Software Integration Testing

Speaker: Jerry Gao Ph.D.

Computer Engineering Department
San Jose State University

email: jerry.gao@sjsu.edu
URL: http://www.engr.sjsu.edu/gaojerry

Presentation Outline

- What is Software Integration Testing?

- Non-Incremental Software Integration

- Incremental Software Integration

- Traditional software Integration Strategies

- Software Integration Test Harness
-Test Stubs and Test Drivers

- Object-Oriented Software Integration Strategies

What is Software Integration Testing?

What is Software Integration Testing?

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.

Major testing focuses:

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

Who perform software integration:

Developers and test engineers

What do you need?:

- ***Integration strategy***
- ***Integration test environment and test suite***
- ***Module (or component) specifications***
- ***Interface and design documents***

Software Integration Strategy

What is a 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.

Typical test models:

- ***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:

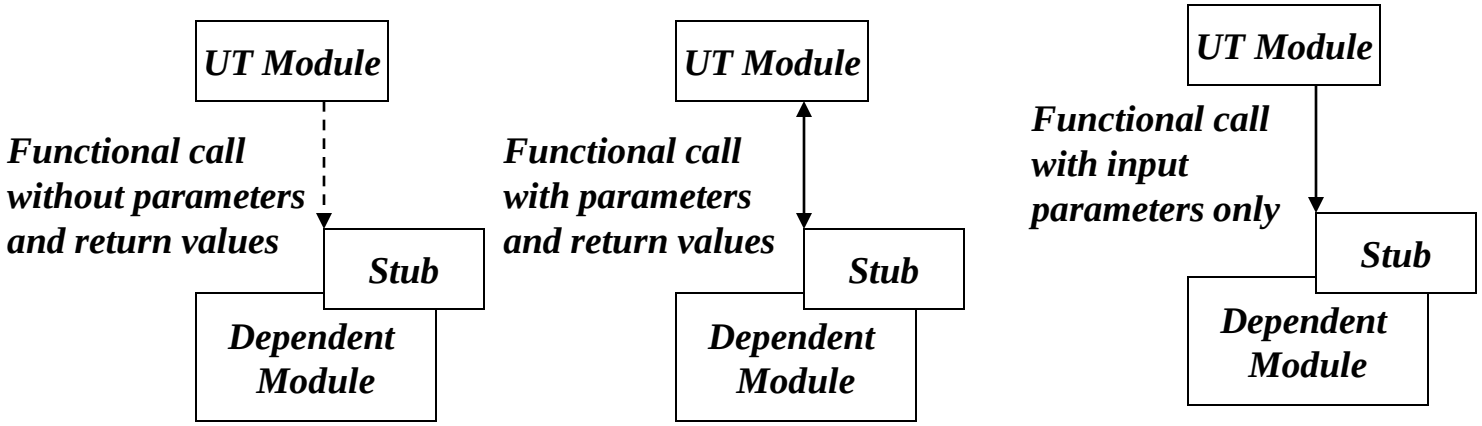
--> *Top- down software integration*
--> *Bottom-up software integration*
--> *Sandwich integration*

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:

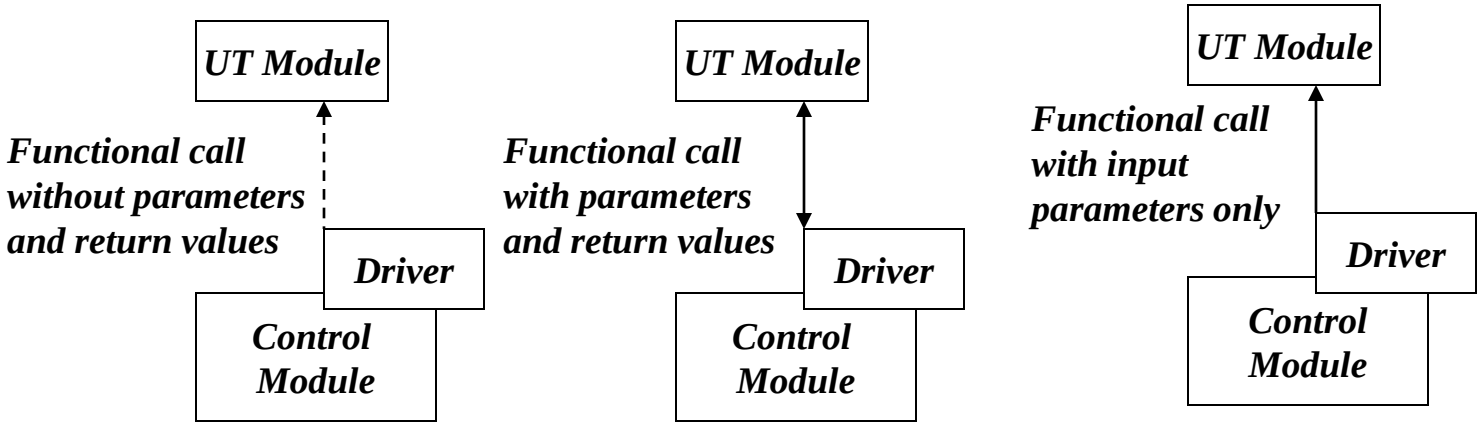


Test Stubs and Test Drivers

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:



Traditional Software Integration Strategy

Non-incremental integration:

- Big Band - combine (or integrate) all parts at once.

Advantages: ***simple***

Disadvantages: ***- hard to debugging, not easy to isolate errors***
- not easy to validate test results
- impossible to form an integrated system

Top-down Integration

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.

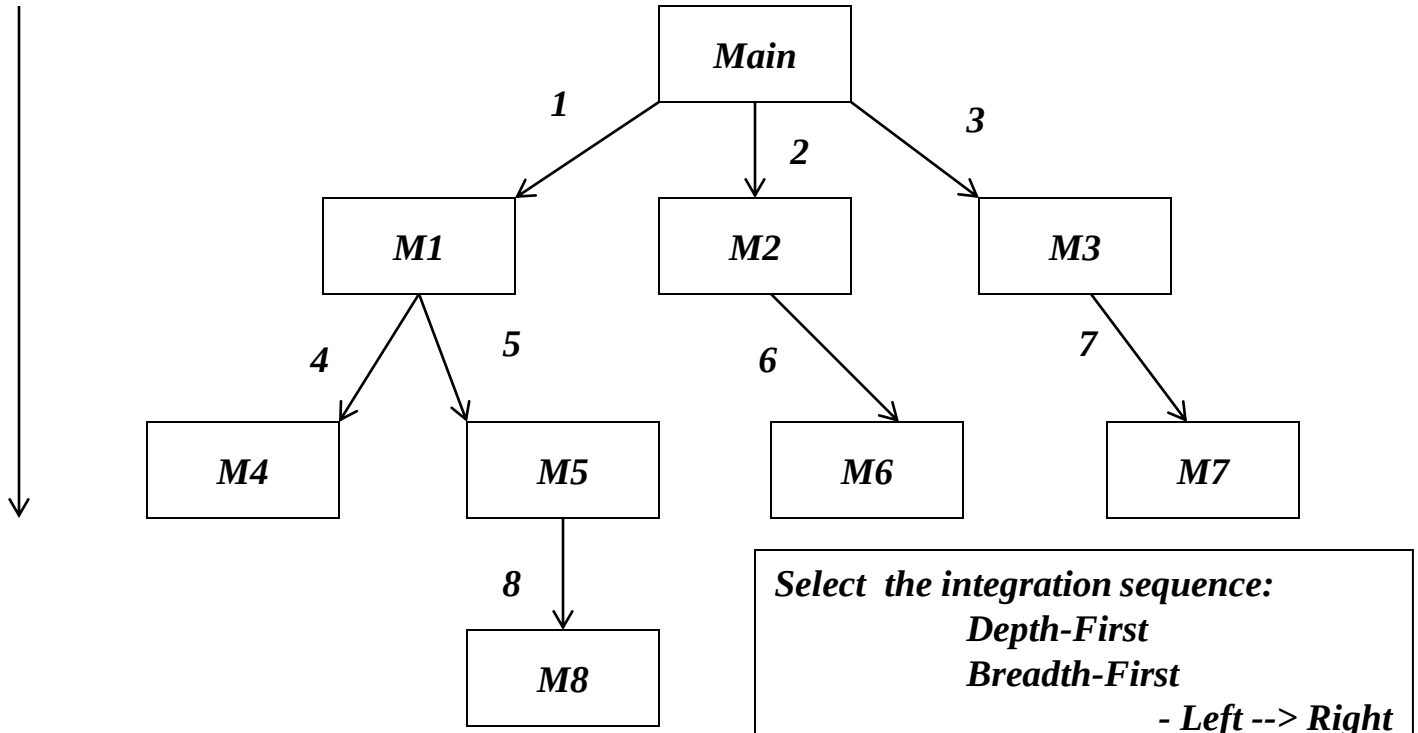
Integration process (five steps):

1. *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.*
2. *subordinate stubs are replaced one at a time with actual modules.*
3. *tests are conducted as each module is integrated.*
4. *On completion of each set of tests, another stub is replaced with the real module.*
5. *regression testing may conducted.*

Pros and cons top-down integration:

- *stub construction cost*
- *major control function can be tested early.*

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***

Bottom-Up Software Integration

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

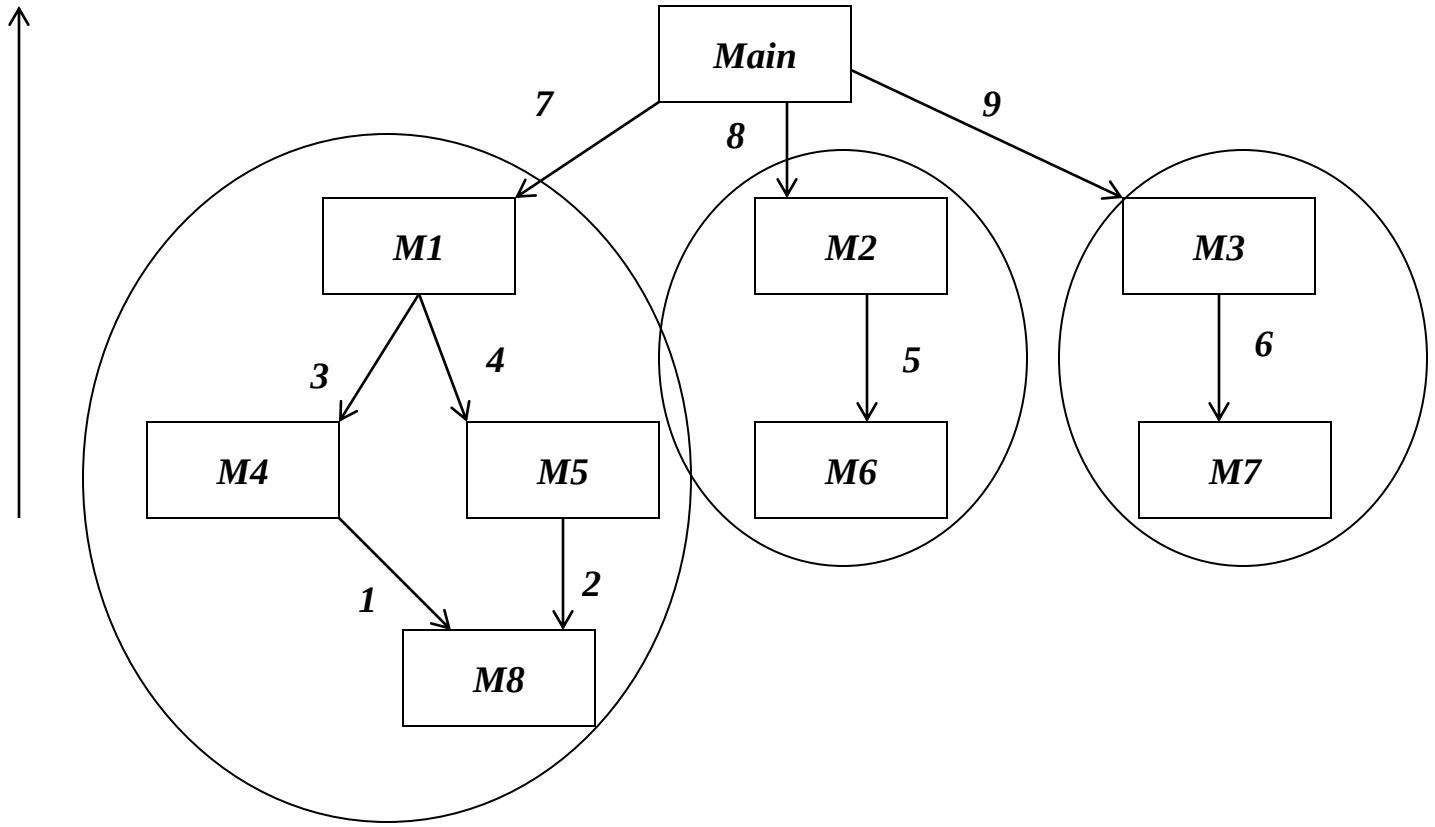
Integration process (five steps):

1. *Low-level modules are combined into clusters that perform a specific software sub-function.*
2. *A driver is written to coordinate test case input and output.*
3. *Test cluster is tested.*
4. *Drivers are removed and clusters are combined moving upward in the program structure.*

Pros and cons of bottom-up integration:

- *no stubs cost*
- *need test drivers*
- *no controllable system until the last step*

Bottom-Up Integration



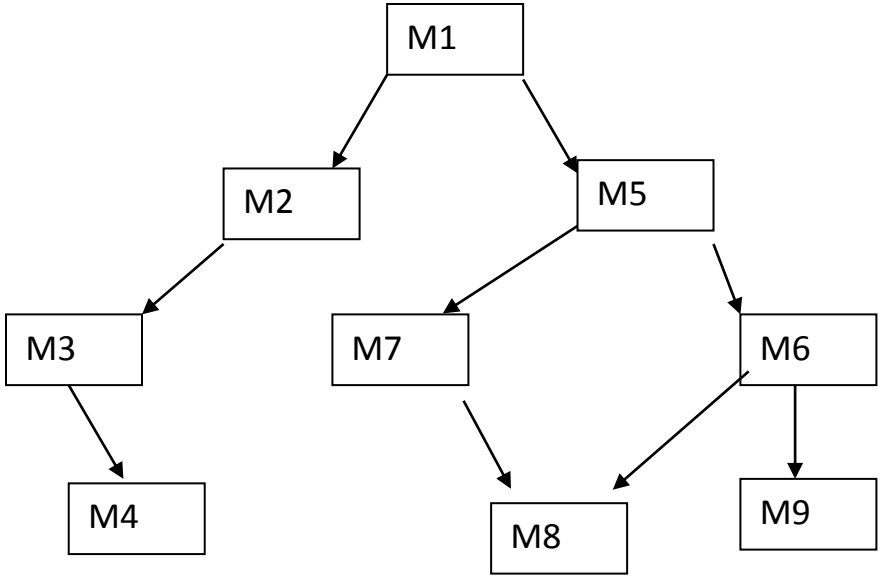
Bottom-Up Integration

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***

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.

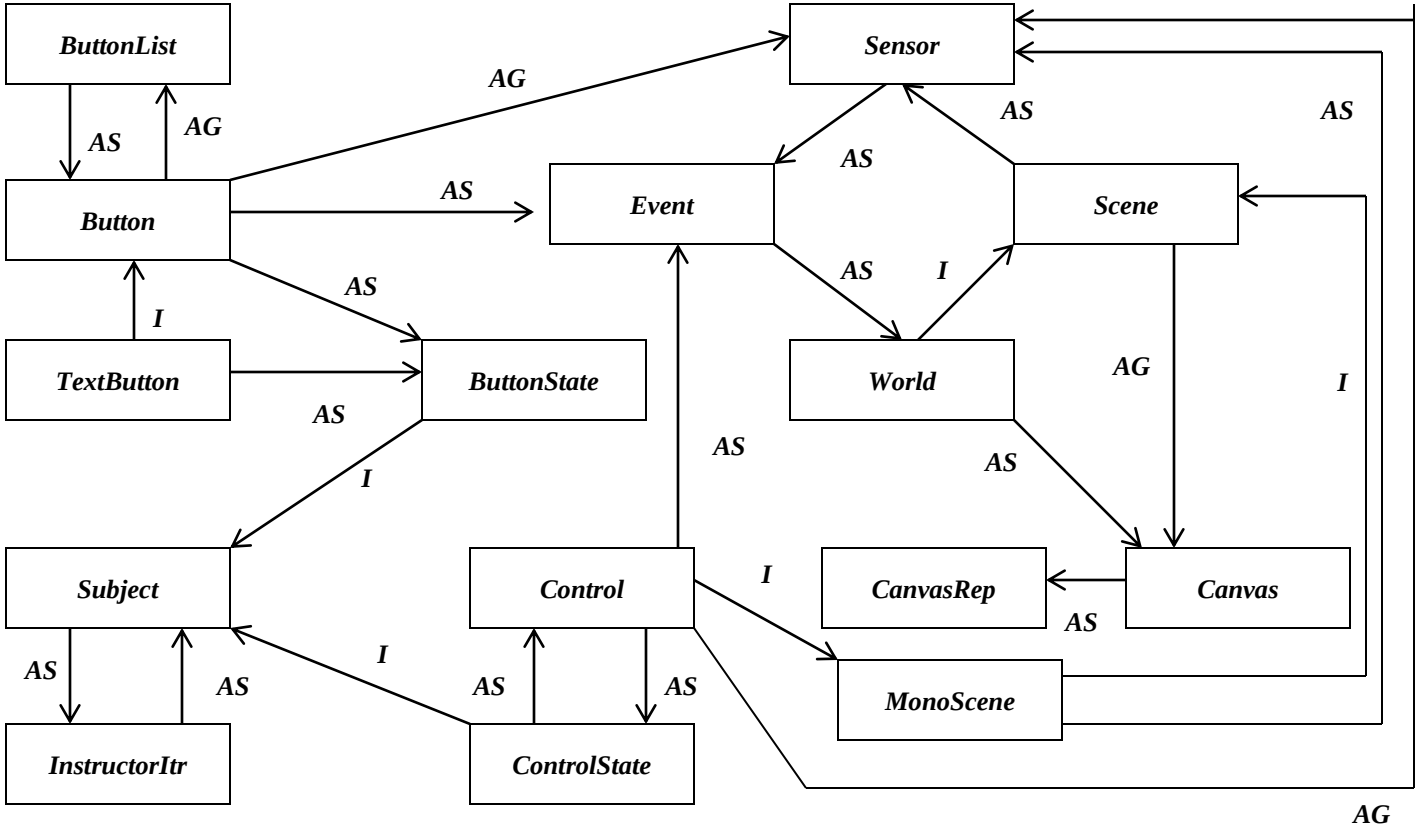
One of them is known as Class Test Order.

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

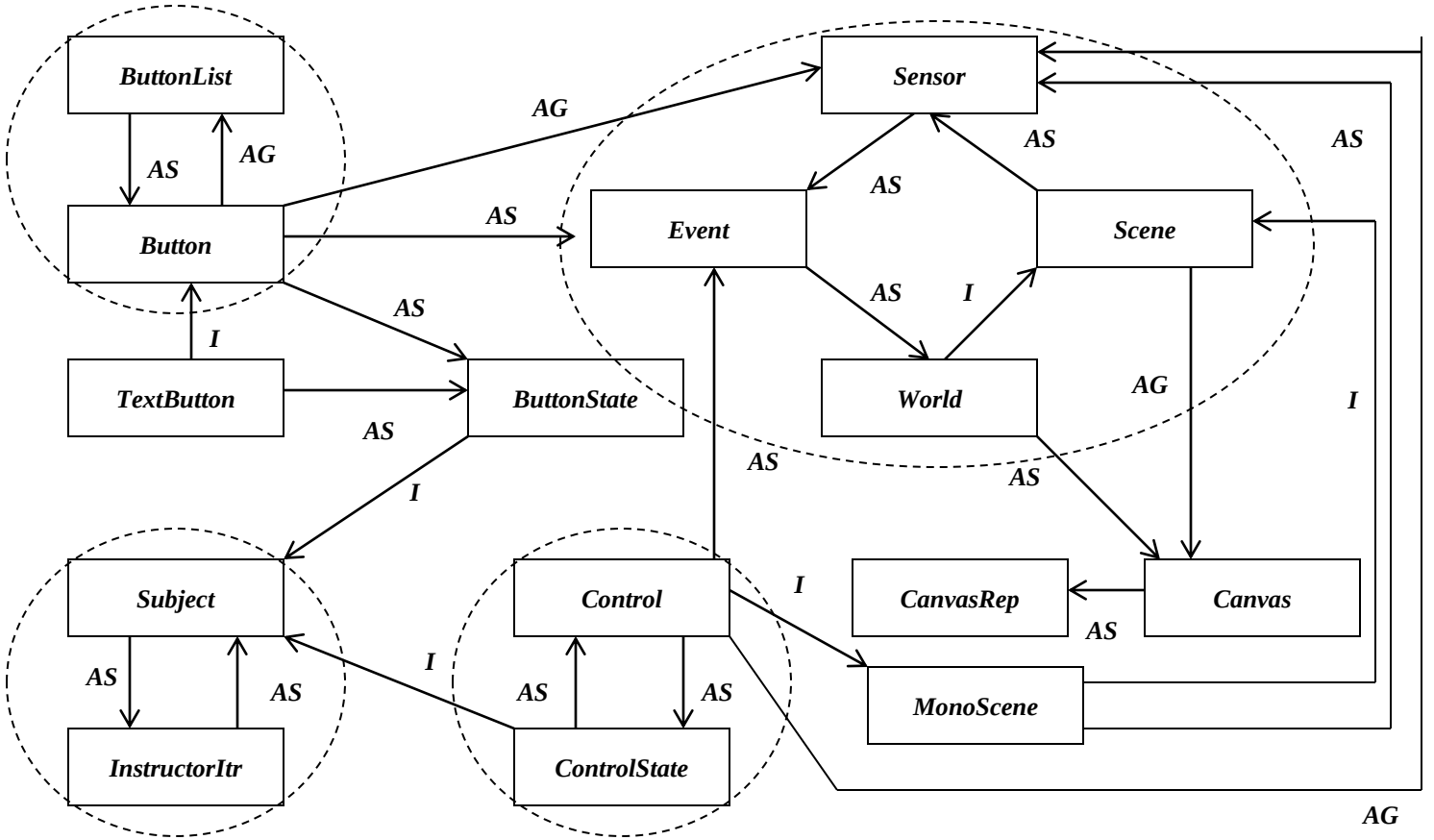
It uses a class relation diagram as its class integration test model.

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

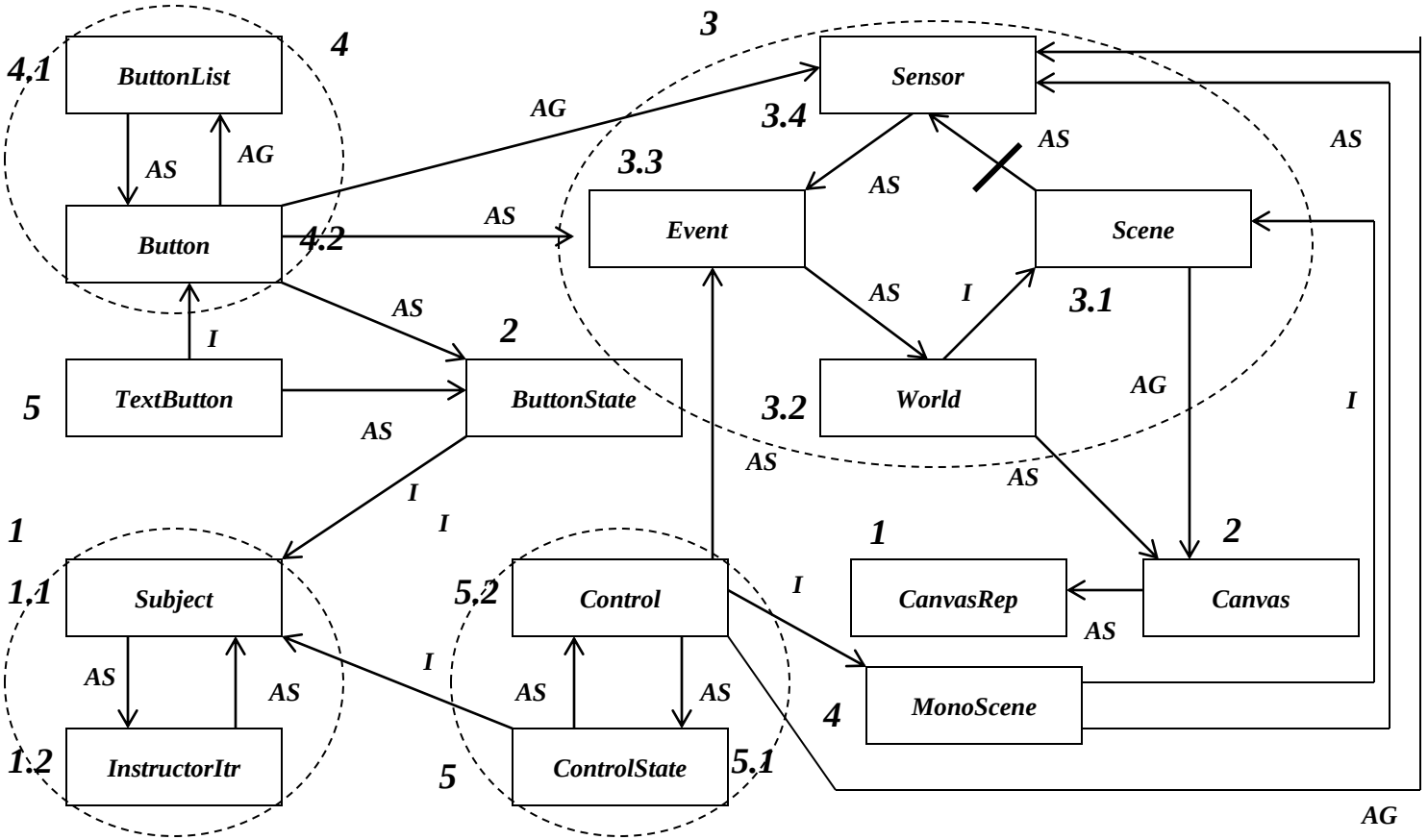
A Class Test Order for Object-Oriented Programs



A Class Test Order for Object-Oriented Programs



A Class Test Order for Object-Oriented Programs



Please find the class test order for the following class relation graph: