Outline

- Introduction
- Basic Concepts
- The Testing Process
- Types of Testing
- Testing Philosophy
- Summary

The Problem

- Given a program, how to check if the program behaves correctly, i.e., as designed?

Software Testing

- A dynamic approach to ensuring software correctness
- Involves sampling the input space, running the test object, and observing the runtime behavior
- Among the most widely used approaches in practice
  - Labor intensive, and often consumes more than 50% of development cost

Static Analysis

- Reason about the behavior of a program based on the source code, i.e., without executing the program
- Question: How do you compare the two approaches?

Testing vs Static Analysis

- Static analysis
  - Like a super compiler (syntax vs semantics), in general very fast, and does not require test cases
  - False positives and negatives
- Testing
  - No false positives, what you see is what you get
  - In general, more time consuming, require test execution setup, depending on the quality of test cases

Outline

- Introduction
- Basic Concepts
- The Testing Process
- Types of Testing
- Testing Philosophy
- Summary

Fault, Error & Failure (1)

- Fault : A static defect in the software
  - Incorrect instructions, missing instructions, extra instructions
- Error : An incorrect internal state that is the manifestation of some fault
- Failure : External, incorrect behavior with respect to the requirements or other description of the expected behavior

Fault, Error, and Failure (2)

```
public static int numZero (int[] x) {
  // effects: if x == null throw NullPointerException
  //          else return the number of occurrences of 0 in x
  int count = 0;
  for (int i = 1; i < x.length; i ++) {
    if (x[i] == 0) {
      count ++;
    }
  }
  return count;
}
```

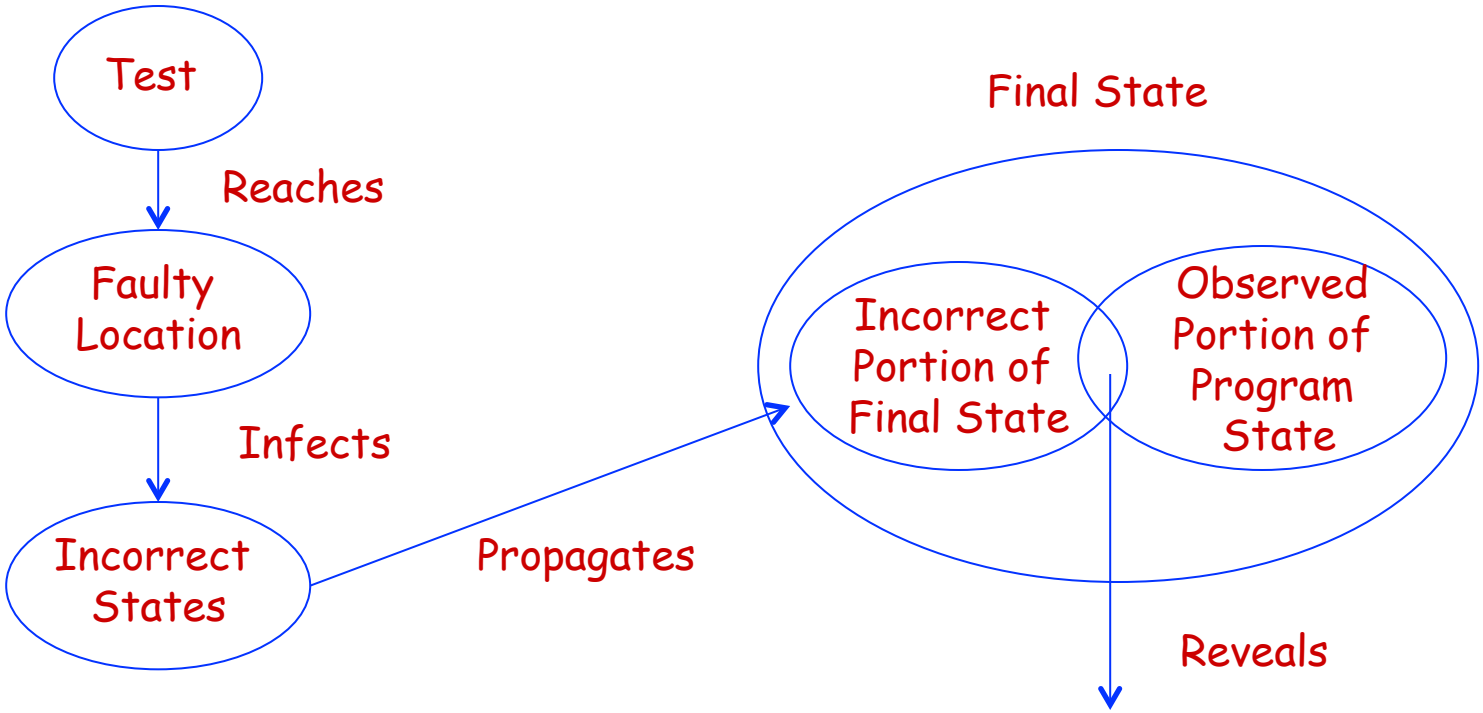
Fault, Error, and Failure (3)

- The state of numZero consists of the values of the variables x, count, i, and the program counter.
- Consider what happens with numZero ([2, 7, 0]) and numZero ([0, 7, 2])?

Fault & Failure Model

- Four conditions must be satisfied for a failure to be observed
  - Reachability : The location or locations in the program that contain the fault must be reached
  - Infection : The state of the program must be incorrect
  - Propagation : The infected state must cause the final state of the program to be incorrect
  - Revealability: The incorrect portion of the final state must be observed from outside

Fault & Failure Model (2)



Static Analysis & Dynamic Testing

- Static Analysis: Testing without executing the program.
  - Code walkthrough & inspection, and various static analysis techniques.
- Dynamic Testing: Testing by executing the program with real inputs
  - Static information can often be used to make dynamic testing more efficient.

Test Case

- Test data: data values to be input to the program under test
- Expected result: the outcome expected to be produced by the program under test

Testing & Debugging

- Testing: Finding inputs that cause the software to fail
- Debugging: The process of finding a fault given a failure
- In practice, testing & debugging are often performed in a cyclic fashion

Verification & Validation

- Verification: Ensure compliance of a software product with its design
- Validation: Ensure compliance of a software product with intended usage
- Question: Which task, validation or verification, is more difficult to perform?
  - Build the right product vs build the product right

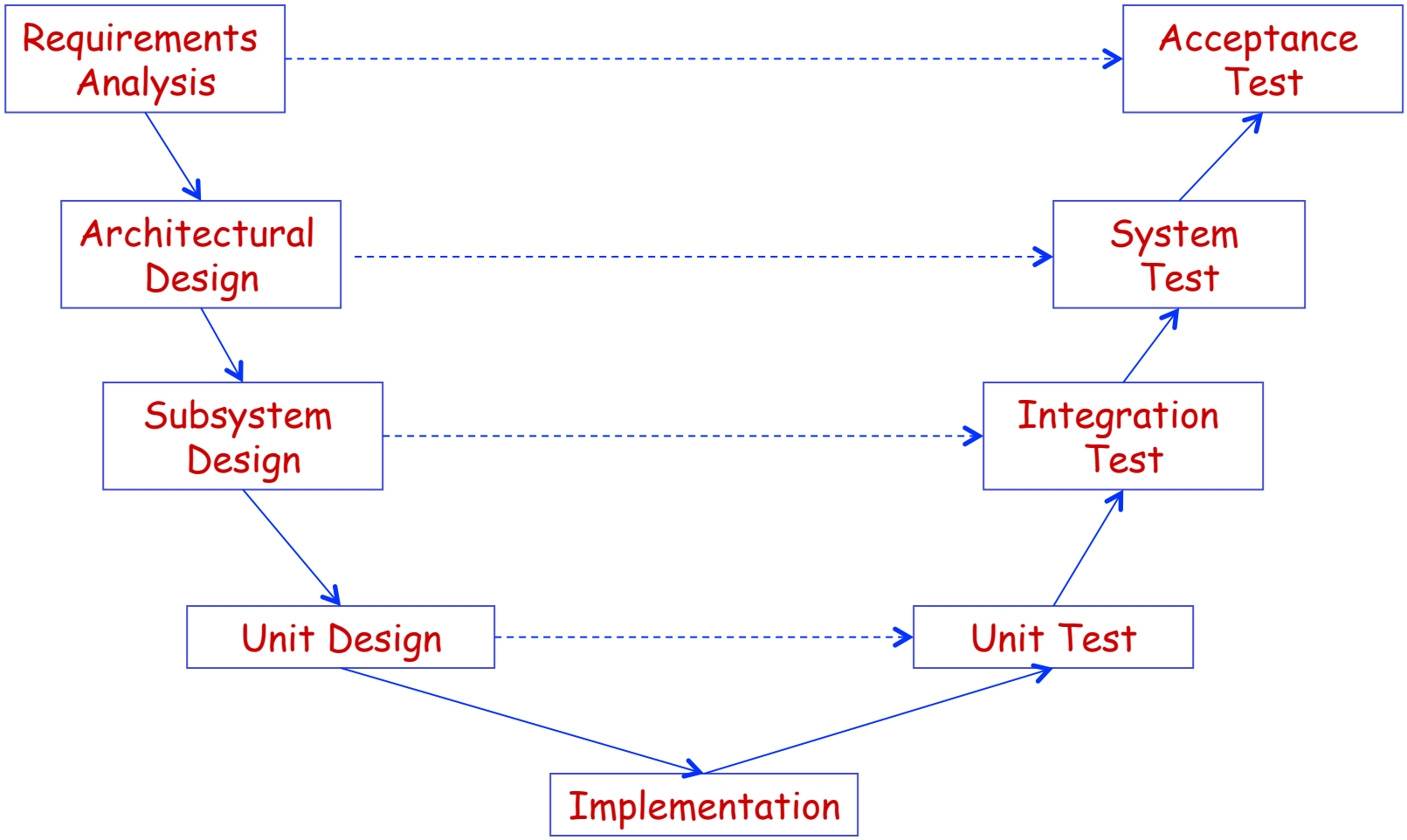
Testability

- The degree to which a system or component facilitates the establishment of test criteria and the performance of tests to determine whether those criteria have been met
- The more complex an application, the lower the testability, i.e., the more effort required to test it
- Design for testability: Software should be designed in a way such that it can be easily tested

Outline

- Introduction
- Basic Concepts
- The Testing Process
- Types of Testing
- Testing Philosophy
- Summary

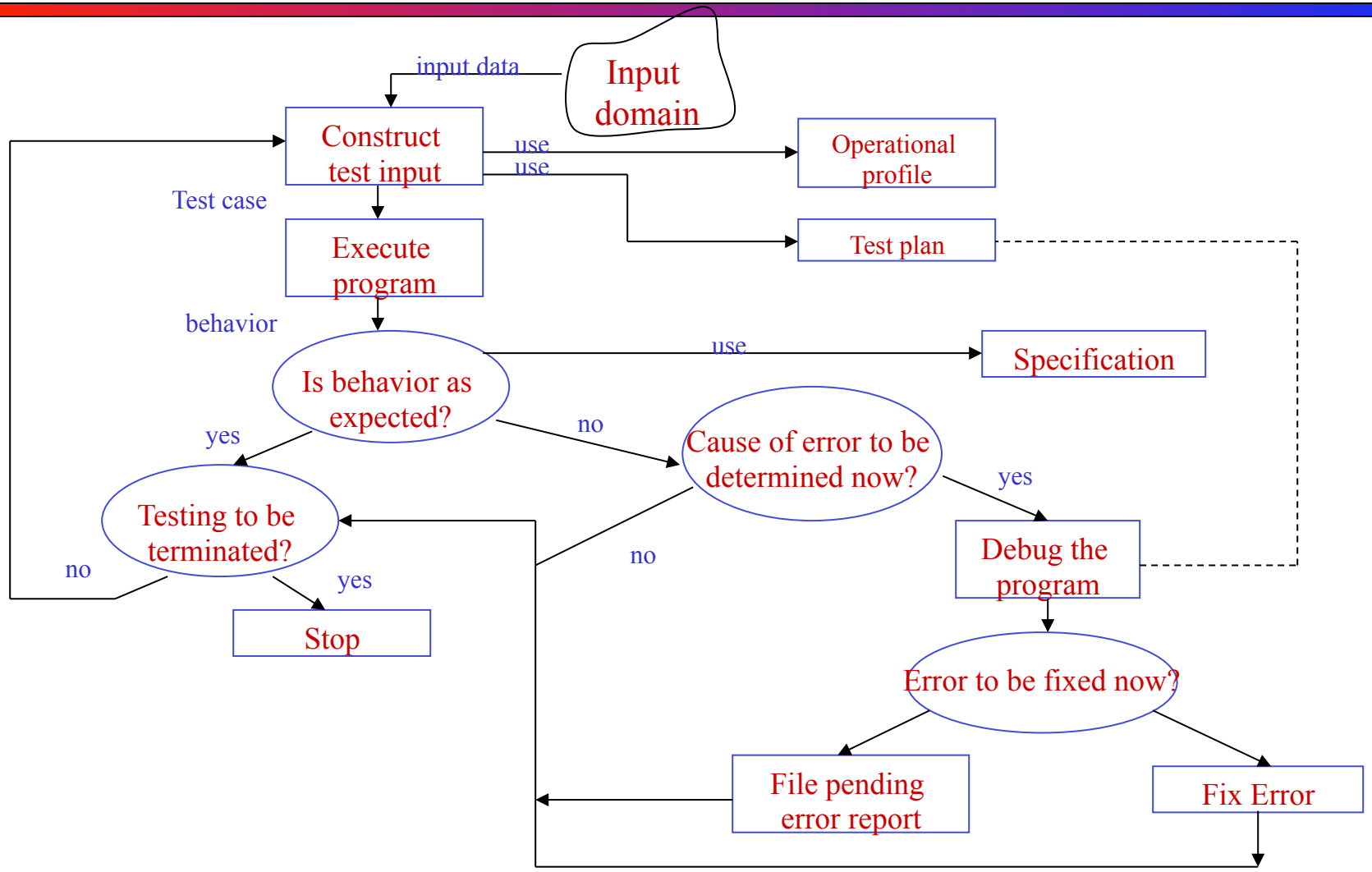
The V Model


Requirements Analysis
Acceptance Test
Architectural Design
System Test
Subsystem Design
Integration Test
Unit Design
Unit Test
Implementation

The Process

- Preparing a test plan
- Constructing test data
- Specifying program behavior
- Executing the program
- Evaluating program behavior
- Construction of automated oracles

Test & Debug Cycle



An Example

- Program sort:
  - Given a sequence of integers, this program sorts the integers in either ascending or descending order.
  - The order is determined by an input request character "A" for ascending or "D" for descending.

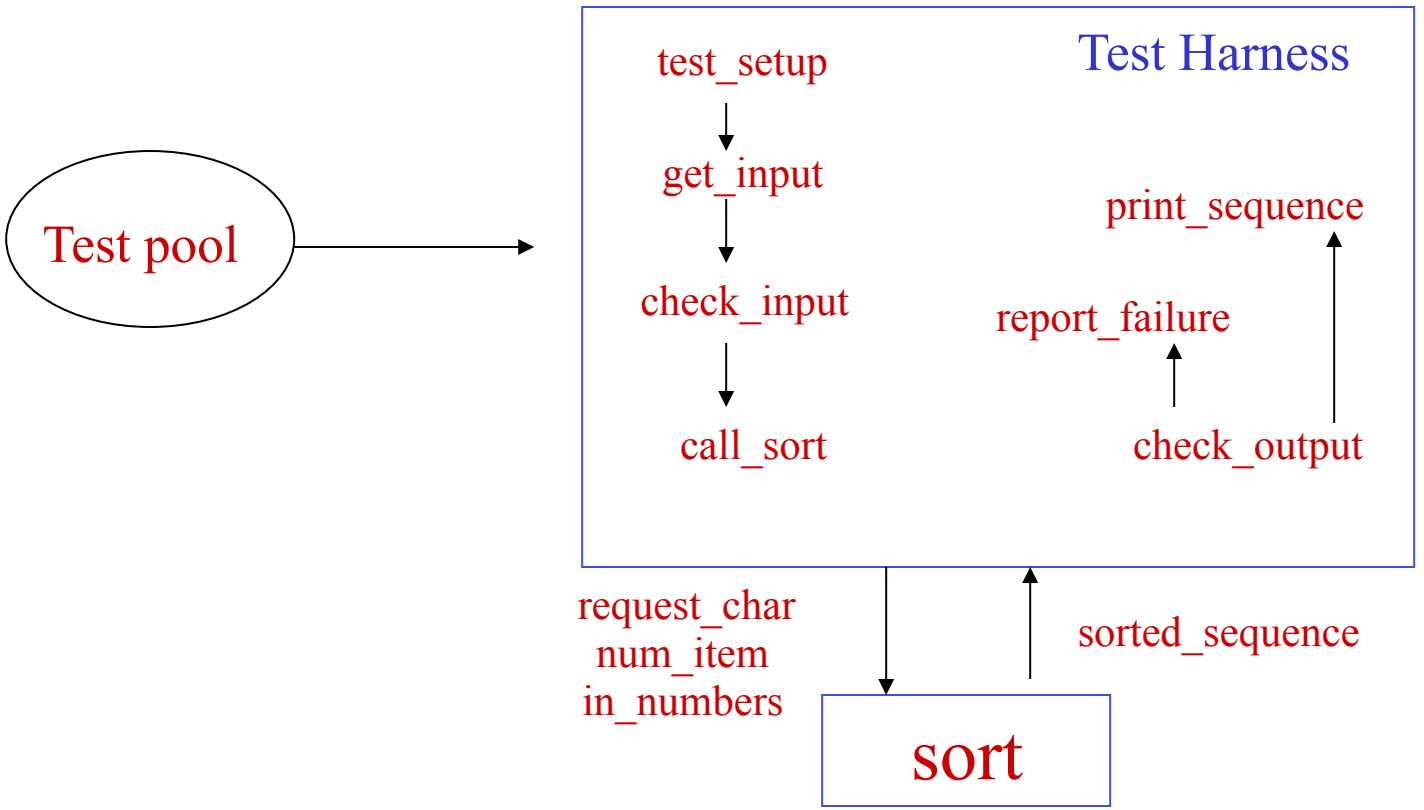
Test plan

1. Execute the program on at least two input sequences, one with "A" and the other with "D" as request characters
2. Execute the program on an empty input sequence
3. Test the program for robustness against invalid inputs such as "R" typed in as the request character
4. All failures of the test program should be reported

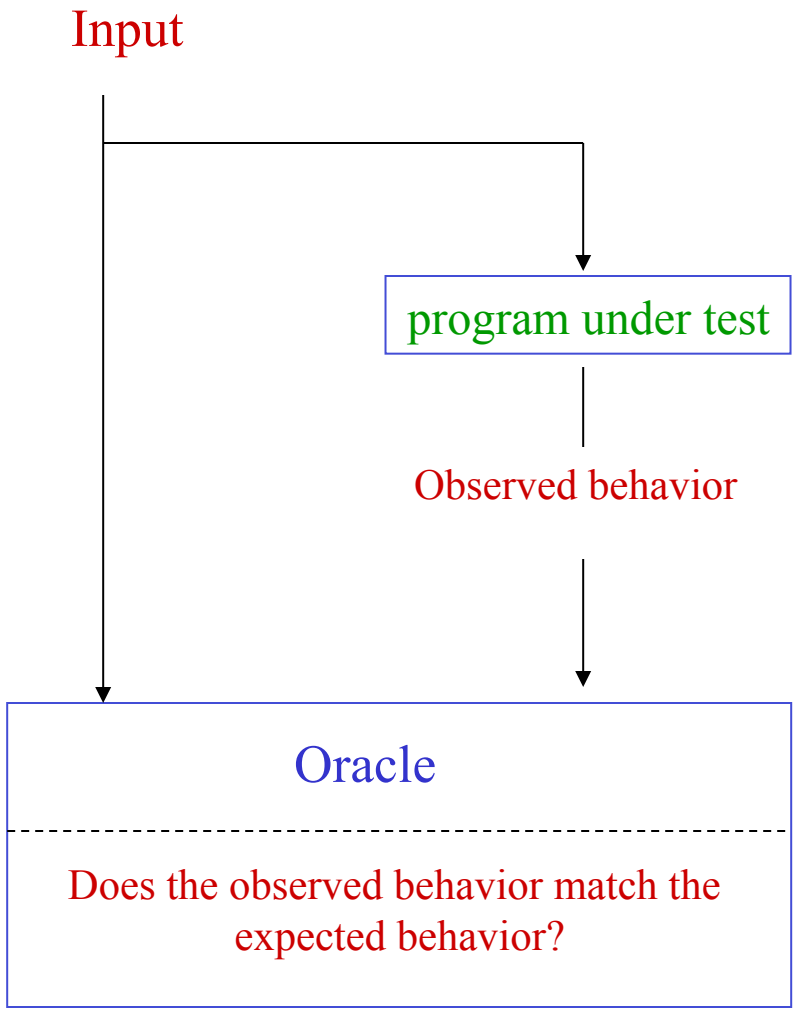
Test Data

- Test case 1:
  - Test data: <“A” 12 -29 32 .>
  - Expected output: -29 12 32
- Test case 2:
  - Test data: <“D” 12 -29 32 .>
  - Expected output: 32 12 -29
- Test case 3:
  - Test data: <“A” .>
  - Expected output: No input to be sorted in ascending order.
- Test case 4:
  - Test data: <“D” .>
  - Expected output: No input to be sorted in ascending order.
- Test case 5:
  - Test data: <“R” 3 17 .>
  - Expected output: Invalid request character
- Test case 6:
  - Test data: <“A” c 17.>
  - Expected output: Invalid number

Test Harness



Test Oracle


Input
program under test
Observed behavior
Oracle
Does the observed behavior match the expected behavior?

Outline

- Introduction
- Basic Concepts
- The Testing Process
- Types of Testing
- Testing Philosophy
- Summary

Classifier C1: Source of Test Generation

- Black-box testing: Tests are generated from informally or formally specified requirements
  - Does not require access to source code
  - Boundary-value analysis, equivalence partitioning, random testing, pairwise testing
- White-box testing: Tests are generated from source code.
  - Must have access to source code
  - Structural testing, path testing, data flow testing

Classifier C2: Life Cycle Phases

PHASE	TECHNIQUE
Coding	Unit Testing
Integration	Integration Testing
System Integration	System Testing
Maintenance	Regression Testing
Postsystem, pre-release	Beta Testing

Classifier C3: Goal Directed Testing

GOAL	TECHNIQUE
Features	Functional Testing
Security	Security Testing
Invalid inputs	Robustness Testing
Vulnerabilities	Penetration Testing
Performance	Performance Testing
Compatibility	Compatibility Testing

Classifier C4: Artifact Under Test

ARTIFACT	TECHNIQUE
OO Software	OO Testing
Web applications	Web Testing
Real-Time software	Real-time testing
Concurrent software	Concurrency testing
Database applications	Database testing

Outline

- Introduction
- Basic Concepts
- The Testing Process
- Types of Testing
- Testing Philosophy
- Summary

Philosophy

- Level 0: Testing is the same as debugging.
- Level 1: Testing aims to show correctness.
- Level 2: Testing aims to show the program under test doesn't work.
- Level 3: Testing aims to reduce the risk of using the software.
- Level 4: Testing is a mental discipline that helps develop higher quality software.

Outline

- Introduction
- Basic Concepts
- The Testing Process
- Types of Testing
- Testing Philosophy
- Summary

Summary

- Quality is the central concern of software engineering.
- Testing is one of the most widely used approaches to ensuring software quality.
- Testing consists of test generation, test execution, and test evaluation.
- Testing can show the presence of failures, but not their absence.