Chapter 6: Testing

Objectives
- After studying this chapter you should understand the following:
  - functional testing and unit testing, their similarities and differences;
  - implementation driven testing vs. test driven implementation;
  - the role of a test plan.
- Also, you should be able to:
  - implement a test plan with a test class.

Functional Testing
- Goal of functional testing: determine system meets customer’s specifications.
- **black box testing**:
  - Test designer ignores internal structure of implementation.
  - Test driven by expected external behavior of the system
  - System is treated as a “black box”: behavior can be observed, but internal structure is unknown.

Test design
- Test design generally begins with an analysis of:
  - Functional specifications of system, and
  - Use cases: ways in which the system will be used.

Test plan and test cases
- A test case is defined by:
  - Statement of case objectives;
  - Data set for the case;
  - Expected results.
- A test plan is a set of test cases.

Unit Testing
- Unit testing: incremental test of classes as they are developed.
- Unit testing and implementation are complementary activities, done concurrently.
- Unit testing is part of system implementation job.
Unit testing
- Requires development of a test plan.
- Test plan is expressed in the testing code itself.

Unit testing
- Implementation driven testing: tests for a module are developed based upon its implementation.
- Knowledge of implementation is used to select and refine test cases.
- Example, testing code that contains an if-then-else statement:
  - Write tests for both branches of if-then-else.

Unit testing
- Test driven implementation: tests for a feature are written before implementing the feature.
- Tests are based on specifications.
- Tests provide a concrete goal for implementation: write implementation that satisfies tests.

Developing test plan
- Analyze feature to identify test cases.
- Consider set of possible states object can assume.
- What are the possible states (situations) related to the CombinationLock2 command isOpen()?

Developing test plan
- Tests must be representative.
- Partition data into equivalency groups.
  - Values are considered to be in the same group if they are not likely to test differently.
  - Test cases are chosen from each equivalency group.
- Consider boundary cases.

Testing digit-by-digit combination lock
- Simplify example with a two digit combination lock.
- Test cases:
  - cases in which lock is open;
  - cases in which the lock is closed;
  - cases in which the first digit of the combination has been entered;
  - cases in which no digit has been entered;
  - cases with various combinations.
Testing digit-by-digit combination lock

- Equivalency groups:
  - combinations specified with a single digit integer (the first digit is implicitly 0);
  - combinations in which the two digits are different,
  - combinations in which the two digits are the same
  - largest (99) combination
  - smallest (00) combination.
- Test locks combinations:
  0 (i.e., 00), 1 (i.e., 01), 12, and 99.

Test initial state: Test the lock is initially open.

Test opening the lock:
  - The lock is already opened: entering an incorrect combination will not cause an open lock to close.

Test closing the lock:
  - Test that an open lock is closed and reset after executing close.
  - Test that a reset lock remains reset after executing close.
  - Test that an almost open lock is reset after executing close.

Number of tests:
- Four combination values,
- Two values for “is open,”
- Two values for “first digit has been entered,”
- 16 interesting states to test.
Summary

- Functional testing: tests system to verify that it meets the customer’s specifications.
- Functional tests are designed from the functional specifications of the system.
- They are generally “black box tests”: test external system behavior while treating implementation structure as unknown.
- Functional testing, is often the responsibility of a separate team different from development team.

Summary

- Unit tests verify that a systems components work properly.
- Unit tests are the responsibility of programmers developing components and are part of the job of implementing the system.

Summary

- All testing requires the development of a test plan.
- A test plan describes tests to be conducted:
  - Purpose of test.
  - Data to be used in the test.
  - Expected results.
- For unit testing, the test plan is formalized in the test code itself.
- For functional testing, test plan is an independent document that directs the testing.

Summary

- Developing a test plan involves identifying a number of cases to be tested.
- In a functional test, these “use cases” describe ways in which the system will be used.
- In a unit test, cases are often determined by possible states of object under test.
- Behavior of object is tested in any number of representative or critical states.