Software Engineering

Empirical Results
Seven Years War
Seven Years War
Actually nine years long
(1754-63)
Seven Years War
Actually nine years long
(1754-63)

Britain lost 1,512 sailors to enemy action
Seven Years War
Actually nine years long (1754-63)

Britain lost 1,512 sailors to enemy action
And almost 100,000 to scurvy
Seven Years War
Actually nine years long
(1754-63)

Britain lost 1,512 sailors to enemy action
And almost 100,000 to scurvy
Unnecessarily
James Lind (1716-94)

1747: first controlled medical experiment in history
James Lind (1716-94)

1747: first controlled medical experiment in history

- cider
- sea water
- vitriol
- oranges
- vinegar
- barley water
James Lind (1716-94)

1747: first controlled medical experiment in history

- cider • sea water
- vitriol • oranges
- vinegar • barley water
James Lind (1716-94)

1747: first controlled medical experiment in history

- cider
- sea water
- vitriol
- oranges
- vinegar
- barley water

Allowed British ships to be effective on long patrols during the Napoleonic Wars
1950: Hill & Doll study comparing smokers and non-smokers
1950: Hill & Doll study comparing smokers and non-smokers

1. Smoking causes lung cancer
1950: Hill & Doll study comparing smokers and non-smokers

1. Smoking causes lung cancer
2. Many people would rather fail than change
1950: Hill & Doll study comparing smokers and non-smokers

1. Smoking causes lung cancer
2. Many people would rather fail than change
1950: Hill & Doll study comparing smokers and non-smokers

1. Smoking causes lung cancer
2. Many people would rather fail than change
A rarity in software engineering until the mid 1990s
A rarity in software engineering until the mid 1990s, but today, papers describing new tools or working practices routinely include results from empirical studies.
A rarity in software engineering until the mid 1990s. But today, papers describing new tools or working practices routinely include results from empirical studies, particularly ones by young researchers.
A rarity in software engineering until the mid 1990s

But today, papers describing new tools or working practices routinely include results from empirical studies Particularly ones by young researchers

Many are flawed or incomplete but standards are constantly improving
Unsurprising in retrospect
Unsurprising in retrospect

Actionable
What goes wrong in developers' first job?

Microsoft
What goes wrong in developers' first job?
What goes wrong in developers' first job?
What goes wrong in developers' first job?

Also unsurprising in retrospect, and actionable
Statistics is just one path
Statistics is just one path
Statistics is just one path
Statistics is just one path

Controlled experiments are expensive...
Statistics is just one path

Controlled experiments are expensive...
...and often eliminate exactly what we want to study
Statistics is just one path

Controlled experiments are expensive...
...and often eliminate exactly what we want to study

Biggest hurdle is re-education
Test-Driven Development
Test-Driven Development

An article of faith among many programmers
Test-Driven Development
An article of faith among many programmers

Meta-analysis of over 30 studies
No consistent effect
Test-Driven Development
An article of faith among many programmers

Meta-analysis of over 30 studies
No consistent effect
• Some positive
Test-Driven Development
An article of faith among many programmers

Meta-analysis of over 30 studies
No consistent effect
• Some positive
• Some negative
Test-Driven Development
An article of faith among many programmers

Meta-analysis of over 30 studies
No consistent effect
- Some positive
- Some negative
- Some inconclusive
Test-Driven Development
An article of faith among many programmers

Meta-analysis of over 30 studies
No consistent effect
• Some positive
• Some negative
• Some inconclusive

The better the study, the weaker the signal
Boehm et al (1975): "Some Experience with Automated Aids to the Design of Large-Scale Reliable Software"
Boehm et al (1975): "Some Experience with Automated Aids to the Design of Large-Scale Reliable Software"

...and many more since
Boehm et al (1975): "Some Experience with Automated Aids to the Design of Large-Scale Reliable Software"

...and many more since

Most errors introduced during requirements analysis and design
Boehm et al (1975): "Some Experience with Automated Aids to the Design of Large-Scale Reliable Software"

...and many more since

Most errors introduced during requirements analysis and design

The later they are removed, the more expensive they are
Pessimists

Optimists
If we tackle the hump in the error injection curve, fewer bugs will get to the expensive part of the fixing curve.

Pessimists

Optimists
If we tackle the hump in the error injection curve, fewer bugs will get to the expensive part of the fixing curve.

Pessimists

If we do shorter iterations, the total cost of fixing bugs will go down.

Optimists
If we tackle the hump in the error injection curve, fewer bugs will get to the expensive part of the fixing curve.

Pessimists

If we do shorter iterations, the total cost of fixing bugs will go down.

Optimists
Fagan 1975: "Design and Code Inspections to Reduce Errors in Program Development"
Fagan 1975: "Design and Code Inspections to Reduce Errors in Program Development"
Reading code carefully is the most cost effective way to find bugs
Fagan 1975: "Design and Code Inspections to Reduce Errors in Program Development"
Reading code carefully is the most cost effective way to find bugs

Cohen 2006: most of the value comes from the first hour and the first pair of eyes
Fagan 1975: "Design and Code Inspections to Reduce Errors in Program Development"
Reading code carefully is the most cost effective way to find bugs

Cohen 2006: most of the value comes from the first hour and the first pair of eyes
Code review now normal in open source
Facts and Fallacies of Software Engineering

Robert L. Glass
Foreword by Alan M. Davis
Do programming languages affect productivity?
Do programming languages affect productivity?

Does using design patterns make for better code?
Do programming languages affect productivity?

Does using design patterns make for better code?

Can we predict software faults statistically?
Do programming languages affect productivity?

Does using design patterns make for better code?

Can we predict software faults statistically?

Is up-front design cost-effective?
Do programming languages affect productivity?

Does using design patterns make for better code?

Can we predict software faults statistically?

Is up-front design cost-effective?

Why is it hard to learn how to program?
Do programming languages affect productivity?

Does using design patterns make for better code?

Can we predict software faults statistically?

Is up-front architecture cost-effective?

Why is it hard to learn how to program?

Is open source software actually better?
Do programming languages affect productivity?

Does using design patterns make for better code?

Can we predict software faults statistically?

Is up-front architecture cost-effective?

Why is it hard to learn how to program?

Is open source software actually better?

Are some programmers 10X better?
narrated by

Greg Wilson

February 2011

Copyright © Software Carpentry 2011
This work is licensed under the Creative Commons Attribution License
See http://software-carpentry.org/license.html for more information.