


Defect testing

- Testing programs to establish the presence of system defects


Objectives

- To understand testing techniques that are geared to discover program faults
- To introduce guidelines for interface testing
- To understand specific approaches to object-oriented testing
- To understand the principles of CASE tool support for testing


Topics covered

- Defect testing
- Integration testing
- Object-oriented testing
- Testing workbenches


The testing process

- Component testing
  - Testing of individual program components
  - Usually the responsibility of the component developer (except sometimes for critical systems)
  - Tests are derived from the developer's experience
- Integration testing
  - Testing of groups of components integrated to create a system or sub-system
  - The responsibility of an independent testing team
  - Tests are based on a system specification


Testing phases

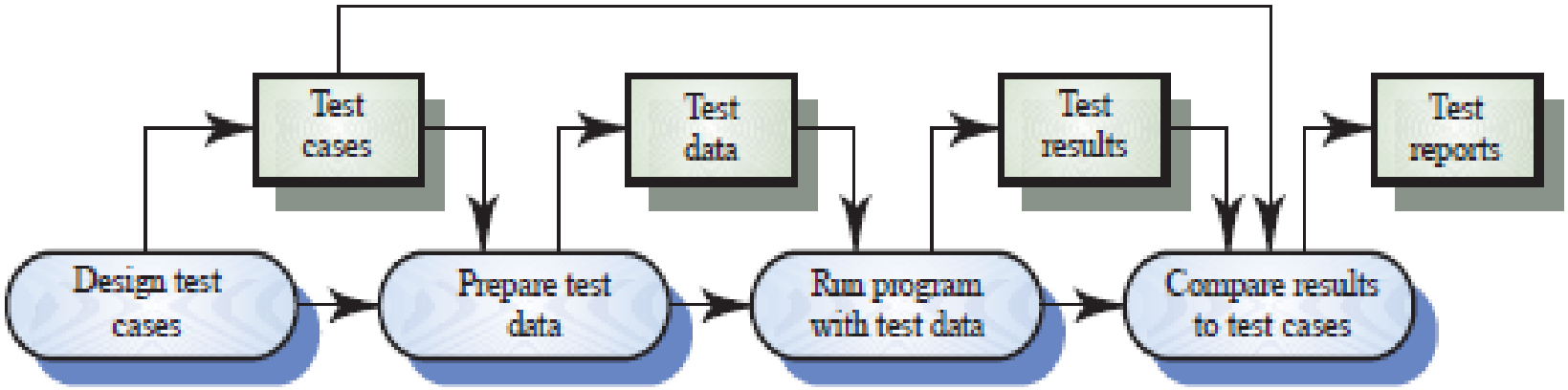
Software developer

Independent testing team


Defect testing

- 결함 테스팅(**defect testing**)의 목표는 프로그램에 있는 결함들을 발견하는 것임.
- *성공적인* 결함 테스트는 프로그램이 비정상적인(**anomalous**) 방식으로 동작하도록 하는 것.
- 테스트는 결함의 부재(**absence**)가 아니라 결함의 존재를 보이는 것이다.

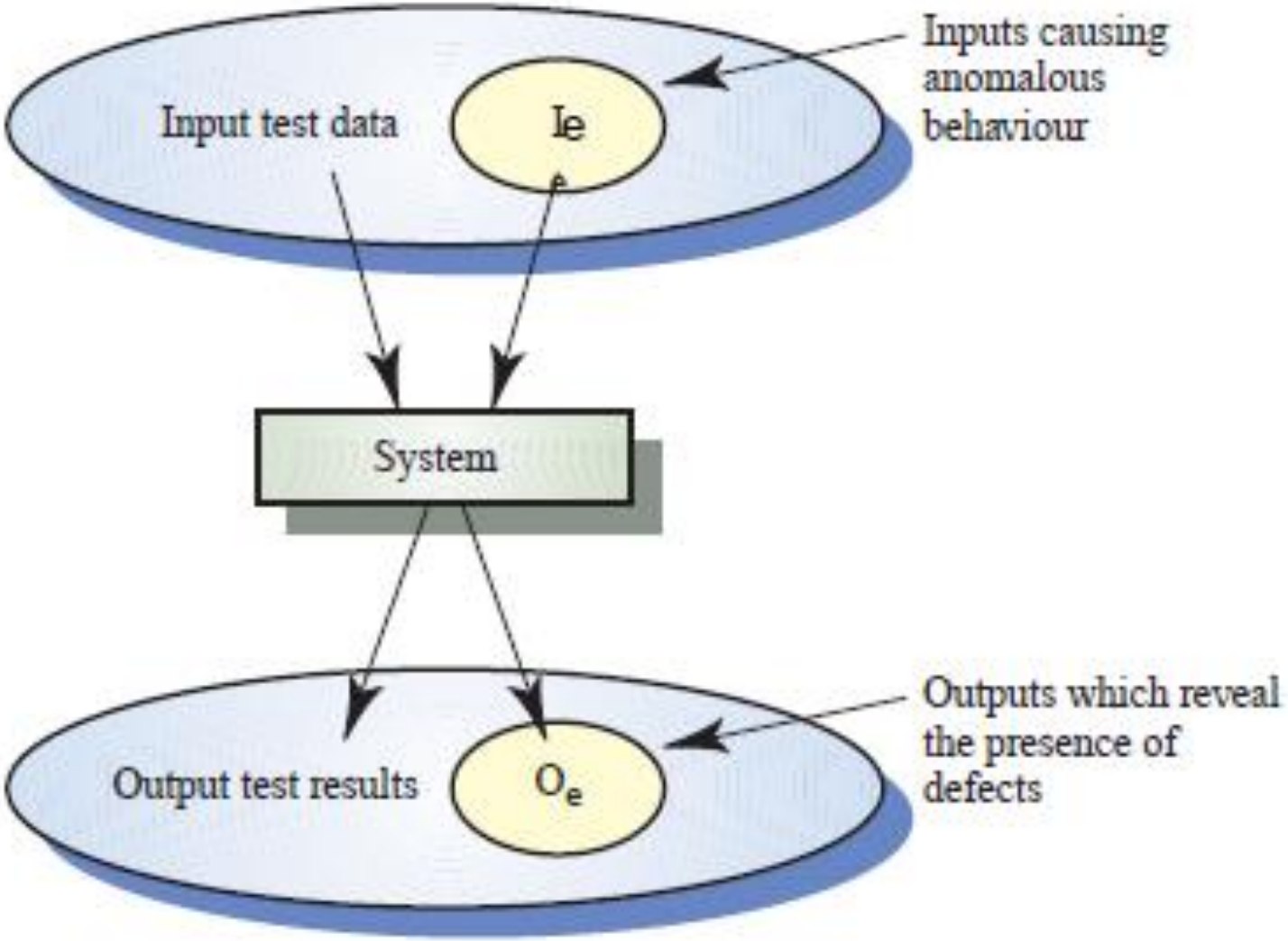
The defect testing process




Black-box testing

- 프로그램을 'black-box'로 생각하고 테스팅을 수행하는 방법.
- The program test cases are based on the system specification
- Test planning can begin early in the software process

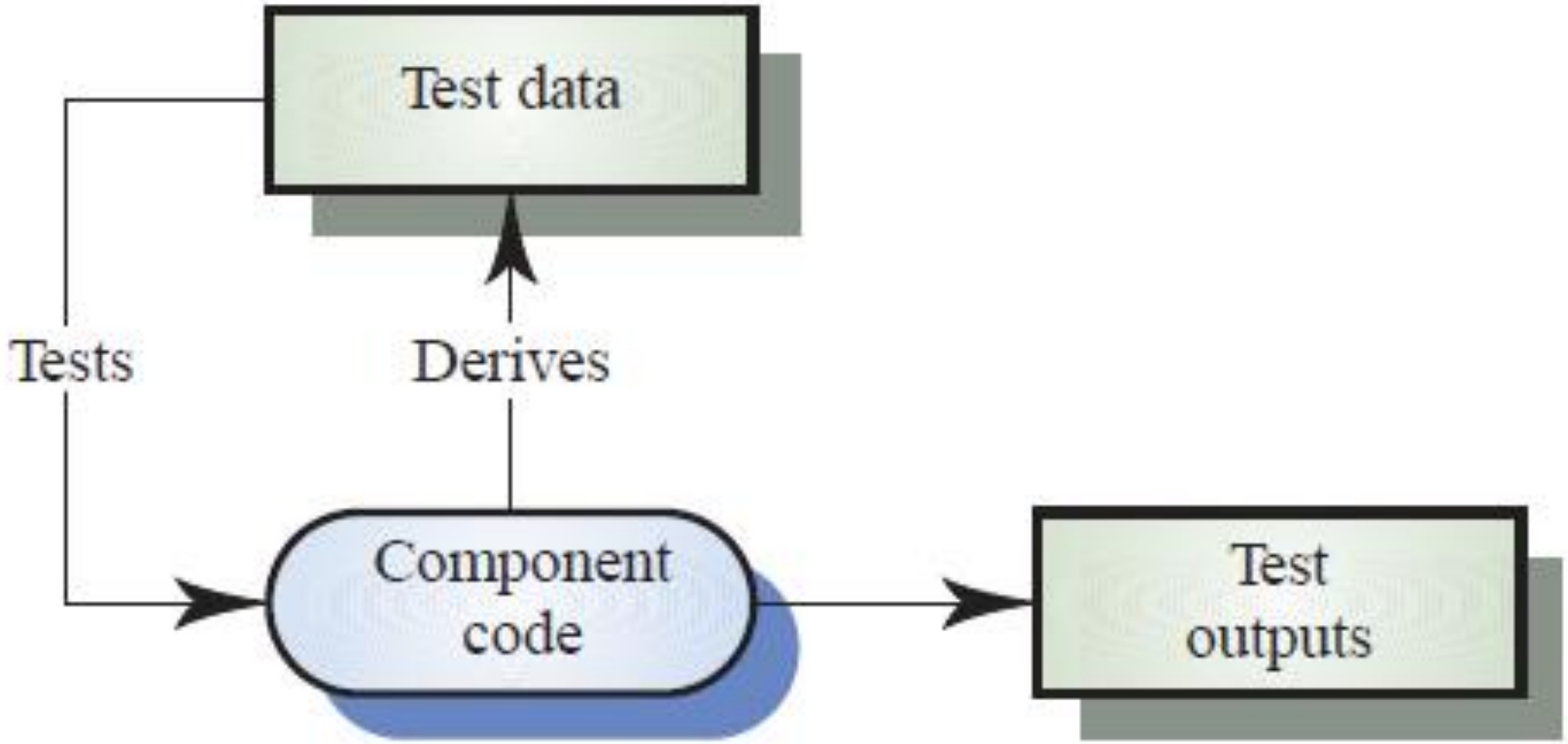
Black-box testing




Structural testing

- Sometime called white-box testing
- 프로그램의 구조에 따라서 테스트 케이스들을 유도함. 부가적인 테스트 케이스들을 식별하기 위해 프로그램에 대한 지식이 사용됨.
- Objective is to exercise all program statements
  (not all path combinations)

White-box testing




Path testing

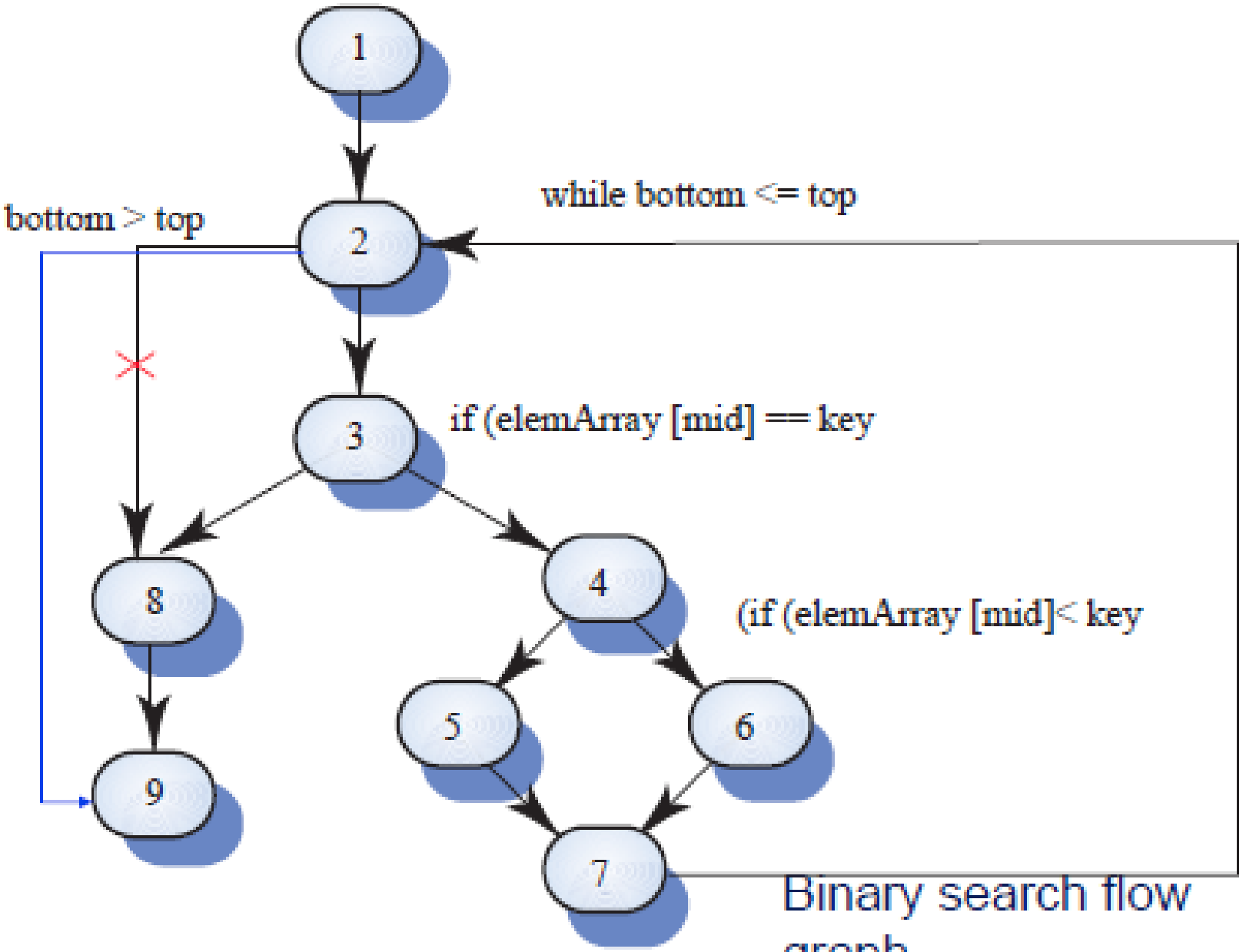
- path testing: 프로그램에 있는 모든 경로가 적어도 한번은 실행되도록 테스트 케이스들의 집합을 구성하는 것.
- path testing의 시작점은 프로그램 판단(decision)들을 나타내는 노드와 제어 흐름을 나타내는 간선(arcs)들을 보여주는 프로그램 흐름 그래프임.
- Statements with conditions are therefore nodes in the flow graph

Program flow graphs

- 프로그램 제어 흐름을 보여줌. Each branch is shown as a separate path and loops are shown by arrows looping back to the loop condition node
- Used as a basis for computing the cyclomatic complexity
- Cyclomatic complexity = Number of edges - Number of nodes +2

Cyclomatic complexity

- 모든 제어 문장들을 시험하기 위한 테스트들의 수는 cyclomatic complexity와 같다.
- Cyclomatic complexity는 프로그램에 있는 조건문의 수 + 1 이다.
- Useful if used with care. Does not imply adequacy of testing.
- Although all paths are executed, all combinations of paths are not executed



Binary search flow graph

Independent paths

- 1, 2, 3, 8, 9
- 1, 2, 3, 4, 6, 7, 2
- 1, 2, 3, 4, 5, 7, 2
- 1, 2, 3, 4, 6, 7, 2, 8, 9
- 테스트 케이스들은 이런 모든 경로들이 실행될 수 있도록 유도되어야 한다.
- 동적 프로그램 분석기가 경로들이 시험되었는지 검사하기 위해 사용될 수 있다.