Outline

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

Software Engineering

- Software has become pervasive in modern society
  - Directly contributes to quality of life
  - Malfunctions cost billions of dollars every year, and have severe consequences in a safe-critical environment
- How to build quality software, especially for large-scale development?
  - Requirements, design, coding, testing, maintenance, configuration, documentation, deployment, and etc.

Software Testing

- Steve Ballmer, 2006: "Let's acknowledge a sad truth about software: any code of significant scope and power will have bugs in it."
- A dynamic approach to detecting software faults
  - Alternatively, static analysis can be performed, which is however often intractable
- Involves sampling the input space, running the test object, and observing the runtime behavior
- The single most widely used approach in practice
  - Labor intensive, and often consumes more than 50% of development cost

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;
}
```

UTA

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])?

UTA

Fault & Failure Model

- Three 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 propagate to cause some output of the program to be incorrect

UTA

Static & Dynamic Testing

- Static Testing: 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.

UTA

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

UTA

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

UTA

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?

UTA

Quality Attributes

- Static attributes refer to the actual code and related documentation
  - Well-structured, maintainable, and testable code
  - Correct and complete documentation
- Dynamic attributes refer to the behavior of the application while in use
  - Reliability, correctness, completeness, consistency, usability, and performance

UTA

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

UTA

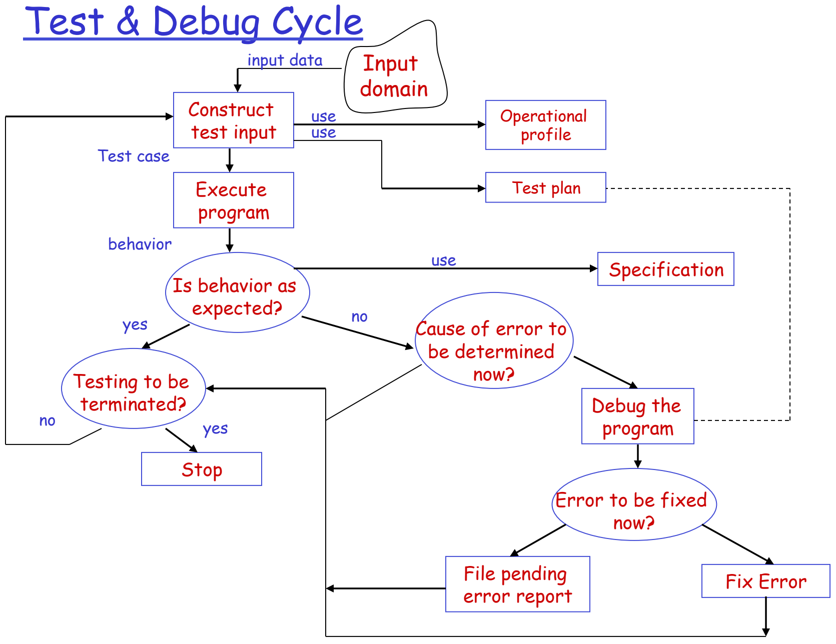
Outline

UTA

The Process

- Preparing a test plan
- Constructing test data
- Executing the program
- Specifying program behavior
- 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.

UTA

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

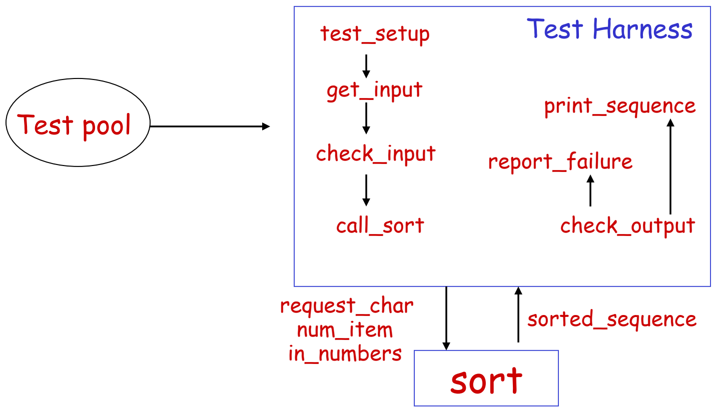
UTA

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

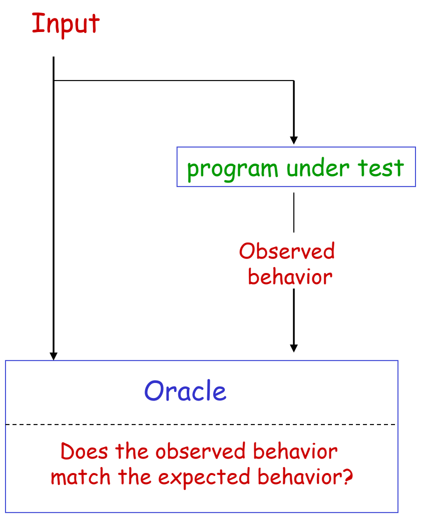
UTA

Test Harness



UTA

Test Oracle



UTA

Outline

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

UTA

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

UTA

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

UTA

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

UTA

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

UTA

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 the single most widely used approach 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.