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

This work is licensed under the Creative Commons Attribution License
Copyright © Software Carpentry and The University of Edinburgh 2011-2012
See http://software-carpentry.org/license.html for more information.
Once upon a time…

• 7 Years War (1754-63)

• Britain loses 1512 sailors to enemy action…
• …and almost 100,000 to scurvy
But before then...

James Lind (1716-94)

1747: (possibly) first ever controlled medical experiment

<table>
<thead>
<tr>
<th>Cider</th>
<th>Sea water</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sulphuric acid</td>
<td>Oranges</td>
</tr>
<tr>
<td>Vinegar</td>
<td>Barley water</td>
</tr>
</tbody>
</table>
Oranges (and lemons)...

James Lind (1716-94)
1747: (possibly) first ever controlled medical experiment

<table>
<thead>
<tr>
<th>Cider</th>
<th>Sea water</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sulphuric acid</td>
<td>Oranges</td>
</tr>
<tr>
<td>Vinegar</td>
<td>Barley water</td>
</tr>
</tbody>
</table>

• 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!
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 about software development?

- “[Using domain specific language] leads to two primary benefits. The first, and simplest, is improved programmer productivity... The second... is... communication with domain experts.”
Just one more thing...

• Two substantive claims...
• ...without a single citation!
• Where’s the proof!
• The Scottish Verdict!
Those are fightin’ words

• “Many software researchers advocate rather than investigate” – Robert L. Glass (2002)
What we think we know…

Times, they are a changin’

- Growing emphasis on empirical studies since the mid-1990s
- Papers describing new tools or practices routinely include results from some kind of field study
- International Conference on Software Engineering
- Empirical Software Engineering
- “It will never work in theory”
Simply the best?

- “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…
- 1968!
  - Designed to compare batch versus interactive
  - 12 programmers for an afternoon
So what do we know?

- “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…
Think before you act…

• 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

• Users often don’t know what they want till you give them something concrete
Why reading is good…

• Rigorous inspections can remove 60-90% of errors before first test is run
  • Fagan (1975) “Design and Code Inspections to Reduce Errors in Program Development"

• The first review and hour matter most
How did you learn to program?

- 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
- 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
Errors of a feather...

- 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!
Aim high...

- Number of lines ... a programmer can write in a fixed period of time is the same independent of the language used
  - Corbato’s law
- Productivity and reliability depend on length of program’s text independent of language abstraction level
  - Lutz Prechelt
- Use the highest-level language you can
  - Trade off performance
And there’s many more…!

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

Shouldn’t our development practices be built around these, and other, facts?
One other thing we know...

• Software developers and researchers differ!

• Saying “this is good because software developers say it is good” is not good enough!

• We say “this is good because it helps your research”
  • Good software development practices contribute to good research

• …which we hope we’ve demonstrated…
Shell

- Reduce errors by re-using tried-and-tested components
- String together components into powerful computational and data processing pipelines
- Avoid re-inventing the wheel and free up time to do research
Make

• Automate repetitive tasks
• Reduce errors
• Document how your software is built…
• …or how your analyses are run
• Free up time to do research
Version control

• Record the provenance of your code, documents or configuration files

• Track which version of your software produced the data you used in a paper, presentation, poster or thesis

• Back up your work

• Productively collaborate
Python

- Make it easier for colleagues to spot bugs in your code
- Help others to validate what you did by peer review
- Help others replicate or reproduce what you did
- Encourage others to reuse what you did
- Reduce your “truck factor”
Unit testing

• Ensure that your algorithms, implementation and data are correct and so that your research is correct
• Make changes without introducing bugs
• Fix bugs without introducing bugs

• Remember Geoffrey Chang!
What we think we know…

- Pair programming and/or code reviews
- Test-driven development
- Red Green Refactor
- Version control
- Continuous integration
- Iteration and release
All work and no play...

- “Why Crunch Modes Doesn't Work: Six Lessons”, Evan Robinson
- Continuous work reduces cognitive function 25% for every 24 hours!
- Our greatest productivity occurs in the first four to six hours
Iterative development does not just apply to software!

What were the good and bad points about this boot camp?

http://www.surveymonkey.com/s/DD9HJNX