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

2016 Q5 a

Levels of Testing (Black Box Testing)

- ***Black-box testing***, also called ***behavioral testing,*** focuses on the **functional requirements of the software.**
- Will check all functional requirements for a program
- **Black-box testing attempts to find errors in the following categories:**
  (1) incorrect or missing functions
  (2) interface errors
  (3) errors in data structures or external database access
  (4) behavior or performance errors
  (5) initialization and termination errors.

Levels of Testing (Black Box Testing)

- Black-box testing is **NOT** an alternative to white-box techniques.

- It is a complementary approach that is likely to uncover a different class of errors

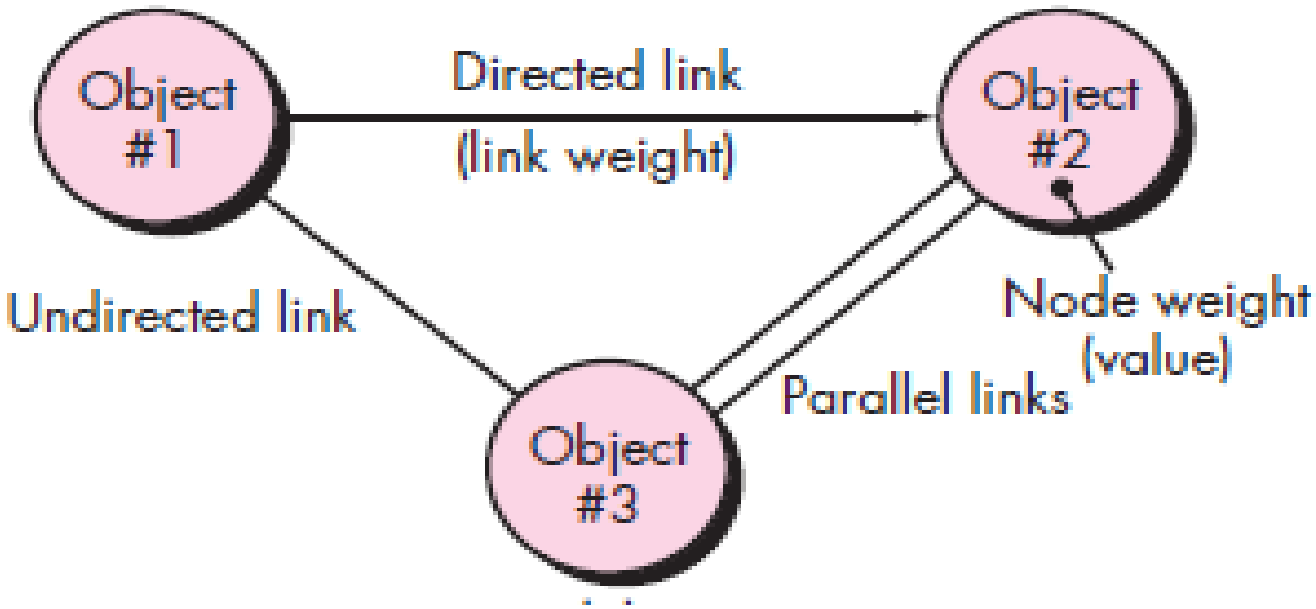
- **Criteria for testing**
  - Identifying classes(types) of errors.
  - additional test cases that must be designed to achieve reasonable testing.

Graph-Based Testing Methods

- **understand the objects** that are modeled in software and the **relationships that connect these objects.**
- Software testing begins by **creating a graph of important objects and their relationships**
- define a series of tests that **verify all objects have the expected relationship to one another.**
- tests that will cover the graph - each object and relationship is exercised and errors are uncovered.

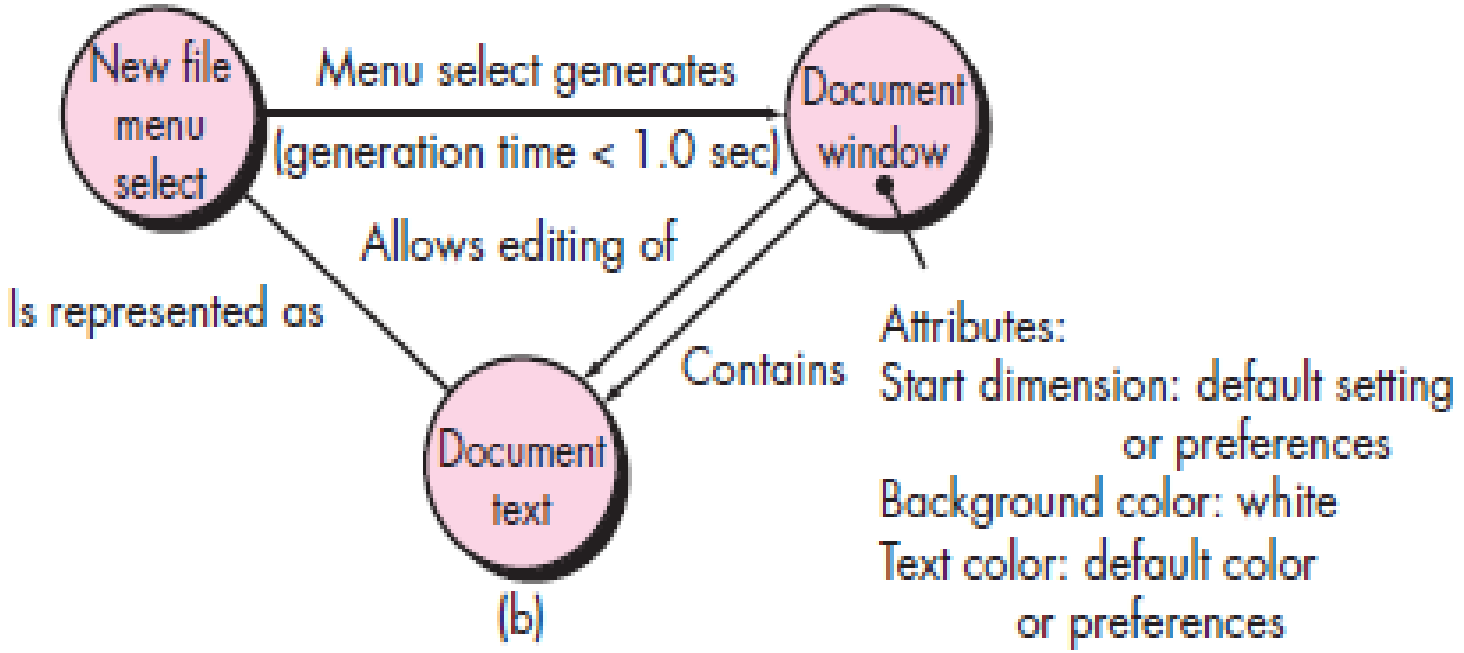
Graph-Based Testing Methods

- Graph notation



Graph-Based Testing Methods

- Example



Graph-Based Testing Methods

- test cases are designed in an attempt to **find errors in any of the relationships.**

- number of behavioral testing methods that can make use of graphs:

1. Transaction flow modeling

Indicate how the different parts/ modules will be invoked

Example: airline reservation with validation

Data flow diagram can be used in creating graphs

Graph-Based Testing Methods

2. Finite state modeling

- different user observable states of the software

- **Example: order-information is verified during *inventory-availability look-up and is followed by* *customer-billing-information***

- **State transition diagram** can be used in creating graphs

Graph-Based Testing Methods

3. Data flow modeling

- transformations that occur to translate one data object into another.

4. Timing modeling

sequential connections between objects

specify the required execution times as the program executes

Equivalence Partitioning

- Equivalence partitioning is a software testing technique that **divides the input and/or output data of a software unit into partitions** of data **from which test cases can be derived**.

- The equivalence partitions are **usually derived from the requirements specification for input.**

- Test cases are designed to **cover each partition at least once.**

Equivalence Partitioning

- Equivalence partitioning technique **uncovers classes of errors.**
- An equivalence class represents a set of **valid or invalid states for input conditions.**
- Typically, an input condition is either a specific **numeric value, a range of values, a set of related values, or a Boolean condition.**

Equivalence Partitioning

- **Guidelines for equivalence classes:**
  1. If an input condition specifies a range, one valid and two invalid equivalence classes are defined.
  2. If an input condition requires a specific value, one valid and two invalid equivalence classes are defined.
  3. If an input condition specifies a member of a set, one valid and one invalid equivalence class are defined.
  4. If an input condition is Boolean, one valid and one invalid class are defined.

Equivalence Partitioning

- Test cases are selected so that the **largest number of attributes of an equivalence class are exercised at once.**
- **For example, a savings account in a bank has a different rate of interest depending on the balance in the account.**

Invalid partition	Valid (for 3% interest)		Valid (for 5%)		Valid (for 7%)
-$0.01	$0.00	$100.00	$100.01	$999.99	$1000.00

Boundary Value Analysis

- A **greater number of errors occurs at the boundaries** of the input domain rather **than in the "center".**

- Boundary value analysis leads to a **selection of test cases that exercise bounding values.**

- BVA leads to the **selection of test cases at the "edges" of the class.**

Boundary Value Analysis

- Rather than **focusing solely on input conditions**, **BVA derives test cases from the output domain as well.**

- **Examples:** temperature versus pressure table
- internal program data structures with prescribed boundaries

- **Guidelines for BVA**
- If an input condition specifies a range bounded by values a and b, test cases should be designed with values a and b and just above and just below a and b.

- If an input condition specifies a number of values then test case designed for maximum and minimum values.