Test Case Selection and Adequacy Criteria

Test adequacy

- A key problem in software testing is **selecting** and **evaluating** test cases
- Software that has passed a thorough set of systematic tests vs. software that has been only superficially or unsystematically tested.
- Each software module should be required undergo **thorough**, systematic testing before being incorporated into the main product.
  - What do we mean by "thorough testing"?
  - What is the criterion by which we can judge the **adequacy** of a suite of tests that a software module has passed?

Adequacy: We can't get what we want

- What we would like:
  - A real way of measuring effective testing
    If the 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

Important 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 pattern for test case specification
  - usually derived from an adequacy criterion (see below)
  - 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.
- **(Test) adequacy criterion**: a predicate that is true (satisfied) or false (not satisfied) of a ⟨program, test suite⟩ pair.

Practical (in)Adequacy Criteria

- Criteria that identify **inadequacies** in test suites, e.g.:
  - *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.*

Practical (in)Adequacy Criteria

- 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 **still do not** know definitively that it is an effective test suite, but we have **some evidence of its thoroughness.**

Test case vs. Test case specification

- Suppose, for example, we are testing a program that sorts a sequence of words.
  - "The input is two or more words" would be a test case specification,
    - Q: What would be the test cases satisfying the case specification?
  - How's about the test case specification "the input is two or more words" and the test case specification "the input contains a mix of lower- and upper-case alphabetic characters."?
    - Q: What would be the test cases satisfying the case specification?

Test case specification

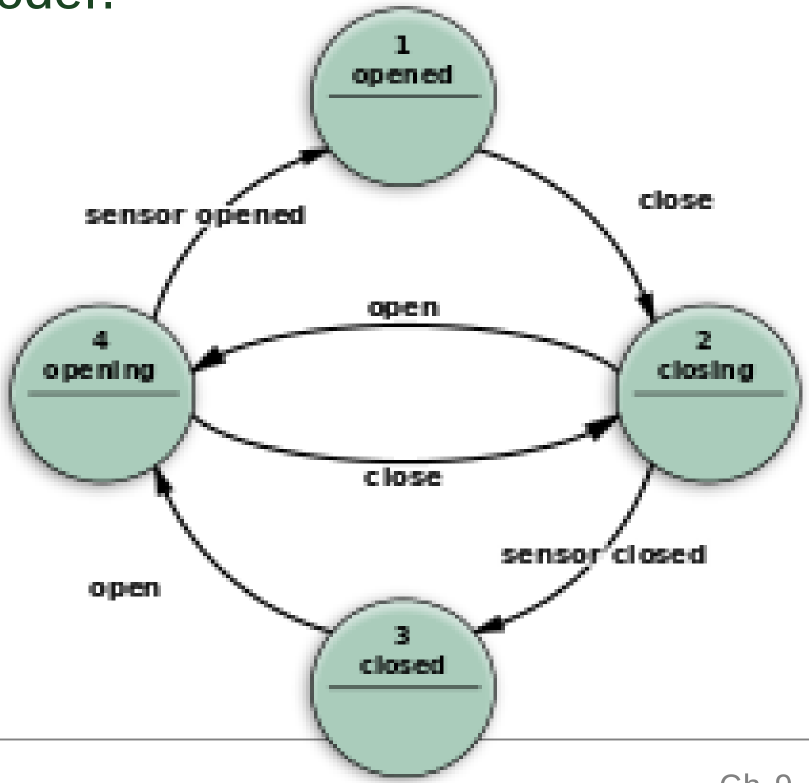
- Characteristics of the input are not the only thing that might be mentioned in a test case specification.
- A complete test case specification includes **pass/fail criteria** for judging test execution
- It may include requirements, drawn from any of several sources of information, such as:
  - system, program, and module interface specifications;
  - source code or detailed design of the program itself;
  - and records of faults encountered in other software systems.

Test case specification (cont.)

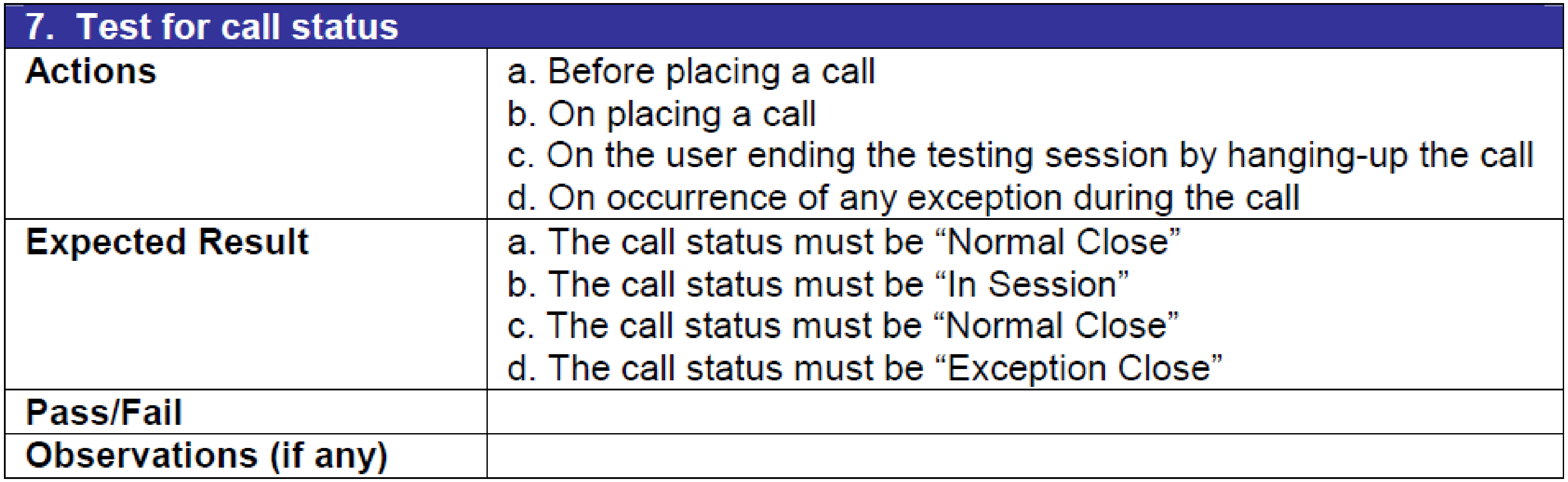
- Test specifications drawn from system, program, and module interface specifications often describe program **inputs**,
- But they can just as well specify any observable behavior that could appear in specifications.
- For example, the specification of a database system might require certain kinds of robust failure recovery in case of power loss,
  - and test specifications might therefore require removing system power at certain critical points in processing.

Test case specification (cont.)

- If a specification describes **inputs and outputs**, a test specification could prescribe aspects of the **input**, the **output**, or both.
- If the specification is modeled as an extended finite state machine, it might require executions corresponding to particular transitions or paths in the state-machine model.



Test case example



7. Test for call status	
Actions	a. Before placing a call b. On placing a call c. On the user ending the testing session by hanging-up the call d. On occurrence of any exception during the call
Expected Result	a. The call status must be “Normal Close” b. The call status must be “In Session” c. The call status must be “Normal Close” d. The call status must be “Exception Close”
Pass/Fail	
Observations (if any)	

IEEE 829 Test Case Specification Template

- **Test Case Specification Identifier**
- **Test Items**
  - Describe features and conditions tested
- **Input Specifications**
  - Data Names
  - Ordering
  - Values (with tolerances or generation procedures)
  - States
  - Timing
- **Output Specifications**
  - Data Names
  - Ordering
  - Values (with tolerances or generation procedures)
  - States
  - Timing
- **Environmental Needs**
  - Hardware
  - Software
  - Other
- **Special Procedural Requirements**
- **Inter-Case Dependencies**

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

Test obligations for white-box testing

- Items to keep in mind during white box testing
  - **Statement coverage**: each statement is executed at least once
  - **Decision (branch) coverage**: each statement ….; each decision takes on all possible outcome at least once
  - **Condition coverage**: each statement…; each decision …; each condition in a decision takes on all possible outputs at least once
  - **Path coverage**: each statement …; all possible combinations of condition outcomes in each decision occur at least once

Quiz

- Identify a test suite that satisfies:
  - Statement coverage
  - Branch coverage
  - Path coverage

```
premium = 500;
if (age < 25) && (gender ==
   male) && !married
{
   premium += 500;
}
else
{
   if (married || (gender ==
   female)
        premium -= 200;
   if (age > 45) && (age <
   65)
                premium -= 100;
}
```

Exercise

- Discuss and identify a number of possible test obligations for this program.

```
/**
  * Remove/collapse multiple spaces.
  *
  * @param String string to remove multiple spaces from.
  * @return String
  */
public static String collapseSpaces(String argStr)
{
    char last = argStr.charAt(0);
    StringBuffer argBuf = new StringBuffer();

    for (int cIdx = 0 ; cIdx < argStr.length(); cIdx++)
    {
        char ch = argStr.charAt(cIdx);
        if (ch != ' ' || last != ' ')
        {
            argBuf.append(ch);
            last = ch;
        }
    }

    return argBuf.toString();
}
```

Figure 9.1: A Java method for collapsing sequences of blanks, excerpted from the StringUtils class of Velocity version 1.3.1, an Apache Jakarta project. © Apache Group, used by permission.

Adequacy criteria

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

    Question: How does test suite S for program P satisfy the statement coverage adequacy criterion?

    Answer: Each executable statement in P is executed by at least one test case in S, and the outcome of each test execution was "pass".

Coping with Unsatisfiability

- Sometimes *no* test suite can satisfy a criterion for a given program
- 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: We say that 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
  - E.g., of progress towards 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 (designing more trivial test cases is 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

Comparing Criteria (cont.)

- Analytic comparisons of the strength of test coverage depends on a precise definition of what it means for one criterion to be “stronger” or “more effective” than another.
- A test suite TA that does not include all the test cases of another test suite TB may fail revealing the potential failure exposed by the test cases that are in TB but not in TA.
- *Question: How to make test suite TA to be stronger than another suite TB?*

Comparing Criteria (cont.)

- Many different test suites might satisfy the same coverage criterion.
- To compare criteria, then, we consider all the possible ways of satisfying the criteria.
- If every test suite that satisfies some criterion A is a superset of some test suite that satisfies criterion B, or equivalently, every suite that satisfies A also satisfies B, then we can say that A **"subsumes"** B.

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.

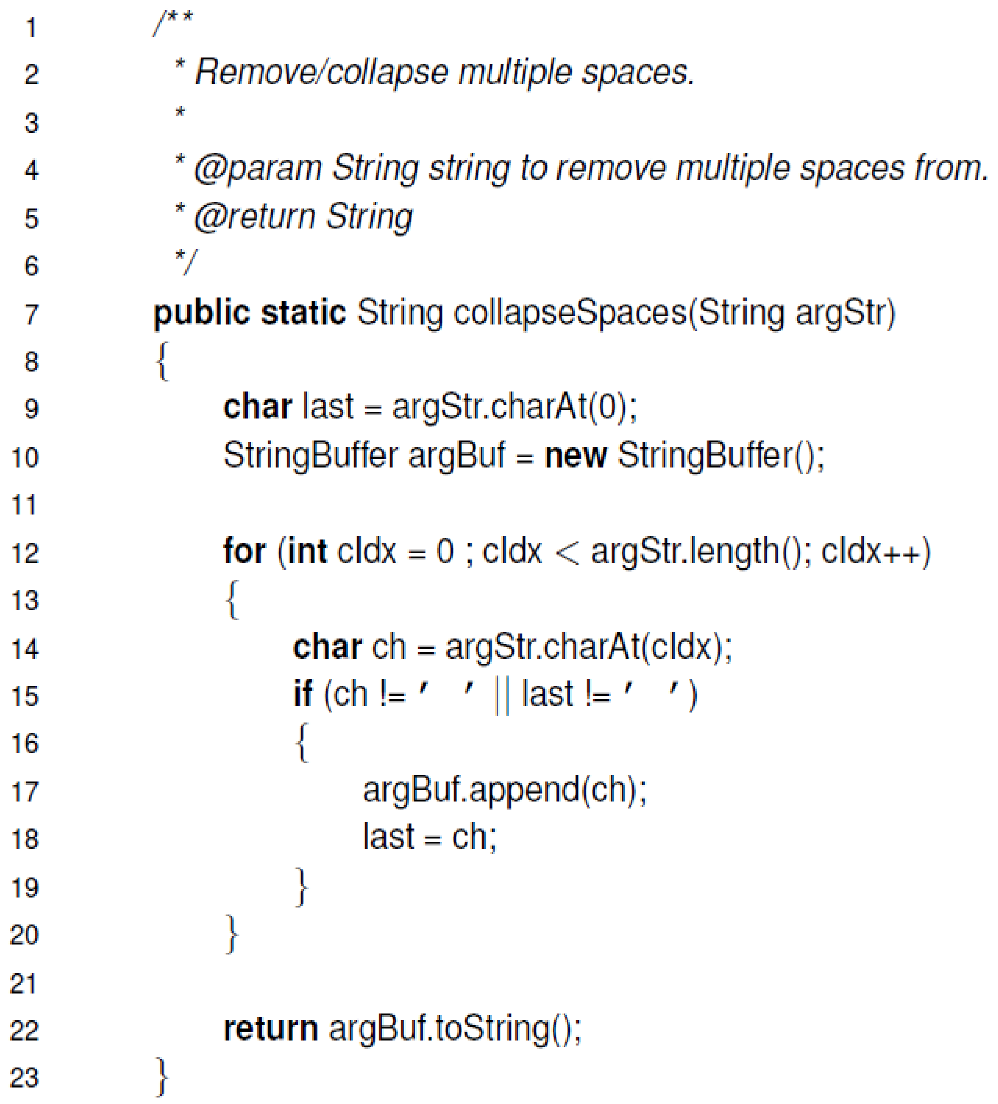
- Example:
  - Exercising all program branches (branch coverage) *subsumes* exercising all program statements

The *subsumes* relation (cont.)

- In this case, if we satisfy criterion C1, there is no point in measuring adequacy with respect to C2 if C1 subsumes C2.
- For example, a structural criterion that requires exploring all outcomes of conditional branches subsumes statement coverage.
- Likewise, a specification-based criterion that requires use of a set of possible values for attribute A and, independently, for attribute B, will be subsumed by a criterion that requires all combinations of those values.

Exercise

- Which adequacy criterion imposed by the test obligation you identified earlier subsumes the others?



```java
/**
  * Remove/collapse multiple spaces.
  *
  * @param String string to remove multiple spaces from.
  * @return String
  */
public static String collapseSpaces(String argStr)
{
    char last = argStr.charAt(0);
    StringBuffer argBuf = new StringBuffer();

    for (int cIdx = 0 ; cIdx < argStr.length(); cIdx++)
    {
        char ch = argStr.charAt(cIdx);
        if (ch != ' ' || last != ' ')
        {
            argBuf.append(ch);
            last = ch;
        }
    }

    return argBuf.toString();
}
```

Figure 9.1: A Java method for collapsing sequences of blanks, excerpted from the StringUtils class of Velocity version 1.3.1, an Apache Jakarta project. © Apache Group, used by permission.

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

Quiz

- Suppose test suite A satisfies adequacy criterion C1. Test suite B satisfies adequacy criterion C2, and C2 subsumes C1. Can we be certain that faults revealed by A will also be revealed by B? Why?

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