Test Case Selection and Adequacy Criteria

Learning objectives

- Be able to define test adequacy criteria, and explain their limitations
- Be able to explain the terminology of test selection and adequacy
- Be able to use the sources of information commonly used to define adequacy criteria
- Be able to use test selection and adequacy criteria

Adequacy: We can’t get what we want

- What we would like:
  - A real way of measuring effective testing
    If the system system passes an adequate suite of test cases, then it must be correct (or dependable)
- But that’s impossible!
  - Adequacy of test suites, in the sense above, is provably undecidable.
- So we’ll have to settle on weaker proxies for adequacy
  - Design rules to highlight inadequacy of test suites

Adequacy Criteria as Design Rules

- Many design disciplines employ *design rules*
  - E.g.: “traces (on a chip, on a circuit board) must be at least ___ wide and separated by at least ___”
  - “The roof must have a pitch of at least ____ to shed snow”
  - “Interstate highways must not have a grade greater than 6% without special review and approval”
- Design rules do not guarantee good designs
  - Good design depends on talented, creative, disciplined designers; design rules help them avoid or spot flaws
  - Test design is no different

Practical (in)Adequacy Criteria

- Criteria that identify inadequacies in test suites.
  - Examples
  - *if the specification describes different treatment in two cases, but the test suite does not check that the two cases are in fact treated differently, we may conclude that the test suite is inadequate to guard against faults in the program logic.*
  - *If no test in the test suite executes a particular program statement, the test suite is inadequate to guard against faults in that statement.*
- If a test suite fails to satisfy some criterion, the obligation that has not been satisfied may provide some useful information about improving the test suite.
- If a test suite satisfies all the obligations by all the criteria, we do not know definitively that it is an effective test suite, but we have some evidence of its thoroughness.

Analogy: Building Codes

- Building codes are sets of design rules
  - Maximum span between beams in ceiling, floor, and walls; acceptable materials; wiring insulation; ...
  - Minimum standards, subject to judgment of building inspector who interprets the code
- You wouldn’t buy a house just because it’s “up to code”
  - It could be ugly, badly designed, inadequate for your needs
- But you might avoid a house because it isn’t
  - Building codes are inadequacy criteria, like practical test “adequacy” criteria
- Building codes are sometimes too restrictive and are relaxed for some buildings that then have to conform to different requirements.

Some useful terminology

- **Test case**: a set of inputs, execution conditions, and a pass/fail criterion.
- **Test case specification**: a requirement to be satisfied by one or more test cases.
- **Test obligation**: a partial test case specification, requiring some property deemed important to thorough testing.
- **Test suite**: a set of test cases.
- **Test** or **test execution**: the activity of executing test cases and evaluating their results.
- **Adequacy criterion**: a predicate that is true (satisfied) or false (not satisfied) of a ⟨program, test suite⟩ pair.

Where do test obligations come from?

- Functional (black box, specification-based): from software specifications
  - Example: If spec requires robust recovery from power failure, test obligations should include simulated power failure
- Structural (white or glass box): from code
  - Example: Traverse each program loop one or more times.
- Model-based: from model of system
  - Models used in specification or design, or derived from code
  - Example: Exercise all transitions in communication protocol model
- Fault-based: from hypothesized faults (common bugs)
  - Example: Check for buffer overflow handling (common vulnerability) by testing on very large inputs

Adequacy criteria

- Adequacy criterion = set of test obligations
- A test suite satisfies an adequacy criterion if
  - all the tests succeed (pass)
  - every test obligation in the criterion is satisfied by at least one of the test cases in the test suite.
  - Example:

    the statement coverage adequacy criterion is satisfied by test suite S for program P if each executable statement in P is executed by at least one test case in S, and the outcome of each test execution was "pass".

Satisfiability

- Sometimes *no* test suite can satisfy a criterion for a given program
  - Example: Defensive programming style includes “can’t happen” sanity checks

```
if (z < 0) {
    throw new LogicError(
      “z must be positive here!”)
  }
```

  No test suite can satisfy statement coverage for this program (if it’s correct)
  - BUT, one might try to get this to happen…

Coping with Unsatisfiability

- Approach A: exclude any unsatisfiable obligation from the criterion.
  - Example: modify statement coverage to require execution only of statements that can be executed.
  - But we can’t know for sure which are executable!
- Approach B: measure the extent to which a test suite approaches an adequacy criterion.
  - Example: if a test suite satisfies 85 of 100 obligations, we have reached 85% *coverage*.
    - Terms: An adequacy criterion is satisfied or not, a coverage measure is the fraction of satisfied obligations

Coverage: Useful or Harmful?

- Measuring coverage (% of satisfied test obligations) can be a useful indicator ...
  - Of progress toward a thorough test suite, of trouble spots requiring more attention
- ... or a dangerous seduction
  - Coverage is only a proxy for thoroughness or adequacy
  - It’s easy to improve coverage without improving a test suite (much easier than designing good test cases)
  - The only measure that really matters is (cost-)effectiveness

Comparing Criteria

- Can we distinguish stronger from weaker adequacy criteria?
- Empirical approach: Study the effectiveness of different approaches to testing in industrial practice
  - What we really care about, but ...
  - Depends on the setting; may not generalize from one organization or project to another
- Analytical approach: Describe conditions under which one adequacy criterion is provably stronger than another
  - Stronger = gives stronger guarantees
  - One piece of the overall “effectiveness” question

The *subsumes* relation

Test adequacy criterion A subsumes test adequacy criterion B iff, for every program P, every test suite satisfying A with respect to P also satisfies B with respect to P.

- Example:

  Exercising all program branches (branch coverage) *subsumes* exercising all program statements (you see this in the structural testing chapter)

- A common analytical comparison of closely related criteria
  - Useful for working from easier to harder levels of coverage, but not a direct indication of quality

Uses of Adequacy Criteria

- Test selection approaches
  - Guidance in devising a thorough test suite
    - Example: A specification-based criterion may suggest test cases covering representative combinations of values
- Revealing missing tests
  - Post hoc analysis: What might I have missed with this test suite?
- Often in combination
  - Example: Design test suite from specifications, then use structural criterion (e.g., coverage of all branches) to highlight missed logic

Summary

- Adequacy criteria provide a way to define a notion of “thoroughness” in a test suite
  - But they don’t offer guarantees; more like *design rules to highlight inadequacy*
- Defined in terms of “covering” some information
  - Derived from many sources: Specs, code, models, ...
- May be used for selection as well as measurement
  - With caution! An aid to thoughtful test design, not a substitute