CS 390 – Lecture 16
Execution-Based Testing

- Organizations spend up to 50% of their software budget on testing
  - But delivered software is frequently unreliable
- Dijkstra (1972)
  - “Program testing can be a very effective way to show the presence of bugs, but it is hopelessly inadequate for showing their absence”

What Should Be Tested?

- Definition of execution-based testing
  - “The process of inferring certain behavioral properties of the product based, in part, on the results of executing the product in a known environment with selected inputs”
- This definition has troubling implications

What Should Be Tested? (2)

- “Inference”
  - We have a fault report, the source code, and — often — nothing else
- “Known environment”
  - We never can really know our environment
- “Selected inputs”
  - Sometimes we cannot provide the inputs we want
  - Simulation is needed

Seven Principles of Software Testing

- Chief architect at Eiffel Software
  - Design by Contract ™, spec is part of the code that can be enforced by compiler
- Professor of Software Engineering at ETH Zurich
  - AutoTest, Automatic test case generation for contract-equipped object-oriented software

Seven Principles of Software Testing (2)

- Execution-based testing is the dominant verification technique
- Need to understand the scope and limitations of testing and perform it right

Principle 1: Definition

- To test a program is to try to make it fail.
  - Does not and cannot determine quality (Wikipedia notwithstanding)
  - Any number of test runs is a miniscule number of possible cases
  - At best can say quality improves if previously discovered fault is removed
  - Focuses testing
Principle 2: Tests versus specs
- Tests are no substitute for specifications.
  - Test suite cannot cover all possible cases.
  - Specs can be used to generate tests, perhaps automatically.

Principle 3: Regression testing
- Any failed execution must yield a test case, to remain a permanent part of the project's test suite.
  - Previously corrected faults often reappear, leading to regression testing.

Principle 4: Applying oracles
- Determining success or failure of tests must be an automatic process.
  - Examining results individually does not scale up.
  - Does not say what is the form of oracles.

Principle 4 (stronger variant): Contracts as oracles
- Oracles should be part of the program text, as contracts. Determining test success or failure should be an automatic process consisting of monitoring contract satisfaction during execution.

Principle 5: Manual and automatic test cases
- An effective testing process must include both manually and automatically produced test cases.
  - Manual tests are good at depth; developers understand problem domain and data structure.
  - Automatic tests are good at breadth; they try many values including extremes that humans might miss.

Principle 6: Empirical assessment of testing strategies
- Evaluate any testing strategy, however attractive in principle, through objective assessment using explicit criteria in a reproducible testing process.
  - Random testing often outperforms supposedly smart ideas.
**Principle 7: Assessment criteria**

- A testing strategy’s most important property is the number of faults it uncovers as a function of time.
  - Other measures used
    - Number of tests to first failure
    - Number of tests
    - Testing time needed to uncover faults
  - Test coverage
  - Fault count function: \( fc(t) \)
    - Assess strategy on a base with known faults
    - Part of reliability model

**What Should Be Tested? (3)**

- We need to test correctness (of course), and also
  - Utility
  - Reliability
  - Robustness, and
  - Performance

**Utility**

- The extent to which the product meets the user’s needs
  - Examples:
    - Ease of use
    - Useful functions
    - Cost effectiveness

**Reliability**

- A measure of the frequency and criticality of failure
  - Mean time between failures
  - Mean time to repair
  - Time (and cost) to repair the results of a failure

**Robustness**

- A function of
  - The range of operating conditions
  - The possibility of unacceptable results with valid input
  - The effect of invalid input

**Performance**

- The extent to which space and time constraints are met
  - Real-time software is characterized by hard real-time constraints
  - If data are lost because the system is too slow
    - There is no way to recover those data
Who Should Perform Execution-Based Testing?

- Programming is constructive
- Testing is destructive
  - A successful test finds a fault
- So, programmers should not test their own code artifacts
  - Another reason to automate test generation as much as possible

Who Should Perform Execution-Based Testing? (2)

- Solution:
  - The programmer does informal testing
  - The SQA group then does systematic testing
  - The programmer debugs the module
- All test cases must be
  - Planned beforehand, including the expected output, and
  - Retained afterwards

When Testing Stops

- Only when the product has been irrevocably discarded