The only valid measurement of code quality: WTFs/minute

Mistakes that matter: Lessons from the Trenches

William Pugh
Univ. of Maryland
Defects that matter
Lessons from the Trenches

> I’m the lead on FindBugs
  • static analysis tool for defect detection
> Visiting scientist at Google for 10 months
  • learned a lot about coding mistakes, which ones matter, how to catch them, how to allow a community to review them
Static analysis

> Analyzes code without running it

> FindBugs is an open source static analysis tool, developed at the University of Maryland
  • with a number of additional contributors
  • Looks for bug patterns, inspired by real problems in real code

> Used at eBay: all new issues must be reviewed before being committed to main branch

> Google’s static analysis effort canceled October 2008, restarted Spring 2009
Google FindBugs fixit

> 4,000 issues to review
  • Bug patterns most relevant to Google
> 8,000+ reviews
  • 81+% must/should fix
  • many issues independently reviewed by multiple engineers

> 1,800 bugs filed
  > more than 600 fixed
> More than 1,500 issues removed in several days
Learned wisdom

> Static analysis typically finds mistakes
  • but some mistakes don’t matter
  • need to find the intersection of stupid and important
> The bug that *matter* depend on context
> Static analysis, *at best*, might catch 5-10% of your software quality problems
  • 80+% for certain specific kinds of defects
  • but overall, not a magic bullet
> Used effectively, static analysis is cheaper than other techniques for catching the same bugs
Null bug

> From Eclipse, 3.5RC3:
org.eclipse.update.internal.ui.views.FeatureStateAction

\[
\text{if (adapters == null && adapters.length == 0)} \\
\text{\quad return;}
\]

> Clearly a mistake

- First seen in Eclipse 3.2
- but in practice, adapters is probably never null

> Is there any impact from this?

- we would probably notice a null pointer exception
- if called with length == 0, we wouldn’t return immediately
Cost when a mistake causes a fault/failure

> How quickly/reliability would you notice?
> What is the impact of the misbehavior caused by the mistake?
> How easily could you diagnose the problem and the fix?
> What is the cost to deliver a fix?
Mistakes in web services

> Some mistakes would manifest themselves by throwing a runtime exception
  • Should be logged and noticed

> If it isn’t happening now, a change might cause it to start happening in the future
  • But if it does, the exception will likely pinpoint the mistake
    • whether in test or in production
  • And pushing a fix into production is cheaper than pushing a fix to desktop or mobile applications
Expensive mistakes (your results may vary)

> Mistakes that might cost millions of dollars on the first day they manifest

> Mistakes that silently cause the wrong answer to be computed
  
  • might be going wrong now, millions of times a day
  • or might be OK now, but when it does go wrong, it won’t be noticed until somewhere downstream of mistake

> Mistakes that are expensive or impossible to fix
More null bugs

> Null pointer issues reported by various tools
  • JSR 308 null pointer dereference checker
  • Accurate Interprocedural Null-Dereference Analysis for Java, Mangala Gowri Nanda and Saurabh Sinha, ICSE 2009
/**
 * Deletes the directory at dirName and all its files.
 * Fails if dirName has any subdirectories.
 */

public static void deleteDir(File dir) {
    File[] files = dir.listFiles();
    for (int i = 0; i < files.length; i++) {
        files[i].delete();
    }
    dir.delete();
}

File.listFiles(): returns null if dir is not a directory
accurate interprocedural null-dereference analysis for Java

/**
 * init this task by creating new instance of the ant task
 * and configuring it's by calling its own init method.
 */

public void init() {
    callee = (Ant) getProject().createTask("ant");
    callee.setOwningTarget(getOwningTarget());
    callee.setTaskName(getTaskName());
    callee.setLocation(getLocation());
    callee.init();
}

Project.createTask: returns null if the task name is not recognised.
Using reference equality rather than `.equals`

from Google’s code (no one is perfect)

```java
class MutableDouble {
    private double value_;

    public boolean equals(final Object o) {
        return o instanceof MutableDouble &&
               ((MutableDouble)o).doubleValue() == doubleValue();
    }

    public Double doubleValue() {
        return value_;
    }
}
```
Using `==` to compare objects rather than `.equals`

> For boxed primitives, `==` and `!=` are computed using pointer equality, but `<`, `<=`, `>`, `>=` are computed by comparing unboxed primitive values

> Sometimes, equal boxed values are represented using the same object
  - but only sometimes

> This can bite you on other classes (e.g., `String`)
  - but boxed primitives is where people get bit
Incomparable equality

```java
org.eclipse.jdt.internal.debug.eval.ast.engine.AstInstructionCompiler

SimpleType simpleType = (SimpleType) type;
if ("java.lang.String".equals(simpleType.getName()))
    return Instruction.T_String;

> SimpleType.getName() returns a org.eclipse.jdt.core.dom.Name
> In Eclipse since 2.0 (June 2002)
```
Many variations, assisted by weak typing in APIs

> Using `.equals` to compare incompatible types
> Using `.equals` to compare arrays
  • only checks if the same array
> Checking to see if a `Set<Long>` contains an `Integer`
  • never found, even if the same integral value is contained in the map
> Calling `get(String)` on a `Map<Integer,String>`
Silent, nasty bugs

> Very hard to find these bugs by inspection
  • types not always visible/explicit

> In some cases, could be introduced by refactoring
  • Change the key type of a Map from Integer to Long
  • Fix all the places where you get type errors
  • Leave behind bugs

> Google had an issue with a refactoring that changed a method to return byte[] rather than String
  • introduced silent errors
Bug introduced between Eclipse 3.5RC1 and RC2

org.eclipse.pde.internal.build.BrandingIron

```java
File rootFolder
    = getCanonicalFile(new File(initialRoot));
if (!rootFolder.equals(target)) {
    rootFolder.delete();
    ...
}
```
Demo

Class search strings: sum.nio

Category | Bug Kind | Bug Pattern | Bug Rank
--- | --- | --- | ---

Bugs (4)
- Correctness (4)
  - Null pointer dereference (2)
    - Possible null pointer dereference (1)
  - Possible null pointer dereference of sc in accept
  - Method call passes null for nonnull parameter (1)
  - Switch case falls through (1)
- Dead store due to switch statement fall through (1)
- Value of result from previous case is overwritten

Classify: should fix

Every single case in the switch statement falls through

First seen 05/14
push @ 05/28. should fix
Every single case in the switch statement falls through

Value of result from previous case is overwritten here due to switch statement fall through
At ScpMultiChannelImpl.java:[line 630]
In method sun.nio.ch.ScpMultiChannelImpl.invokeNotificationHandler(ScpResultContainer, NotificationHandler, Object) [Lines 617 - 656]
Local variable named result
Another occurrence at ScpMultiChannelImpl.java:[line 630]
Another occurrence at ScpMultiChannelImpl.java:[line 628]

Dead store due to switch statement fall through
A value stored in the previous switch case is overwritten here due to a switch fall through. It is likely that you forgot to put a break or return at the end of the previous case.

267 issues synchronized with database

267 issues synchronized with database

UNIVERSITY OF MARYLAND
Questions?

William Pugh
pugh@cs.umd.edu
http://findbugs.sourceforge.net/