Software Testing

King Saud University
College of Computer and Information Sciences
Department of Computer Science

Dr. S. HAMMAMI

S.H

Objectives

- To discuss the distinctions between validation testing and defect testing

- To describe the principles of system and component testing

- To describe strategies for generating system test cases

Defect testing

- The goal of defect testing is to discover defects in programs

- A *successful* defect test is a test which causes a program to behave in an anomalous way

- Tests show the presence not the absence of defects

Testing & Verification & Validation

- Testing = Verification + Validation

- **Verification**: Static Testing (no run)

- **Validation**: Dynamic Testing (Run code)

Who Tests the Software

Developer

- understands the system
- has the source code
- white-box 'Unit' testing
- will test "gently"
- driven by delivery 'schedule' constraint

Independent tester

- must learn about the system
- has no source code
- black-box 'Acceptance' testing
- will attempt to break the sys (ME!!)
- driven by quality constraint

Testing policies

- Only exhaustive testing can show a program is free from defects. However, exhaustive testing is impossible.

- Testing policies define the approach to be used in selecting system tests:

  - All functions accessed through menus should be tested;
  - Combinations of functions accessed through the same menu should be tested;
  - Where user input is required, all functions must be tested with correct and incorrect input.

The testing process

- Component (Unit) testing: needs source code (White-box)
  - Testing of individual program components
  - Usually the responsibility of the component developer
  - Tests are derived from the developer's experience

- System Testing: Involves integrating components to create a system or sub-system. May involve testing an increment to be delivered to the customer.

  - Integration testing - the test team have access to the system source code. The system is tested as components are integrated.
  - Release testing - the test team test the complete system to be delivered as a black-box.

Component testing

- Component or unit testing is the process of testing individual components in isolation.

- It is a defect testing process.

- Components may be:

  - Individual functions or methods within an object;
  - Object classes with several attributes and methods;
  - Composite components with defined interfaces used to access their functionality.

System testing

- Involves integrating components to create a system or sub-system.

- May involve testing an increment to be delivered to the customer.

- Two phases:
  - Integration testing - the test team have access to the system source code. The system is tested as components are integrated.
  - Release testing - the test team test the complete system to be delivered as a black-box.

Release testing

- The process of testing a release of a system that will be distributed to customers.

- Primary goal is to increase the supplier's confidence that the system meets its requirements.

- Release testing is usually black-box or functional testing
  - Based on the system specification only;
 - Testers do not have knowledge of the system implementation.

Black-box testing *functionality testing*

- Program is considered as a 'black-box'
- No need to know or access source code
- Functionality testing
- No implementation testing (implementation testing needs source code)
- Test cases are based on the system specification
- Test planning can begin early in the software process

Black-box testing

- Testers provide the system with inputs and observe the outputs

  - They can see none of:
    - The source code
    - The internal data
    - Any of the design documentation describing the system's internals

Black-box testing
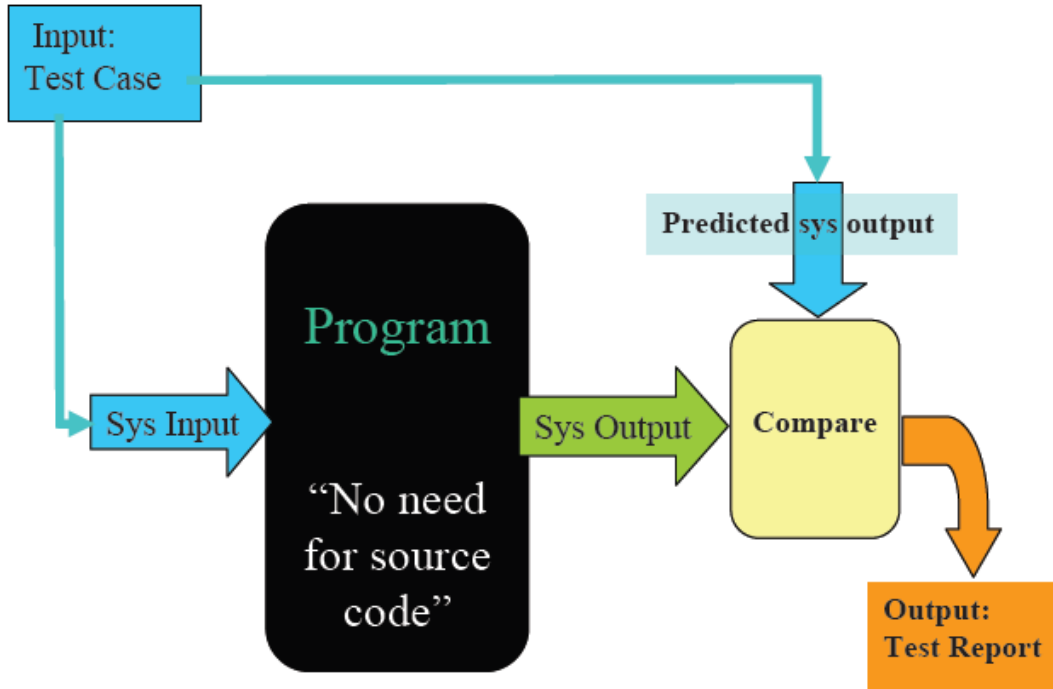
Test case design

- Involves designing the test cases (inputs and outputs) used to test the system.

- The goal of test case design is to create a set of tests that are effective in validation and defect testing.

  validation or defect *white or Black box* ← *app roach* قسم على

- Design approaches:
  - Requirements-based testing; *validation, Black box*
  - Partition testing; *defect, Black box*
  - Structural testing. *defect, white box*   ← *white testing*
  - Path testing *Both validation + defect, white box*

Requirements based testing

- A general principle of requirements engineering is that requirements should be testable.

- Requirements-based testing is a validation testing technique where you consider each requirement and derive a set of tests for that requirement.

Partition testing

- Input data and output results often fall into different classes where all members of a class are related.

- Each of these classes is an equivalence partition or domain where the program behaves in an equivalent way for each class member.

- Test cases should be chosen from each partition.

$-\infty$ 0 20 $+\infty$

partition   partition

Equivalence partitioning

- Objective:

Reduce the number of test cases
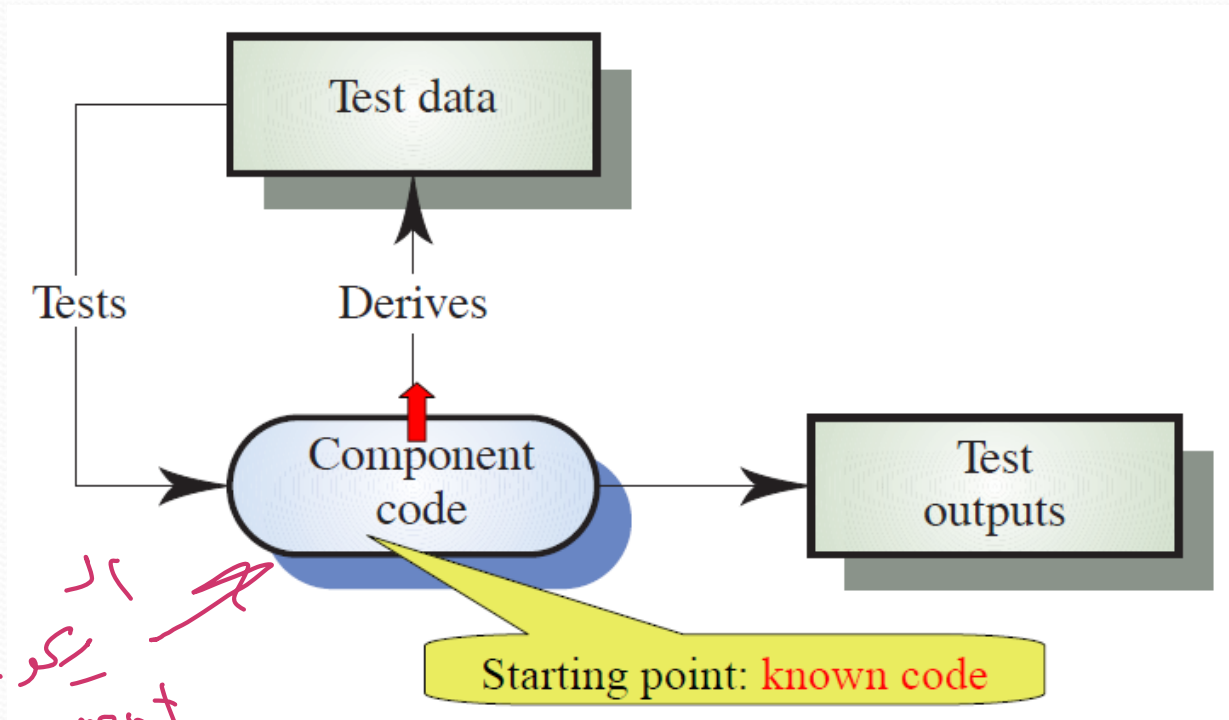
Structural testing: White-box testing

- Sometime called white-box testing.

- Derivation of test cases according to program structure. Knowledge of the program is used to identify additional test cases.

- Objective is to exercise all program statements (not all path combinations).

اذا كان كل paths هو في validation
تتم في Structural و component اذا

Structural testing: White-box testing

- **Synonyms:**
  - **Glass-box, Clear-box, Transparent-box**
- For small program units
- Needs **source code**

- Objective: is to exercise **all program statements**
- (not all path combinations)

Structural testing: White-box testing

Tests → Derives → Test data

Component code → Test outputs

Starting point: known code

test كل تاثير
component على تركيز

Path testing

- The objective of path testing is to ensure that the set of test cases is such that each path through the program is executed at least once.

- The starting point for path testing is a program flow graph that shows nodes representing program decisions and arcs representing the flow of control.

  if condition

- Statements with conditions are therefore nodes in the flow graph.

Program flow graphs

- **Flow Graph**:
  - nodes representing program decisions
  - arcs representing the flow of control
  - Ignore sequential statements (assignments, procedures calls, I/O)

- Statements with conditions are therefore nodes in the flow graph

- **Cyclomatic complexity =**

  Number of edges − Number of nodes + 2 = number of paths

  = number of test cases

Cyclomatic complexity

- Cyclomatic complexity = **number of tests** to test all control statements

  if condition

- Cyclomatic complexity = number of conditions in a program

- Although all paths are executed, all combinations of paths are not executed

Example

initial

لا نزم نرجع للأعل

نرجع كل node من كل الأعل
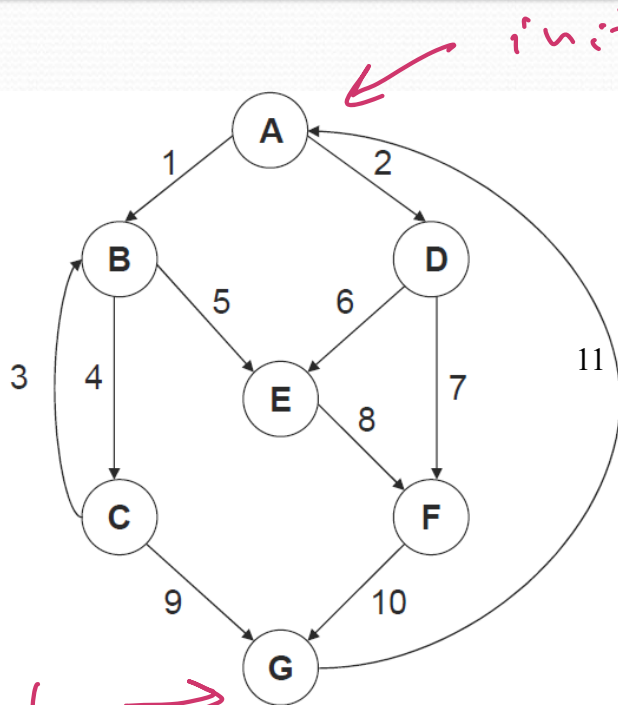
Path 1: A, B, C, G.
Path 2: A, B, C, B, C, G. or A B C B E FG
Path 3: A, B, E, F, G.
Path 4: A, D, E, F, G.
Path 5: A, D, F, G.
Path 6: A, D, F, G, A, B, C, G

كلهم نفس بعض

final →

Cyclomatic Complexity = 11 -7 + 2 = 6

→ 6   paths

Binary search flow graph

start

final
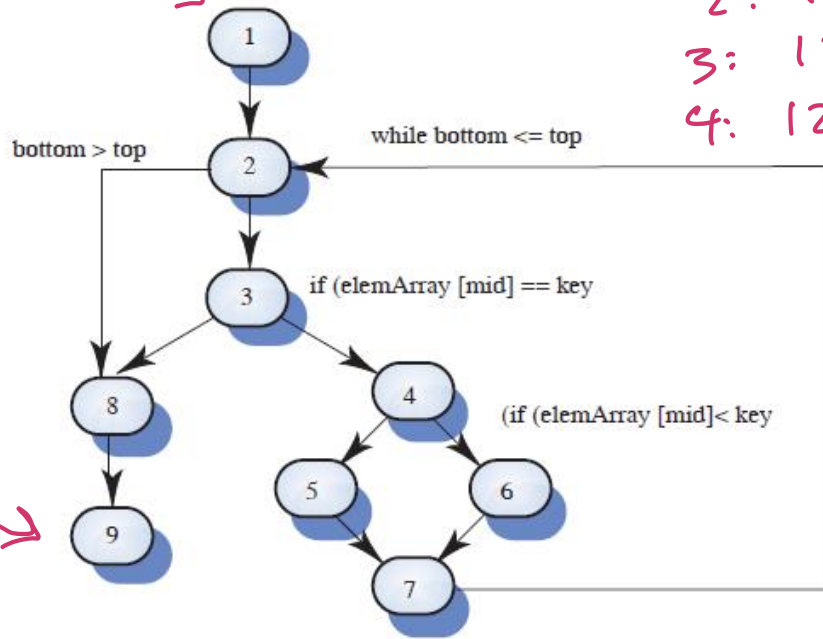
path 1: 1 2 8 9
2: 1 2 3 8 9
3: 1 2 3 4 5 7 2 8 9
4: 1 2 3 4 6 7 2 8 9

C C = 11 - 9 + 2 = 4

Independent paths

Cyclomatic Complexity = 11 -9 + 2 = 4

Independents Paths:

1-2-8-9
1-2-3-8-9
1-2-3-4-5-7-2-8-9
1-2-3-4-6-7-2-8-9

Key points

- Testing can show the presence of faults in a system; it cannot prove there are no remaining faults.

- Component developers are responsible for component testing; system testing is the responsibility of a separate team.

- Integration testing is testing increments of the system; release testing involves testing a system to be released to a customer.

Key points

- Use experience and guidelines to design test cases in defect testing.

- Equivalence partitioning is a way of discovering test cases - all cases in a partition should behave in the same way.

- Structural analysis relies on analysing a program and deriving tests from this analysis.