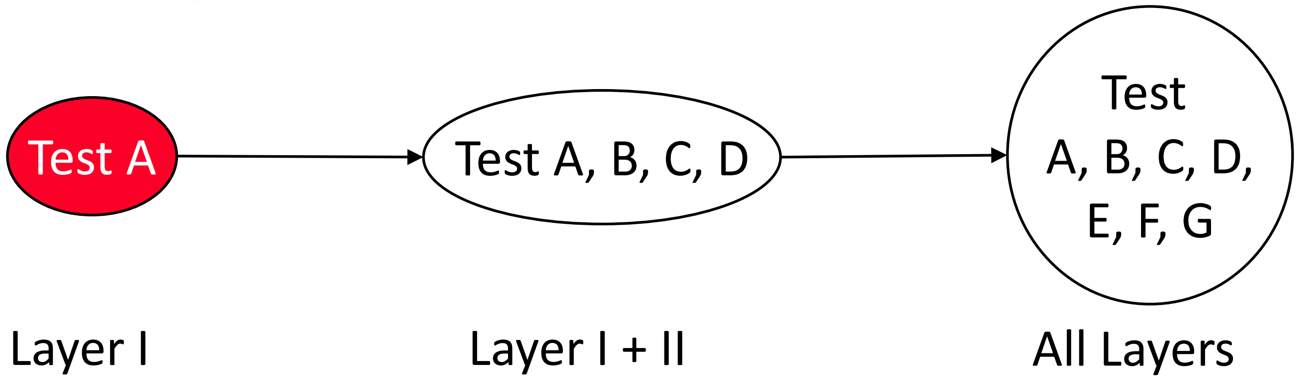
Typical drivers relates to a under test module in the following ways:


UT Module
Functional call without parameters and return values
Driver
Control Module
UT Module
Functional call with parameters and return values
Driver
Control Module
UT Module
Functional call with input parameters only
Driver
Control Module

Top-down Integration Approach

Idea:-Modules are integrated by moving downward through the control structure.

Modules subordinate to the main control module are incorporated into the system
in either a depth-first or breadth-first manner.



Top-down Integration Approach

Integration process (five steps):

✓The main control module is used as a test driver, and the stubs are substituted for all modules directly subordinate to the main control module.

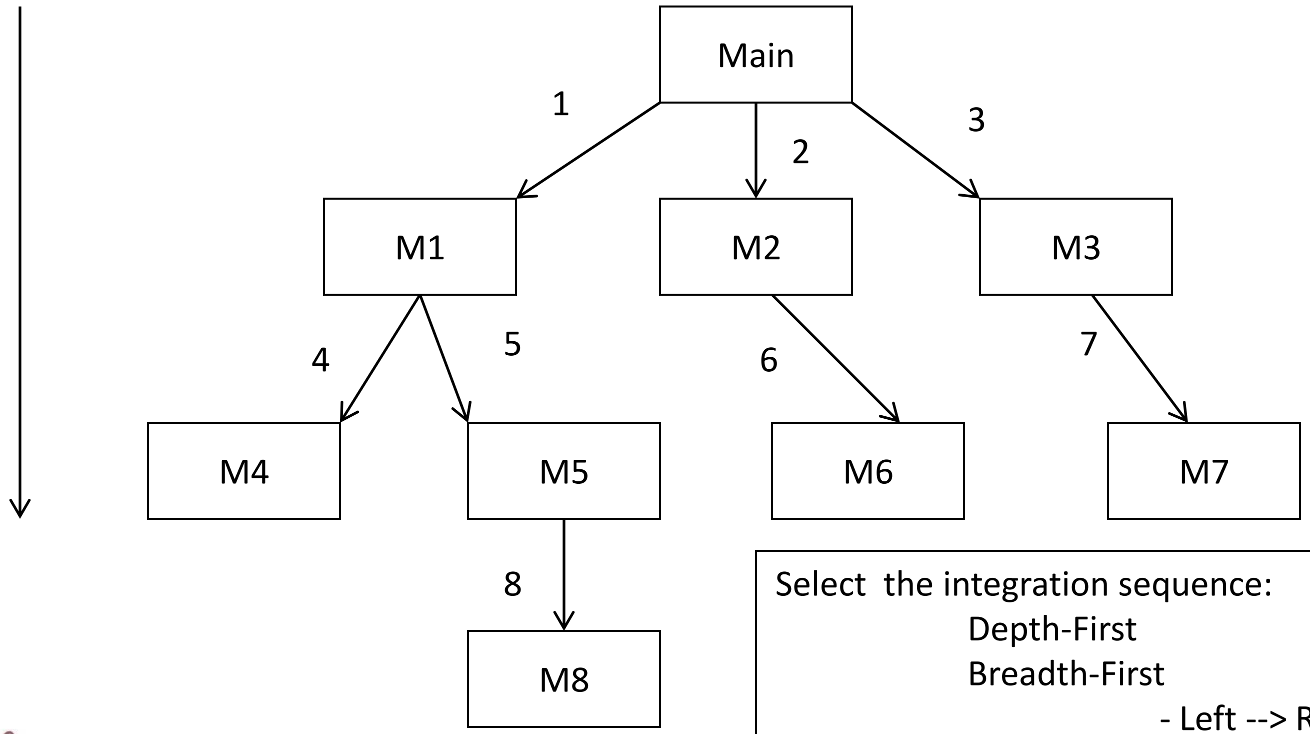
✓Subordinate stubs are replaced one at a time with actual modules.

✓Tests are conducted as each module is integrated.

✓ On completion of each set of tests, another stub is replaced with the real module.

✓ Regression testing may conducted.

Top-Down Integration



Select the integration sequence:

Depth-First

Breadth-First

- Left --> Right
- Right --> Left

Top-Down Integration

Integration Order: Breadth-First (Left Order)

IS: Integrated System

Mi ’: software stub for Module Mi.

Step #1: IS = Main + M1 (need: M2’, M3’, M4’ and M5’)
Step #2: IS = IS + M2 (need: M4’, M5’, M6’, and M3’)
Step #3: IS = IS + M3 (need: M4’, M5’, M6’, and M7’)
Step #4: IS = IS + M4 (need: M5’, M6’,and M7’)
Step #5: IS = IS + M5 (need: M8’, M6’, and M7’)
Step #6: IS = IS + M6 (need: M7’, and M8’)
Step #7: IS = IS + M7 (need: M8’)
Step #8: IS = IS + M8

Pros and Cons of Top-down Integration Approach

Pros

✓Test cases can be defined in terms of the functionality of the system (functional requirements)

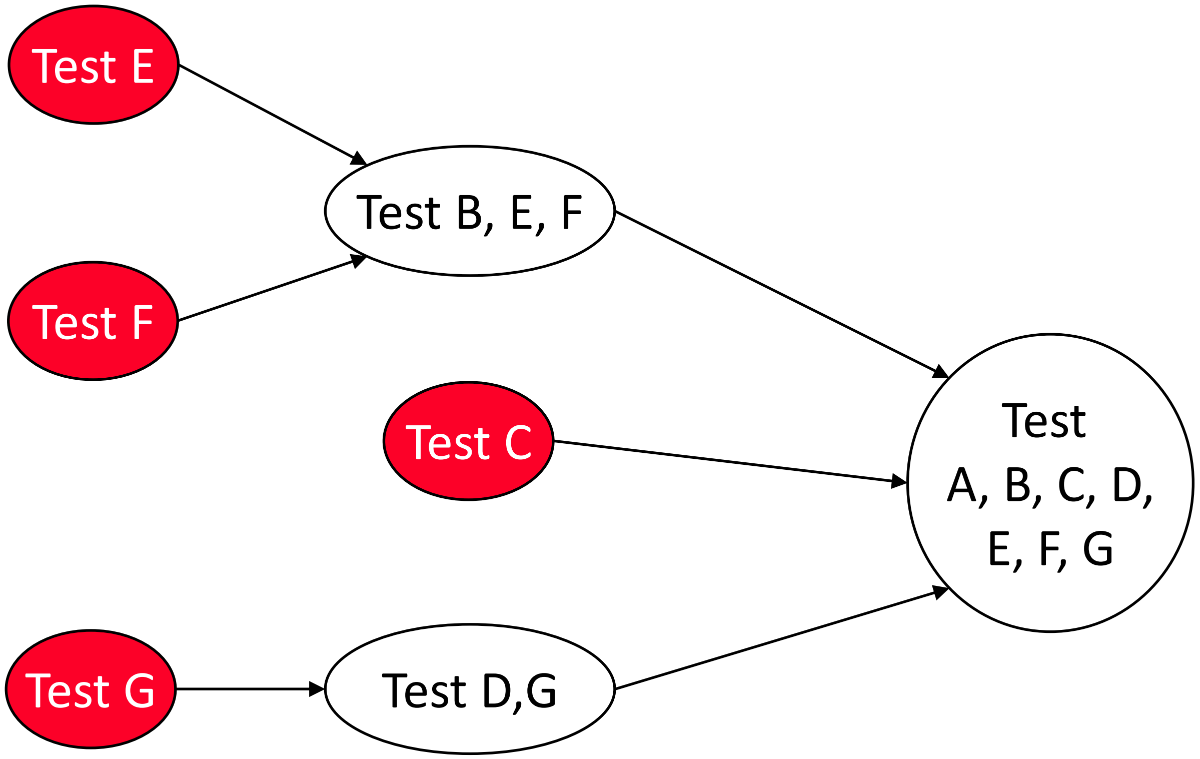
✓No drivers needed

Cons

✓Writing stubs is difficult: Stubs must allow all possible conditions to be tested.
✓Large number of stubs may be required, especially if the lowest level of the system contains many methods.
✓Some interfaces are not tested separately.

Bottom-Up Software Integration

Idea:- Modules at the lowest levels are integrated at first, then by moving upward through the control structure.



Bottom-Up Software Integration

Integration process (five steps):

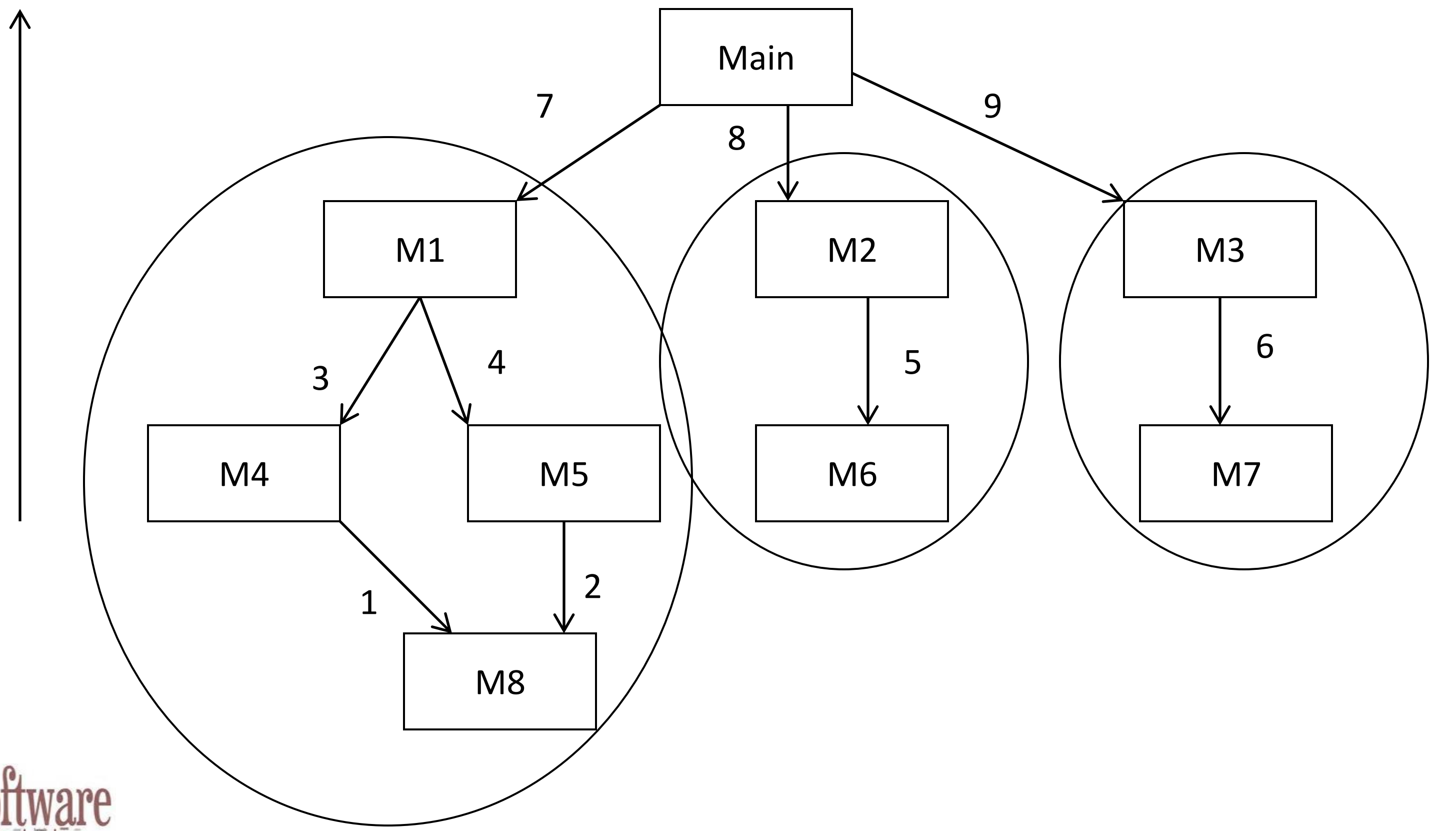
✓ Low-level modules are combined into clusters that perform a specific software sub-function.

✓ A driver is written to coordinate test case input and output.

✓Test cluster is tested.

✓ Drivers are removed and clusters are combined moving upward in the program structure.

Bottom-Up Software Integration



Bottom-Up Software Integration

Integration Order: Breadth-First (Left Order)

IS: Integrated System

Mi": software driver for Module Mi.

Step #1: IS1 = M8 + M4 (need: M5" and M1")
Step #2: IS1 = IS1 + M5 (need: M1")
Step #3+4: IS1 = IS1 + M1 (need: Main")
Step #5: IS2 = M2 + M6 (need: Main")
Step #6: IS3 = M3 + M7 (need: Main")
Step #7: IS = IS1 + Main (need: M2', M3')
Step #8: IS = IS + IS2 (Need: M3')
Step #9: IS = IS + IS3

Pros and Cons of Bottom-Up Integration Approach

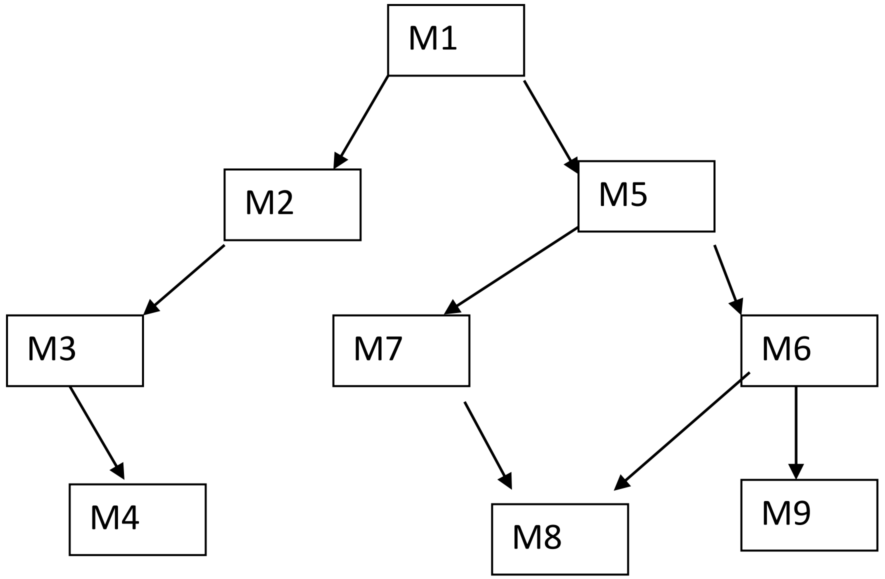
Cons:

- ✓Tests the most important subsystem (user interface) last
- ✓Drivers needed

Pros:

- ✓No stubs needed
- ✓Useful for integration testing of the following systems
- ✓Object-oriented systems
- ✓Real-time systems
- ✓Systems with strict performance requirements.

Integration Example



Please find the integration test order using the top-down approach.

Please find the integration sequence using the bottom-up approach.

Object-Oriented Software Integration

There are a number of proposed integration test strategies for object-oriented software.

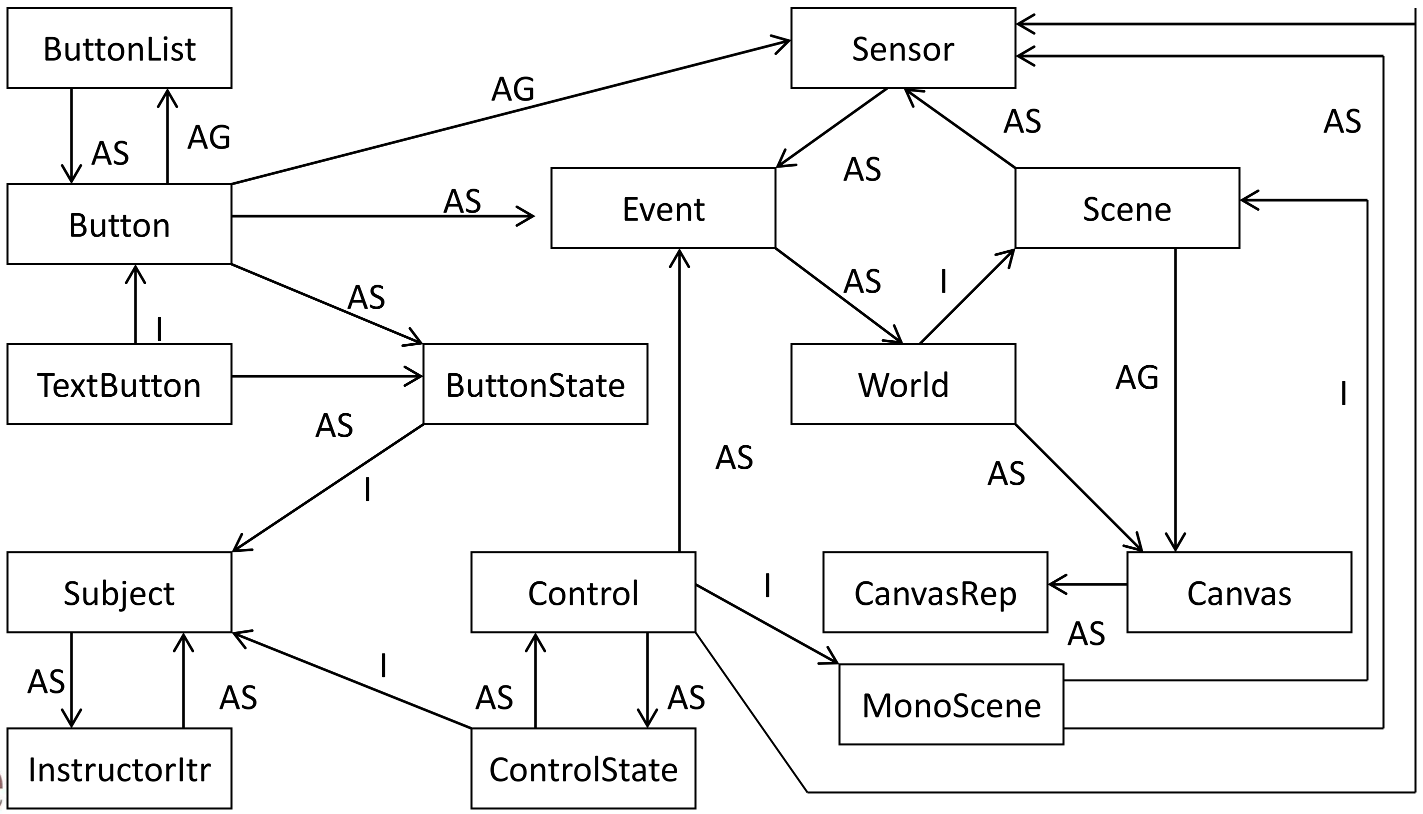
One of them is known as Class Test Order.

What is class test order?
- It is a class test sequence order for a class library or an OO program.

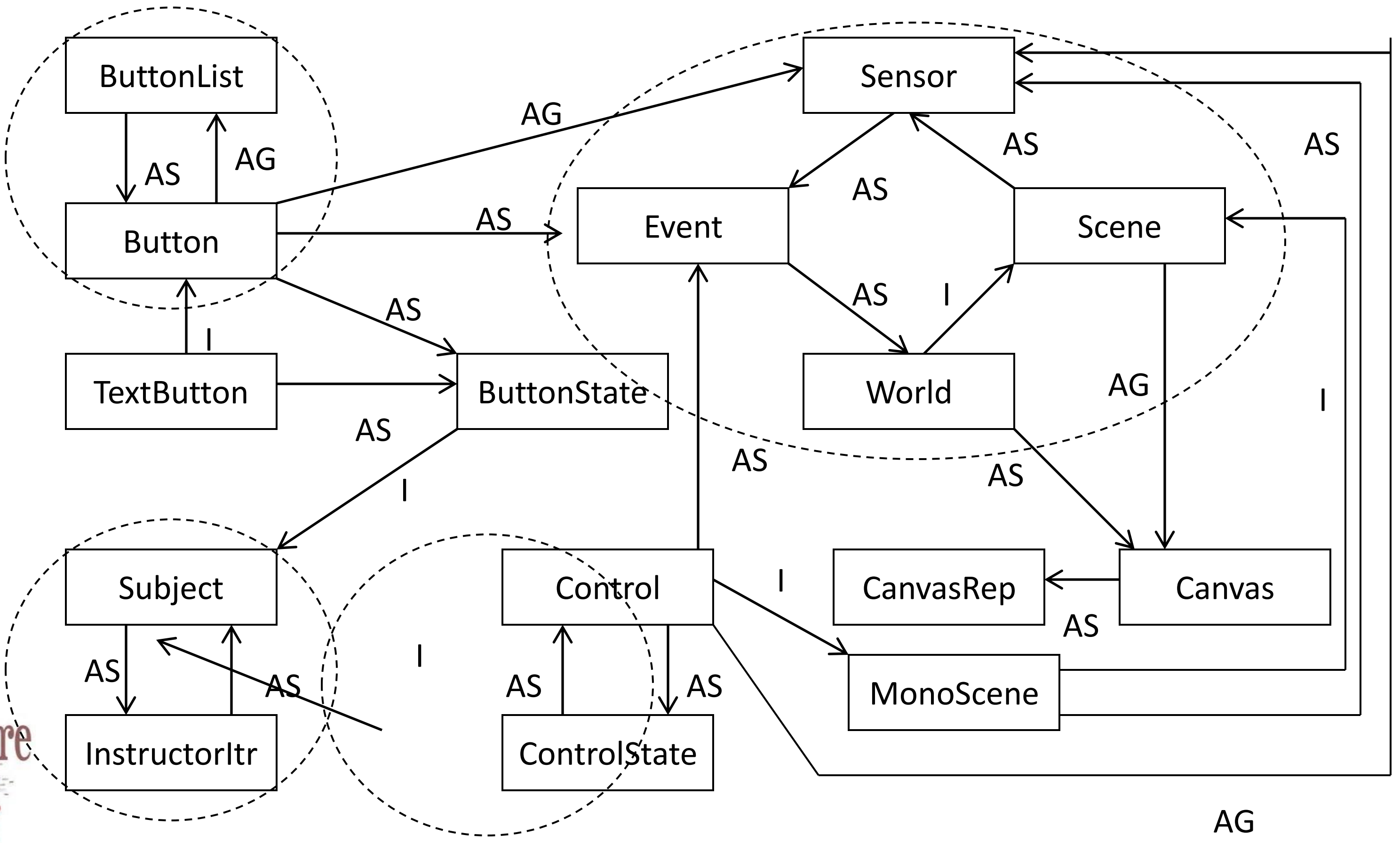
It uses a class relation diagram as its class integration test model.

This class test order provides a unit test sequence for classes in a class library

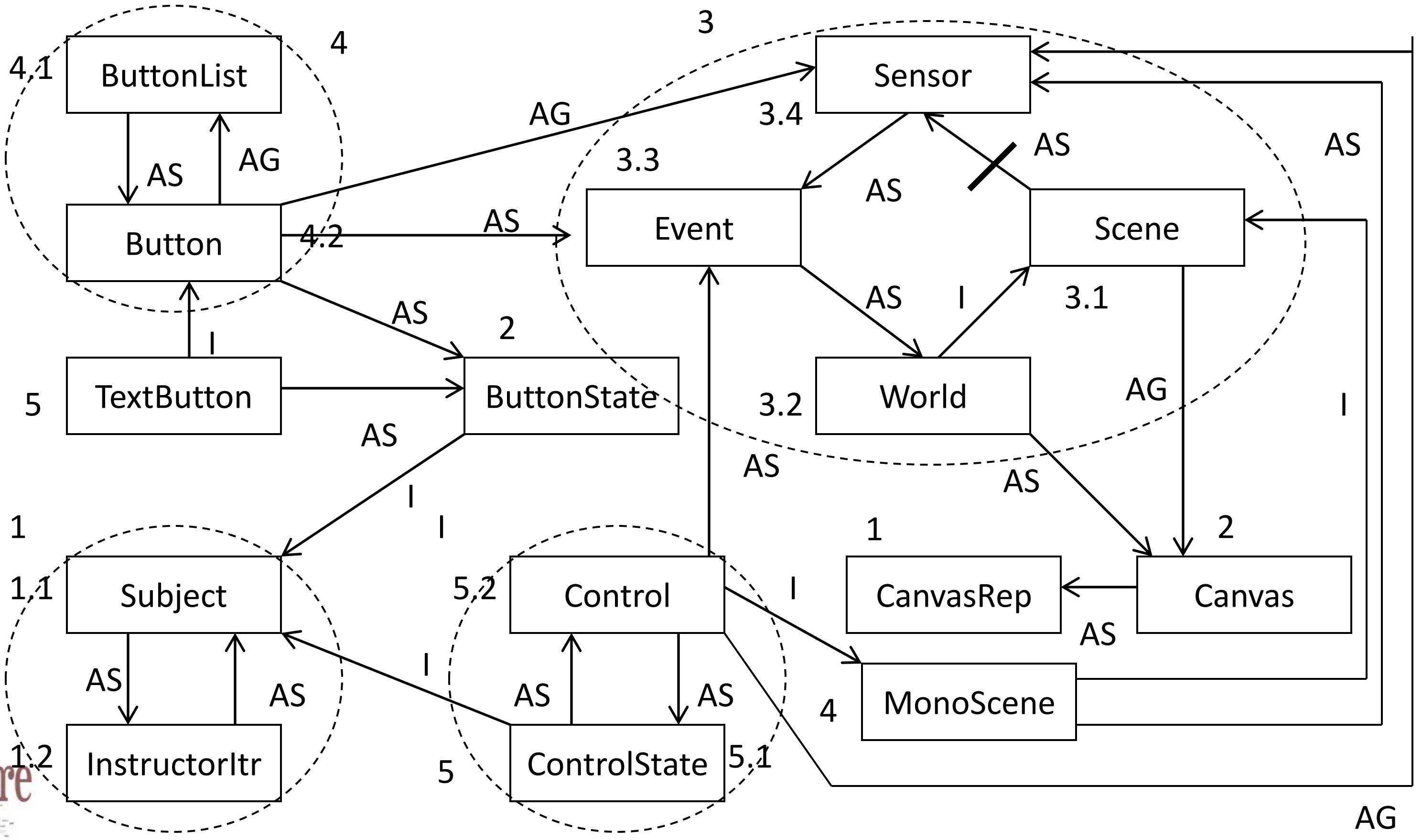
A Class Test Order for Object-Oriented Programs



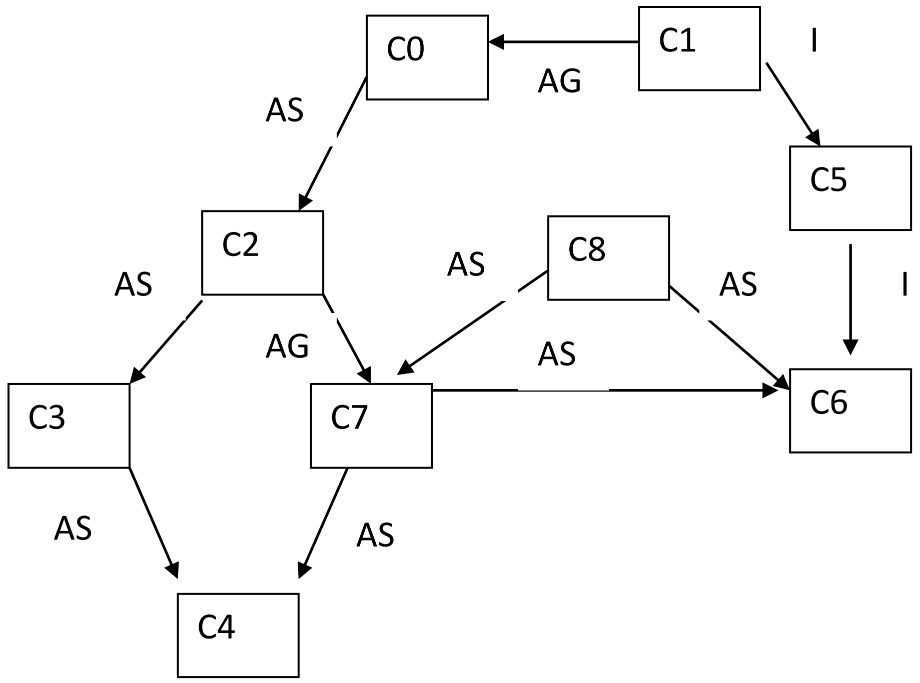
A Class Test Order for Object-Oriented Programs



A Class Test Order for Object-Oriented Programs



Please fine the class test order for the following


C0
C1
AG
I
AS
C5
C2
C8
AS
AS
AS
I
AG
AS
C3
C7
C6
AS
AS
C4