Evidence-Based Software Engineering

Bogdan Vasilescu

11/30/2017
Slides from:

- Thomas Zimmermann, Microsoft Research: https://speakerdeck.com/tomzimmermann
- Greg Wilson, Mozilla https://www.slideshare.net/gvwilson/presentations
- Laurie Williams, NC State https://www.slideshare.net/laurieannwilliams/writing-good-software-engineering-research-papers-revisited
- Prem Devanbu, UC Davis https://www.slideshare.net/pdevanbu/belief-evidence/icse
Once Upon a Time...

Seven Years’ War (1754-63)
Britain loses 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    × sea water
× sulfuric acid    ✓ oranges
× vinegar    × barley water

No-one paid attention until a proper Englishman repeated the experiment in 1794...
The British Doctors Study

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

1951: start the British Doctors Study (which runs until 2001)
Two Discoveries

#1: Smoking causes lung cancer

#2: Many people would rather fail than change

“...what happens ‘on average’ is of no help when one is faced with a specific patient...”
Like Water on Stone

1992: Sackett coins the term “evidence-based medicine”

Randomized double-blind trials are accepted as the gold standard for medical research

The Cochrane Collaboration (http://www.cochrane.org/) now archives results from hundreds of medical studies
What about Software Engineering?
What metrics are the **best predictors of failures**?

What is the **data quality** level used in empirical studies and how much does it actually matter?

How can I tell if a piece of software will have **vulnerabilities**?

Do **cross-cutting concerns** cause defects?

Does **Test Driven Development (TDD)** produce better code in shorter time?

I just submitted a **bug report**. Will it be fixed?

If I increase **test coverage**, will that actually increase software quality?

Are there any metrics that are indicators of **failures** in both Open Source and Commercial domains?

**Should I be writing unit tests** in my software project?

Is strong **code ownership** good or bad for software quality?

Does **Distributed/Global software development** affect quality?
Software Engineering is becoming more like modern medicine
The Times They Are A-Changin’

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

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

Yes, many are flawed or incomplete, but standards are constantly improving
## Contributions (RQ2)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Procedure or technique</td>
<td>152 (44%)</td>
<td>195 (37%)</td>
<td>28 (51%)</td>
<td>35 (35%)</td>
<td>18%</td>
<td>18%</td>
</tr>
<tr>
<td>Qualitative or descriptive model</td>
<td>50 (14%)</td>
<td>22 (4%)</td>
<td>4 (7%)</td>
<td>4 (4%)</td>
<td>8%</td>
<td>18%</td>
</tr>
<tr>
<td>Empirical model</td>
<td>4 (1%)</td>
<td>29 (5%)</td>
<td>1 (2%)</td>
<td>5 (5%)</td>
<td>25%</td>
<td>17%</td>
</tr>
<tr>
<td>Analytic model</td>
<td>48 (14%)</td>
<td>54 (10%)</td>
<td>7 (13%)</td>
<td>8 (8%)</td>
<td>15%</td>
<td>15%</td>
</tr>
<tr>
<td>Tool or notation</td>
<td>49 (14%)</td>
<td>83 (16%)</td>
<td>10 (18%)</td>
<td>16 (16%)</td>
<td>20%</td>
<td>19%</td>
</tr>
<tr>
<td>Specific solution</td>
<td>34 (10%)</td>
<td>14 (3%)</td>
<td>5 (9%)</td>
<td>2 (2%)</td>
<td>15%</td>
<td>14%</td>
</tr>
<tr>
<td>Empirical Report</td>
<td>11 (3%)</td>
<td>103 (19%)</td>
<td>0 (0%)</td>
<td>31 (31%)</td>
<td>0%</td>
<td>30%</td>
</tr>
</tbody>
</table>
Validation (RQ3)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis</td>
<td>48 (16%)</td>
<td>72 (14%)</td>
<td>11 (26%)</td>
<td>19 (19%)</td>
<td>23%</td>
<td>26%</td>
</tr>
<tr>
<td>Evaluation</td>
<td>21 (7%)</td>
<td>188 (35%)</td>
<td>1 (2%)</td>
<td>65 (64%)</td>
<td>5%</td>
<td>35%</td>
</tr>
<tr>
<td>Experience</td>
<td>34 (11%)</td>
<td>19 (4%)</td>
<td>8 (19%)</td>
<td>4 (4%)</td>
<td>24%</td>
<td>21%</td>
</tr>
<tr>
<td>Example</td>
<td>82 (27%)</td>
<td>61 (12%)</td>
<td>16 (37%)</td>
<td>1 (1%)</td>
<td>20%</td>
<td>2%</td>
</tr>
<tr>
<td>Underspecified</td>
<td>6 (2%)</td>
<td>94 (18%)</td>
<td>1 (2%)</td>
<td>11 (11%)</td>
<td>17%</td>
<td>12%</td>
</tr>
<tr>
<td>Persuasion</td>
<td>25 (8%)</td>
<td>37 (7%)</td>
<td>0 (0%)</td>
<td>4 (4%)</td>
<td>0%</td>
<td>3%</td>
</tr>
<tr>
<td>No validation</td>
<td>84 (28%)</td>
<td>31 (6%)</td>
<td>6 (14%)</td>
<td>0 (0%)</td>
<td>7%</td>
<td>0%</td>
</tr>
</tbody>
</table>

Analysis/Evaluation/Experience becoming ICSE requirement compared to 2002
What enabled this?
data science / analytics 101
Use of data, analysis, and systematic reasoning to [inform and] make decisions
EARLY “GLOBAL” MODELS AND SOFTWARE ANALYTICS

As soon as people started programming, it became apparent that programming was an inherently buggy process. As recalled by Maurice Wilkes, speaking of his programming experiences from the early 1950s: “It was on one of my journeys between the EDSAC room and the punching equipment that ‘hesitating at the angles of stairs’ the realization came over me with full force that a good part of the remainder of my life was going to be spent in finding errors in my own programs.”

It took several decades to gather the experience required to quantify the size/defect relationship. In 1971, Fumio Akiyama described the first known “size” law, saying the number of defects $D$ was a function of the number of LOC; specifically, $D = 4.86 + 0.018 \times i$. In 1976, Thomas McCabe argued that the number of LOC was less important than the complexity of that code. He argued that code is more likely to be defective when his “cyclomatic complexity” measure was over 10.

Not only is programming an inherently buggy process, it’s also inherently difficult. Based on data from 63 projects, Barry Boehm proposed in 1981 an estimator for development effort that was exponential on program size: $\text{effort} = a \times \text{KLOCb} \times \text{EffortMultipliers}$, where $2.4 \leq a \leq 3$ and $1.05 \leq b \leq 1.2$.

References
the many names
software intelligence
software analytics
software development analytics
analytics for software development
empirical software engineering
mining software repositories
<table>
<thead>
<tr>
<th>Authors</th>
<th>Title</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ahmed E. Hassan, Tao Xie</td>
<td>Software intelligence: the future of mining software engineering data. FoSER 2010: 161-166</td>
<td>[Software Intelligence] offers software practitioners (not just developers) up-to-date and pertinent information to support their daily decision-making processes. [...]</td>
</tr>
<tr>
<td>Raymond P. L. Buse, Thomas Zimmermann</td>
<td>Analytics for software development. FoSER 2010: 77-80</td>
<td>The idea of analytics is to leverage potentially large amounts of data into real and actionable insights.</td>
</tr>
</tbody>
</table>
| Dongmei Zhang, Yingnong Dang, Jian-Guang Lou, Shi Han, Haidong Zhang, and Tao Xie | Software Analytics as a Learning Case in Practice: Approaches and Experiences. MALETSS 2011 | Software analytics is to enable software practitioners to perform data exploration and analysis in order to obtain insightful and actionable information for data-driven tasks around software and services.  
1 Software practitioners typically include software developers, testers, usability engineers, and managers, etc. |
| Raymond P. L. Buse, Thomas Zimmermann | Information needs for software development analytics. ICSE 2012: 987-996 | Software development analytics [...] empower[s] software development teams to independently gain and share insight from their data without relying on a separate entity. |
| Tim Menzies, Thomas Zimmermann | Software Analytics: So What? IEEE Software 30(4): 31-37 (2013) | Software analytics is analytics on software data for managers and software engineers with the aim of empowering software development individuals and teams to gain and share insight from their data to make better decisions. |
| Dongmei Zhang, Shi Han, Yingnong Dang, Jian-Guang Lou, Haidong Zhang, Tao Xie | Software Analytics in Practice. IEEE Software 30(5): 30-37 (2013) | With software analytics, software practitioners explore and analyze data to obtain insightful, actionable information for tasks regarding software development, systems, and users. |
Data Science
trinity of software analytics


MSR Asia Software Analytics group: http://research.microsoft.com/en-us/groups/sa/
Data Scientist: The Sexiest Job of the 21st Century

by Thomas H. Davenport and D.J. Patil

W
When Jonathan Goldman arrived for work in June 2006 at LinkedIn, the business networking site, the place still felt like a start-up. The company had just under 8 million accounts, and the number was growing quickly as existing members invited their friends and colleagues to join. But users weren’t seeking out connections with the people who were already on the site at the rate executives had expected. Something was apparently missing in the social experience. As one LinkedIn manager put it, “It was like arriving at a conference reception and realizing you don’t know anyone. So you just stand in the corner sipping your drink—and you probably leave early.”

Goldman, a PhD in physics from Stanford, was intrigued by the linking he did see going on and by the richness of the user profiles. It all made for messy data and unwieldy analysis, but as he began exploring people’s connections, he started to see possibilities. He began forming theories, testing hunches, and finding patterns that allowed him to predict whose networks a given profile would land in. He could imagine that new features capitalizing on the heuristics he was developing might
Obsessing over our customers is everybody's job. I'm looking to the engineering teams to build the experiences our customers love. [...] In order to deliver the experiences our customers need for the mobile-first and cloud-first world, we will modernize our engineering processes to be **customer-obsessed, data-driven, speed-oriented and quality-focused.**

Each engineering group will have **Data and Applied Science resources** that will focus on measurable outcomes for our products and predictive analysis of market trends, which will allow us to innovate more effectively.

http://news.microsoft.com/ceo/bold-ambition/index.html
Typical data science workflow

[Diagram with steps:
- Acquire data
  - Reformat and clean data
- Preparation
  - Explore alternatives
- Analysis
  - Edit analysis scripts
  - Execute scripts
  - Inspect outputs
  - Debug
- Dissemination
  - Make comparisons
  - Write reports
    - Deploy online
    - Archive experiment
    - Share experiment
- Reflection
  - Take notes
  - Hold meetings]

Background of Data Scientists

Most CS, many interdisciplinary backgrounds
Many have higher education degrees
Strong passion for data

I love data, looking and making sense of the data. [P2]

I’ve always been a data kind of guy. I love playing with data. I’m very focused on how you can organize and make sense of data and being able to find patterns. I love patterns. [P14]

“Machine learning hackers”. Need to know stats

My people have to know statistics. They need to be able to answer sample size questions, design experiment questions, know standard deviations, p-value, confidence intervals, etc.
Background of Data Scientists

PhD training contributes to working style

It has never been, in my four years, that somebody came and said, “Can you answer this question?” I mostly sit around thinking, “How can I be helpful?” Probably that part of your PhD is you are figuring out what is the most important questions. [P13]

I have a PhD in experimental physics, so pretty much, I am used to designing experiments. [P6]

Doing data science is kind of like doing research. It looks like a good problem and looks like a good idea. You think you may have an approach, but then maybe you end up with a dead end. [P5]
Working Styles of Data Scientists

**Insight Provider**

**Specialists**

**Platform Builder**

**Polymath**

**Team Leader**
# Types of data scientists

<table>
<thead>
<tr>
<th><strong>Generalists</strong></th>
<th><strong>Specialists</strong></th>
<th><strong>Manager</strong></th>
<th><strong>Moonlighter</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Polymath</strong></td>
<td><strong>Data Preparer</strong></td>
<td><strong>Data Evangelist / Team Leader</strong></td>
<td><strong>50% Moonlighter</strong></td>
</tr>
<tr>
<td>“describes data scientists who ‘do it all’”</td>
<td><strong>Data Shaper</strong></td>
<td><strong>‘senior data scientists who run their own data science teams act as data science ‘evangelists’”</strong></td>
<td><strong>20% Moonlighter</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Data Analyzer / Insight Provider</strong></td>
<td><strong>Data Businesspeople</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>“main task is to generate insights and to support and guide their managers in decision making”</td>
<td><strong>people who “are most focused on the organization and how data projects yield profit”</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Platform Builder</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>“build shared data platforms used across several product teams”</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Modelling Specialist</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>“data scientists who act as expert consultants and build predictive models”</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>HARRIS ET AL. 2013</strong></td>
<td>Data Creatives “data scientists [who] can often tackle the entire soup-to-nuts analytics process on their own”</td>
<td>Data Developer “people focused on the technical problem of managing data”</td>
<td></td>
</tr>
</tbody>
</table>
Do we really need evidence?

“We hold these Truths to be self-evident, …”
Engineering software is inherently a human venture
My Favorite Little Result


“How long do you think it will take to make a change to this program?”

Control Group: “I’d like to give an estimate for this project myself, but I admit I have no experience estimating. We’ll wait for your calculations for an estimate.”

Group A: “I admit I have no experience with software projects, but I guess this will take about 2 months to finish.”

Group B: “…I guess this will take about 20 months…”
## Results

<table>
<thead>
<tr>
<th>Group</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group A (lowball)</td>
<td>5.1 months</td>
</tr>
<tr>
<td>Control Group</td>
<td>7.8 months</td>
</tr>
<tr>
<td>Group B (highball)</td>
<td>15.4 months</td>
</tr>
</tbody>
</table>

The anchor mattered more than experience, how formal the estimation method was, or anything else.

Q: Are agile projects similarly afflicted, just on a shorter and more rapid cycle?
40 percent of major decisions are based not on facts, but on the manager’s gut.

Accenture survey among 254 US managers in industry. 
http://newsroom.accenture.com/article_display.cfm?article_id=4777
Belief vs evidence

I’m going to do $X$ to avoid bugs
I’m going to *do X* to avoid bugs

**Belief**

But..data says Otherwise!

**Evidence**
Our Approach
Our Approach

Survey
Programmers

Belief
Our Approach

Survey Programmers

Mine+Analyze Repositories

Belief

Evidence
Our Approach

Survey Programmers

Mine+Analyze Repositories

Belief Evidence
Our Approach

Survey
Programmers

Belief
Our Approach

Survey Programmers

Belief

What are the Beliefs?
Our Approach

Survey Programmers

What are the Beliefs?

How strong?

Belief
Our Approach

Survey Programmers

What are the Beliefs?

How strong?

Where do they originate?

Belief
Sample Question
Sample Question

Geographically distributed teams produce code of as good quality as non-distributed teams.
Sample Question

Geographically distributed teams produce code of as good quality as non-distributed teams.

1. Strongly Agree
2. Agree
3. Neutral
4. Disagree
Sample Question

Geographically distributed teams produce code of as good quality as non-distributed teams.

1. Strongly Agree
2. Agree
3. Neutral
4. Disagree
5. Strongly Disagree
Responses

• 2500 surveyed, 564 Responses (22%)

• 497 Male, 53 Female

• 267 Bachelors, 211 Masters, 29 PhD

• 386 US, 66 EU, 48 IN, 39 CN, 25 (Other).
Least Controversial
1. Code Reviews improve Code Quality
Least Controversial

1. Code Reviews improve Code Quality

2. Coding Standards improve code quality
Least Controversial

1. Code Reviews improve Code Quality
2. Coding Standards improve code quality
3. Static Analysis tools improve code quality
Most Controversial
Most Controversial

1. Code Quality depends on programming language
Most Controversial

1. Code Quality depends on programming language
2. Fixing Defects is riskier than adding new features
Most Controversial

1. Code Quality depends on programming language
2. Fixing Defects is riskier than adding new features
3. Geographically distributed teams produce code of as good quality as non-distributed teams.
Opinion Source
Opinion Source

Code quality (defect occurrence) depends on which programming language is used.

1. Strongly Agree
2. Agree
3. Neutral
4. Disagree
5. Strongly Disagree
Opinion Source

Code quality (defect occurrence) depends on which programming language is used.

1. Strongly Agree
2. Agree
3. Neutral
4. Disagree
5. Strongly Disagree
Opinion Source

Code quality (defect occurrence) depends on which programming language is used.

1. Strongly Agree
2. Agree
3. Neutral
4. Disagree
5. Strongly Disagree

Where do they originate?
Opinion Source
Opinion Source

What factors played a role in your previous answer? Please choose the relevant factors, and rank them:

1. Research Papers
2. Articles in Industry Magazines
3. What I hear from my mentors/managers
4. What I hear from my peers
5. Personal Experience
6. Other
Opinion Formation
Belief

Evidence
I BELIEVE

Belief

Evidence
Belief

Evidence

I BELIEVE

..but EVIDENCE Says....
I believe...

but evidence says....

Belief

Evidence

Same? Different?
A Tale of Two Projects

- **Project-A**: Operating System; 400,000 files, 150 Million SLOC, began in the Puget Sound area.

- **Project-B**: Web Service, 430,000 files, 85 Million SLOC, always distributed.

- Both practice distributed (many buildings, cities, regions, and countries) development.

- **Project-B** is a bit more distributed than **Project-A**.
"Geographically distributed teams produce code whose quality (defect occurrence) is just as good as teams that are not geographically distributed”

- Project-A members tended to **DISAGREE**
- Project-B members tended to **AGREE**

$p < 0.001$
A Tale of Two Projects

- **Project-A**: Operating System; 400,000 files, 150 Million SLOC, began in the Puget Sound area.

- **Project-B**: Web Service, 430,000 files, 85 Million SLOC, always distributed.

- Both practice distributed (many buildings, cities, regions, and countries) development.

- **Project-B** is a bit more distributed than **Project-A**.
A Tale of Two Projects

- Project-A: Operating System; 400,000 files, 150 Million SLOC, began in the Puget Sound area.

- Project-B: Web Service, 430,000 files, 85 Million SLOC, always distributed.

- Both practice distributed (many buildings, cities, regions, and countries) development.

- Project-B is a bit more distributed than Project-A.
A Tale of Two Projects

- **Project-A**: Operating System; 400,000 files, 150 Million SLOC, began in the Puget Sound area.
- **Project-B**: Web Service, 430,000 files, 85 Million SLOC, always distributed.
- Both practice distributed (many buildings, cities, regions, and countries) development.
- **Project-B** is a bit more distributed than **Project-A**.

Evidence?

Statistical analysis —> practically no difference
Another example:

Perl - low entry barrier
The Biggest Challenge

http://tinyurl.com/nwit-randomo

Stefik et al: “An Empirical Comparison of the Accuracy Rates of Novices using the Quorum, Perl, and Randomo Programming Languages.” PLATEAU'11

We present here an empirical study comparing the accuracy rates of novices writing software in three programming languages: Quorum, Perl, and Randomo. The first language, Quorum, we call an evidence-based programming language, where the syntax, semantics, and API designs change in correspondence to the latest academic research and literature on programming language usability. Second, while Perl is well known, we call Randomo a Placebo-language, where some of the syntax was chosen with a random number generator and the ASCII table. We compared novices that were programming for the first time using each of these languages, testing how accurately they could write simple programs using common program constructs (e.g., loops, conditionals, functions, variables, parameters). Results showed that while Quorum users were afforded significantly greater accuracy compared to those using Perl and Randomo, Perl users were unable to write programs more accurately than those using a language designed by chance.
A few success stories
DEFECT PREDICTION
Bugs are everywhere
Bugs are everywhere
Quality assurance is limited...

...by time...
Quality assurance is limited...

...by time...

...and by money.
Spent QA resources on the components/files that need it most, i.e., are most likely to fail.
Defect prediction

Software element → Model → Prediction

Model:
- PCA
- Regression
- Bayes
Defect prediction

Model
- PCA
- Regression
- Bayes

Software element
- Metrics
- Churn
- Dependencies
- ...
- ...

Prediction
Defect prediction

Software element

Metrics
Churn
Dependencies
...

Model
PCA
Regression
Bayes

Prediction
Classification
Ranking
...

...
Classification

Has a binary a defect or not?
Ranking

Which binaries have the most defects?

or or ... or
Defect prediction

• Learn a prediction model from historic data
• Predict defects for the same project
• Hundreds of prediction models exist
• Models work fairly well with precision and recall of up to 80%.
Defect prediction

- Learn a prediction model from historic data
- Predict defects for the same project
- Hundreds of prediction models exist
- Models work fairly well with precision and recall of up to 80%.

<table>
<thead>
<tr>
<th>Predictor</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-Release Bugs</td>
<td>73.80%</td>
<td>62.90%</td>
</tr>
<tr>
<td>Test Coverage</td>
<td>83.80%</td>
<td>54.40%</td>
</tr>
<tr>
<td>Dependencies</td>
<td>74.40%</td>
<td>69.90%</td>
</tr>
<tr>
<td>Code Complexity</td>
<td>79.30%</td>
<td>66.00%</td>
</tr>
<tr>
<td>Code Churn</td>
<td>78.60%</td>
<td>79.90%</td>
</tr>
<tr>
<td>Org. Structure</td>
<td>86.20%</td>
<td>84.00%</td>
</tr>
</tbody>
</table>

How many projects to work on at the same time?

Multitasking is common
**EXAMPLE:** GitHub developer (25 Nov 2013 – 18 May 2014)

<table>
<thead>
<tr>
<th></th>
<th>Nov</th>
<th>Dec</th>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mon</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Tue</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Wed</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Thu</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Fri</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sat</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sun</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- **#Projects**
  - 0
  - 1
  - 3
  - 5
  - 8
EXAMPLE: GitHub developer (25 Nov 2013 – 18 May 2014)
**EXAMPLE:** GitHub developer (25 Nov 2013 – 18 May 2014)

<table>
<thead>
<tr>
<th>Day</th>
<th>Nov</th>
<th>Dec</th>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mon</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tue</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wed</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thu</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Fri</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Sat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>Sun</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Software developers multitask too

**EXAMPLE:** GitHub developer (25 Nov 2013 – 18 May 2014)

<table>
<thead>
<tr>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Sat</th>
<th>Sun</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="#" alt="Projects" /></td>
<td><img src="#" alt="Projects" /></td>
<td><img src="#" alt="Projects" /></td>
<td><img src="#" alt="Projects" /></td>
<td><img src="#" alt="Projects" /></td>
<td><img src="#" alt="Projects" /></td>
<td><img src="#" alt="Projects" /></td>
</tr>
</tbody>
</table>

#Projects
- 0
- 1
- 3
- 5
- 8

<table>
<thead>
<tr>
<th>Nov</th>
<th>Dec</th>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="#" alt="Projects" /></td>
<td><img src="#" alt="Projects" /></td>
<td><img src="#" alt="Projects" /></td>
<td><img src="#" alt="Projects" /></td>
<td><img src="#" alt="Projects" /></td>
<td><img src="#" alt="Projects" /></td>
</tr>
</tbody>
</table>
Software developers multitask too

**EXAMPLE:**  GitHub developer (25 Nov 2013 – 18 May 2014)

![Calendar diagram showing #Projects over time]
Software developers multitask too

**EXAMPLE:** GitHub developer (25 Nov 2013 – 18 May 2014)

<table>
<thead>
<tr>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Sat</th>
<th>Sun</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**WHY?**

- Request from other dev’s / management
Software developers multitask too

**EXAMPLE:** GitHub developer (25 Nov 2013 – 18 May 2014)

- **WHY?**
  - Request from other dev’s / management
  - Dependencies
Software developers multitask too

**EXAMPLE:** GitHub developer (25 Nov 2013 – 18 May 2014)

**WHY?**

- Request from other dev’s / management
- Dependencies
- Being “stuck”
- Downtime
Software developers multitask too

**EXAMPLE:** GitHub developer (25 Nov 2013 – 18 May 2014)

![Weekly project tasks](image)

**WHY?**

- Request from other dev’s / management
- Dependencies
- Being “stuck”
- Personal interest
- Downtime
# Software developers multitask too

**EXAMPLE:** GitHub developer (25 Nov 2013 – 18 May 2014)

<table>
<thead>
<tr>
<th></th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Sat</th>
<th>Sun</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nov</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>Dec</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jan</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feb</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mar</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apr</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**WHY?**

- Request from other dev’s / management
- Dependencies
- Being “stuck”
- Downtime
- Personal interest
- Signaling
Hardly any empirical evidence

Rule of thumb (Weinberg, 1992) - not based on data

- Working time available per project
- Loss to context switching

Percent of time

Number of simultaneous projects

From: http://blog.codinghorror.com/the-multi-tasking-myth/
Hardly any empirical evidence

Rule of thumb (Weinberg, 1992) - *not* based on data

- Working time available per project
- Loss to context switching

<table>
<thead>
<tr>
<th>Number of simultaneous projects</th>
<th>Percent of time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>40</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>
Hardly any empirical evidence

Rule of thumb (Weinberg, 1992) - not based on data

![Bar graph showing working time available per project and loss to context switching.](http://blog.codinghorror.com/the-multi-tasking-myth/)

- **Working time available per project**
- **Loss to context switching**

<table>
<thead>
<tr>
<th>Number of simultaneous projects</th>
<th>Working time available per project (%)</th>
<th>Loss to context switching (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>60</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>80</td>
</tr>
</tbody>
</table>
Effects: perception vs. data

“When contributing to multiple projects in parallel, I:”

- 15% increase project success
- 23% resolve more issues
- 29% feel more productive
- 31% contribute more code overall
- 34% review more pull requests
- 52% introduce fewer bugs

Percentage:
- Strongly disagree
- Disagree
- Neutral
- Agree
- Strongly agree
Effects: perception vs. data

“When contributing to multiple projects in parallel, I feel more productive.”

<table>
<thead>
<tr>
<th>Perception</th>
<th>Strongly disagree</th>
<th>Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>15%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>52%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Percentage: 100
50
0
50
100

Effects: perception vs. data

“When contributing to multiple projects in parallel, I feel more productive.”

<table>
<thead>
<tr>
<th>Perception</th>
<th>Strongly disagree</th>
<th>Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>15%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>52%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Percentage: 100
50
0
50
100
Effects: perception vs. data

“When contributing to multiple projects in parallel, I:

- resolve more issues: 15% strongly disagree, 23% disagree, 29% neutral, 31% agree, 34% strongly agree, 52% total
- contribute more code overall: 15% strongly disagree, 23% disagree, 29% neutral, 31% agree, 34% strongly agree, 52% total
- review more pull requests: 15% strongly disagree, 23% disagree, 29% neutral, 31% agree, 34% strongly agree, 52% total
- resolve more issues: 15% strongly disagree, 23% disagree, 29% neutral, 31% agree, 34% strongly agree, 52% total
- introduce fewer bugs: 15% strongly disagree, 23% disagree, 29% neutral, 31% agree, 34% strongly agree, 52% total
- increase project success: 15% strongly disagree, 23% disagree, 29% neutral, 31% agree, 34% strongly agree, 52% total

[Diagram showing the distribution of responses for each statement]
Effects: perception vs. data

“When contributing to multiple projects in parallel, I:”

- Increase project success:
  - 15% Strongly disagree
  - 23% Disagree
  - 29% Neutral
  - 31% Agree
  - 34% Strongly agree
  - 52% Agree

- Introduce fewer bugs:
  - 15% Strongly disagree
  - 23% Disagree
  - 29% Neutral
  - 31% Agree
  - 34% Strongly agree
  - 52% Strongly agree

Percentage:
- 100
- 50
- 0
- 50
- 100
# Effects: perception vs. data

**PERCEPTION**

“When contributing to multiple projects in parallel, I:

<table>
<thead>
<tr>
<th>Percentage</th>
<th>Effect</th>
<th>Agree</th>
<th>Strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>15%</td>
<td>increase project success</td>
<td>47%</td>
<td></td>
</tr>
<tr>
<td>23%</td>
<td>resolve more issues</td>
<td>40%</td>
<td></td>
</tr>
<tr>
<td>29%</td>
<td>feel more productive</td>
<td>33%</td>
<td></td>
</tr>
<tr>
<td>31%</td>
<td>contribute more code overall</td>
<td>29%</td>
<td></td>
</tr>
<tr>
<td>34%</td>
<td>review more pull requests</td>
<td>23%</td>
<td></td>
</tr>
<tr>
<td>52%</td>
<td>introduce fewer bugs</td>
<td>5%</td>
<td></td>
</tr>
<tr>
<td>5%</td>
<td>resolve more issues</td>
<td>40%</td>
<td></td>
</tr>
<tr>
<td>100%</td>
<td>perceive more productive</td>
<td>33%</td>
<td></td>
</tr>
</tbody>
</table>

**Effects: perception vs. data**

- **Perception**: When contributing to multiple projects in parallel, I feel more productive and contribute more code overall.
- **Data**: Increase project success, resolve more issues, and introduce fewer bugs.
Modeling multitasking

- Period matters
Modeling multitasking

- Period matters
- Effort matters
  (A vs. B)
Modeling multitasking

- Period matters
- Effort matters
- Break matters (A vs. D)

<table>
<thead>
<tr>
<th>MON</th>
<th>TUE</th>
<th>WED</th>
<th>THU</th>
<th>FRI</th>
<th>SAT</th>
<th>SUN</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>C</td>
<td>A</td>
<td>A</td>
<td>D</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Modeling multitasking

- Period matters
- Effort matters
- Break matters

**Day-to-day**

**Daily**

**Weekly**

WE MODELED:

- One-week panels
- Three dimensions
Multitasking dimensions

1. Projects per day

Working sequentially vs. Within-day multitasking
Multitasking dimensions

1. PROJECTS PER DAY

Working **sequentially** vs. Within-day **multitasking**

AvgProjectsPerDay = 1

AvgProjectsPerDay = 2.2
Multitasking dimensions

2. WEEKLY FOCUS

Focusing on one project vs. Contributing evenly to all

High focus vs. Low focus
Multitasking dimensions

2. WEEKLY FOCUS

Focusing on one project vs. Contributing evenly to all

High focus

Low focus

\[ S_{\text{Focus}} = 0.25 \]

Shannon entropy:

\[ S_{\text{Focus}} = - \sum_{i=1}^{N} p_i \log_2 p_i \]
Multitasking dimensions

3. DAY-TO-DAY FOCUS

Repetitive day-to-day

<table>
<thead>
<tr>
<th>Project</th>
<th>Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>7</td>
</tr>
</tbody>
</table>

AvgProjectsPerDay = 1
$S_{Focus} = 1$

vs.

Switching focus

<table>
<thead>
<tr>
<th>Project</th>
<th>Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>7</td>
</tr>
</tbody>
</table>

AvgProjectsPerDay = 1
$S_{Focus} = 1$
Multitasking dimensions

3. DAY-TO-DAY FOCUS

Repetitive day-to-day vs. Switching focus

Focus shifting networks
(Xuan et al, 2014)

AvgProjectsPerDay = 1
$S_{Focus} = 1$

AvgProjectsPerDay = 1
$S_{Focus} = 1$
Multitasking dimensions

3. DAY-TO-DAY FOCUS

Repetitive day-to-day vs. Switching focus

<table>
<thead>
<tr>
<th>Project</th>
<th>Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>1, 2, 3, 4, 5, 6, 7</td>
</tr>
</tbody>
</table>

Day 1 to 7 correspond to the columns, and Projects A to D correspond to the rows.
# Multitasking dimensions

## 3. Day-to-day Focus

<table>
<thead>
<tr>
<th>Repetitive day-to-day</th>
<th>vs.</th>
<th>Switching focus</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Project</th>
<th>Day</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

- **Project A**: Day 5
- **Project B**: Day 6
- **Project C**: Day 7
- **Project D**: Day 2, 5, 6, 7
### Multitasking dimensions

#### 3. Day-to-Day Focus

**Repetitive day-to-day** vs. **Switching focus**

<table>
<thead>
<tr>
<th>Project</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>D</td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td>C</td>
</tr>
</tbody>
</table>

**Day**

1  2  3  4  5  6  7
Multitasking dimensions

3. DAY-TO-DAY FOCUS

Repetitive day-to-day vs. Switching focus

Project

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Day

Day 4-5

A

B

C

D

Switching focus

A

C

B

D
Multitasking dimensions

3. DAY-TO-DAY FOCUS

Repetitive day-to-day vs. Switching focus

<table>
<thead>
<tr>
<th>Project</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Day

A

B

C

D
Multitasking dimensions

3. DAY-TO-DAY FOCUS

Repetitive day-to-day vs. Switching focus

Project

<table>
<thead>
<tr>
<th>Day</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Day

A

B

C

D

A

B

C

D
Multitasking dimensions

3. DAY-TO-DAY FOCUS

Repetitive day-to-day vs. Switching focus

Project

1 2 3 4 5 6 7

Day

A
B
C
D

1 1 1 1

A
B
C
D

1 1 1 1 5
Multitasking dimensions

3. DAY-TO-DAY FOCUS

Repetitive day-to-day vs. Switching focus

Day

<table>
<thead>
<tr>
<th>Day</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Switching focus

- Repetitive day-to-day
- Switching focus
Multitasking dimensions

3. DAY-TO-DAY FOCUS

Repetitive day-to-day vs. Switching focus

Markov entropy: \[ S_{\text{Switch}} = - \sum_{i=1}^{N} p_i \sum_{j \in \pi_i} p(j|i) \log_2 p(j|i) \]

How predictable is my focus tomorrow if today I work on project j?
Linear mixed-effects regression

Response:
LOC added / week

Controls:
- time
- total projects
- programming languages

Longitudinal data
- 1,200 developers
- 5+ years each: multiple weeks of observation

Random effect: developer
- developer-to-developer variability in the response

Random slope: time | developer
- developers more productive initially may be less strongly affected by time passing
Multitaskers do more; scheduling matters

Projects per day

Weekly focus

Day-to-day focus (repeatability)
Multitaskers do more; scheduling matters

Projects per day

Weekly focus

Day-to-day focus (repeatability)
Multitaskers do more; scheduling matters

Projects per day

Higher LOC added

Weekly focus

More within-day multitasking

Day-to-day focus (repeatablility)

Higher focus

More repetitive day-to-day work
Multitaskers do more; scheduling matters

Projects per day

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

vs.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

Weekly focus

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>100%</td>
<td>80%</td>
<td>60%</td>
<td>40%</td>
</tr>
</tbody>
</table>

vs.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>100%</td>
<td>80%</td>
<td>60%</td>
<td>40%</td>
</tr>
</tbody>
</table>

Day-to-day focus (repeatability)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

vs.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

Higher focus

More repetitive day-to-day work

Interaction effects:
No scheduling is productive over 5 projects/week

More within-day multitasking
How long will my pull request take?

The pull-based model

… traditionally

Octocats from https://octodex.github.com/
The pull-based model

... traditionally
The pull-based model

... traditionally

Octocats from https://octodex.github.com/
The pull-based model

... traditionally

Octocats from https://octodex.github.com/
The Pull Request process
The Pull Request process

Create a branch
The Pull Request process

Add commits
The Pull Request process

Open a Pull Request
The Pull Request process

Pull Request updates
The Pull Request process

Merge
The pull-based model

submit pull requests

push

... modernly
The pull-based model

... modernly

- Open source-style collaborative development practices in commercial projects using GitHub
  E Kalliamvakou, D Damian, K Blincoe, L Singer, DM German. ICSE 2015
- Work practices and challenges in pull-based development: the integrator's perspective
  G Gousios, A Zaidman, MA Storey, A Van Deursen. ICSE 2015
Considerable review load

- **467 Open** / **12,551 Closed**

<table>
<thead>
<tr>
<th>Pull Request Description</th>
<th>Comments</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move Integer#positive? and Integer#negative? query methods to Numeric</td>
<td>2</td>
<td>meinac</td>
</tr>
<tr>
<td>Deprecate <code>assert_template</code></td>
<td>8</td>
<td>tgxworld</td>
</tr>
<tr>
<td>Add Enumerable#map_with to ActiveSupport</td>
<td>0</td>
<td>mlarraz</td>
</tr>
<tr>
<td>Allow creating a save callback for same name with parent association</td>
<td>2</td>
<td>meinac</td>
</tr>
<tr>
<td>ActiveSupport::HashWithIndifferentAccess select and reject should return enumerator if called without block</td>
<td>0</td>
<td>imanel</td>
</tr>
<tr>
<td>Don’t ignore false values for <code>include_blank</code> passed to Tags::Base#select_content_tag</td>
<td>9</td>
<td>greysteil</td>
</tr>
<tr>
<td>Fix for irregular inflection inconsistency</td>
<td>0</td>
<td>yoongkang</td>
</tr>
<tr>
<td>Add openssl_verify_mode and sync other smtp_settings with API docs</td>
<td>0</td>
<td>jfine</td>
</tr>
</tbody>
</table>
The Pull Request process
... with Travis-CI

- Continuous integration in a social-coding world: Empirical evidence from GITHUB
  B Vasilescu, S van Schuylenburg, J Wulms, A Serebrenik, MGJ van den Brand. ICSME 2014
The Pull Request process
... with Travis-CI

Pull Request is automatically merged into testing branch

- Continuous integration in a social-coding world: Empirical evidence from GITHUB
  B Vasilescu, S van Schuylenburg, J Wulms, A Serebrenik, MGJ van den Brand. *ICSME 2014*
The Pull Request process
… with Travis-CI

Test suite runs automatically

- Continuous integration in a social-coding world: Empirical evidence from GITHUB
  B Vasilescu, S van Schuylenburg, J Wulms, A Serebrenik, MGJ van den Brand. ICSME 2014
The Pull Request process … with Travis-CI

Pull Request is updated in response to test failures

- Continuous integration in a social-coding world: Empirical evidence from GITHUB
  B Vasilescu, S van Schuylenburg, J Wulms, A Serebrenik, MGJ van den Brand. ICSME 2014
The Pull Request process
... with Travis-CI

Tests rerun after update

- Continuous integration in a social-coding world: Empirical evidence from GITHUB
  B Vasilescu, S van Schuylenburg, J Wulms, A Serebrenik, MGJ van den Brand. ICSME 2014
The Pull Request process
... with Travis-CI

More updates

- Continuous integration in a social-coding world: Empirical evidence from GITHUB
  B Vasilescu, S van Schuylenburg, J Wulms, A Serebrenik, MGJ van den Brand. ICSME 2014
The Pull Request process
... with Travis-CI

Tests finally pass

- Continuous integration in a social-coding world: Empirical evidence from GITHUB
  B Vasilescu, S van Schuylenburg, J Wulms, A Serebrenik, MGJ van den Brand. ICSME 2014
The Pull Request process … with Travis-CI

- Continuous integration in a social-coding world: Empirical evidence from GitHub
  B Vasilescu, S van Schuylenburg, J Wulms, A Serebrenik, MGJ van den Brand. ICSME 2014
Merge after CI tests pass

8,842 Total

- **removing unnecessary default parameter in private method**
  #18356 opened on Jan 6 by georgemillo
- **Documenting 'remove_possible_method' and 'redefine_method' [ci skip]**
  #18355 opened on Jan 6 by georgemillo
- **Improve protect_from_forgery documentation.**
  #18354 opened on Jan 6 by simi
- **Propagate bind_values from join in subquery**
  #18350 opened on Jan 5 by brainopia
- **Fix rollback of primarykey-less tables**
  #18349 opened on Jan 5 by jdelStrother
- **Switching SecureTokens to Base58**
  #18347 opened on Jan 5 by robertomiranda
- **Fix TypeError in Fixture creation**
  #18345 opened on Jan 5 by mtthgn
- **Clean up secure_token_test**
  #18344 opened on Jan 5 by jonatack
Merge after CI tests pass
Code review

11.2 hours

16 mins

39 mins

• Wait for it: Determinants of pull request evaluation latency on GitHub
  Y Yu, H Wang, V Filkov, P Devanbu, B Vasilescu. MSR 2015
Merge after CI tests pass
Code review

- Wait for it: Determinants of pull request evaluation latency on GitHub
  Y Yu, H Wang, V Filkov, P Devanbu, B Vasilescu. MSR 2015
PULL REQUEST EVALUATION TIME

HOW TO PREDICT?
**Hypothesis:**

Technical attributes dominate: Size, Complexity, Having Tests
PULL REQUEST EVALUATION TIME

FACTORS

Size
• n_additions
• n_commits

Review
• n_comments

Experience & Social Connections
• merge_rate
• connection_strength
• n_followers

[Gousios et al, ICSE’14, ICSE’15]
[Tsay et al, ICSE’14, FSE’14]
M1: Previously-identified factors

$\checkmark \quad R^2 = 36.2\%$

Size
- n_additions
- n_commits

Review
- n_comments

Experience & Social Connections
- merge_rate
- connection_strength
- n_followers

[Gousios et al, ICSE’14, ICSE’15]
[Tsay et al, ICSE’14, FSE’14]
PULL REQUEST EVALUATION TIME

M2: M1 + process-related factors + continuous integration

Title & description
• n_tokens

[MSR 2015]
M2: M1 + process-related factors + continuous integration

PULL REQUEST EVALUATION TIME

MODELS

Title & description
- n_tokens

Management
- workload
- availability

[MSR 2015]
M2: M1 + process-related factors + continuous integration

Title & description
- n_tokens

Priority
- time_to_first_response

Continuous Integration
- response time

Management
- workload
- availability

[MSR 2015]
PULL REQUEST EVALUATION TIME

M2: M1 + process-related factors + continuous integration
✓ \( R^2 = 58.7\% \)

MODELS

Title & description
- n_tokens

Priority
- time_to_first_response

Continuous Integration
- response time

Management
- workload
- availability

Social tagging
- @mention
- #issue

[MSR 2015]
PULL REQUEST EVALUATION TIME

- Pull request received
- First human response: 16 mins
- CI response: 39 mins
- 11.2 hours for Pull request evaluation
- Pull request closed
Pull request evaluation time

- First human response
- CI response

11.2 hours

16 mins
39 mins

Pull request closed

- Submitter is core developer
- Number of followers
- Strength of social connection

... all stronger predictors than including tests
Science is hard to get right
Two classes of students at Miami University of Ohio that studied object-oriented (OO) design in a one semester course:

- Control group (random sample): OO design class
- Treatment group (volunteers): OO design class + formal methods
  - No statistical difference between the abilities of the two groups on standardized ACT pre-tests

As project, both classes were assigned the development of an elevator system:

- Hand in functioning executable + source code (+ formal specification written using first-order logic)
Standard set of test cases:
- 45.5% of control teams passed all tests
- 100% of treatment teams

Conclusions:
- “formal methods students had increased complex-problem solving skills”
- “the use of formal methods during software development produces ‘better’ programs”

- “Unfortunately, the paper contains several subtle problems. The reader unfamiliar with the basic principles of experimental psychology may easily miss them and interpret the results incorrectly. Not only do we wish to point out these problems, but we also aim to illustrate what to look for when drawing conclusions from controlled experiments.”
• Confounding variables:
  • differences in motivation (treatment group volunteers more motivated)
  • differences in exposure (treatment group more instruction)
  • differences in learning style (treatment group better learners)
  • differences in skills (outside of ACT)
  • Novelty effects
  • …
Why big data needs thick data
Why big data needs thick data
Why big data needs thick data

• Looking for answers in the wrong places:
  
  • A/B testing doesn’t say anything about why users prefer a certain feature
Why big data needs thick data

• Looking for answers in the wrong places:

• A/B testing doesn't say anything about why users prefer a certain feature

• Reality distortion field:

• From 2006 to 2011 the US murder rate was well correlated with the market share of Internet Explorer: Both went down sharply
Why big data needs thick data

• Looking for answers in the wrong places:

  • A/B testing doesn’t say anything about why users prefer a certain feature

• Reality distortion field:

  • From 2006 to 2011 the US murder rate was well correlated with the market share of Internet Explorer: Both went down sharply

  • “Data is like people – interrogate it hard enough and it will tell you whatever you want to hear”
Anscombe's quartet
Why big data needs thick data

• Looking for answers in the wrong places:
  • A/B testing doesn’t say anything about why users prefer a certain feature

• Reality distortion field:
  • From 2006 to 2011 the US murder rate was well correlated with the market share of Internet Explorer: Both went down sharply
  • “Data is like people – interrogate it hard enough and it will tell you whatever you want to hear”

• Which data should we collect? What is the meaning of the data that is collected? How should the insights be shared and used?
<table>
<thead>
<tr>
<th>Question</th>
<th>Unwise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Which individual measures correlate with employee productivity (e.g. employee age, tenure, engineering skills, education, promotion velocity, IQ)?</td>
<td>25.5%</td>
</tr>
<tr>
<td>Which coding measures correlate with employee productivity (e.g. lines of code, time it takes to build software, particular tool set, pair programming, number of hours of coding per day, programming language)?</td>
<td>22.0%</td>
</tr>
<tr>
<td>What metrics can use used to compare employees?</td>
<td>21.3%</td>
</tr>
<tr>
<td>How can we measure the productivity of a Microsoft employee?</td>
<td>20.9%</td>
</tr>
<tr>
<td>Is the number of bugs a good measure of developer effectiveness?</td>
<td>17.2%</td>
</tr>
<tr>
<td>Can I generate 100% test coverage?</td>
<td>14.4%</td>
</tr>
<tr>
<td>Who should be in charge of creating and maintaining a consistent company-wide software process and tool chain?</td>
<td>12.3%</td>
</tr>
<tr>
<td>What are the benefits of a consistent, company-wide software process and tool chain?</td>
<td>10.4%</td>
</tr>
<tr>
<td>When are code comments worth the effort to write them?</td>
<td>9.6%</td>
</tr>
<tr>
<td>How much time and money does it cost to add customer input into your design?</td>
<td>8.3%</td>
</tr>
<tr>
<td>Turkish</td>
<td>English</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>o bir aşıçıcı</td>
<td>she is a cook</td>
</tr>
<tr>
<td>o bir mühendis</td>
<td>he is an engineer</td>
</tr>
<tr>
<td>o bir doktor</td>
<td>he is a doctor</td>
</tr>
<tr>
<td>o bir hemşire</td>
<td>she is a nurse</td>
</tr>
<tr>
<td>o bir temizlikçi</td>
<td>he is a cleaner</td>
</tr>
<tr>
<td>o bir polis</td>
<td>He-she is a police</td>
</tr>
<tr>
<td>o bir asker</td>
<td>he is a soldier</td>
</tr>
<tr>
<td>o bir öğretmen</td>
<td>She's a teacher</td>
</tr>
<tr>
<td>o bir sekreter</td>
<td>he is a secretary</td>
</tr>
<tr>
<td>o bir arkadaş</td>
<td>he is a friend</td>
</tr>
<tr>
<td>o bir sevgili</td>
<td>she is a lover</td>
</tr>
<tr>
<td>onu sevmiyor</td>
<td>she does not like her</td>
</tr>
<tr>
<td>onu seviyor</td>
<td>she loves him</td>
</tr>
<tr>
<td>onu görüyor</td>
<td>she sees it</td>
</tr>
<tr>
<td>onu göremiyor</td>
<td>he can not see him</td>
</tr>
<tr>
<td>o onu kucakliyor</td>
<td>she is embracing her</td>
</tr>
<tr>
<td>o onu kucaklamiyor</td>
<td>he does not embrace it</td>
</tr>
<tr>
<td>o evli</td>
<td>she is married</td>
</tr>
<tr>
<td>o bekar</td>
<td>he is single</td>
</tr>
<tr>
<td>o mutlu</td>
<td>he's happy</td>
</tr>
<tr>
<td>o mutsuz</td>
<td>she is unhappy</td>
</tr>
<tr>
<td>o çalışkan</td>
<td>he is hard working</td>
</tr>
<tr>
<td>o tembel</td>
<td>she is lazy</td>
</tr>
</tbody>
</table>