

CMPE 276 Software Engineering

Lecture 7 Testing

Luca de Alfaro, CMPE 276, Lecture 7

1

Administrativa

- Any objections to moving the project presentations next Tue to after the class?
 - It would help to introduce Verisoft sooner, and have more time to play with it.

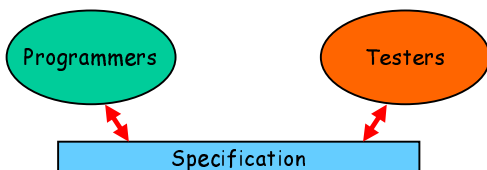
Luca de Alfaro, CMPE 276, Lecture 7

2

What is Testing?

Testing: checking that a program meets its specification

- No specification, no testing: only debugging by the programmers themselves.



Luca de Alfaro, CMPE 276, Lecture 7

3

Testing Phases

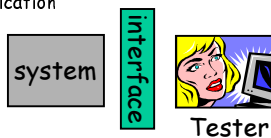
- Model the software's environment
- Select the test scenarios
- Running and evaluating test scenarios
- Measuring test progress

Luca de Alfaro, CMPE 276, Lecture 7

4

Testing requires interfaces

- Testing occurs along an interface of the system:
 - User interface
 - File interface
 - API, procedural interface
 - Communication interface
- Testing: checking that the system respects its interface specification



Luca de Alfaro, CMPE 276, Lecture 7

5

How do we test?

We must somehow:

- select inputs for the system that trigger relevant behaviors;
- observe the system response;
- check whether it satisfies the specification;
 - if it does not:
 - minimize example
 - communicate problem to programmers
- measure the progress of the testing effort (coverage).

We will examine these aspects in turns.

Luca de Alfaro, CMPE 276, Lecture 7

6

Checking that it satisfies the specification

- Checking that the specification is met sometimes entails further inputs to the system
 - Example: in a database, first insert, then search, etc...
- Testing also clarifies the specification
 - Very common to find ambiguities at this stage!
 - **Testers**: the program fails 50% of the tests!
 - **Programmers**: the tests are wrong!
- Good testing is difficult, and requires high-quality human work
 - Problem: often testers are those that "don't qualify to be programmers"

Luca de Alfaro, CMPE 276, Lecture 7

7

What to check for?

- Test case
 - Add a child to Mary Brown's record*
- Version 1
 - Check that Ms. Brown's # of children is one more
- Version 2
 - Also check Mr. Brown's # of children
- Version 3
 - Check that no one else's child counts changed

Luca de Alfaro, CMPE 276, Lecture 7

8

Selecting inputs - manually

- Some ideas are simple:
 - Test boundary values
 - Check for ± 1 type of errors, and the like
- But the most effective tool is still a clever tester:
 - Understands the problem
 - Has some idea of the technique used to solve it
 - Uses the knowledge to test the boundary cases for the technique, and to anticipate possible programming errors

Luca de Alfaro, CMPE 276, Lecture 7

9

Producing a test script

- It is a sequence of actions that should lead to an expected result.
- It can be used in regression testing.
- How precise to make it?
 - Precise test cases can be automated, but are difficult to maintain.
- Example: "draw a green rectangle" vs.
 - click on rectangle
 - draw a rectangle
 - select the palette option
 - click on "More colors"...

Luca de Alfaro, CMPE 276, Lecture 7

10

Other test generation techniques

- Look at traces occurring while using past versions of the product, or similar products
 - TCAS, the collision avoidance algorithm for planes, was/is checked in this fashion.
 - Especially useful for software that interacts with a physical environment (e.g., embedded software)
 - Can be also used for web servers, etc.

Luca de Alfaro, CMPE 276, Lecture 7

11

Random test generation

- Generate random input, and feed it to the program.
- Requires a probability space from which to draw the samples
 - Easy for strings, numbers, etc.
 - Difficult for text, data structures (random trees, ...), things that must be parsed (random grammar must be written)...

Luca de Alfaro, CMPE 276, Lecture 7

12

Random test generation (cont.)

Advantages:

- Can do a lot of testing unattended.
- Tries many "crazy" combinations that may not be tried by a human tester
 - 50% of Unix routines used to fail under random input; now "reduced" to 30%
 - Chief problems: buffer overruns, printf, pointers
 - Example: `csch !0%8f`
- Good for "shallow" programs

Luca de Alfaro, CMPE 276, Lecture 7

13

Random testing (cont.)

Disadvantages:

- Not good for uncovering faults that are due to correlated action
 - Example (in a database):
 - Add a child to the mother
 - Subtract a child to the father
 - The probability is too small, humans are better at thinking of these corner cases

When most used:

- Hardware verification
- Some GUI testing, web testing, ...

Luca de Alfaro, CMPE 276, Lecture 7

14

Can we do better?

- Instead of random tests, can we steer tests for "full coverage"?
- Can we generate tests from the property we are trying to show?

Luca de Alfaro, CMPE 276, Lecture 7

15

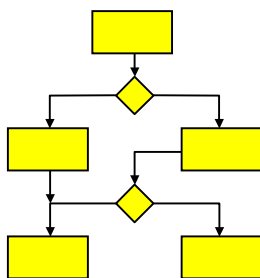
Yes, we can do better

- We can try to steer tests to the sections of the code that have changed
 - See next
- We can try to steer tests to full coverage
- We can try to generate tests from the property
- The latter two are more complex, and are among the techniques known as **Formal Verification**
 - We will study how to do this later in the course.

Luca de Alfaro, CMPE 276, Lecture 7

16

Directing tests to modified code

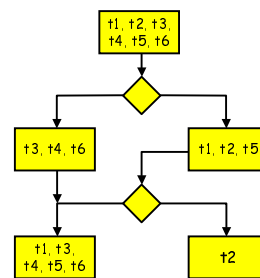


- Consider the flow chart of the program
 - Every region is a region that cannot be left, nor entered, internally

Luca de Alfaro, CMPE 276, Lecture 7

17

Directing tests to modified code



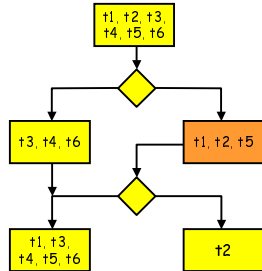
- Mark, for each region, the set of tests that reach it.

- Given a region, synthesizing a test that reaches it, is a hard problem!

Luca de Alfaro, CMPE 276, Lecture 7

18

Directing tests to modified code



- When a region changes, you know which tests to re-run.
 - Powerful technique
 - Automatized by various tools

Luca de Alfaro, CMPE 276, Lecture 7

19

Automating Testing

- Record a test, and play it back later.
- Enables regression testing.
- Test recording is usually very fragile
 - Breaks if environment changes anything
 - E.g., location, background color of textbox
- More generally, automation tools cannot generalize
 - They literally record exactly what happened
 - If anything changes, the test breaks
- A hidden strength of manual testing
 - Because people are doing the tests, ability to adapt tests to slightly modified situations is built-in

Luca de Alfaro, CMPE 276, Lecture 7

20

Regression testing

- Can be done also in conjunction with nightly builds
- What to do when the specification changes, and large numbers of tests become invalidated?
 - Repair them?
 - Drop them?

Luca de Alfaro, CMPE 276, Lecture 7

21

Assertions

- Insert in the code
`assert (<predicate>)`
- Example: `assert (len >= 0 && head != NULL)`
- Finds the error earlier - hugely important!
 - More likely to find it
 - More likely to know where it was caused
- It also checks that the programmer has a good understanding of the code
 - And of how it will be used
- Programmers don't use this nearly enough!

Luca de Alfaro, CMPE 276, Lecture 7

22

Unit Testing

- Done by programmers on one unit at a time
- More language support would be desirable
 - Difficult to build complex data structures
 - Algebraic data types are not catching on in mainstream languages!
 - Difficult to visualize them.

Luca de Alfaro, CMPE 276, Lecture 7

23

Code reviews

- They have been shown to find 70%-90% of the bugs.
- The code is distributed in advance, and people are requested to read it in advance.
- Expensive, but well worth it.
- Limit code to a few pages.
- Added benefit: people now understand the code!
 - and check the quality of the documentation

Luca de Alfaro, CMPE 276, Lecture 7

24

Testing software configurations

- How to test that some software works on linux, Windows, ...?
- Problems:
 - Is the OS initialized to a known state (repeatability)?
 - How can we quickly test many different platforms?
- A sample solution:
 - Using the OS on a virtual machine (as in VMWare).

Luca de Alfaro, CMPE 276, Lecture 7

25

Two nasty categories of bugs

- Concurrency
- Real-time

They are:

- Difficult to think about.
- Difficult to design
 - How much concurrency? Do we need a lock? Can this cause deadlocks?
 - What the worst-case execution time (WCET)? What timing relations can occur?
- Difficult to stimulate bugs (timing coincidences, scheduling peculiarities).
- Difficult to reproduce, when a bug arises.

Many formal approaches are geared to these bugs.

Luca de Alfaro, CMPE 276, Lecture 7

26

Design for testability

- Avoid unpredictable results
 - No unnecessary non-deterministic behavior
- Each piece of functionality in one place only
 - If two pieces of code have overlapping functionality, one often masks bugs of the other
- Design in self-checking
 - At appropriate places have system check its own work
 - Asserts
 - May require adding some redundancy to the code

[from: Aiken]

Luca de Alfaro, CMPE 276, Lecture 7

27

Design for testability

- Avoid system state
 - Retains nothing across units of work
 - A transaction, a session, etc.
 - System returns to well-known state after each task is complete
 - Easiest system to test
- Minimize interactions between features
 - Number of interactions can easily grow huge
 - Rich breeding ground for bugs
- Have a test interface

[from: Aiken]

Luca de Alfaro, CMPE 276, Lecture 7

28

Coverage

Measures of test coverage have many uses:

- They can point out to untested regions of code, and to weaknesses in the test suite.
- They help to decide when to terminate the test phase, and ship the product (goal: 85% coverage).
- They quantify the progress in testing
- And they can even be used to steer testing to modified regions of code! (see later)

Unfortunately:

- They are all heuristics, and they are difficult to compare one to the other.

Luca de Alfaro, CMPE 276, Lecture 7

29

Statement Coverage

- Statement coverage: which statements have been executed during testing. Aka: line coverage
- Advantages:
 - Can be applied directly to object code
 - Does not require any code analysis
 - If bugs are spread homogeneously, then the % coverage is a good indicator.
 - Easy, cheap to obtain.

Luca de Alfaro, CMPE 276, Lecture 7

30

Statement Coverage

Drawbacks:

- Insensitive to logical tests:

```
if (x == 0) p* = 7;
```
- It does not test that both ways are taken
- Expensive (lots of bookkeeping)
- Bugs are more related to logical decisions than to straight-line code (is it true?).

Luca de Alfaro, CMPE 276, Lecture 7

31

Branch Coverage

- Aka: decision coverage.
- Keeps track of which branches of if-then-elses, while and for-loops, etc, are taken.
- Advantages:
 - Simple, requires little code analysis
 - Covers the decision portion of the program
- Drawbacks:
 - Insensitive to logical expression evaluation:

```
if (q1 || q2 && q3) { ...
```

Luca de Alfaro, CMPE 276, Lecture 7

32

Condition Coverage

- Checks whether all truth-values for each of the q1, q2, q3 have occurred:

```
if (q1 || q2 && q3) { ...
```

- Advantages:
 - More accurate test analysis
 - Not very complicated to implement (compiler help needed)
- But it does not check the combination of these values.

Luca de Alfaro, CMPE 276, Lecture 7

33

Multiple Condition Coverage

- Checks whether all combinations of truth-values for each of the q1, q2, q3 have occurred:

```
if (q1 || q2 && q3) { ...
```

- Advantages:
 - Very accurate test analysis
- Drawbacks:
 - Expensive!
- But can be helped by randomization techniques

Luca de Alfaro, CMPE 276, Lecture 7

34

Path Coverage

- Path = sequence of branches.
- Have all possible paths in a function been taken?
- This is the history-dependent equivalent of branch coverage.
- For loops: explore to a bounded depth.
- Disadvantages:
 - Your opinion here...

Luca de Alfaro, CMPE 276, Lecture 7

35

Path Coverage

- Path = sequence of branches.
- Have all possible paths in a function been taken?
- This is the history-dependent equivalent of branch coverage.
- For loops: explore to a bounded depth.
- Disadvantages:
 - The number of paths is exponential in the number of branches.
 - Not all paths can be followed, due to data dependencies!
 - Difficult therefore to establish when complete coverage is achieved.

Luca de Alfaro, CMPE 276, Lecture 7

36

Data Flow Coverage

- Similar to path coverage, but remember the "history" only as long in the past as there is some data that can still be influenced by it.
 - Lots of research has been done.
 - I will instead concentrate on formal verification (this topic borders on it).

Luca de Alfaro, CMPE 276, Lecture 7

37

Loop Coverage

- Did you execute loops:
 - 0 iterations?
 - 1 iteration?
 - > 1 iterations?

Luca de Alfaro, CMPE 276, Lecture 7

38

Race Coverage

- Eraser! - remind me to tell you about this.

Luca de Alfaro, CMPE 276, Lecture 7

39

Mutation Analysis

- Inject errors in the code, producing variant implementations, and check whether the test suite is able to distinguish between the original, and the modified, variants.
- Idea: the injected bugs are representatives of the real ones.
 - But is this true? Difficult to inject.
- Drawbacks:
 - Your opinions here...

Luca de Alfaro, CMPE 276, Lecture 7

40

Mutation Analysis

- Inject errors in the code, producing variant implementations, and check whether the test suite is able to distinguish between the original, and the modified, variants.
- Idea: the injected bugs are representatives of the real ones.
 - But is this true? Difficult to inject.
- Drawbacks:
 - Not all injections lead to errors!
 - This undermines the metric
 - Difficult to establish if a modification was not found because it is not an error, or because the tests were insufficient.

Luca de Alfaro, CMPE 276, Lecture 7

41

Comparing Coverage Measures

- Decision coverage includes statement coverage, since exercising every branch must lead to exercising every statement.
- Path coverage includes decision coverage.

Luca de Alfaro, CMPE 276, Lecture 7

42

What to do once a bug is found?

- **Minimization:**
 - Find a minimal sequence of actions that reproduces the error.
 - A wonderful topic for research work, but in practice mostly done by hand.
- **Communicate to the programmers**
 - A "socially sensitive" job...

Luca de Alfaro, CMPE 276, Lecture 7

43

When to decide to ship?

A mix of:

- **Coverage**
- **Bug trends** (has the rate at which we find bugs decreased enough?)
- Each one of these two criteria, in isolation, would not make sense!

Luca de Alfaro, CMPE 276, Lecture 7

44

Reading

- Start to read the papers on Verisoft, on the web page. Verisoft will soon be presented in class, but it helps if you have some idea already.

Luca de Alfaro, CMPE 276, Lecture 7

45