What We Think We Know About Software Development and Why We Believe it's True

Greg Wilson
Software Carpentry
Steve Crouch, Mike Jackson
The Software Sustainability Institute

Once upon a time…

- 7 Years War (1754-63)

- Britain loses 1512 sailors to enemy action…
- …and almost 100,000 to scurvy

What we think we know…

Symptoms: Horrible - Gums go black swell up, fingernails and teeth fall out, then the bad stuff starts!

But before then…

James Lind (1716-94)

1747: (possibly) first ever controlled medical experiment

Cider	**Sea water**
Sulphuric acid (!)	**Oranges**
Vinegar	**Barley water**

What we think we know…

~9 years before that, a Scottish surgeon James Lind found a solution.
If you pickle food, it doesn't go bad…so can we pickle, well, preserve, sailors?
Took twelve sailors, divided them into six pairs, and gave each pair something different: cider, sea water, vitriol (which is a weak solution of sulphuric acid—bad day to be those guys), oranges, vinegar, and barley water.

Oranges (and lemons)…

James Lind (1716-94)

1747: (possibly) first ever controlled medical experiment

Cider	**Sea water**
Sulphuric acid	**Oranges**
Vinegar	**Barley water**

- And no-one listened…he's not an English gentleman!
- But in 1794 the Admiralty gave it a go…
- …and England came out on top!

What we think we know…

Sailors who were given oranges were back on their feet in just a few days, while the others continued to sicken.
Many historians believe that one of the key reasons England came out of Napoleonic Wars on top.
British expeditions that went into the South Pacific and around Africa took lime trees with them.

Modern medicine

- Medical profession eventually realised controlled studies were the right way to go
- David Sackett
  - Pioneer of “evidence-based medicine”
  - Randomised double-blind test is a “gold standard”
- Cochrane Collaboration
  - http://www.cochrane.org/
  - Largest collection of records randomised controlled trials the world

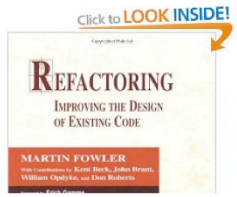
What we think we know…

Randomised double-blind test – no drug/medicine released to market until this happens.

What about software development?

software carpentry

- Martin Fowler IEEE Software, July/Aug 2009
- “[Using domain specific language] leads to two primary benefits. The first, and simplest, is improved programmer productivity... The second... is... communication with domain experts.”



What we think we know…

He wrote the book on refactoring, literally!

Just one more thing…

- Two substantive claims…
- …without a single citation!
- Where's the proof!
  - The Scottish Verdict!
- "Many software researchers advocate rather than investigate" – Robert L. Glass (2002)

What we think we know…

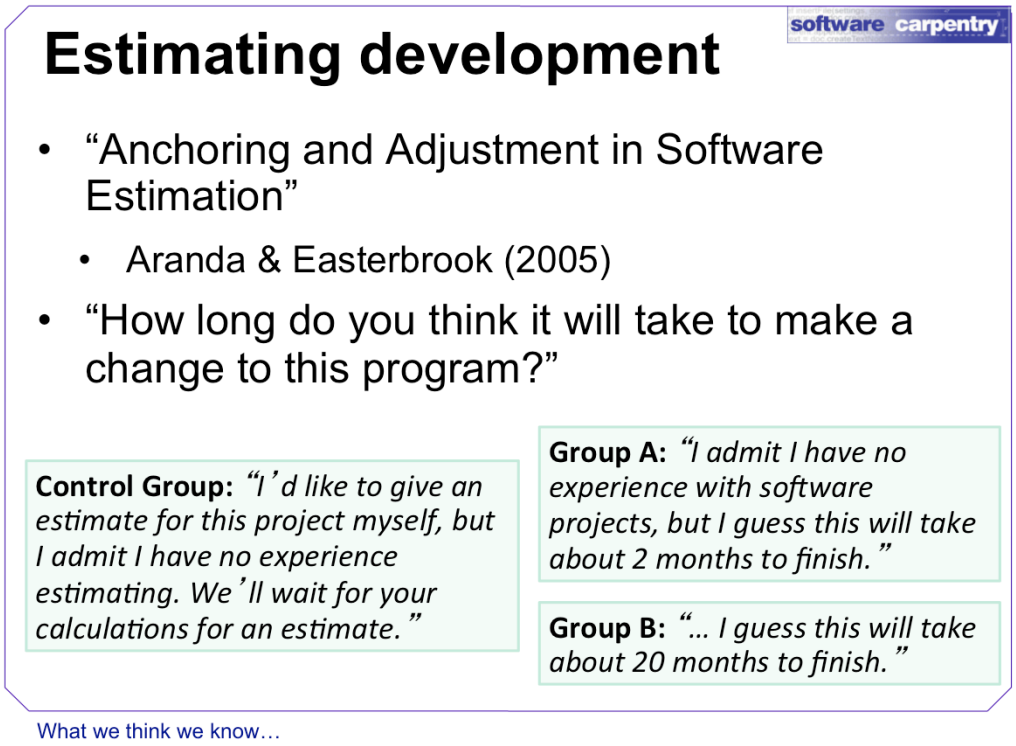
Perhaps seems like a good idea – you can express the problem you want to solve in a language that is closer to the area in which you want to express it.
In software engineering this kind of thing is par for the course, perhaps with a brief reference to an article from some guy over there who happened to agree. Not proof.
Advocate: RLG, Facts and Fallacies of Software Engineering – "probably true that this is the most contentious fact in this book".

Times, they are a changin'

- Growing emphasis on empirical studies in software research since the mid-1990s
- Papers describing new tools or practices routinely include results from some kind of field study
- International Conference on Software Engineering
  - http://www.icse-conferences.org/
- It will never work in theory
  - http://www.neverworkintheory.org/
- Almost always must include some field testing results in papers now

What we think we know…

Often expensive to do, many flawed or incomplete, but standards are improving.
A cultural shift.



Participants were recruited through email invitations sent to graduate computer science students and software developers.

23 participants, 57% had previous estimation experience.

Each group gets a 2 page specification, one variation.

Groups don't know about the different guesses.

Anchoring effect

Group A (lowball)	5.1 months
Control Group	7.8 months
Group B (highball)	15.4 months

- The anchor mattered more than…
  - Experience in software engineering
  - Tools used
  - Formality of estimation
- Are agile projects similarly afflicted, just on a shorter and more rapid cycle?

What we think we know…

Anchoring effect – cognitive bias, rely too heavily on one piece of information, in this case the estimate from the novice!
Agile methods have shorter timescales but could still be prone to the same percentage mis-estimates. Not (yet) studied but would be an argument in favour of agile methods IF people were more accurate at shorter period estimations.

A popular quote

software carpentry

- "Exploratory experimental studies comparing online and offline programming performance"
  - Sackman, Erikson and Grant (1968)
- "The best programmers are up to 28 times more productive than the worst"
- So all we need are a few good people…
  - …like ourselves!
- 1968!
  - Designed to compare batch versus interactive
  - 12 programmers for an afternoon

What we think we know…

Dozens of books quote this.

Was done in 1968 and not designed to measure productivity but batch versus interactive systems!

And just used 12 programmers – bad use of statistics.

So what *do* we know?

software carpentry

- Lutz Prechelt
  - Productivity variations between programmers
  - Effects of language and web programming frameworks
- Productivity and reliability depend on length of program's text
  - Independent of language abstraction level
- Go high-level
  - Trading off performance

What we think we know…

Studies are hard to do and expensive but, compared to clinical trials, nothing.

Software controls planes, financial transactions, the results of your research! It's important.

Consider a 5% productivity improvement in a typical programmer. Apply that to a £300 billion industry…

A classic result…

- "Some Experience with Automated Aids to the Design of Large-Scale Reliable Software"
  - Boehm et al (1975)
- Most errors are introduced during requirements analysis and design
- The later an error is detected the more costly it is to address
  - 1 hour to fix in the design
  - 10 hours to fix in the code
  - 100 hours to fix after it's gone live…

What we think we know…

Validated by many others.

After it's gone live when you've been hit with a lawsuit!

Which can be viewed by…

- Pessimists
  - Traditionalists
  - “If we tackle the jump in the error injection curve, fewer bugs will get to the expensive part of the fixing curve”
- Optimists
  - Agile
  - “If we do lots of short iterations, the total cost of fixing bugs will go down”

What we think we know…

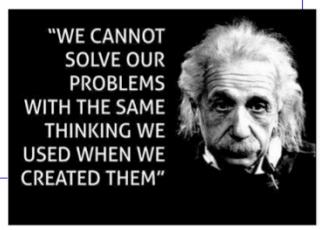
Pessimists: fix the bugs earlier in a large cycle, cheaper than fixing later. Total cost will come down.

Optimists: let’s reduce the entire cycle to a couple of weeks, and stitch a lot of them together. Total area under the fixing effort curve is reduced. Essentially fixing ‘later’ becomes ‘very soon’ and overall, effort for fixing is reduced.

Greatest hits I

software carpentry

- For every 20% increase in problem complexity, there is a 100% increase in solution complexity
  - Woodfield (1979)
- The two biggest causes of project failure are poor estimation and unstable requirements
  - van Genuchten (1991) and many others
- Development teams do not benefit from existing experience, and they repeat mistakes over and over again
  - Brossler (1999)



What we think we know…

Woodfield was replicated a number of times.

Non-linear because of interaction effects.

Add a few more features, could equal five-fold increase in effort required

If you reduce the problem space i.e. by simplification or assumption, you can significantly reduce the solution required.

Perhaps this is why agile methods work so well.

Greatest hits II

software carpentry

- Rigorous inspections can remove 60-90% of errors before first test is run
  - Fagan (1975)
- The first review and hour matter most
  - Cohen (2006)
- Physical distance doesn't affect post-release fault rates, distance in organisational chart does
  - Nagappan et al (2007) & Bird et al (2009)

What we think we know…

Fagan – don't run the code, don't even test it yet…Read it!

Cohen's: footnote…

Having two or more people read the code, or for longer, doesn't really yield a good enough return to make it worthwhile.

For best value, have one other read through the code for an hour at a time (after an hour, brain full – around 200 lines of code; take into account interactions.

Argues that we should be developing code in little steps. Look at Mozilla, Linux. Most progress in little additions.

Greatest hits III

software carpentry

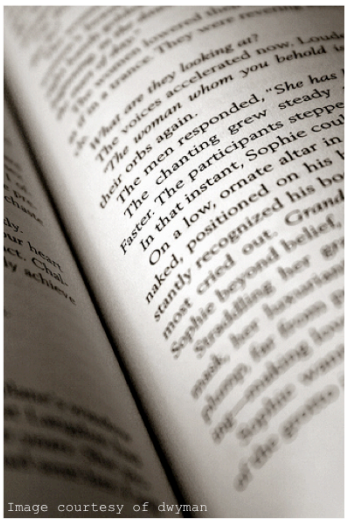
- Half the errors are found in 15% of the modules
  - Davis (1995) quoting Endres (1975)
- About 80% of the defects come from 20% of the modules, and about half the modules are error free
  - Boehm and Basili (2001)
- When you identify more errors than expected in some program module, keep looking!
- Shouldn't our development practices be built around these **facts**?

What we think we know…

Errors cluster and we've known this since 1975!

How did you learn to program?

software carpentry



- You don't just go out and write War and Peace
  - You'd read other skilled writers first
- You don't just go out and write a foreign language
  - You learn to read it first
- Foreign programming languages!
- Teach maintenance first
  - Harlan Mills (1990)

"The best way to prepare [to be a programmer] is to write programs and to study great programs that other people have written" Susan Lammers, 1986, quoting a younger Bill Gates

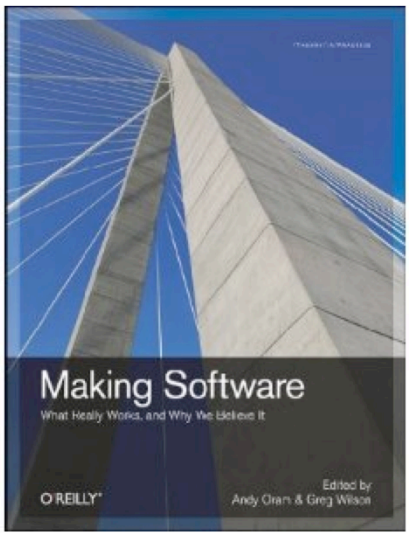
What we think we know…

Quentin Tarantino made Reservoir Dogs…after watching De Palma, Leone, Scorsese, Hawks.

See how its done well (and badly) and learn, then write it yourself.

In more mature subjects they teach reading before writing.
SEI (Carnegie Mellon) tried to collect examples of code to read, gave up. Software engineers cannot decide on what is easy to read.

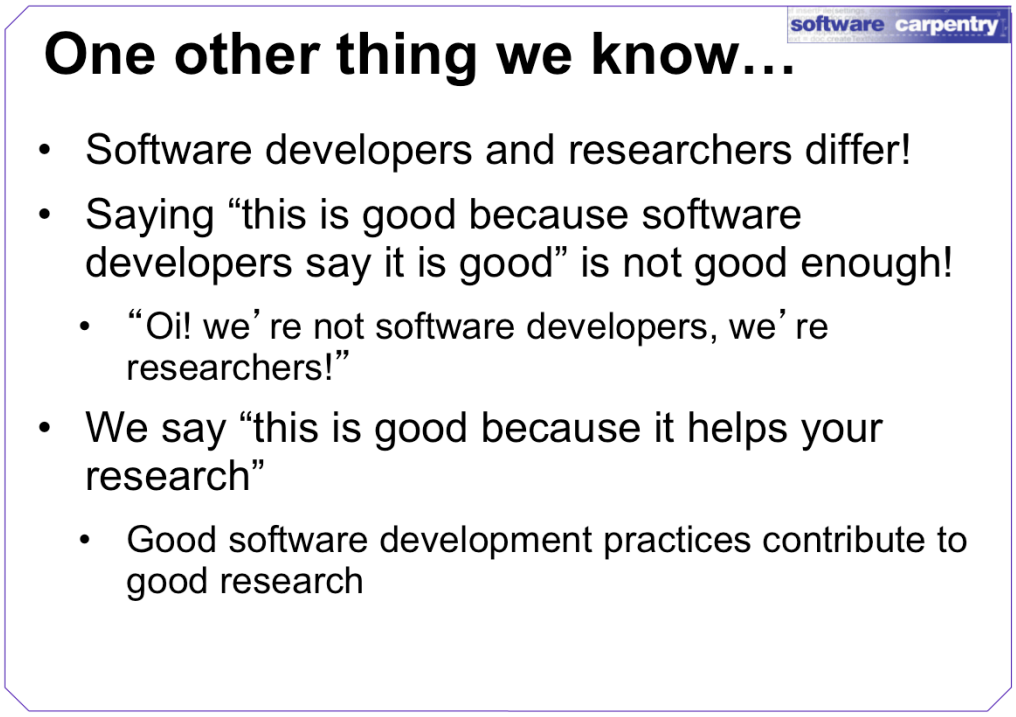
And there's many more…!



http://software-carpentry.org/4_0/softeng/ebse

What we think we know…

Based on empirical results and qualitative studies.
Not just hearsay, "common sense" or "what we've always done".



What we think we know…

Differ in nature of job, outputs (software versus papers, release and users versus peer review) and recognition.

Though we are advocating the recognition of software and data as first-class research outputs in their own right.