A Topics Course in Empirical Software Engineering: Bridging Research and Practice

Week 2, Sept 18th 2020

Instructor: Margaret-Anne (Peggy) Storey

Part 2: Beliefs and Evidence in Software Engineering

"A paradigm is a shared world view that represents the beliefs and values in a discipline and that guides how problems are solved."

– Schwandt, 2001

Scientific method

Evidence-based reality

Theory verification and falsification

Quantitative over qualitative

Paradigms – **Postpositivism**

Creswell, 2014

Reality is subjective and experiential

Theory generation

Biases are expected and made explicit

Qualitative over quantitative

Paradigms – **Constructivism**

Creswell, 2014

Change oriented

Collaborative

Shaped by political and social lenses

Qualitative and quantitative

Paradigms – **Advocacy / Participatory**

Problem centered

Real-world practice oriented

Chooses methods as needed

Paradigms – **Pragmatism**

Creswell, 2014

Nature of science...

“Once this comparison took hold, no one bothered checking its validity or utility” [Gould]

Myth 1: Hypotheses --> theories --> laws

Myth 2: Scientific laws and ideas are absolute

Myth 3: A hypothesis is an educated guess (generalizing vs. explanatory hypotheses)

Myth 4: A general and universal scientific method exists (will discuss next week!)

Myth 6: Science and its methods provide absolute proof

Myth 7: Science is procedural more than creative (yeah induction!)

Myth 8: Science and its methods can answer all question (for example?)

Myth 9: Scientists are particularly objective

Myth 10: Experiments are the principal route to scientific knowledge

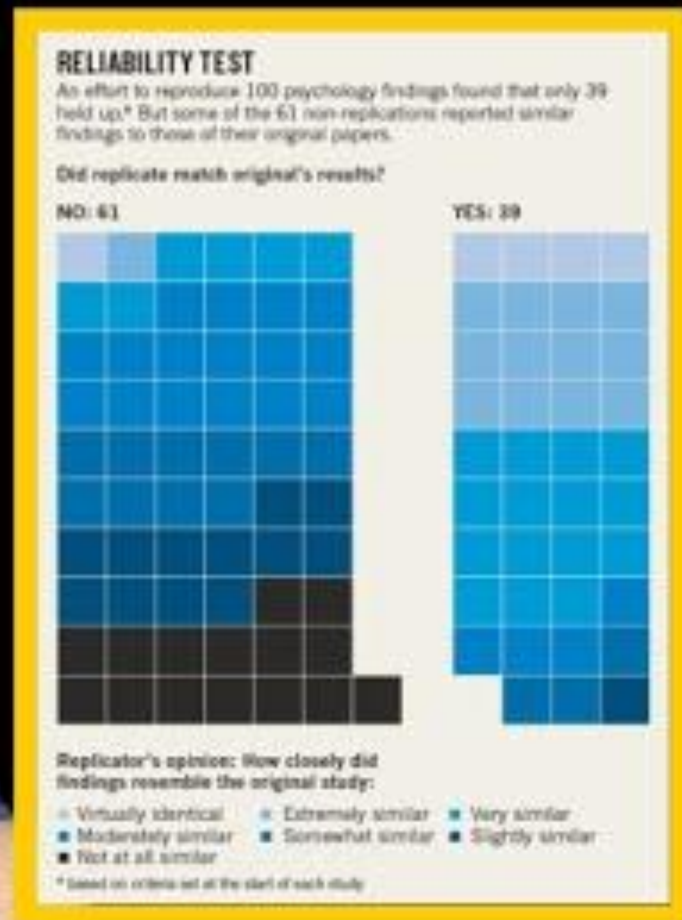
Myth 11: Scientific conclusions are reviewed for accuracy

Myth 13: Science models represent reality

Myth 14: Science and technology are identical (enter Design Science)

The Principal Elements of the Nature of Science: Dispelling the Myths by William F. McComas,1998

Replication crisis in psychology (in software engineering?)



The Changelog – Episode #339

Why smart engineers write bad code

featuring Adam Barr

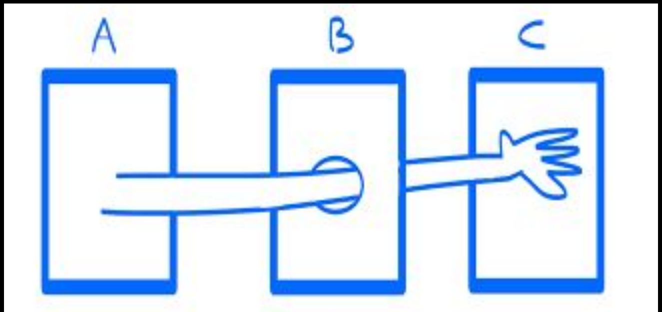

Adam


Adam


Jerod

PLAY DISCUSS SUBSCRIBE SHARE

Is this a law?



Law of Demeter

https://medium.com/@evan.hopkins.us/the-law-of-demeter-and-its-application-to-react-ab1e054f13c5
See also: https://productcoalition.com/ten-laws-of-software-development-cbd72db0f85c

Activity

Let’s replicate a study from a paper!

http://thomas-zimmermann.com/publications/files/devanbu-icse-2016.pdf

Mentimeter link...

(results posted separately online!)

Beliefs and Evidence in SE Activity

Go to menti.com

37 56 99 0

Sheet1

Question
Fixing defects is riskier (more likely to cause future defects) than adding new features.
Code quality (defect occurrence) depends on which programming language is used.
Geographically distributed teams produce code whose quality is just as good as that of teams that aren't geographically d
When it comes to producing code with fewer defects, specific experience in the project matters more than overall progran
Stronger code ownership (fewer people owning a module or a file) leads to better code quality.
Merge commits are buggier than other commits.
Components with more unit tests have fewer customer-found defects.
More defects are found in more complex code.
Using assertions improves code quality.
Using static analysis improves code quality.
Coding standards help improve code quality.
Code review improves code quality.

Beliefs and Evidence in Software Engineering

Question	Score	Variance
Code quality (defect occurrence) depends on which programming language is used [46]	3.17	1.16
Fixing defects is riskier (more likely to cause future defects) than adding new features [34, 48]	2.63	1.08
Geographically distributed teams produce code whose quality (defect occurrence) is just as good as teams that are not geographically distributed [29, 6]	2.86	1.07
When it comes to producing code with fewer defects specific experience in the project matters more than overall general experience in programming [39]	3.5	1.06
Well commented code has fewer defects [52]	3.4	1.05
Code written in a language with static typing (e.g., C#) tends to have fewer bugs than code written in a language with dynamic typing (e.g., Python) [46, 15]	3.75	1.02
Stronger code ownership (i.e, fewer people owning a module or file) leads to better software quality [7, 57, 15]	3.75	1.02
Merge commits are buggier than other commits.	3.4	0.97
Components with more unit tests have fewer customer-found defects [22].	3.85	0.95
More experienced programmers produce code with fewer defects. [34, 39]	3.86	0.94
More defects are found in more complex code. [25]	4.0	0.93
Factors affecting code quality (defect occurrence) vary from project to project. [59, 42]	3.8	0.92
Using asserts improves code quality (reduces defect occurrence) [4, 3]	3.78	0.89
The use of static analysis tools improves end user quality (fewer defects are found by users) [53, 58]	3.77	0.87
Coding standards help improve software quality [8]	4.18	0.79
Code reviews improve software quality (reduces defect occurrence) [38]	4.48	0.64

http://thomas-zimmermann.com/publications/files/devanbu-icse-2016.pdf

Beliefs and Evidence in Software Engineering