Section 6.3
Integration Testing

1. Overview
2. Horizontal integration
3. Vertical integration

6.3.1 Overview

- Focus of integration testing
  - small groups of components
    - components already unit tested
- Approach
  - unit testing verifies individual components
  - tested components can be grouped and tested together
  - more complex parts can be tested incrementally

Overview (cont.)

- Ordering of component testing can optimize the process
  - horizontal integration testing strategies
    - components are integrated according to layers
  - vertical integration testing strategies
    - components are integrated according to functionality

6.3.2 Horizontal Integration

- Horizontal integration testing strategies
  - big bang testing
  - bottom-up testing
  - top-down testing
  - sandwich testing
  - modified sandwich testing

Big Bang Testing

- Approach
  - unit test each component
  - throw everything together
- Disadvantage
  - difficult to determine:
    - where the faults occur
    - which components fail

Bottom-Up Testing

- Approach
  - unit test all the bottom layer components
  - integrate with the next layer up
  - repeat
- Characteristics of bottom-up testing
  - requires that test drivers be implemented
  - requires no test stubs

Bottom-Up Testing (cont.)

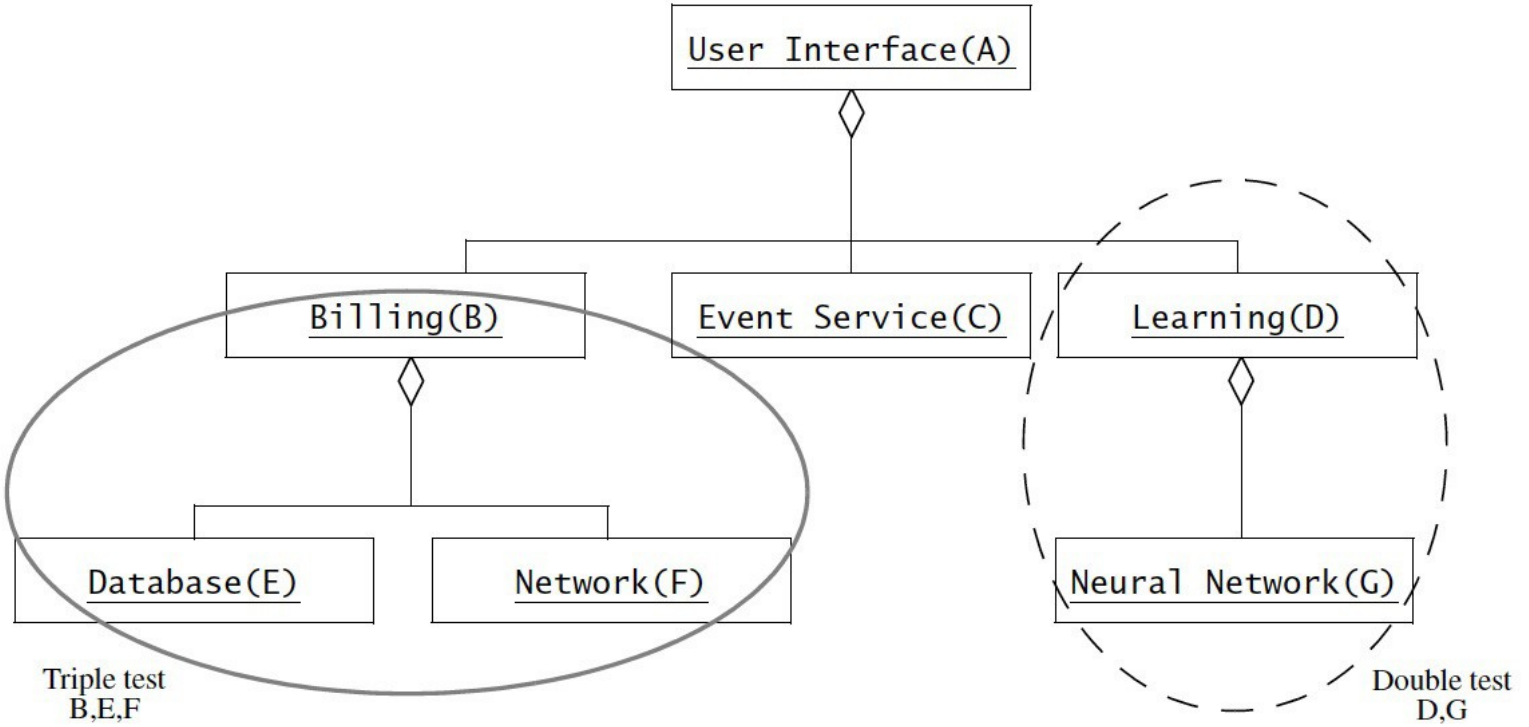

Figure 11-19 Bottom-up test strategy. After unit testing subsystems E, F, and G, the bottom up integration test proceeds with the triple test B-E-F and the double test D-G.

Top-Down Testing

- Approach
  - unit test all the top layer components
  - integrate with the next layer down
  - repeat
- Characteristics of top-down testing
  - requires that test stubs be implemented
  - requires no test drivers

Top-Down Testing (cont.)

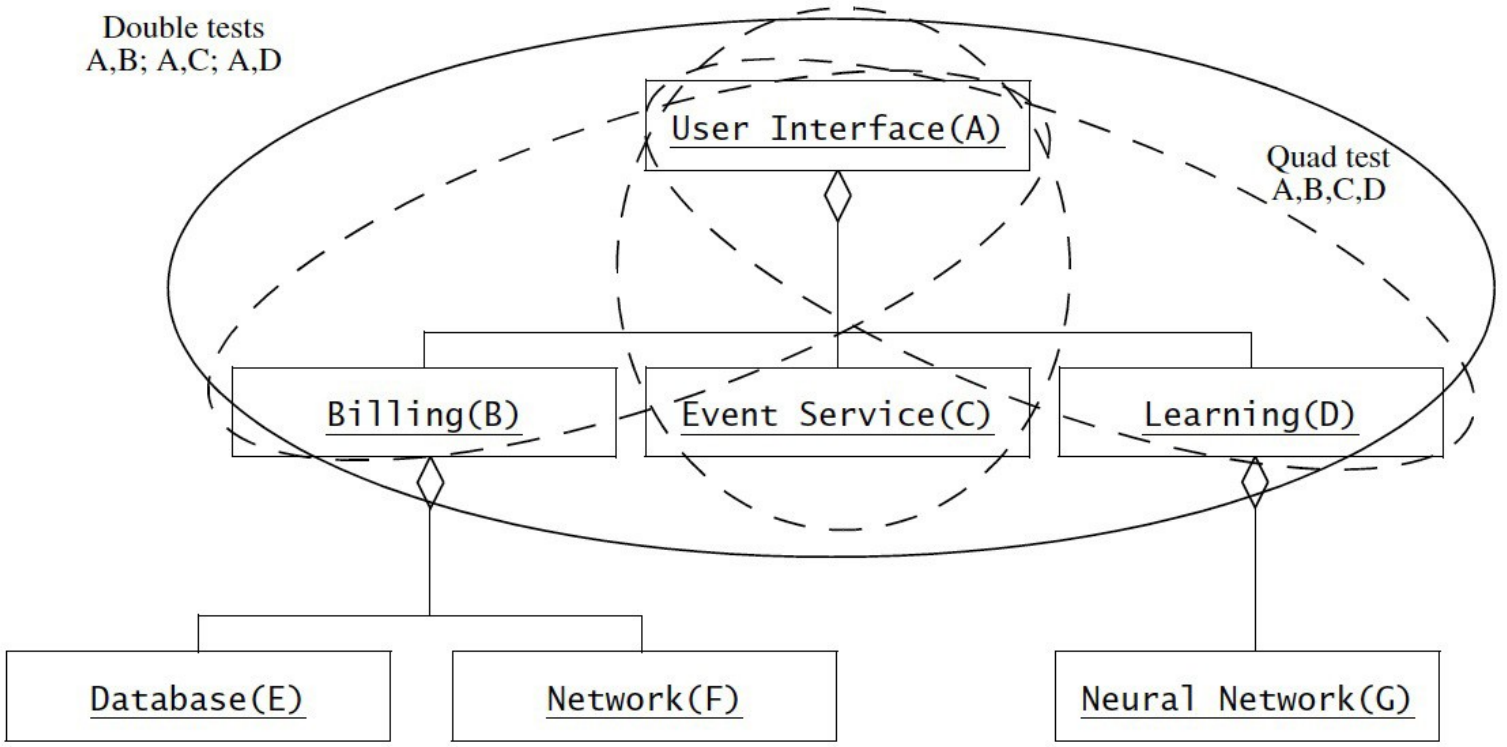

Figure 11-20 Top-down test strategy. After unit testing subsystem A, the integration test proceeds with the double tests A-B, A-C, and A-D, followed by the quad test A-B-C-D.

Bottom-Up vs. Top-Down Testing

- Bottom-up testing
  - advantage: finds interface faults more easily
  - disadvantage: UI subsystems are tested last
- Top-down testing
  - advantage: starts with UI testing
  - disadvantage: large number of test stubs must be developed

Sandwich Testing

- Approach
  - combines top-down and bottom-up
  - system is divided into:
    - target layer
    - layer above target layer
    - layer below target layer

Sandwich Testing (cont.)

- Approach (cont.)
  - top-down and bottom-up testing conducted in parallel
  - top-down:
    - top layer is tested incrementally with target components
  - bottom-up:
    - bottom layer is tested incrementally with target components
- Characteristics
  - requires no test drivers and no test stubs
  - disadvantage: target components are not unit tested

Sandwich Testing (cont.)

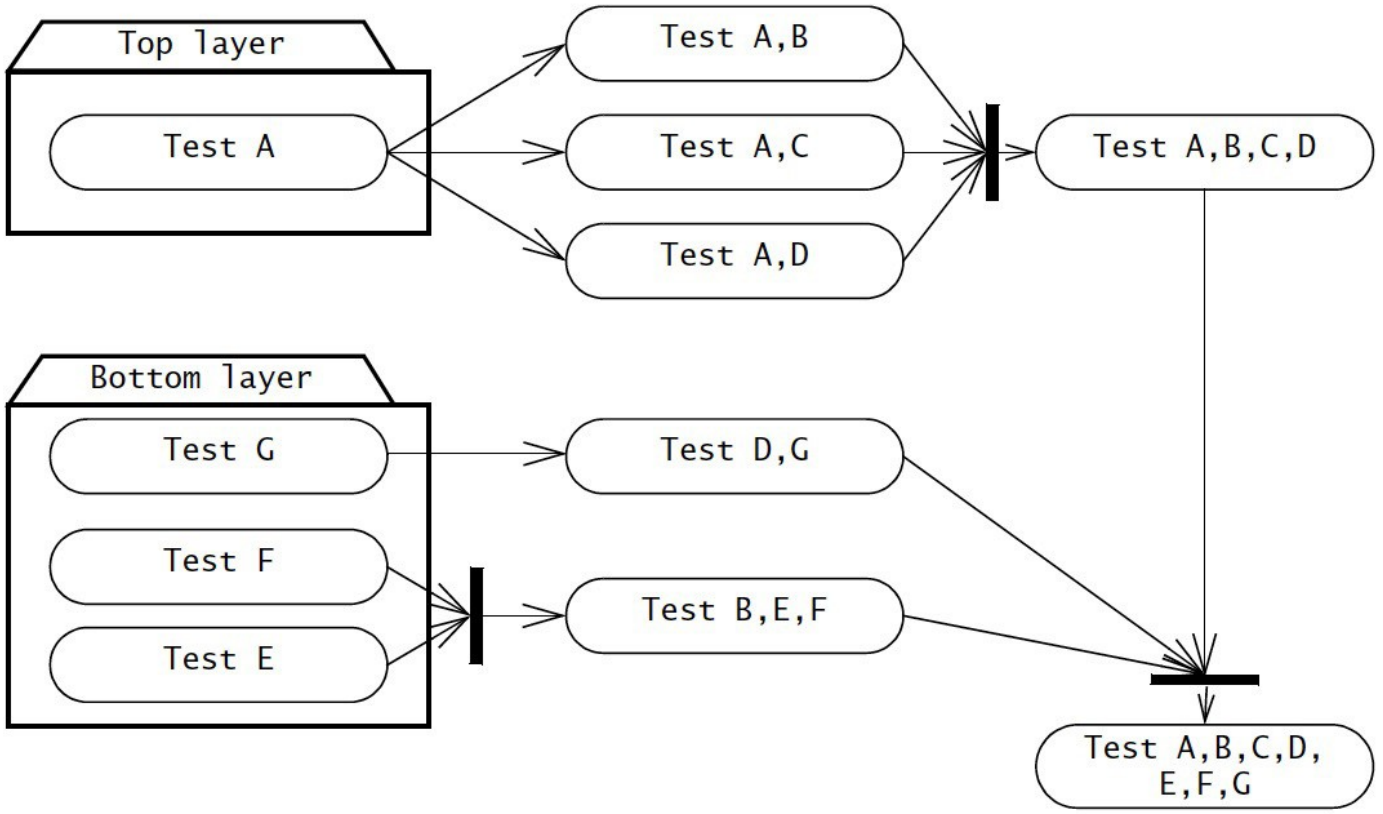

Figure 11-21 Sandwich testing strategy (UML activity diagram). None of the components in the target layer (i.e., B, C, D) are unit tested.

Modified Sandwich Testing

- Approach
  - similar to sandwich testing
  - test the three layers individually before testing together
  - first step: individual layer tests
    - top layer tested, with stubs for target layer
    - target layer tested, with drivers for top layer and stubs for bottom
    - bottom layer tested, with drivers for target layer

Modified Sandwich Testing (cont.)

- Approach (cont.)
  - second step: combined layer tests
    - top layer accesses target layer
      - replace the test drivers with top layer components
    - bottom layer is accessed by target layer
      - replace the test stubs with bottom layer components
- Characteristics
  - advantage: parallelism
  - disadvantage: requires additional test drivers and stubs

Modified Sandwich Testing (cont.)

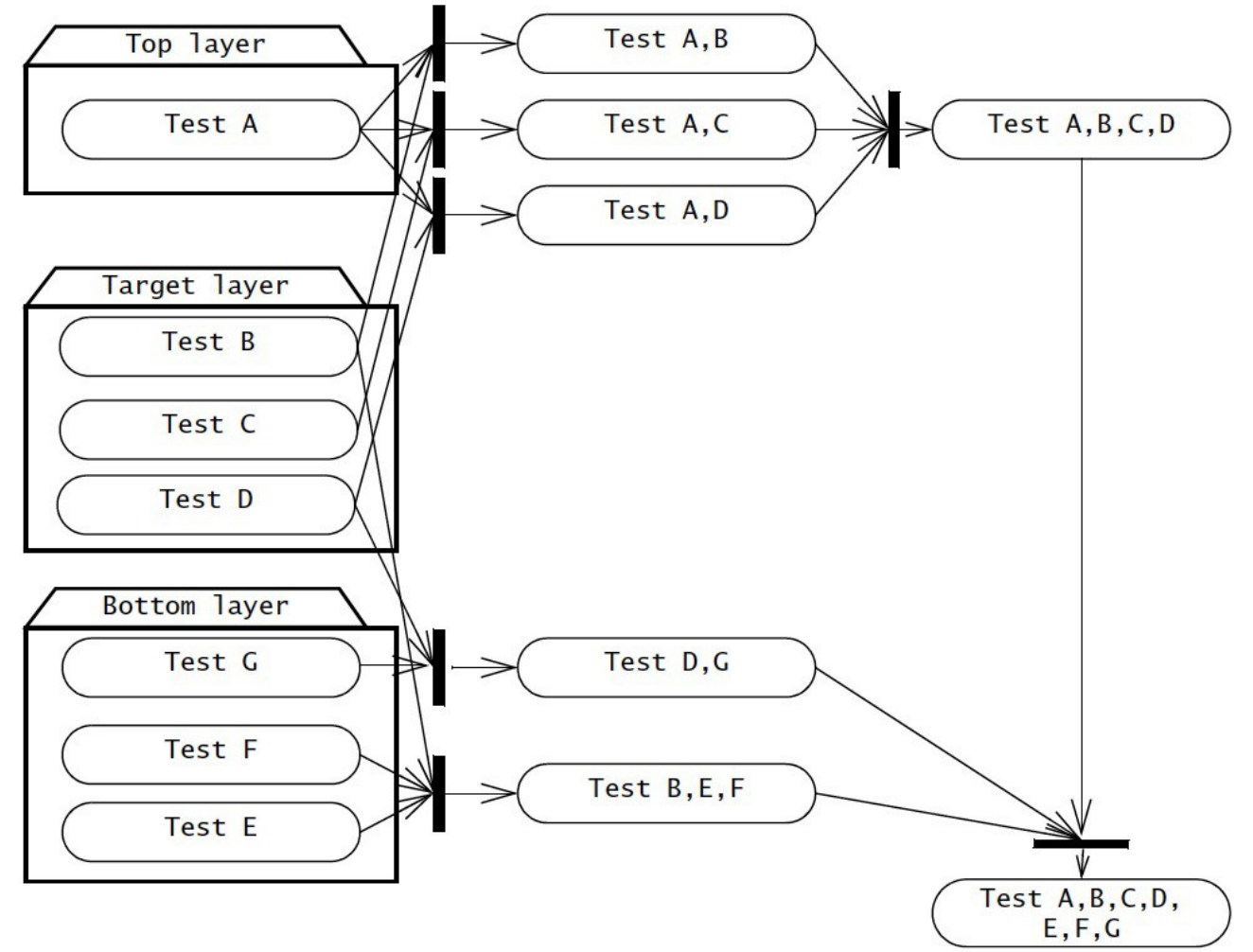

Figure 11-22 An example of modified sandwich testing strategy (UML activity diagrams). The components of the target layer are unit tested before they are integrated with the top and bottom layers.

6.3.3 Vertical Integration

- Vertical vs. horizontal integration testing strategies
  - in horizontal integration:
    - components are integrated in layers, based on subsystem decomposition
  - in vertical integration:
    - all components for a given use case are fully implemented
    - these components are tested together
    - similar to prototyping, but prototypes are not releasable
    - disadvantages:
      - system evolves more incrementally
      - design is more subject to change