Lecture 09

Software Testing Techniques and Strategies

Introduction

- What is software testing ?
  - Once source code has been generated software must tested to uncover (correct) as many errors as possible before delivery to the customer.
    - Goal→ design a series of test cases
  - Two way testing can be done
    1. Exercise the internal logic of the software components
    2. Exercise the input and output domain of the program to uncover errors in program function, behavior and performance
- Who does it ?
  - Early by software engineer
  - Now a days testing specialist may become involved.

Introduction

- Important of software testing
  - Review
  - Software Quality Assurance activity can be done
  - Uncover errors

Software Testing

1. Module (Unit) Testing
   - 1.1 Black Box
     - 1.1.1 Functional Check
     - 1.1.2 Equivalence Partitioning
     - 1.1.3 Boundary Value Analysis (BVA)
   - 1.2 White Box
     - 1.2.1 Path testing
2. System (Integrity) Testing
   - 2.1 Top down
   - 2.2 Bottom up
   - 2.3 Mixed

Black Box(Behavioral) Testing

- Test that are conducted at the software interface
- Design to uncover errors
- Test that the software functions are operational, that the inputs are properly accepted and outputs are correctly produced. That the integrity of the external information (e.g. database) is maintained.
- Focuses on the functional requirements of the software
- Black Box testing attempt to find errors in the following categories
  1. Incorrect or missing functions
  2. Interface errors
  3. Errors in the data structures or external data base access
  4. Behavior or performance errors
  5. Initialization and termination errors

B.B. Testing

- Test case criteria
  1. Test cases that are reduce, by a count that is greater than one, the number of additional test cases that must be designed to achieve reasonable testing.
  2. Test case that tells us something about the presence or absence of classes of errors, rather than an error associated only with the specific test at hand.
- Functional check
  - Check the specified functioins that are designed to perform

B.B. Testing

- Equivalence Partitioning (EP) :
  - Testing method that divided the input domain of a program into classes of data from which test cases can be derived.
  - EP strives to define a test case that uncovers classes of errors, thereby reducing the total number of test cases that must be developed.
  - Based on equivalence classes for an input condition
  - Equivalence class represents a set of valid or invalid states for input conditions.

B.B. Testing

- Equivalence class may be defined according to the guidelines as follows :
  1. If an input condition specifies a range of value, 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 number of a set, one valid and one invalid equivalence classes are defined.
  4. If an input condition is boolean, one valid and one invalid equivalence classes are defined.

B.B. Testing

- Boundary Value Analysis (BVA) :
  - A greater number of errors tends to occur at the boundaries of the input domain rather than in the centre.
  - Complement of the equivalent partitioning.
  - BVA leads to selection of test cases at the edges of the class.
  - BVA derives test cases from the output domain as well.
  - BVA are similar in many respects to those provided for equivalence partitioning.

B.B. Testing

- BVA may be defined according to the guidelines as follows :

1. If an input condition specifies a range of 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.
2. If an input condition specifies a number of values, test cases should be developed that exercise the minimum and the maximum numbers. Values just above and below minimum and maximum are also tested.
3. Apply guidelines 1 and 2 to output conditions.
4. If the internal program data structures have prescribed boundaries (e.g. an array has a defined limit of 100 entries), be certain to design a test case to exercise the data structure at its boundary.

White Box (Glass) Testing

- On close examination of control structure of the procedural details.
- Logical paths through the software are tested by providing test cases that exercise specific sets of conditions and / or loops.
- The status of the program may be examined at various points to determine, if the expected or asserted status corresponds to actual status.

Using White Box testing derive test cases are,

1. Guarantee that all independent paths within a module have been excercised at least once.
2. Exercise all logical decisions on their true and false sides

White Box (Glass) Testing

3. Exercise all loops at their boundaries and within their operational bounds.
4. Exercise internal data structures to ensure their validity

Path Testing

1. Geographic (Flowgraph)
2. Matrix (Graph Matrix)

Geographic (flowgraph):

- **Sequence**
- **Decision**
- **Looping**

Sequence of process boxes and a decision diamond can map into a single node

W.B. Testing

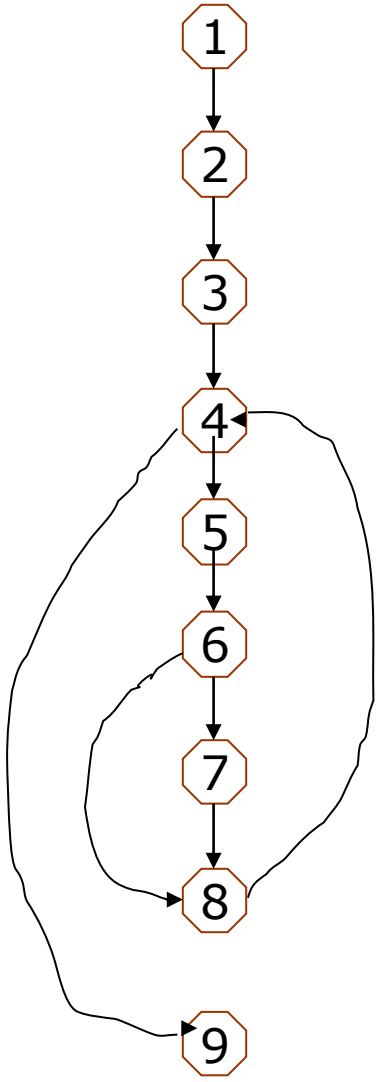
- Region : Area bounded by edges and nodes are called regions.
- Predicate node : Each node that contains a condition is called predicate node.
- **Cyclomatic Complexity** : is a software metric that provides a quantitative measure of the logical complexity of the program.
- Defined number of independent paths in the basis set of a program and provides us with an upper bound for the number of tests that must be conducted to ensure that all statements have been executed at least once.

W.B. Testing

- Complexity is computed in one of three ways :
  1. The number of regions of the flowgraph correspond to the cyclomatic complexity
  2. Cyclomatic complexity, V(G), for a flow graph, G, is defined as V(G)=E-N+2 (E→edge, N→node)
  3. V(G)=P+1 (P → predicate node)
- Link weight : How many paths between two nodes is determined.
- Rules :
  1. If two nodes are connected then weight is 1, otherwise 0.
  2. Addition for parallel connection

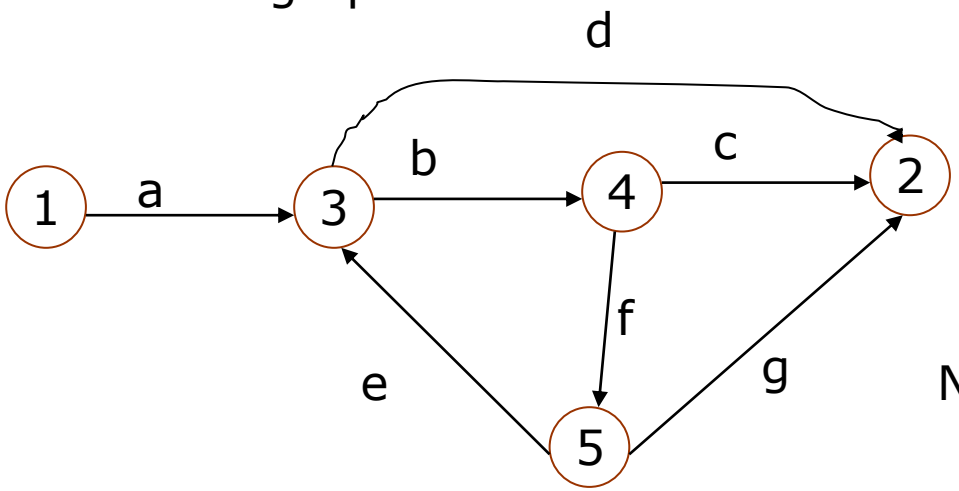
W.B. Testing

- Rules :
  3. Multiplication for series connection
  4. Exponentiation for looping
- Example of Flowgraph: (program segment)
  1. Read N
  2. MAX = 0
  3. I = 1
  4. While I <= N
  5. Read X(I)
  6. If X(I) > MAX
  7. Then MAX = X(I)
  8. I = I + 1
  9. Print MAX



Graph Matrix

- Directed graph



Connected to Node

Node

	1	2	3	4	5
1			a		
2					
3		d		b	
4		c			f
5		g	e		

W.B. Testing

- Control structure testing
  1. Condition testing
  2. Dataflow testing
  3. Loop testing

Condition testing :

E1 < relational operator > E2

- A test case design method
- Exercises the logical conditions contained in a program module/unit

W.B. Testing

Dataflow testing :

Loop testing :

Test validity of loop constructs

Simple loop, concatenated loop, nested loop and unstructured loops

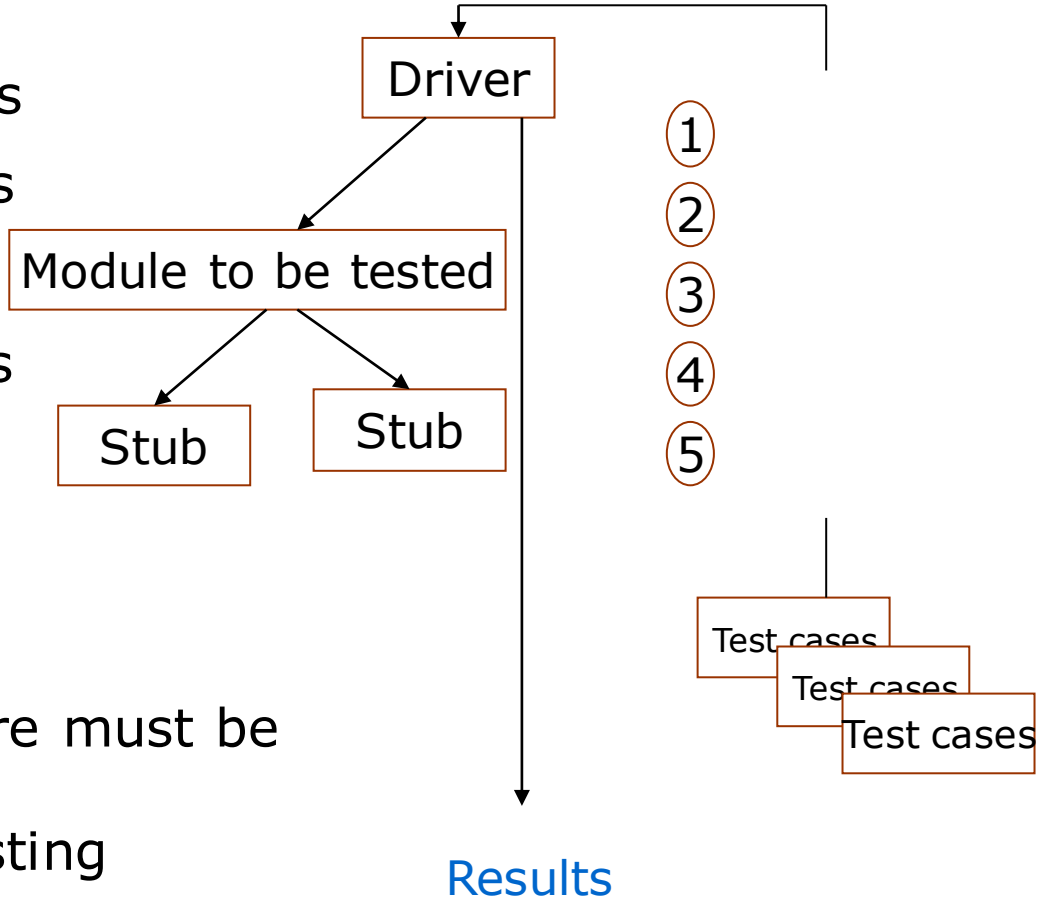
Software testing strategy

- Unit testing
  1. Interface testing
  2. Local data structures
  3. Boundary conditions
  4. Independent paths
  5. Error handling paths

A component is not a stand alone program,

Driver and / or stub software must be developed for each unit testing

Unit test procedure



Software testing strategy

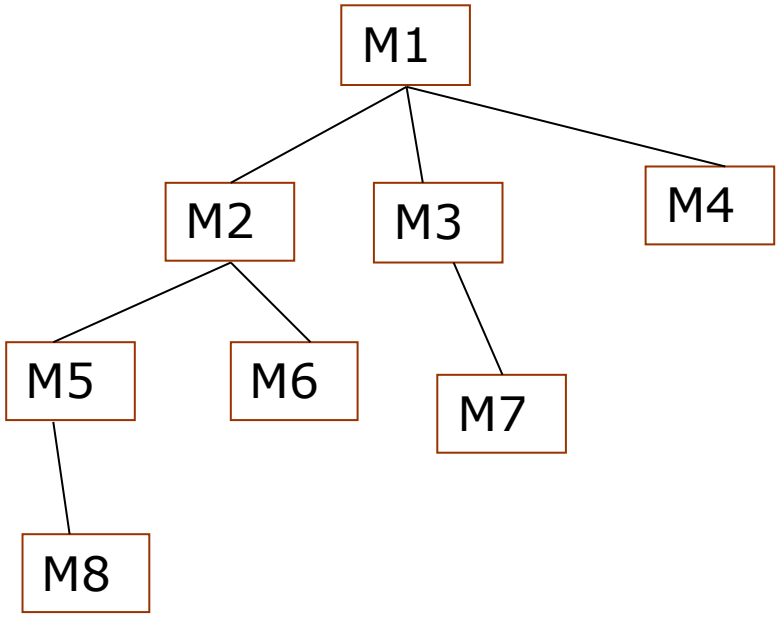
- Driver nothing more than a " main Program"
- Stub or " dummy subprogram", uses the subordinate modules interface, may do minimum data manipulation, prints verification of entry, and returns control to the module undergoing testing.
- Integration testing:
  - The problem, of course, is "putting them together" – interface (data can be lost across an interface)
  - Global data structures can present problems.
  - One module can have an inadvertent, adverse affect on another.
  - Subfunctions when combined may not produce desired functions

Software testing strategy

- Two types of Integration testing
  1. Non incremental integration (all components are combined in advance and the entire program is tested as a whole)
  2. Incremental integration
- Top Down
  - Modules subordinate to the main control module are incorporated into the structure in either a depth-first or breadth-first manner.
- Depth-first integration
  - Would integrate all components on a major control path of the structure

Software testing strategy

- Breadth-first integration :
  - Incorporate all components directly subordinate at each level, moving across the structure horizontally.



- Steps :
  1. The main control module is used as a test driver and stubs are substituted for all components directly subordinate to the main control module.

Software testing strategy

2. Depending upon the integration approach selected (depth or breadth first) subordinate stubs are replaced one at a time with actual components.

3. Test are conducted as each component is integrated.

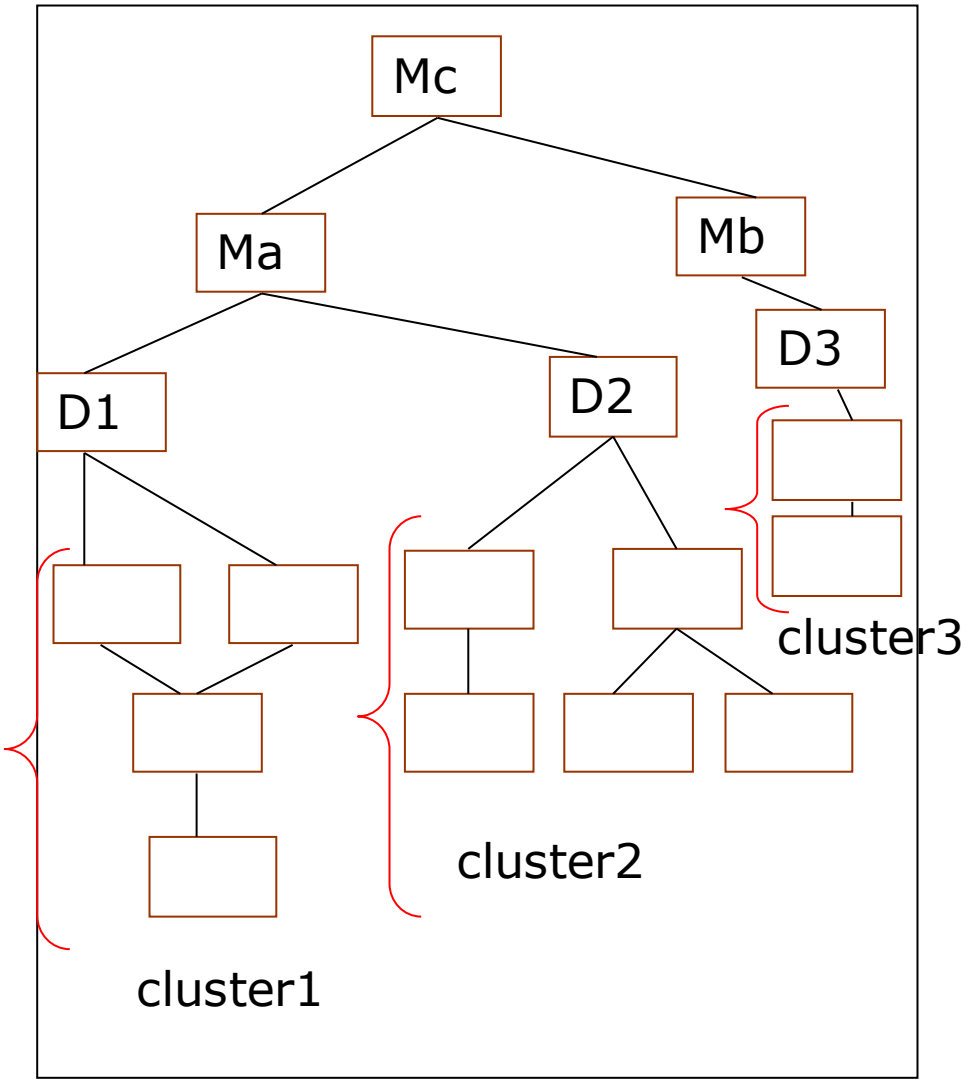
4. On completion of each set of tests, another stub is replaced with the real component.

5. Regression testing may be conducted to ensure that new errors have not been introduced.

6. Process continues from step2 until the entire program structure is built.

Software testing strategy

- Bottom-up Integration :
- Steps :
  1. Low level components are combined into clusters that perform a specific software subfunction.
  2. A driver is written to coordinate test case input and output.
  3. The cluster is tested
  4. Drivers are removed and clusters are combined moving upward in the program structures.



System Testing

- **Recovery testing :**
  - A system must be fault tolerant.
  - Recovery testing is a system that forces the software to fail in a variety of ways and verifies that recovery is properly performed.
  - Recovery is (i) Automatic (by the system itself), (ii) Reinitialization, (iii) Checkpointing mechanisms, (iv) Data recovery and (v) restart are evaluate for correctness
  - MTTR to calculate the mean time to recovery of a software.

System Testing

- **Security Testing :**
  - Improper or illegal penetration (Hackers).
  - Security testing verifies that the protection mechanism build into a system will, in fact, protect it from improper penetration.
  - password
- **Stress Testing :** (a verification of stress is called sensitivity test)
  - Stress test are designed to confront programs with abnormal situations.
  - Stress testing execute a system in a manner that demands resources in abnormal quantity, frequency or volume.
  - Special tests may be designed that generate ten interrupts per second, when one or two is the average rate.

System Testing

- Input data rate may be increased by an order of magnitude to determine how input functions will respond.
- Test cases that are require maximum memory or other resources are executed.
- Test cases that may be cause memory management problems are designed.
- Test cases may cause excessive hunting for disk-resident data are created.

Performance testing :

- Design to test the run time performance of software.
- This testing is performance throughout all steps in testing process

System Testing

- **Regression Testing :**

- Each time a new module is added as part of integration testing, the software changes
  1. New data flow paths are established.
  2. New I/O may occur.
  3. New control logic is invoked.
  4. Functions may not work flawlessly

- Re-execution of some subset of tests that have already been conducted to ensure that changes have not propagated unintended side effects.

System Testing

- Classes of test case :
  1. Will exercise all software functions.
  2. Functions that are likely to be affected by the change.
  3. Software components that have been changed.

- **Smoke testing** :

- It is designed as a packing mechanism for time critical projects, allowing software team to assess its project on a frequent basis.

- Activities :
  - All data files, libraries, reusable modules, components are checked.
  - Expose errors that will keep the build from properly performing its functions.
  - Test daily basis.

System Testing

- Advantages :
  - Integration risk minimize
  - Quality of the end product is improved.
  - Error diagnosis and correction are simplified.
  - Progress is easier to assess.
- **Verification & Validation** (requirement, α- testing & β - testing):
- **Acceptance testing** :
  - Specified by the customer and focus on overall system features, functionality that are visible and reviewable by the customer.