Software Testing, Quality Assurance & Maintenance—Lecture 11

Patrick Lam
University of Waterloo

January 28, 2015

Course roadmap

- ✓ Introduction (faults etc)
- ✓ Graph coverage
- □ Testing Concurrent Programs (wrap-up)
- □ Tools

Assertions

statements about the program state that are true, e.g.

doubly-linked list property: `prev` is the inverse of `next`

Assertions in This Course's Context

We also use assertions in unit tests
to say what's supposed to be true.

Plus, last time, we asserted about a lock being held upon entry to a method. Or not.

Preconditions, Postconditions

More generally, we can express

- what is supposed to be true upon entry & exit from a method.

We saw this code in Linux.

```
/* LOCKING: caller. */
void ata_dev_select(...) { ...}
```

= an assertion that the lock is held upon entry.

Assume/Guarantee Reasoning

Why would you use preconditions and postconditions?

When reasoning about the callee:
assume the precondition holds upon entry;

When reasoning about the caller:
guarantee the precondition holds before the call.

The reverse holds about the postcondition.

What aComment actually does

- extract locking-related annotations from code;
- extract locking-related annotations from comments;
- propagate annotations to callers.

Part I

Tools

OS X Mavericks goto fail bug

```
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
    goto fail;  /* MISTAKE! THIS LINE SHOULD NOT BE HERE */
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;
    err = sslRawVerify(...);

fail:
    return err;
```

Source and writeup: goto fail

The bug:

`opensource.apple.com/source/Security/Security-55471/libsecurity_ssl/lib/sslKeyExchange.c`

No bug:

`www.opensource.apple.com/source/Security/Security-55179.13/libsecurity_ssl/lib/sslKeyExchange.c`

Writeup:

`nakedsecurity.sophos.com/2014/02/24/anatomy-of-a-goto-fail-apples-ssl-bug-explained-plus-an-unofficial-patch/`

Detecting goto fail

In retrospect, a number of options:

- compiler `-Wunreachable-code` option
- PC-Lint:warning 539: Did not expect positive indentation
- PVS-Studio:V640: Logic does not match formatting

[slide credit: contents from Sye Van De Veen]

Testing and Static Analysis Tools

The continuum:

- manual testing;
- running a JUnit test suite, manually generated;
- running automatically-generated tests;
- running static analysis tools.

We'll examine several points on this continuum today.

Tools in Practice

More on this later (March), thanks to guest lecturer.

- Coverity—static analysis—used by 900+ companies, including BlackBerry, Mozilla, etc.
- Microsoft requires Windows device drivers to pass the Static Driver Verifier for certification.

Tools you can Download

We'll survey some tools for:

- Java
- C/C++

(or, use e.g. a programming language with stronger types!)

findbugs.sourceforge.net

Open-source static bytecode analyzer for Java.
Finds bug patterns:

- off-by-one;
- null pointer dereference;
- ignored `read()` return value;
- ignored return value (immutable classes);
- uninitialized read in constructor;
- and more. . .

FindBugs gives some false positives.

Some techniques to avoid them:

`patricklam.ca/papers/14.msr.saa.pdf`

Java Path Finder (JPF), NASA

Key Idea:

Implement a Java Virtual Machine,
but explore many thread interleavings,
looking for concurrency bugs.

"JPF is an explicit state software model checker for Java™ bytecode."

JPF can also search for

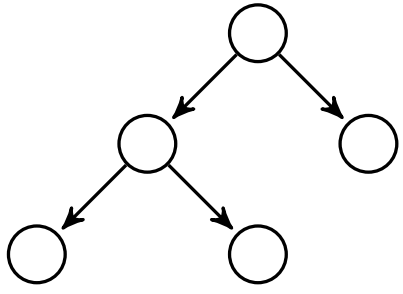
- deadlocks and unhandled exceptions
  (`NullPointerException`, `AssertionError`);
- race conditions, heap bounds checks.

`javapathfinder.sourceforge.net`

Korat (University of Illinois)

Korat

Key Idea:
Generate Java objects from a representation invariant specification written as a Java method.



Binary Tree!

One characteristic of a Binary Tree:

- left & right pointers don't refer to same node.

Korat repOk for Binary Tree

```
boolean repOk() {
  if (root == null) return size == 0;            // empty tree has size 0
  Set visited = new HashSet(); visited.add(root);
  List workList = new LinkedList(); workList.add(root);
  while (!workList.isEmpty()) {
    Node current = (Node)workList.removeFirst();
    if (current.left != null) {
      if (!visited.add(current.left)) return false; // acyclicity
      workList.add(current.left);
    }
    if (current.right != null) {
      if (!visited.add(current.right)) return false; // acyclicity
      workList.add(current.right);
    }
  }
  if (visited.size() != size) return false;      // consistency of size
  return true;
}
```

What Korat Does

Generates all distinct (“non-isomorphic”) trees, up to a given size (say 3).

Use these trees as inputs for testing the `add()` method of the tree.
(or for any other methods)

`korat.sourceforge.net/index.html`

Randoop (MIT)

Key Idea:

"Writing tests is a difficult and time-consuming activity, and yet it is a crucial part of good software engineering.

Randoop automatically generates unit tests for Java classes."

How Randoop Works

Generate random sequence of method calls,
looking for object contract violations.

Point it at a program & let it run.

Discard bad method sequences
(e.g. illegal argument exceptions).
Remember method sequences that create complex objects,
and sequences that result in object contract violations.

`code.google.com/p/randoop/`

An Example Generated by Randoop

```
public static void test1() {
    LinkedList list = new LinkedList();
    Object o1 = new Object();
    list.addFirst(o1);

    TreeSet t1 = new TreeSet(list);
    Set s1 = Collections.synchronizedSet(t1);

    // violated in the Java standard library!
    Assert.assertTrue(s1.equals(s1));
  }
```