Click here to review SW Testing Strategies

Object Oriented Testing

Chapter 23

OO Testing

- **Class Testing:** Equivalent to unit testing for conventional SW.
  - The concept of a unit changes in the OO context.
  - The class or object is the smallest testable unit.
  - Testing must focus on each method of the class and the state behavior of the class.
  - It is important to consider the state of the class because this will effect the behavior, methods, of the class.
  - It is often difficult to determine the state of the object.
    - You may have to introduce new methods to look at the state variables for testing purposes.

OO Class Testing Example

- A bank account class.
  - It may have methods to open the account, deposit funds, withdraw funds and close the account.
  - The states of the account include
    - open with positive balance,
    - open with negative or zero balance and
    - closed.
  - How the methods behave depends on the state of the account.
    - An account with a zero or negative balance will not allow the customer to withdraw funds.
    - If positive it might allow customer to go to overdraft once.
    - You could introduce a new method to determine the if the account is open or closed and if balance is positive.

OO Testing

- **Integration Testing:** Because OO SW does not have a hierarchical control structure, conventional top-down or bottom-up integration strategies are not applicable.

Two approaches for OO **Integration testing**

- **Thread-based testing-**
  - integrate the set of classes required to respond to one input or event of the system.
  - Each thread is integrated and tested individually.
- **Use-based testing –**
  - first construct and test the classes which use very few (if any) server classes (these are called the independent classes).
  - The next step is to test the next layer of classes, the dependent classes, that use the independent classes.

Validation Testing

- Conventional black-box testing derived from the analysis model can be used to test the OO software.
- Use cases are a good place to look when developing test cases.

Test cases and the Class hierarchy

- Methods that are redefined in a subclass must be tested because it represents a new design and new code.

- Methods that are inherited must also be tested.
  - A subset of the original tests can be executed to ensure it works in the derived class.

- **Random Testing:** When a variety of different operation sequences are randomly generated.
  - Keep in mind the behavior life sequence of the class.

- **Partition testing:** Very similar to equivalence class partitioning for conventional SW. Partition the input and output of the class and design test cases to exercise each class.
- 3 Examples:
  - **State-based**
  - **Attribute-based**
  - **Category-based**

- **State-based partitioning:**
  - Look at the states for the class.
  - Determine which operations change the state of the class and which do not and design test cases to exercise the class.
  - Test each method, while object is in each state.
  - Design test cases to do this.

- **Attribute-based partitioning:**
  - Look at the attributes of the class.
  - Partition methods into those that use the attribute, modify it and those that do not use it.
  - Design test cases for each partition.

- **Category-based partitioning:**
  - Look at the methods for the class.
  - Partition the methods into categories based on their function.
  - Examples:
    - initialization operations, computational, queries and termination operations.
  - Design test cases for each partition

Student Registration Example

- A student registration class.
  - The methods include adding a class, dropping a class, transferring to a different section of the class and list classes.
  - A student must first be registered with the university (opened as a student).
  - Holds can be placed on a student and this effects whether he/she can register for a class.
  - There are limits on how many credit hours a student may register for.
  - A student may graduate (close a student).

Student Registration Example

- **Random testing** may generate:
  - register, add class, transfer, add class, drop class, add class, hold, release hold, add class, …

Student Registration Example

- **State-based partitioning** :
  - States:
    - Registered
    - Hold
    - Full Load
    - Partial Load
    - Freedom
  - Events:
    - Register, Add, Drop, List, transfer, Hold,Release, Graduate
  - Added methods:
    - Display Hold
    - Display credits

OO Test Case Design

- Given the differences between conventional and OO SW, test case design is slightly different.
- OO test cases should be defined in the following way:
  - Each test case should be uniquely identified and associated with the class to be tested.
  - The purpose of the test should be stated.
  - The following list should be developed for each test:
    1. List of object states to be tested.
    2. List of messages and operations (methods) to be tested
    3. List of exceptions that may occur as the object is tested.
    4. List of external conditions (think about system testing) that may change.
- Anything else needed for the test?

OO Test Case Design

- Instead of testing each component, you will test each class.
- Make sure you identify the class you are testing.
- You must test each method, in each state for the class.
  - This information will be included in the purpose/condition.
- If needed, add methods to check state of the object.
  - You could use the result of these methods in your expected outcome.

Click to go to SW Metrics