Lecture 7: Mostly Integrated Development Environments; About Bugs

Engineering Design with Embedded Systems

Patrick Lam
University of Waterloo

January 21, 2013

Variable Naming

Choose your names wisely.

- Single-letter names (`i`) are great for short-lived variables.
- Longer, more descriptive, names are better for longer-lived variables.

Miscellaneous: Real-Time Systems

Must respond to an external event in a fixed amount of time.

This fixed amount of time is not necessarily small.

- may potentially be fixed and large.

Many embedded systems must satisfy real-time constraints.

In upper-year courses, you'll see both embedded systems and real-time systems in more detail.

Real-Time System Example

(credit `digitaljournal.com`, from flickr)

Blu-Ray player must:

- read compressed video data from a media disk;
- decompress the video; and
- output it to a HDMI interface,

all within a fixed amount of time, to avoid a degradation of video quality.

Part I

Integrated Development Environments

What is an IDE?

Contains:

- editor, plus
- compiler, plus
- debugger

Integrated into a single environment.

The Eclipse IDE

Eclipse is a fully-featured modern IDE.

Notes:

- runs on Linux, Mac, Windows;
- it is free software: you can extend and modify it;
- initially developed by IBM Ottawa.

Beyond Core IDE Components

Eclipse (and other IDEs) also support:

- revision control systems (you've used this);
- documentation and modelling (e.g. UML);
- autocomplete and refactoring.

Project Development Workflow

0. Figure out what you'll need to do.
1. Start a new project from a template or, more realistically, check it out from a version control repository.
2. Make the edits that you need.
3. Test your edits by running the application.
4. Debug your edits.
5. Commit your files to the version control repository.

Eclipse demo

OK, this time we'll really do the demo. Plan:

1. Create a new Android project from a **template**.
2. Add an **`EditText`** to the main **`Activity`**.
3. Use **`addTextChangedListener`** to watch for changes in the text.
4. Use **Content Assist** to get method stubs in the **`TextWatcher`** inner class.
5. Add code to the **`afterTextChanged`** method.
6. **Commit** it to the repository.

Part II

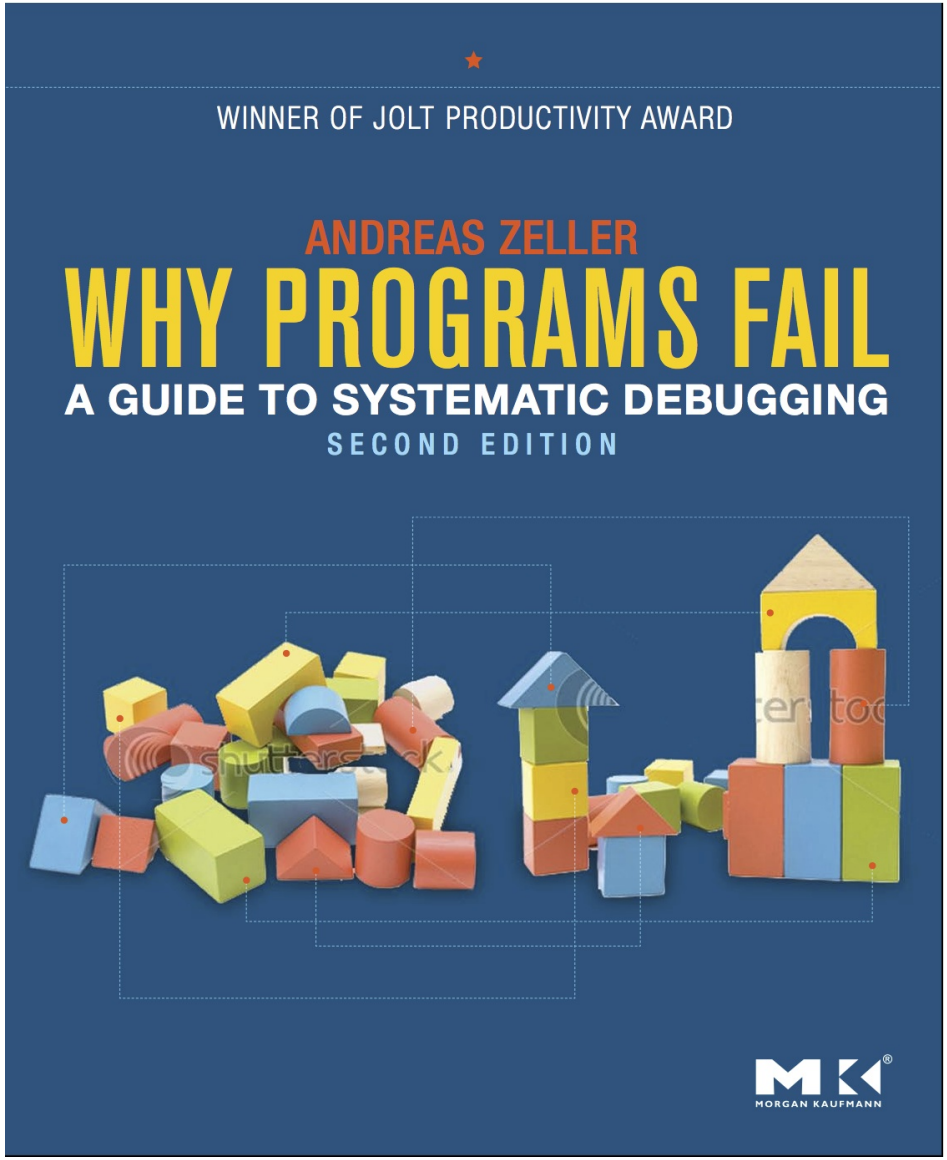
Fixing Bugs

9/9

0800 Arctan started
1000 " stopped - arctan ✓ { 1.2700 9.037 847 025
9.037 846 795 correct
13"uc (032) MP - MC ~~1.9821447000~~ ~~2.13047641S (-3)~~ 4.615925059(-2)
(033) PRO 2 2.13047641S
correct 2.13067641S
Relays 6-2 in 033 failed special speed test
In relay " 11.00 test.
Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Mult + Adder Test.
1545  Relay #70 Panel F
(moth) in relay.
First actual case of bug being found.
~~1625~~ 1630 Arctangent started.
1700 closed down.

Relay 2145
Relay 3370

About Bugs: Recommended Reading



Finding Bugs

Three things have to happen before you can observe a bug:

1. A programmer puts a defect in the code—some code doesn't do what it's supposed to do.
2. This defect sets some program state (e.g. a variable) to an incorrect, or "infected" value.
3. The infected value has to propagate to program output to cause an observable failure.

The Debugging Process: High-Level Overview

- Identify the bug and steps to reproduce it.
- Figure out the cause of the bug.
- Correct the bug.

Closely related to the scientific method.

More next time!