Section 6.4
System Testing

1. Overview
2. Functional testing
3. Performance testing
4. Acceptance testing

6.4.1 Overview

- Focus of system testing
  - the complete system
- System testing ensures that the system complies with:
  - the functional requirements
  - the non-functional requirements

Overview (cont.)

- System testing activities include:
  - functional testing
  - performance testing
  - field testing
  - acceptance testing
  - installation testing

6.4.2 Functional Testing

- Characteristics of functional testing
  - it's also known as *requirements testing*
  - it's a blackbox technique
  - test cases are derived from the Requirements Analysis Document
- Goal is to find differences between:
  - the system and the functional requirements
  - the use case model and the observed system behaviour

Functional Testing (cont.)

- Strategy
  - inspect the use case model
  - identify scenarios that are likely to cause failures
  - exercise common and exceptional use cases

Functional Testing (cont.)

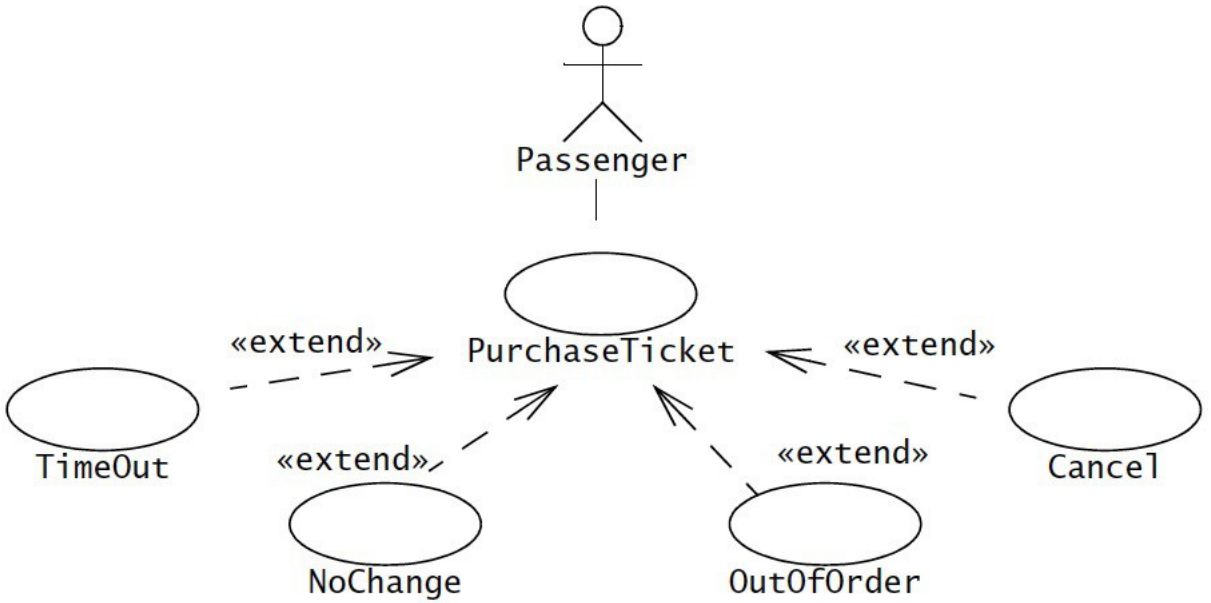

Figure 11-23 An example of use case model for a subway ticket distributor (UML use case diagram).

Functional Testing (cont.)

Use case name	`PurchaseTicket`
Entry condition	The `Passenger` is standing in front of ticket `Distributor`. The `Passenger` has sufficient money to purchase ticket.
Flow of events	1. The `Passenger` selects the number of zones to be traveled. If the `Passenger` presses multiple zone buttons, only the last button pressed is considered by the `Distributor`. 2. The `Distributor` displays the amount due. 3. The `Passenger` inserts money. 4. If the `Passenger` selects a new zone before inserting sufficient money, the `Distributor` returns all the coins and bills inserted by the `Passenger`. 5. If the `Passenger` inserted more money than the amount due, the `Distributor` returns excess change. 6. The `Distributor` issues ticket. 7. The `Passenger` picks up the change and the ticket.
Exit condition	The `Passenger` has the selected ticket.

Figure 11-24 An example of use case from the ticket distributor use case model `PurchaseTicket`.

Functional Testing (cont.)

Test case name	`PurchaseTicket_CommonCase`
Entry condition	The `Passenger` standing in front of ticket `Distributor`. The `Passenger` has two $5 bills and three dimes.
Flow of events	1. The `Passenger` presses in succession the zone buttons 2, 4, 1, and 2. 2. The `Distributor` should display in succession $1.25, $2.25, $0.75, and $1.25. 3. The `Passenger` inserts a $5 bill. 4. The `Distributor` returns three $1 bills and three quarters and issues a 2-zone ticket. 5. The `Passenger` repeats steps 1–4 using his second $5 bill. 6. The `Passenger` repeats steps 1–3 using four quarters and three dimes. The `Distributor` issues a 2-zone ticket and returns a nickel. 7. The `Passenger` selects zone 1 and inserts a dollar bill. The `Distributor` issues a 1-zone ticket and returns a quarter. 8. The `Passenger` selects zone 4 and inserts two $1 bills and a quarter. The `Distributor` issues a 4-zone ticket. 9. The `Passenger` selects zone 4. The `Distributor` displays $2.25. The `Passenger` inserts a $1 bill and a nickel, and selects zone 2. The `Distributor` returns the $1 bill and the nickel and displays $1.25.
Exit condition	The `Passenger` has three 2-zone tickets, one 1-zone ticket, and one 4-zone ticket.

Figure 11-25 An example of test case derived from the `PurchaseTicket` use case.

6.4.3 Performance Testing

- Goal is to find differences between:
  - the system and the design goals selected during design
- Test cases are derived from:
  - the System Design Document
  - the Requirements Analysis Document

Performance Testing (cont.)

- Performance testing includes:
  - stress testing
    - checks if the system can respond to multiple simultaneous requests
  - volume testing
    - finds faults associated with large amounts of data
  - security testing
    - finds security faults in the system
    - one approach: white hat hackers try to break in

Performance Testing (cont.)

- Performance testing includes (cont.):
  - timing testing
    - find system behaviours that violate timing constraints
  - recovery testing
    - evaluate the ability of the system to recover from error states
- System is deemed *validated* if:
  - functional and performance testing produce no failures

6.4.4 Acceptance Testing

- Final phases of testing before client accepts the system
  - field testing
  - acceptance testing
  - installation testing

Acceptance Testing (cont.)

- Field testing
  - goal
    - install the system for a selected group of users
    - collect feedback from those users
  - types of field testing
    - alpha test
      - field test with the system in the development environment
    - beta test
      - field test with the system in the target environment
  - unlike usability tests, user behaviour is not observed

Acceptance Testing (cont.)

- Acceptance testing
  - client evaluates the system
  - three possible tests:
    - benchmark testing
      - selected users evaluate the system against the requirements
    - competitor testing
      - the system is tested against another product
    - shadow testing
      - new and legacy systems are executed in parallel

Acceptance Testing (cont.)

- Installation testing
  - the system is installed in the target environment
  - functional and performance tests are repeated

Testing Recap

- What we learned:
  - understand the main categories of testing
  - unit testing:
    - understand creation of test cases for blackbox and whitebox testing
  - integration testing:
    - select testing integration strategy
    - understand creation of test cases using stubs and drivers
  - system testing:
    - understand creation of test cases based on the functional model