Software Testing

Testing: Our Experiences

When to Stop?

A Real Testing Example

Automated Testing

Automated Testing
Testing the New Version

Regression Testing

What is Testing?
- Process of determining whether a task has been correctly carried out [Schach '96]
- Goals of testing
  - Reveal Faults
    - Correctness
    - Reliability
    - Usability
    - Robustness
    - Performance

Types of Testing
- Execution-based Testing
- Non-execution based Testing
- Discussion

Execution-based Testing
- Generating and Executing Test Cases on the Software
- Types of Execution-based Testing
  - Testing to Specifications
    - Black-box Testing
  - Testing to Code
    - Glass-box (White-box) Testing

Black-box Testing
- Discussion: MAC/ATM Machine Example
  - Specs
    - Cannot withdraw more than $300
    - Cannot withdraw more than your account balance
  
  ![Diagram of MAC/ATM Machine Example]
White-box Testing

- Example
  
x: 1..1000;
  1. INPUT-FROM-USER(x);
  2. If (x <= 300) {
        2.1. INPUT-FROM-FILE(BALANCE);
        2.2. If (x <= BALANCE)
                2.2.1. GiveMoney x;
        2.3. else Print "You don't have $x in your account!!";
    3. else Print "You cannot withdraw more than $300";
  4. Eject Card;

Discussion

- Which is superior?
- Each technique has its strengths – Use both

Determining Adequacy

- Statement coverage
- Branch coverage
- Path coverage
- All-def-use-path coverage

Surprise Quiz

- Determine test cases so that each print statement is executed at least once

```
input(x);
if (x < 100)
  print "Line 1";
else {
  if (x < 50) print "Line 2"
  else print "Line 3";
}
```

Non-execution Based

- Walkthroughs
  - Manual simulation by team leader
- Inspections
  - Developer narrates the reading
- Key Idea
  - Review by a team of experts: Syntax checker?
- Code Readings
- Formal Verification of Correctness
  - Very Expensive
  - Justified in Critical Applications
- Semi-formal: Some Assertions

Simulation

- Integration with system hardware is central to the design
- Model the external hardware
- Model the interface

- Examples
- Discussion
Boundary-value Analysis

- Partition the program domain into input classes
- Choose test data that lies both inside each input class and at the boundary of each class
- Select input that causes output at each class boundary and within each class
- Also known as stress testing

Testing Approaches

- Top-down
- Bottom-up
- Big Bang
- Unit testing
- Integration testing
- Stubs
- System testing

Mutation Testing

- Errors are introduced in the program to produce “mutants”
- Run test suite on all mutants and the original program

Test Case Generation

- Test Input to the Software
- Some researchers/authors also define the test case to contain the expected output for the test input

Category-partition Method

- Key idea
  - Method for creating functional test suites
  - Role of test engineer
    - Analyze the system specification
    - Write a series of formal test specifications
  - Automatic generator
    - Produces test descriptions

Steps

- Decompose the functional specification into functional units
  - Characteristics of functional units
    - They can be tested independently
    - Examples
      - A top-level user command
      - Or a function
  - Decomposition may require several stages
  - Similar to high-level decomposition done by software designers
    - May be reused, although independent decomposition is recommended
Steps

- Examine each functional unit
  - Identify parameters
    - Explicit input to the functional unit
  - Environmental conditions
    - Characteristics of the system's state
- Test Cases
  - Specific values of parameters
  - And environmental conditions

Test Cases

- "Test cases are chosen to maximize chances of finding errors"
- For each parameter & environmental condition
  - Find categories
    - Major property or characteristic
    - Examples
      - Browsers, Operating System, array size
      - For each category
        - Find choices
          - Examples: (IE 5.0, IE 4.5, Netscape 7.0), (Windows NT, Linux), (100, 0, -1)

Steps

- Develop “Formal Test Specification” for each functional unit
  - List of categories
  - Lists of choices within each category
- Constraints
  - Automatically produces a set of “test frames”
    - Consists of a set of choices

AI Planning Method

- Key Idea
  - Input to Command-driven software is a sequence of commands
  - The sequence is like a plan
- Scenario to test
  - Initial state
  - Goal state

Example

- VCR command-line software
- Commands
  - Rewind
    - If at the end of tape
  - Play
    - If fully rewound
  - Eject
    - If at the end of tape
  - Load
    - If VCR has no tape

Preconditions & Effects

- Rewind
  - Precondition: If at end of tape
  - Effects: At beginning of tape
- Play
  - Precondition: If at beginning of tape
  - Effects: At end of tape
- Eject
  - Precondition: If at end of tape
  - Effects: VCR has no tape
- Load
  - Precondition: If VCR has no tape
  - Effects: VCR has tape
Preconditions & Effects

- **Rewind**
  - Precondition: \(\neg\text{end\_of\_tape}\)
  - Effects: \(\neg\text{end\_of\_tape}\)

- **Play**
  - Precondition: \(\text{end\_of\_tape}\)
  - Effects: \(\neg\text{end\_of\_tape}\)

- **Eject**
  - Precondition: \(\text{end\_of\_tape}\)
  - Effects: \(\neg\text{has\_tape}\)

- **Load**
  - Precondition: \(\neg\text{has\_tape}\)
  - Effects: \(\text{has\_tape}\)

Initial and Goal States

- **Initial State**
  - \(\text{end\_of\_tape}\)

- **Goal State**
  - \(\neg\text{end\_of\_tape}\)

- **Plan?**
  - **Rewind**

Initial and Goal States

- **Initial State**
  - \(\neg\text{end\_of\_tape} \& \text{has\_tape}\)

- **Goal State**
  - \(\neg\text{has\_tape}\)

- **Plan?**
  - **Rewind**

Test Coverage & Adequacy

- **How much testing is enough?**
- **When to stop testing**
- **Test data selection criteria**
- **Test data adequacy criteria**
  - **Stopping rule**
  - **Degree of adequacy**
- **Test coverage criteria**
- **Objective measurement of test quality**

Preliminaries

- **Test data selection**
  - What test cases
- **Test data adequacy criteria**
  - When to stop testing
- **Examples**
  - Statement Coverage
  - Branch coverage
  - Def-use coverage
  - Path coverage

Goodenough & Gerhart ['75]

- **What is a software test adequacy criterion**
  - Predicate that defines "what properties of a program must be exercised to constitute a thorough test", i.e., one whose successful execution implies no errors in a tested program
Uses of test adequacy

- Objectives of testing
- In terms that can be measured
  - For example branch coverage
- Two levels of testing
  - First as a stopping rule
  - Then as a guideline for additional test cases

Categories of Criteria

- Specification based
  - All-combination criterion
  - Each-choice-used criterion
- Program based
  - Statement
  - Branch
- Note that in both the above types, the correctness of the output must be checked against the specifications

Others

- Random testing
- Statistical testing
- Interface based