Bits of Evidence

What We Actually Know About
Software Development,
Why We Believe It's True,
And What It Has To Do With You

Greg Wilson

http://software-carpentry.org

March 2015

Once Upon a Time…

1986: Start programming first-generation parallel computers

1992: "HPC Considered Harmful"

1996: "What Should Computer Scientists Teach Physical Scientists and Engineers?" (*IEEE CS&E*)

1998: "OK, show us." (John Reynders, LANL)

What I Learned

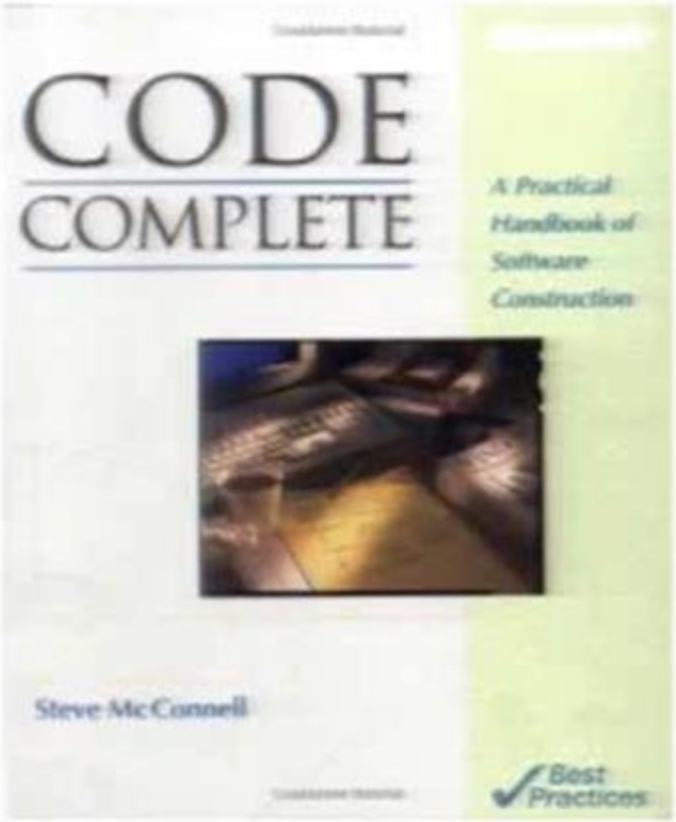
The problem with teaching scientists is they know what evidence looks like

The problem with computer science is *I didn't have any*

I had never done an experiment in a computer science course

Or even analyzed someone else's experimental data

But Other People Had



Hey, we actually know stuff about things!

Arrr, Matey

Seven Years' War (actually 1754-63)

Britain lost 1,512 sailors to enemy action...

...and almost 100,000 to scurvy

Oh, the Irony

James Lind (1716-94)

1747: (possibly) the first-ever controlled medical experiment

× *cider*
× *sulfuric acid*
× *vinegar*
× *sea water*
√ *oranges*
× *barley water*

Of course, no-one paid attention until a proper Englishman repeated the experiment in 1794...

It Took a While...

1950: Hill & Doll publish a *case-control study* comparing smokers with non-smokers

Now called the "British Doctors" study, it ran until 2001

What They Found

#1: Smoking causes lung cancer

#2: Most people would rather fail than change

"What happens 'on average' is of no help when one is faced with a specific patient…"

The Cochrane Collaboration (http://www.cochrane.org/) now archives results from hundreds of medical studies

So Where Are We?

"[Using domain-specific languages] leads to two primary benefits. The first, and simplest, is improved programmer productivity... The second...is...communication with domain experts."

– *Martin Fowler,*
IEEE Software,
July/August 2009

Look Closer

One of the smartest guys in the industry...

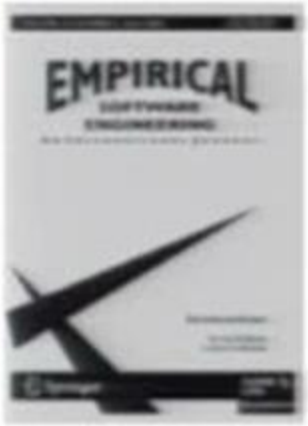
...made two substantive claims of fact...

...in a peer-reviewed journal...

...*without a single citation*...

...because nobody expected one

A New Hope



Growing emphasis on empirical studies in software engineering since the mid-1990s

Papers describing new tools or practices routinely include results from some kind of field study

Many are flawed or incomplete,
but standards are constantly improving

A Classic Result

Rigorous inspections can remove 60-90% of errors before the first test is run. (Fagan 1975)

Isn't *That* Interesting…

Nagappan et al (2007) & Bird et al (2009):

Geography has little correlation with software quality

All Together Now

Andy Oram & Greg Wilson (ed): *Making Software: What Really Works, and Why We Believe It.* O'Reilly, 2010, 978-0596808327.

What You Hear

Reproducible
peta-scale
GPU workflows –
in the cloud!

Reality

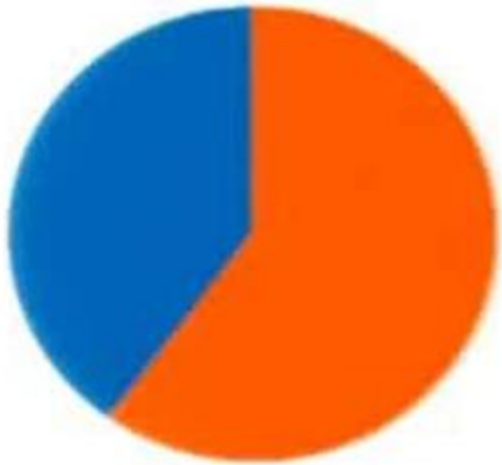
5-10%

90-95%

Hannay et al, 2009,
Prabhu et al, 2011

Surely You're Exaggerating

1. How many graduate students write shell scripts to analyze each new data set instead of running those analyses by hand?



It Is Therefore Obvious That…

Put more computing in the curriculum!

But it's already full

It Is Therefore Obvious That…

Put a little computing in every course!

Still adds up: 5 minutes/lecture = 4 courses/degree

First thing cut when the lecturer is running late

The blind leading the blind…

It Is Therefore Obvious That…

Partner with computer scientists!

"Gosh, I'd like some clean water."

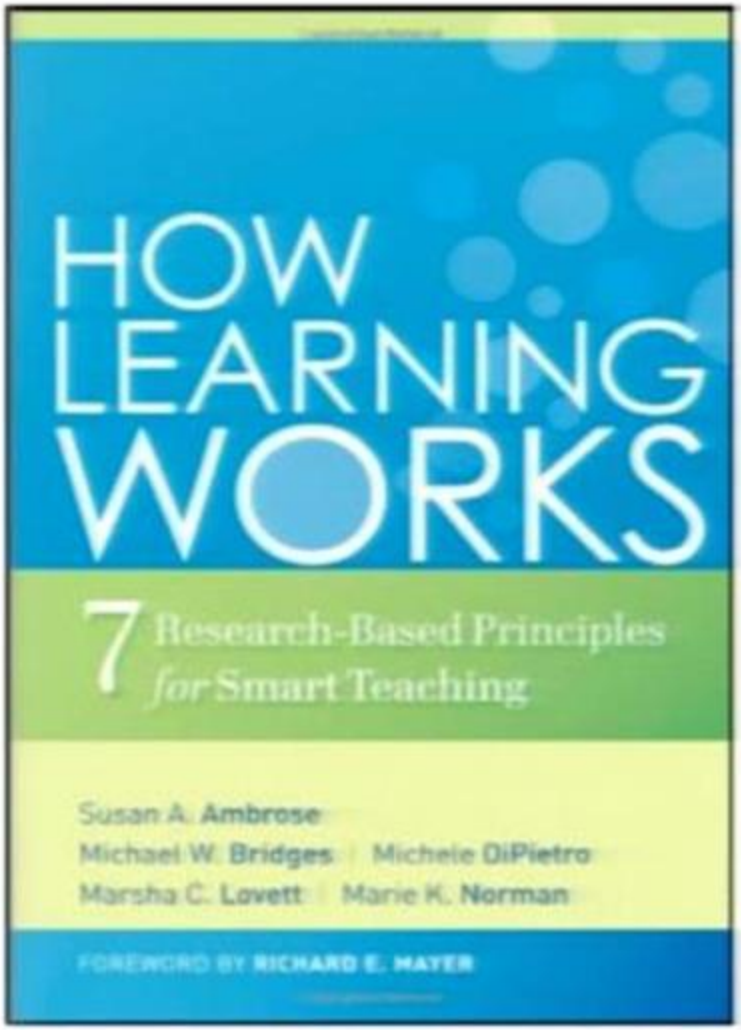
"Let's talk about brain scans."

Bait and Switch

Unix shell	Task automation
Python/R/Matlab	Structured programming
Git and GitHub	Provenance and collaboration
SQL	Data management

http://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.1001745

Evidence


HOW LEARNING WORKS
7 Research-Based Principles for Smart Teaching
Susan A. Ambrose
Michael W. Bridges
Michele DiPietro
Marsha C. Lovett
Marie K. Norman
FOREWORD BY RICHARD E. MAYER

A Few More Results

Are any software metrics better at predicting effort or faults than counting lines of code?

No.

My Personal Favorite

We first present two surveys conducted with students on the intuitiveness of syntax, which we used to garner formative clues on what words and symbols might be easy for novices to understand. We followed up with two studies on the accuracy rates of novices using a total of six programming languages: Ruby, Java, Perl, Python, Randomo, and Quorum. To our surprise, we found that languages using a more traditional C-style syntax (both Perl and Java) did not afford accuracy rates significantly higher than a language with randomly generated keywords, but that languages which deviate (Quorum, Python, and Ruby) did.

My Personal Favorite

*We first present **two surveys conducted with students on the intuitiveness of syntax**, which we used to garner formative clues on what words and symbols might be easy for novices to understand. We **followed up with two studies on the accuracy rates of novices** using a total of six programming languages: Ruby, Java, Perl, Python, Randomo, and Quorum. To our surprise, we found that languages using a more traditional C-style syntax (both Perl and Java) did not afford accuracy rates significantly higher than a language with randomly generated keywords, but that languages which deviate (Quorum, Python, and Ruby) did.*

Words to Live By

If you build a man a fire,
you'll keep him warm for a night.
If you set a man on fire,
you'll keep him warm for the rest of his life.

— Terry Pratchett

software carpentry

gvwilson@software-carpentry.org

Thank You