Software Engineering 2UCCE501

Module 5

Module 5 Testing & Maintenance

5.1 Testing Concepts: Purpose of Software Testing, Testing Principles, Goals of Testing, Testing aspects: Requirements, Test Scenarios, Test cases, Test scripts/procedures,

5.2 Strategies for Software Testing, Testing Activities: Planning Verification and Validation, Software Inspections, FTR

5.3 Levels of Testing : unit testing, integration testing, regression testing, product testing, acceptance testing and White-Box Testing

5.4 Black-Box Testing: Test case design criteria, Requirement based Testing, Boundary value analysis, Equivalence Class Partitioning

5.5 Object Oriented Testing: Review of OOA and OOD models, class testing, integration testing, validation testing

5.6 Reverse & Reengineering, types of maintenance

Review of OOA and OOD models

- The **construction of object-oriented** software begins with the **creation of requirements (analysis) and design models.**

- **review of OO analysis and design** models is especially useful because the same semantic **constructs various levels of software product.**

Review of OOA and OOD models

- **Earlier review may help in avoiding following problems in analysis:**

1. Unnecessary creation of special subclasses to **accommodate invalid attributes** is avoided.
2. A misinterpretation of the class definition may lead to **incorrect or extraneous class relationships.**
3. The **behavior of the system** or its classes may be **improperly characterized** to accommodate the extraneous attribute.

Review of OOA and OOD models

- **Earlier review may help in avoiding following problems in <u>design:</u>**

1. **Improper allocation of the class to subsystem** and/or tasks may occur during system design.
2. **Unnecessary design work** to create the procedural design for the operations that address the **extraneous attribute.**
3. The messaging model will be incorrect

Review of OOA and OOD models

- latter stages of their development, (OOA) and (OOD) **models provide substantial information about the structure and behavior of the system.**

- models should be **subjected to rigorous review** prior to the generation of code.

Review of OOA and OOD models

- **Correctness of OOA and OOD Models**
  - syntactic correctness is judged on proper use of the symbols
  - each model is reviewed to ensure that proper modeling conventions have been maintained.
  - If the model accurately reflects the real world , then it is semantically correct.

Review of OOA and OOD models

- **Consistency of Object-Oriented Models**
  - The consistency judged by “considering the relationships among entities in the model.
  - Each class and its connections to other classes – examine consistency
  - class-responsibility-collaboration (CRC) model used to measure consistency.

Review of OOA and OOD models

- **Steps to evaluate class model:**

1. Revisit the CRC model and the object-relationship model – **requirements**

2. **Inspect** the description of each CRC index card to determine if a delegated responsibility is part of the collaborator’s definition.

3. **Invert** the connection to ensure that each collaborator that is asked for service is receiving requests from a reasonable source.

Review of OOA and OOD models

4. Determine whether classes are valid or whether responsibilities are properly grouped among the classes.

5. Determine whether widely requested responsibilities might be combined into a single responsibility.

Object Oriented Testing Strategies

2016 Q5 a

- classical software testing strategy begins with “testing in the small” and works outward toward “testing in the large.”

1. **Unit Testing in the OO Context**
    - Classes and objects
    - Each class and object have attributes and operations
    - Here smallest testable unit is class.

- Class (superclass) has operations defined , which are inherited by subclasses.
- Because **operation *X()*** is used varies in subtle (**indirect) ways**, it is necessary to test operation ***X()* in the context of each of the subclasses.**
- This means that testing operation *X()* in a vacuum (the traditional unit-testing approach) is ineffective in the object-oriented context.
- class testing for OO software is driven by the operations encapsulated by the class and the state behavior of the class.

Object Oriented Testing Strategies

2. Integration Testing in the OO Context

- There are two different strategies for integration testing of OO systems

Object Oriented Testing Strategies

1. Thread-based testing

- integrates the set of classes required to respond to one input or event for the system.

- Each thread is integrated and tested individually to ensure that no side effects occur.

Object Oriented Testing Strategies

2. **Use-based testing**
   - begins the construction of the system by testing independent classes.
   - Dependent classes, that use the independent classes are tested.
   - This sequence of testing layers of dependent classes continues until the entire system is constructed.
   - Use of drivers and stubs is to be avoided.

Object Oriented Testing Strategies

3. Validation Testing in an OO Context

- Like conventional validation, the validation of OO software **focuses on user-visible actions and user-recognizable outputs from the system.**

- Tester should draw upon **use cases based on requirement model.**

- The use case provides a scenario that has a **high likelihood of uncovered errors in user-interaction requirements.**