CSC323 2010S *Software Design*

Class 21: Testing, Revisited

Held: Tuesday, April 20, 2010

Summary: We return to the issue of testing, considering kinds of testing one might employ. We also visit two views of handling errors, programming by contract and defensive programming.

Related Pages:

- EBoard.

Notes:

- One more chapter of HFOOAD! (Due Thursday.)
- So, what did you think of the great effort imparted on the minor joke?
- Buy one get one free EBook sale at ORA until April 30. Code BYEBK.

Overview:

- Models of Testing.
- Programming-by-Contract.
- Defensive Programming.

Models of Testing

- We've seen a variety of models of testing in this book and in our own experience.
  - We can write *output-oriented non-interactive tests*, in which we print out results from a variety of calls to a procedure and evaluate correctness by looking at the result.
  - We can write *output-oriented interactive tests*, in which we type an expression and evaluate correctness by looking at the result. (These are easiest to do in an interactive environment, but it is possible to write your own interactive tester.)
  - We can write *self-checked non-interactive tests*, in which the test looks at the result and verifies that it's correct.
  - We can write *self-checked non-interactive tests in a testing framework*, in which the framework provides a harness for the tests.
- Why might we use ech kind of test?

Dealing with Potential Errors

- Note that almost every operation or procedure call has implicit or explicit preconditions.
  - Looking up a value by key in a dictionary/table/map.
  - Indexing into an array.
  -
- One important part of thinking about the architecture of a system is dealing with those preconditions.

    - Who, if anyone, is responsible for checking them?
    - What should happen if the preconditions are not met?
- Who is responsible for checking those preconditions?
- That's at the heart of the PbC vs. Defensive Programming debate.
    - I'd argue that the authors got it a bit wrong.

Programming by Contract

- What is your understanding of programming by contract? (PbC)
    - Think back to the six P's of CSC 151, if that helps.
    - *Note that the book takes a very informal approach to PbC.*
- In what other ways you might represent the contract?

Defensive Programming

- What is your understanding of defensive programming?
- Why might we care about defensive programming?
- As one of you asked, what are the benefits of forcing the client to deal with an exception?
    - Note that "forcing the client to deal with an exception" is a core strategy in Java.
- The authors present defensive programming in a somewhat negative light. Why might someone practice defensive programming?

An example of non-defensive programming in C.

```
int
main (int argc, char **argv)
{
  int i;
  FILE *outfile = fopen ("/tmp/params");
  for (i = 0; i < argc; i++)
    {
      fprintf (outfile, "%d: %s\n", i, argv[i]);
    }
  fclose (outfile);
  exit (0);
} /* main */
```

How might you make it defensive?

An Independent Exercise

- Suppose you wanted to diagnose what is going on with a slow file server and you did not have root permissions on that file server.
- What would you do?
- Exercise: Write a program that queries all the clients of a file server to see what processes are currently running on all of those clients.

Copyright © 2010 Samuel A. Rebelsky. This work is licensed under a Creative Commons Attribution-NonCommercial 2.5 License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc/2.5/ or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.