Emacs or Vi

9 things your editor says about you!

You won't believe number 7!

Discourses and Dialogs on Debugging

Perry Kivolowitz

~~University of [illegible] – Madison~~

~~Compute[illegible]epartment~~

CARTHAGE COLLEGE

On tap tonight:

Part 1 Discourses

Rules, that if followed, will change your life

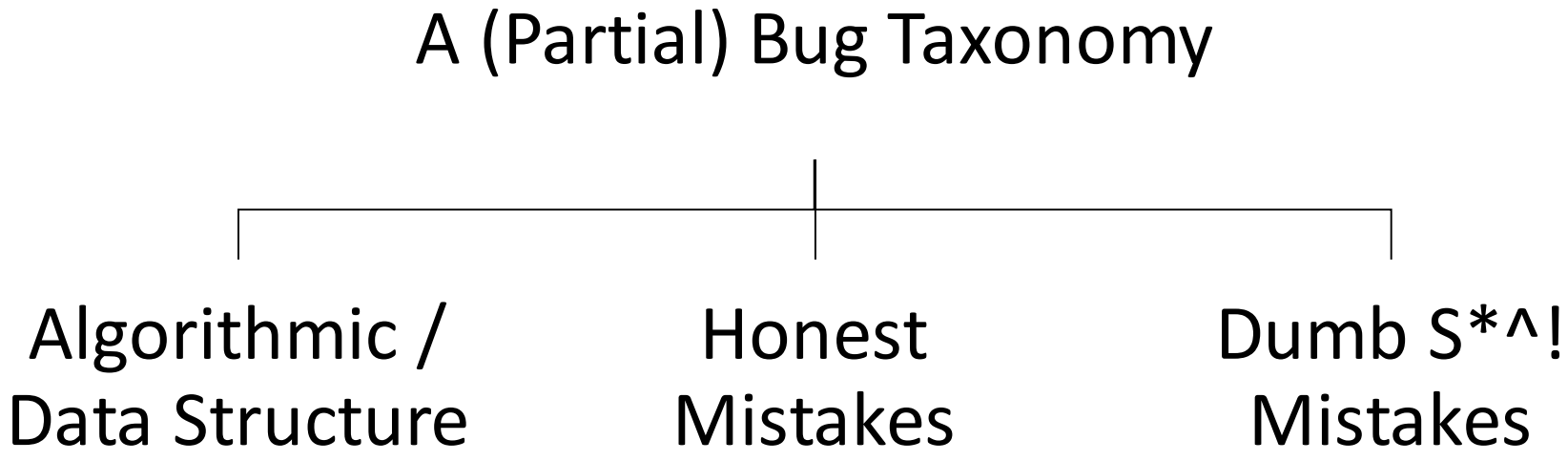
Part 2 Dialogs

Real world applications of the discourses

Every dialog actually happened*

As we talk, something you might think about:

How does this relate to each method, maxim, corollary or dialog?



Part 1 - Discourses

Motivation

We **all** write buggy code

Few of us are

taught how to debug

Some say debugging cannot be taught

"Debugging is an art"

"Either you get it or you don't"

The thesis of this talk is:

debugging is based in *science*

(and science can be taught)

The Science of Debugging

The Scientific Method

and also…

Kivolowitz Corollary

Kivolowitz's Maxim 1

Maxwell Cohen's Law

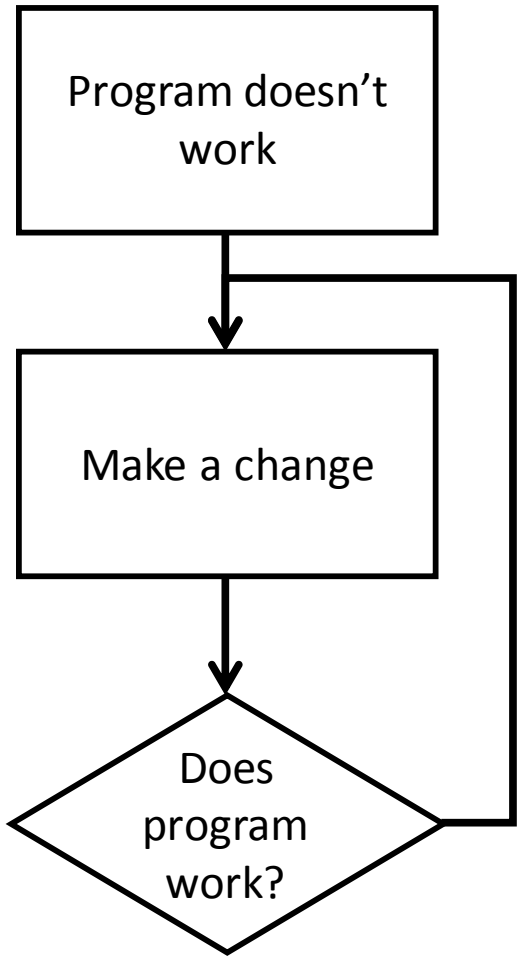
Kivolowitz's Maxim 2

Conan Doyle's Law

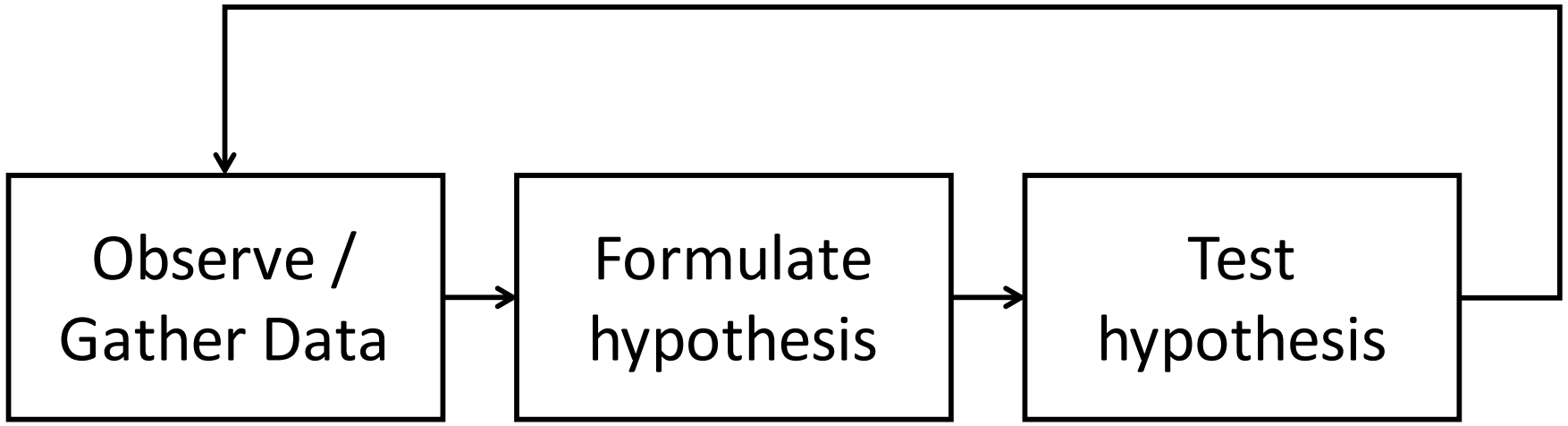
Kivolowitz's Maxim 3

Kivolowitz's Maxim 4

Most Common Debugging Algorithm



The Scientific Method

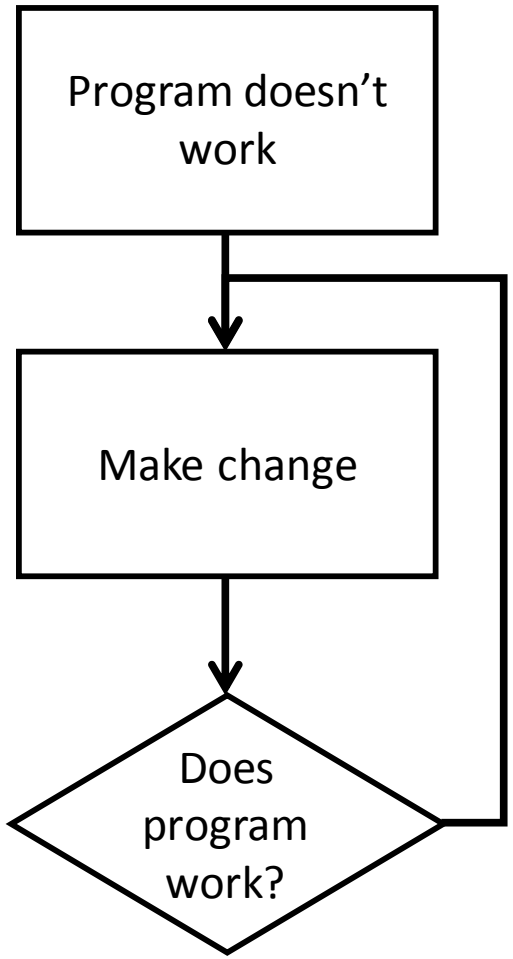


Debugging should be methodical not knee-jerk or reactionary

Kivolowitz Corollary

"A fix is not a fix until you completely understand why it is a fix"

What else is wrong with this?



Kivolowitz Maxim #1

"Debugging is about the elimination of unknowns,

not their introduction"

Kivolowitz Maxim #1

If you make a change with no beneficial result, back it out*

Else, you are chasing a moving target

*unless you are certain you're fixing a different bug

Maxwell (Mickey) Cohen's Law

"Where there is one, there are likely many"

Always be mindful that you may be seeing a

cascade of errors

(Note to self: Describe how debugging is like a train wreck)

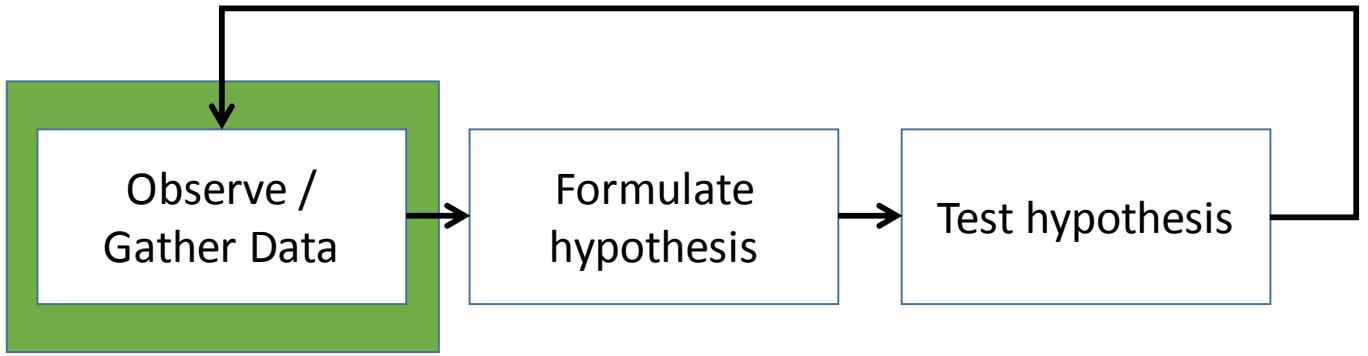
(Note to self: Describe how debugging is different from looking for your phone)

Kivolowitz's Maxim #2

"Bugs want to be found"

They often announce themselves clearly

if you make the effort to listen



Conan Doyle's Law

"When you have eliminated the impossible, whatever remains, however improbable, must be the truth."

The Sign of Four, 1890

Kivolowitz Maxim #3

"Write in small units.
Test in small units."

Kivolowitz Maxim #4

"Always play defense"

"Extra work" is finite.

Debugging time, not so much.

Part 2 - Dialogs *(all actually happened)*

http://images6.fanpop.com/image/photos/33100000/Kung-Fu-Panda-3-random-33170400-1920-810.jpg

Kivolowitz Corollary Example

Student: My code crashed. I added a print out. Now it doesn't crash. I fixed it, right?

Master: Sigh

No understanding of the "fix"

Kivolowitz's Maxim #2 Example

Student:	I get "Seg fault. Core dumped." Can you help me?
Master:	What do you think the problem is?
Student:	How should I know? That's all it says.
Master:	Sigh.

"Bugs want to be found"

They often announce themselves clearly

if you make the effort to listen

Kivolowitz's Maxim #2 Example

Student: I have an error. Can you fix it?

Master: What does the error say?

Student: I didn't read it. I just clicked "OK"

Master: Sigh

"Bugs want to be found"

They often announce themselves clearly

if you make the effort to listen

Kivolowitz's Maxim #2 Example

Student: My code crashes. Can you fix it?

Master: You get compiler warnings.

Student: They're just warnings.

Master: Sigh

"Bugs want to be found"

They often announce themselves clearly

if you make the effort to listen

Kivolowitz's Maxim #2 Example

Student: It's like this code isn't executed!

Master: Have you proved it is executed?

Student: No

Master: Sigh

"Bugs want to be found"

They often announce themselves clearly

if you make the effort to listen

Kivolowitz's Maxim #2 Example

Student: It crashes after running for a while.

Master: Do you have a memory leak?

Student: A what now?

Master: Sigh

"Bugs want to be found"

Use leak detection tools.

Leave code running for ____ if warranted

Kivolowitz's Maxim #2 Example

Student: After a while my code stops responding to me.

Master: Do you have an infinite loop?

Student: A what now?

Master: Sigh

"Bugs want to be found"

Not responding? Check CPU utilization.

First code to write in a loop is how it exits.

Kivolowitz's Maxim #2 Example

Student: My code works great except on this input.
Master: Have you single-stepped that input?
Student: Single… what?
Master: Sigh

"Bugs want to be found"

Your environment should be your friend.

Get to know it – USE IT!

Debugging tools and techniques

- Single step
- Breakpoints
  - Always
  - Conditional
    - In debugger
    - In code
- Call stack
- Re-execution of code
- Immediate modes

- Value inspection
- Value modification
- Console output
  - Binary search
  - Entry / exit
- assert() [discuss later]

Conan Doyle's Law Example

Student: I looked everywhere.

Master: Did you check here?

Student: The bug can't possibly be there.

Master: Sigh

"When you have eliminated the impossible, whatever remains, however improbable, must be the truth."

Kivolowitz Maxim #3 Example

Student: It worked yesterday. Then I wrote 5000 lines of code. Now it doesn't work. Can you fix it?

Master: Sigh

"Write in small units. Test in small units."

Kivolowitz Maxim #3 Example

Student: I must have a compiler / OS bug.

Master: Have you written a minimal test harness that manifests the bug?

Student: No.

Master: Sigh

A minimal test harness saves time (and face).

"Write in small units. Test in small units"

Kivolowitz Maxim #3 - Role of testing

Testing can only prove the presence
of bugs, not their absence.

Edsger W. Dijkstra

Beware of bugs in the above code; I have
only proved it correct, not tried it.

Donald Knuth

Maxim #4 - Defensive programming example

Student: I commented like you told me.

```
// Increment j
```

Master: Sigh

Comment your thought process.

Not minutia.

Maxim #4 - Defensive programming example

Student: I commented like you told me.

```
// Increment j
```

Master: Sigh

When commenting, think of the next person to read your code.
It might be you.

Maxim #4 - Defensive programming example

Student: My code doesn't work.

Master: And there is no way it ever would.

Did you test it *before* you wrote it?

Student: Wait… What?

Master: Sigh

Writing comments before you write your code is like a one-person code review.

Maxim #4 and Scientific Method

Student: My code doesn't work.

Master: Did you try ____ approach?

Student: I don't remember.

Master: Sigh.

Comment what **didn't work**

so you don't try it again.

Scientific method learns from history. Preserve it.

Maxim #4 - Defensive programming example

Student: My code doesn't work.

Master: What does variable `ineedsleep` do?

Student: I don't remember. I was tired.

Master: Sigh

`descriptive_variable_names_really_help`

Keystrokes are finite. Debugging time isn't.

Maxim #4 - Defensive programming example

Student: ____; ____; ____; ____;

Master: What's that in the middle?

Student: Where?

Master: Sigh

Newlines are free. Resist temptation for multiple statements on one line.

Maxim #4 - Defensive programming example

Student: `__ += __ ? (__ = --__ , __) : __ = __;`

Master: What's that in the middle?

Student: Where?

Master: Sigh

Use side effects with caution

(Do this and be despised by your peers)

Cohen's Law, Defensive programming *and* Doyle's Law

Student: I looked everywhere.

Master: Did you check here?

Student: The bug can't possibly be there.

Master: Sigh.

If you "know" a condition to be true, `assert()` it!

Stops a cascade in its tracks

Eliminates the impossible and identifies the improbable

Cohen's Law, Defensive programming *and* Doyle's Law

```
#include <assert.h>

assert(condition I know to be true);
```

assert enforces "contracts"

Defensive programming *and* Doyle's Law

Student: It can't be there. I pasted that code from over there.

Master: Really? This is different from that.

Student: Oh yeeaahhhh, I fixed a bug over there.

Master: Sigh.

Code abstraction is actually a defensive technique

Code in one place, test in one place, fix in one place

Defensive programming *and* Doyle's Law

Student: foo has to be one of these values.

Master: And if it isn't?

Student: That's impossible.

Master: Sigh.

Always code the "default" case

Always code the "impossible else"

(tell Ben Liblit's story)

Maxim #4 and Scientific Method

Student: My code doesn't work.

Master: Wouldn't ____ approach be more likely to work?

Student: I tried that, it didn't. I deleted it.

Master: Sigh.

#ifdef and source code control

Scientific method learns from history. Preserve it.

Maxim #4 - Defensive programming example

Student: My code doesn't work.
Can you fix it?
Master: Are you checking return values?
Student: No.
Master: Sigh.

Return values are meant to be checked

Maxim #4 - Defensive programming example

Student: My code doesn't work.
Can you fix it?
Master: Are you checking return values?
Student: That makes my code look ugly.
Master: Sigh.

Use exceptions
(when performance requirements allow)

Maxim #4 - Defensive programming example

Student: I wrote my own _____.

Master: Why didn't you use _____ or STL?

Student: I wanted to write my own.

Master: Sigh.

A framework vetted by thousands of programmers is *probably* better than yours

One last dialog

Student: My code crashes every time it gets here!

Master: May you always be so fortunate.

Student: Wait. What?

Summary

Debugging as art: Skill grows over time.

Debugging as science: Every bug can be found.

We all write buggy code.

No! to desperation. Yes! to challenges.

Enjoy the craft.