Principles of Software Construction: Objects, Design, and Concurrency

Part 4: Et cetera

Toward SE in practice: Empiricism in SE

Josh Bloch **Charlie Garrod**

Carnegie Mellon University
School of Computer Science

institute for SOFTWARE RESEARCH

Administrivia

- Homework 6 available
  - Checkpoint deadline tonight
  - Due Wednesday, December 9th
- Final exam due 11:59 pm EST Tuesday, December 15th
  - Will be released on the evening (EST) of Monday, December 14th
  - Review session Sunday, December 13th, 7:30-9:30 pm EST
  - Practice exam released late next week

Key concepts from Tuesday

- SE as a sociotechnical system

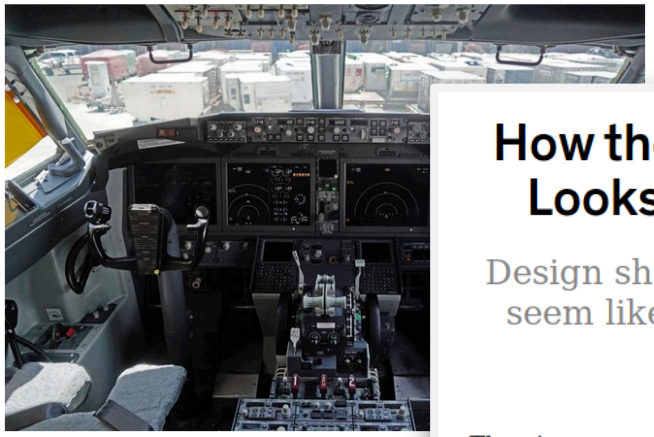
Search Bloomberg Sig

Technology

Boeing's 737 Max Software Outsourced to $9-an-Hour Engineers

By Peter Robison
June 28, 2019, 4:46 PM EDT

- Planemaker and suppliers used lower-paid temporary workers
- Engineers feared the practice meant code wasn't done right



The cockpit of a grounded 737 Max 8 aircraft. *Photographer: Dimas*

It remains the mystery at the hea
crisis: how a company renowned
made seemingly basic software n
deadly crashes. Longtime Boeing
was complicated by a push to ou
contractors.

The Max software -- plagued by is
planes grounded months longer
week revealed a new flaw -- was
was laying off experienced engin
suppliers to cut costs.

https://spectrum.ieee.org/aerospace/aviation/h
developer

How the Boeing 737 Max Disaster Looks to a Software Developer

Design shortcuts meant to make a new plane seem like an old, familiar one are to blame

By **Gregory Travis**

The views expressed here are solely those of the author and do not represent positions of IEEE Spectrum *or the IEEE.*

Photo: Jemal Countess/Getty Images

This is part of the wreckage of Ethiopian Airlines Flight ET302, a Boeing 737 Max

A year after the first 737 Max crash, it's unclear when the plane will fly again

Two crashes of Boeing's 737 Max 8 killed 346 people, and authorities are blaming Boeing's design, a faulty sensor and airline staff. Plus: Everything you need to know about the plane.

Kent German November 1, 2019 9:01 AM PDT

ed killing 346 people.

ts 737 Max 8 that killed 346 people, Boeing is facing
its newest and most critical aircraft models. The
und the world, and the Federal Aviation

institute for SOFTWARE RESEARCH

Major topics in 17-313 (Foundations of SE)

- Process considerations for software development
- Requirements elicitation, documentation, and evaluation
- Design for quality attributes
- Strategies for quality assurance
- Empirical methods in software engineering
- Time and team management
- Economics of software development

SE as a sociotechnical system summary

- Software engineering requires consideration of many issues, social and technical, above code-level considerations
- Interested? Take 17-313
- Shameless plug: Take API Design, 17-480

Today: Software engineering in practice

- Empiricism in SE
  - Mob programming
  - Test-driven development

Volunteer?

Mob programming

Mob programming

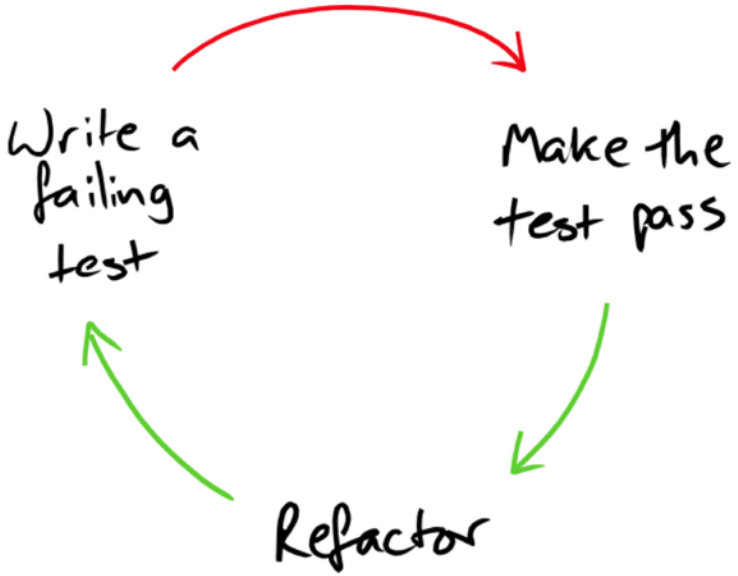
- Like pair programming, but with more people
  - Driver vs. navigators (a.k.a. the typist vs. everyone else)
  - Group decision-making
  - Frequent rotation

Today: Software engineering in practice

- Empiricism in SE
  - Mob programming
  - Test-driven development

Test-driven development (TDD)

Test-driven development (TDD), informally



From Growing Object-Oriented Software by Nat Pryce and Steve Freeman
http://www.growing-object-oriented-software.com/figures.html

@sebrose

http://cucumber.io

Formal test-driven development rules

1. You may only write production code to make a failing test pass
2. You may only write a minimally failing unit test
3. You may only write minimal code to pass the failing test

Test-driven development as a design process

"The act of writing a unit test is more an act of design and documentation than of verification. It closes a remarkable number of feedback loops, the least of which pertains to verification."

Advantages of test-driven development

- Clear place to start
- Iterative, agile design process
- Less wasted effort?
- Robust test suite, including regression tests

A test-driven development demo: Diamond Kata

- Given a letter, generate a diamond starting at ‘A’, with the given letter at the widest point.
  - e.g., `diamond('C')` would generate:

```
  A
 B B
C   C
 B B
  A
```

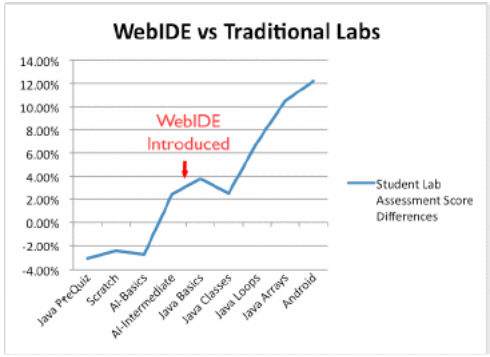
Formal test-driven development: Your impressions?

Empirical methods in software engineering

- How do we study the effectiveness of mob programming or test-driven development compared to other methodologies?
  - Note: Mix of social and technical issues

Research on test-driven development (1/2)

- Hilton et al.: Students learn better when forced to write tests first
- Bhat et al.: At Microsoft, projects using TDD had greater than two times code quality, but 15% more upfront setup time



- George et al.: TDD passed 18% more test cases, but took 16% more time
- Scanniello et al.: Perceptions of TDD include: novices believe TDD improves productivity at the expense of internal quality

Research on test-driven development (2/2)

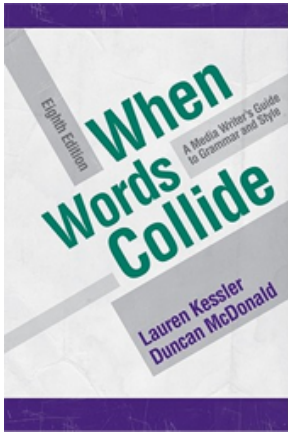
- Fucci et al.: Results: The Kruskal-Wallis tests did not show any significant difference between TDD and TLD in terms of testing effort (p-value = .27), external code quality (p-value = .82), and developers' productivity (p-value = .83).

- Fucci et al.: Conclusion: The claimed benefits of TDD may not be due to its distinctive test-first dynamic, but rather due to the fact that TDD-like processes encourage fine-grained, steady steps that improve focus and flow.

Summary

- Software engineering as an empirical field
  - Quantitative and qualitative methodologies

6. “When Words Collide”




```
public class PrintWords {
  public static void main(String[] args) {
    System.out.println(
      Words.FIRST + " " + Words.SECOND + " " + Words.THIRD);
  }
}

public class Words { // Compile PrintWords against this version
  public static final String FIRST  = "the";
  public static final String SECOND = null;
  public static final String THIRD  = "set";
}

public class Words { // Run against this version
  public static final String FIRST  = "physics";
  public static final String SECOND = "chemistry";
  public static final String THIRD  = "biology";
}
```

What does it print?

(a) the null set
(b) physics chemistry biology
(c) Throws exception
(d) None of the above

```
public class PrintWords {
  public static void main(String[] args) {
    System.out.println(
      Words.FIRST + " " + Words.SECOND + " " + Words.THIRD);
  }
}

public class Words { // Compile PrintWords against this version
  public static final String FIRST  = "the";
  public static final String SECOND = null;
  public static final String THIRD  = "set";
}

public class Words { // Run against this version
  public static final String FIRST  = "physics";
  public static final String SECOND = "chemistry";
  public static final String THIRD  = "biology";
}
```

What does it print?

(a) `the null set`

(b) `physics chemistry biology`

(c) Throws exception

(d) None of the above: `the chemistry set`

Java inlines *constant variables*

What exactly is a constant variable?

- Loosely speaking, a final primitive or `String` variable whose value is a *compile-time constant*
  - See JLS3 4.12.4, 13.4.9, 15.28 for gory details
- Surprisingly, `null` *isn't* a compile-time constant

Another look

```
public class PrintWords {
  public static void main(String[] args) {
    System.out.println(
      Words.FIRST + " " + Words.SECOND + " " + Words.THIRD);
  }
}

public class Words { // Compile PrintWords against this version
  public static final String FIRST  = "the";   // Constant variable
  public static final String SECOND = null;    // Not a constant variable!!!
  public static final String THIRD  = "set";   // Constant variable
}

public class Words { // Run against this version
  public static final String FIRST  = "physics";
  public static final String SECOND = "chemistry";
  public static final String THIRD  = "biology";
}
```

How do you prevent constants from being inlined?

```
// Utility function that simply returns its argument
private static String ident(String s) {
  return s;
}

// None of these fields are constant variables!
public class Words {
  public static final String FIRST  = ident("the");
  public static final String SECOND = ident(null);
  public static final String THIRD  = ident("set");
}
```

Prints physics chemistry biology

The Moral

- Constant variable references are inlined
  - Only primitives and strings can be constant variables
  - **`null`** is not a constant variable (neither are enums)
- **If you change a constant's value without recompiling its clients, they break!**
  - Use constant variable only if value will *never* change
  - Use **`ident`** method for final primitive or string fields whose value may change
- For language designers
  - Don't inline constants in a late-binding language
  - More generally, be consistent!