Carnegie Mellon

institute for SOFTWARE RESEARCH

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/beliefevidenceicse

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

× sulfuric acid

× vinegar

× sea water

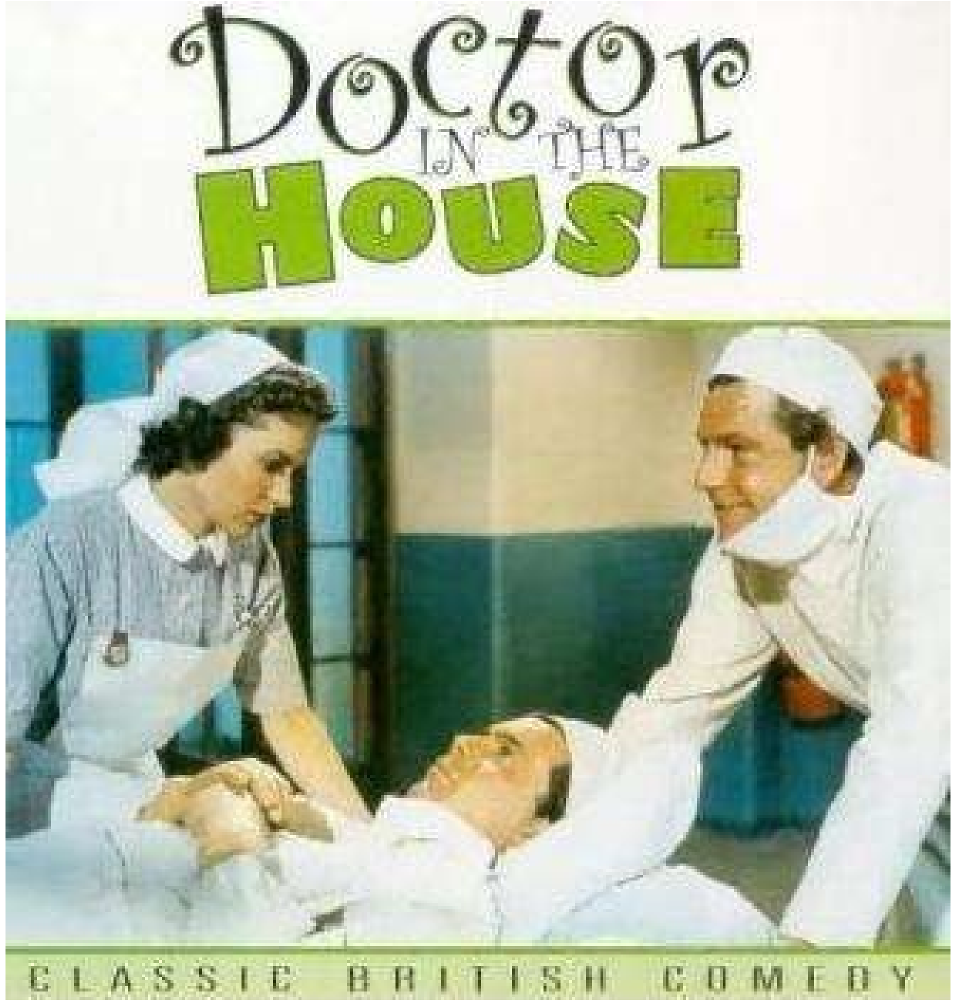
√**oranges**

×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**?

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

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

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

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

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

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

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

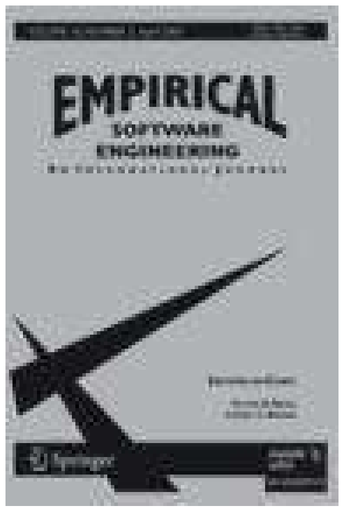
Is strong **code ownership** good or bad for software quality?

Does **Distributed/Global software development** affect quality?

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

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

NC STATE UNIVERSITY

Contributions (RQ2)

Types of research contribution in ICSE 2016 submissions and acceptances						
Type of contribution	Submitted (2002)	Submitted (2016)	Accepted (2002)	Accepted (2016)	Ratio (2002)	Ratio (2016)
Procedure or technique	152 (44%)	195 (37%)	28 (51%)	35 (35%)	18%	18%
Qualitative or descriptive model	50 (14%)	22 (4%)	4 (7%)	4 (4%)	8%	18%
Empirical model	4 (1%)	29 (5%)	1 (2%)	5 (5%)	25%	17%
Analytic model	48 (14%)	54 (10%)	7 (13%)	8 (8%)	15%	15%
Tool or notation	49 (14%)	83 (16%)	10 (18%)	16 (16%)	20%	19%
Specific solution	34 (10%)	14 (3%)	5 (9%)	2 (2%)	15%	14%
Empirical Report	11 (3%)	103 (19%)	0 (0%)	31 (31%)	0%	**30%**

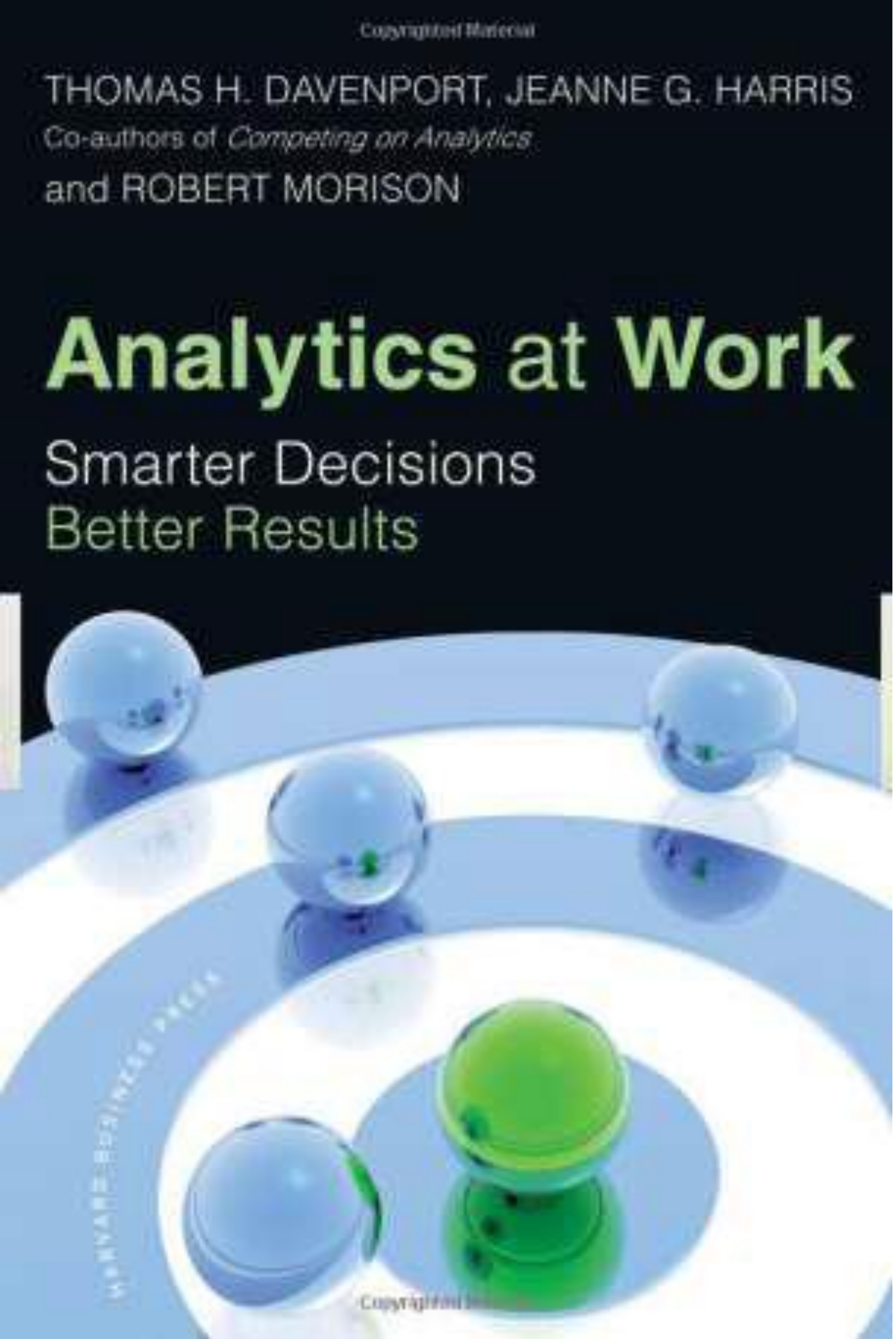
Validation (RQ3)

Types of Validation in ICSE 2016 submissions and acceptances						
Type of result	Submitted (2002)	Submitted (2016)	Accepted (2002)	Accepted (2016)	Ratio (2002)	Ratio (2016)
Analysis	48 (16%)	72 (14%)	11 (26%)	19 (19%)	23%	26%
Evaluation	21 (7%)	188 (35%)	1 (2%)	65 (64%)	5%	35%
Experience	34 (11%)	19 (4%)	8 (19%)	4 (4%)	24%	21%
Example	82 (27%)	61 (12%)	16 (37%)	1 (1%)	20%	2%
Underspecified	6 (2%)	94 (18%)	1 (2%)	11 (11%)	17%	12%
Persuasion	25 (8%)	37 (7%)	0 (0%)	1 (1%)	0%	3%
No validation	84 (28%)	31 (6%)	6 (14%)	0 (0%)	7%	0%

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

history of software analytics

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,[1] 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[2] described the first known "size" law, saying the number of defects D was a function of the number of LOC; specifically, $D = 4.86 + 0.018 * i$. In 1976, Thomas McCabe argued that the number of LOC was less important than the complexity of that code.[3] 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[4] proposed in 1981 an estimator for development effort that was exponential on program size: $\text{effort} = a * \text{KLOC}b * \text{EffortMultipliers}$, where $2.4 \le a \le 3$ and $1.05 \le b \le 1.2$.

References

1. M. Wilkes, *Memoirs of a Computer Pioneer*, MIT Press, 1985.
2. F. Akiyama, "An Example of Software System Debugging," *Information Processing*, vol. 71, 1971, pp. 353–359.
3. T. McCabe, "A Complexity Measure," *IEEE Trans. Software Eng.*, vol. 2, no. 4, 1976, pp. 308–320.
4. B. Boehm, *Software Engineering Economics*, Prentice-Hall, 1981.

Tim Menzies, Thomas Zimmermann: Software Analytics: So What?
IEEE Software 30(4): 31-37 (2013)

Microsoft Research

the many names

software intelligence
software analytics
software development analytics
analytics for software development
empirical software engineering
mining software repositories

Microsoft Research

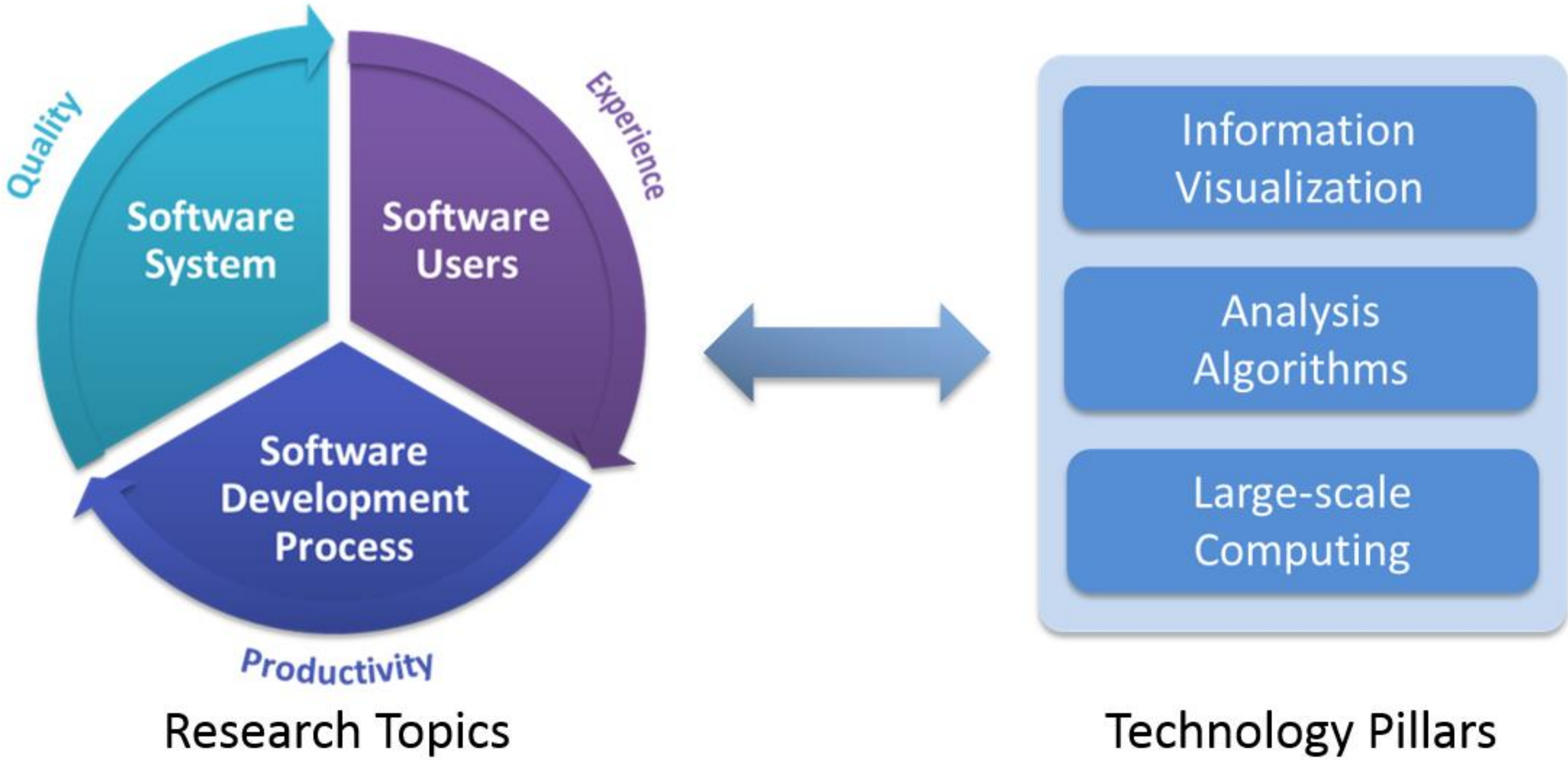
STRUDEL

SOCIO-TECHNICAL RESEARCH USING DATA EXCAVATION LAB

Ahmed E. Hassan, Tao Xie: Software intelligence: the future of mining software engineering data. FoSER 2010: 161-166	[Software Intelligence] offers software practitioners (not just developers) up-to-date and pertinent information to support their daily decision-making processes. […]
Raymond P. L. Buse, Thomas Zimmermann: Analytics for software development. FoSER 2010: 77-80	The idea of analytics is to leverage potentially large amounts of data into real and actionable insights.
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. MALETS 2011	*Software analytics* is to enable software practitioners[1] 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



Dongmei Zhang, Shi Han, Yingnong Dang, Jian-Guang Lou, Haidong Zhang, Tao Xie: Software Analytics in Practice. IEEE Software 30(5): 30-37, September/October 2013.

MSR Asia Software Analytics group: http://research.microsoft.com/en-us/groups/sa/

Microsoft Research

ARTWORK: TAMAR COHEN, ANDREW J BUBOLTZ, 2011, SILK SCREEN ON A PAGE FROM A HIGH SCHOOL YEARBOOK, 8.5" X 12"

DATA

Data Scientist: The Sexiest Job of the 21st Century

by Thomas H. Davenport and D.J. Patil

FROM THE OCTOBER 2012 ISSUE

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

WHAT TO READ NEXT
Big Data: The Management Revolution
Making Advanced Analytics Work for You
Google Flu Trends' Failure Shows Good Data > Big Data

VIEW MORE FROM THE
October 2012 Issue

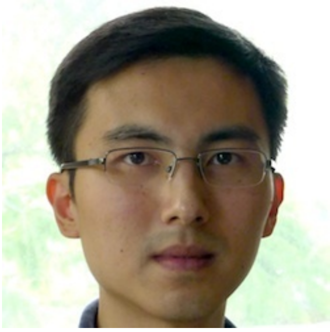
Obsessing over our customers is everybody's job. I'm looking to the engineering teams to **build the experiences our customers love**. […] In order to deliver the experiences our customers need for the mobile-first and cloud-first world, we will modernize our engineering processes to be **customer-obsessed, data-driven, speed-oriented and quality-focused.**

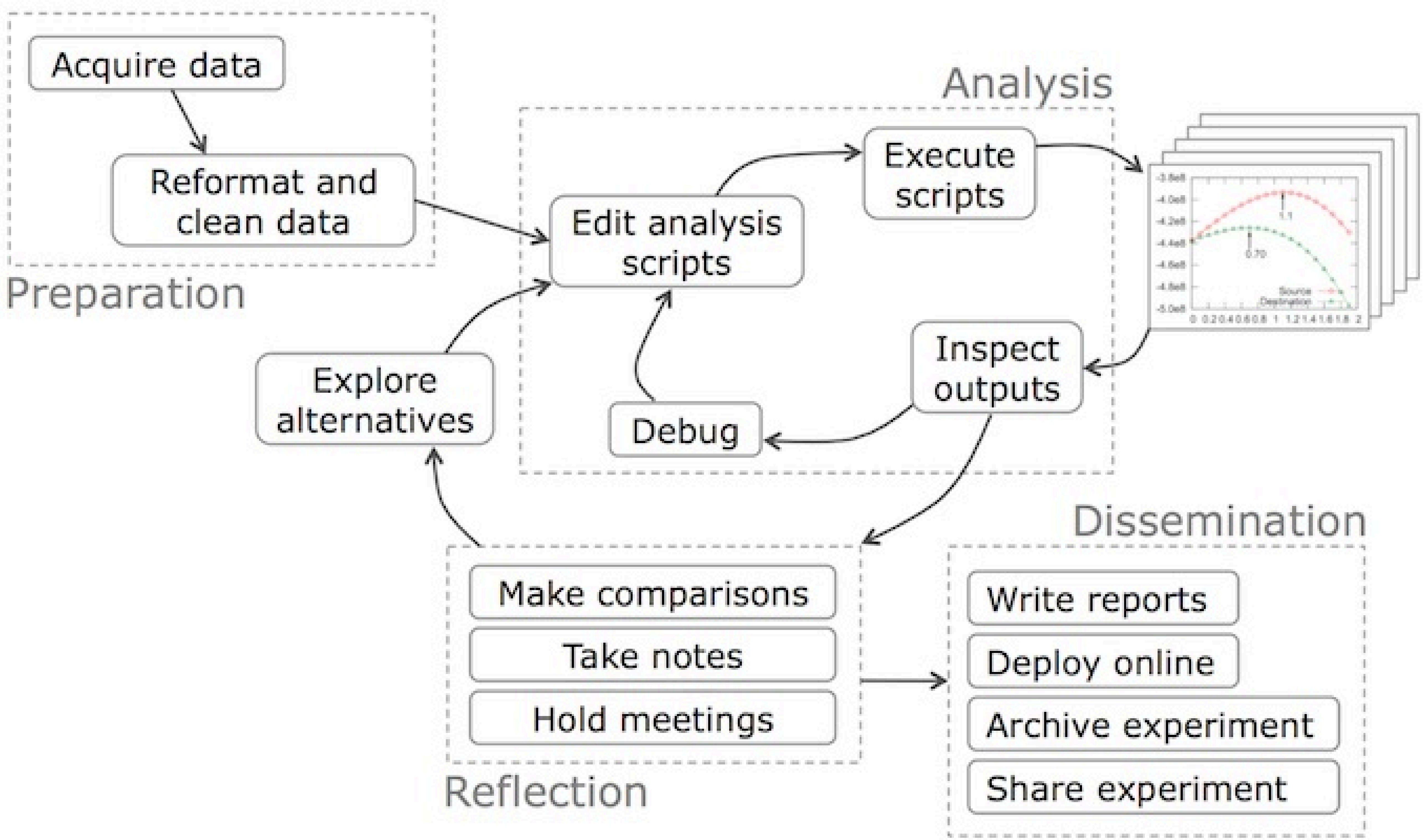
http://news.microsoft.com/ceo/bold-ambition/index.html

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





https://cacm.acm.org/blogs/blog-cacm/169199-data-science-workflow-overview-and-challenges/

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

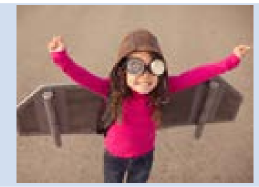
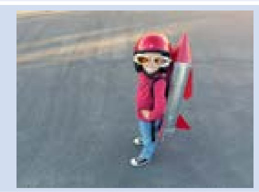
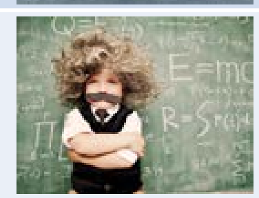
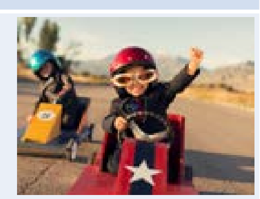
Specialists

Platform Builder

Polymath

Team Leader

Types of data scientists

			Harris et al. 2013
Generalists	Polymath “describes data scientists who ‘do it all’ ”		Data Creatives “data scientists [who] can often tackle the entire soup-to-nuts analytics process on their own”
Specialists	Data Preparer		
	Data Shaper		
	Data Analyzer / Insight Provider “main task is to generate insights and to support and guide their managers in decision making”		
	Platform Builder “build shared data platforms used across several product teams”		Data Developer “people focused on the technical problem of managing data”
	Modelling Specialist “data scientists who act as expert consultants and build predictive models”		
			Data Researcher people with “deep academic training in the use of data to understand complex processes”
Manager	Data Evangelist / Team Leader “senior data scientists who run their own data science teams act as data science ‘evangelists’ ”		Data Businesspeople people who “are most focused on the organization and how data projects yield profit”
	Insight Actor		
Moonlighter	50% Moonlighter		
	20% Moonlighter		

Do we <u>really</u> need evidence?

“We hold these Truths to be **self-evident**, …”

Engineering software is inherently a human venture

My Favorite Little Result

Aranda & Easterbrook (2005): "Anchoring and Adjustment in Software Estimation"

"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

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

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

Microsoft® Research

Belief vs evidence

Devanbu, P., Zimmermann, T., & Bird, C. (2016, May). Belief & evidence in empirical software engineering. In *Proceedings of the 38th international conference on software engineering* (pp. 108-119). ACM.

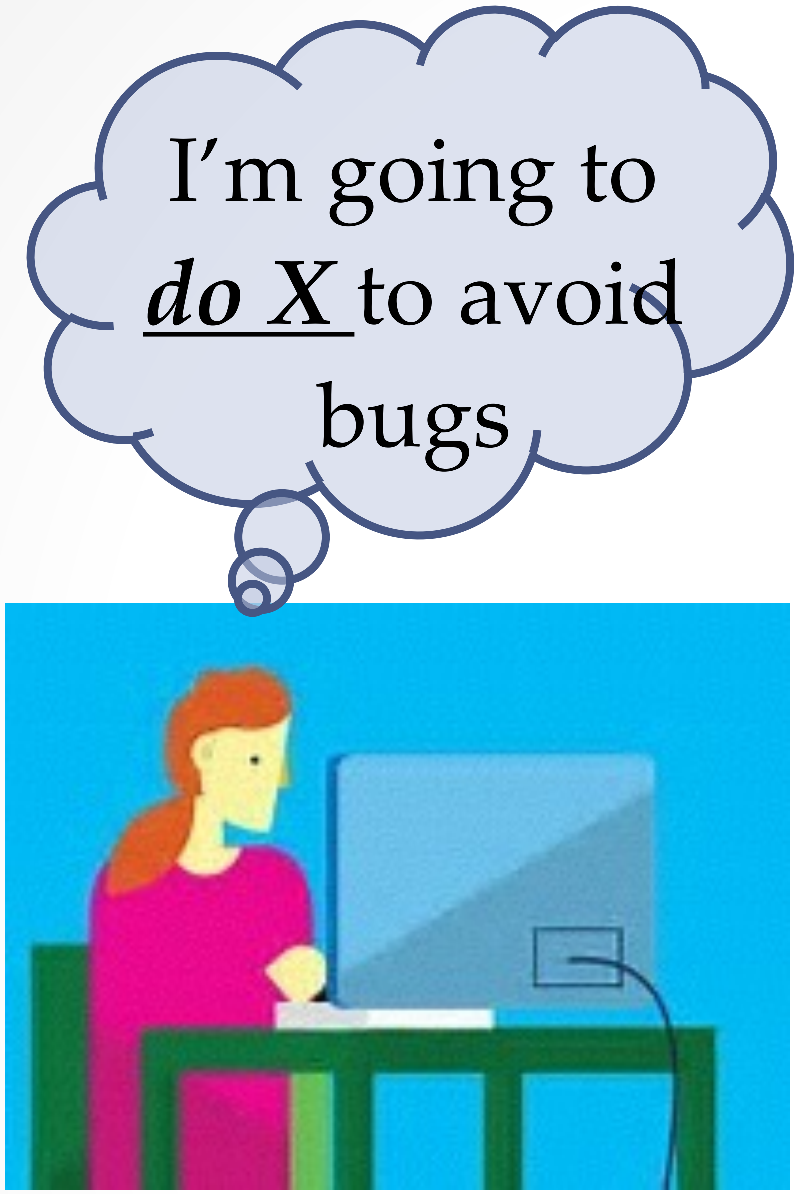
Belief

Evidence

I’m going to ***do X*** to avoid bugs

Belief

Evidence



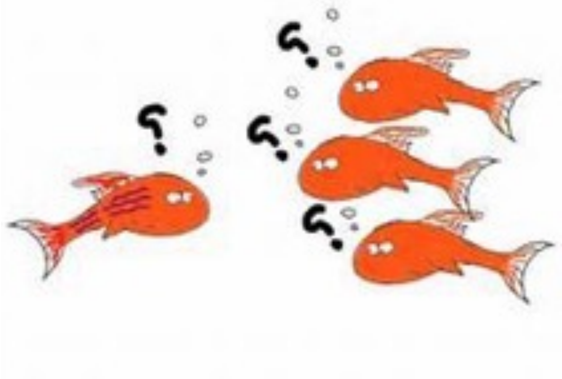
Belief

Evidence

Our Approach

Our Approach

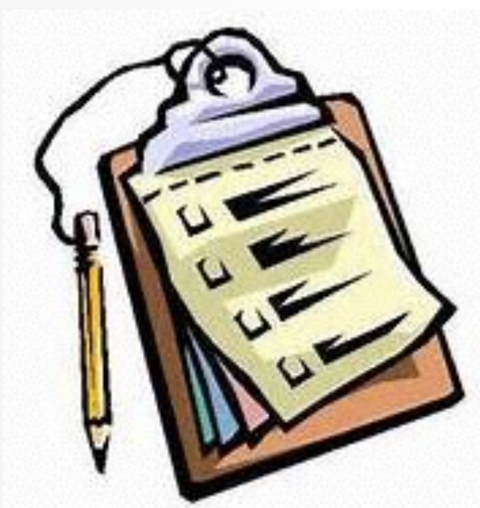
Survey
Programmers

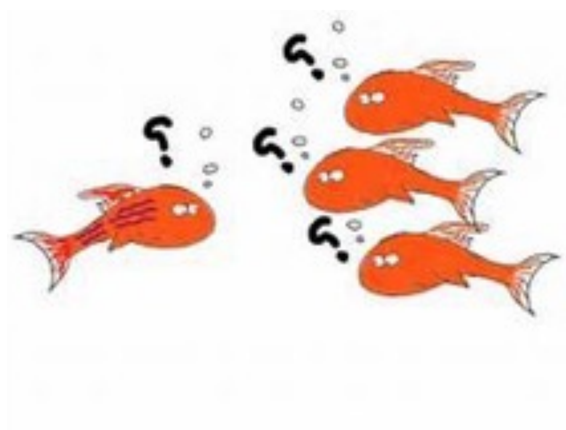


Belief

Our Approach

Survey Programmers





Belief

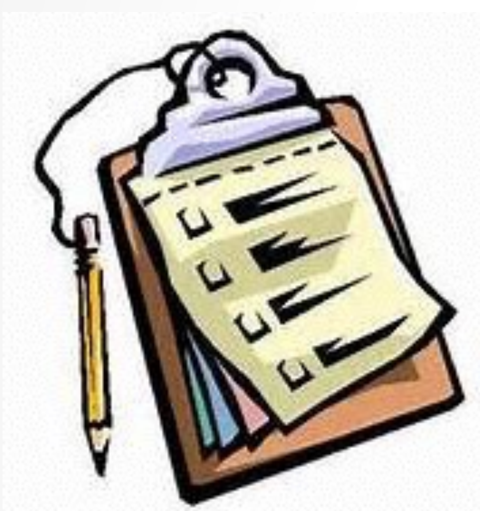
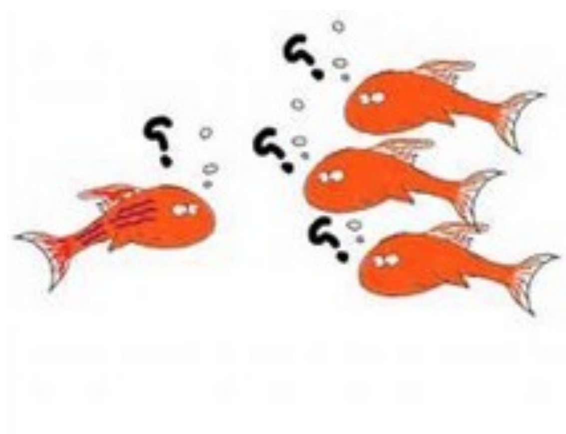
Mine+Analyze Repositories

Evidence

Our Approach

Survey Programmers

Mine+Analyze Repositories

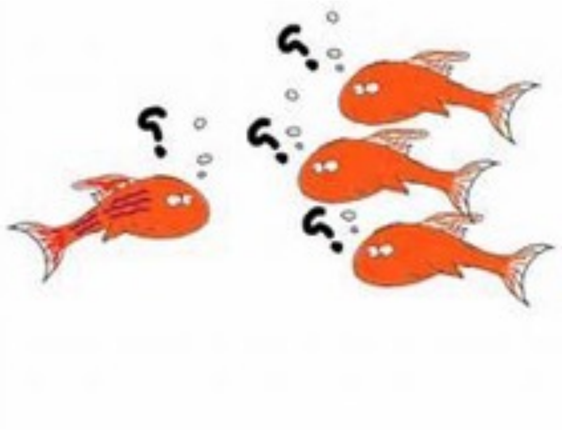
Belief

Evidence

??

Our Approach

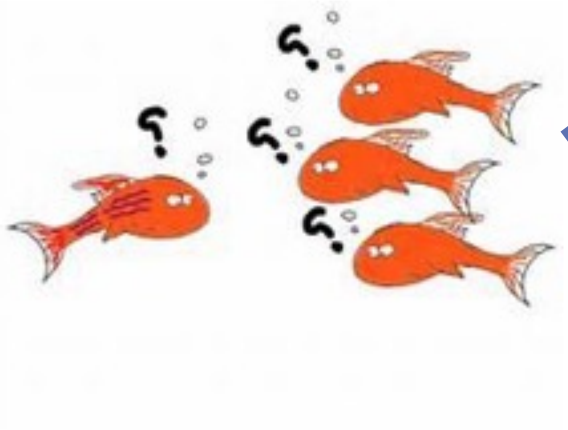
Survey
Programmers

Belief

Our Approach

Survey Programmers

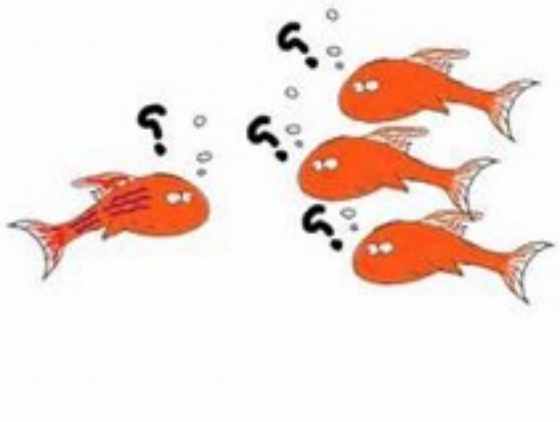


What are the Beliefs?

Belief

Our Approach

Survey
Programmers



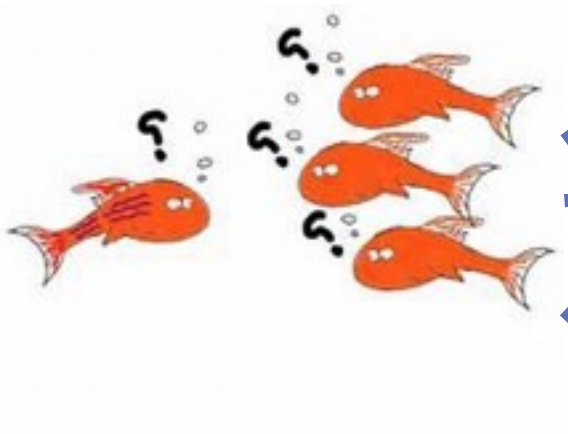
What are the Beliefs?

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

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?

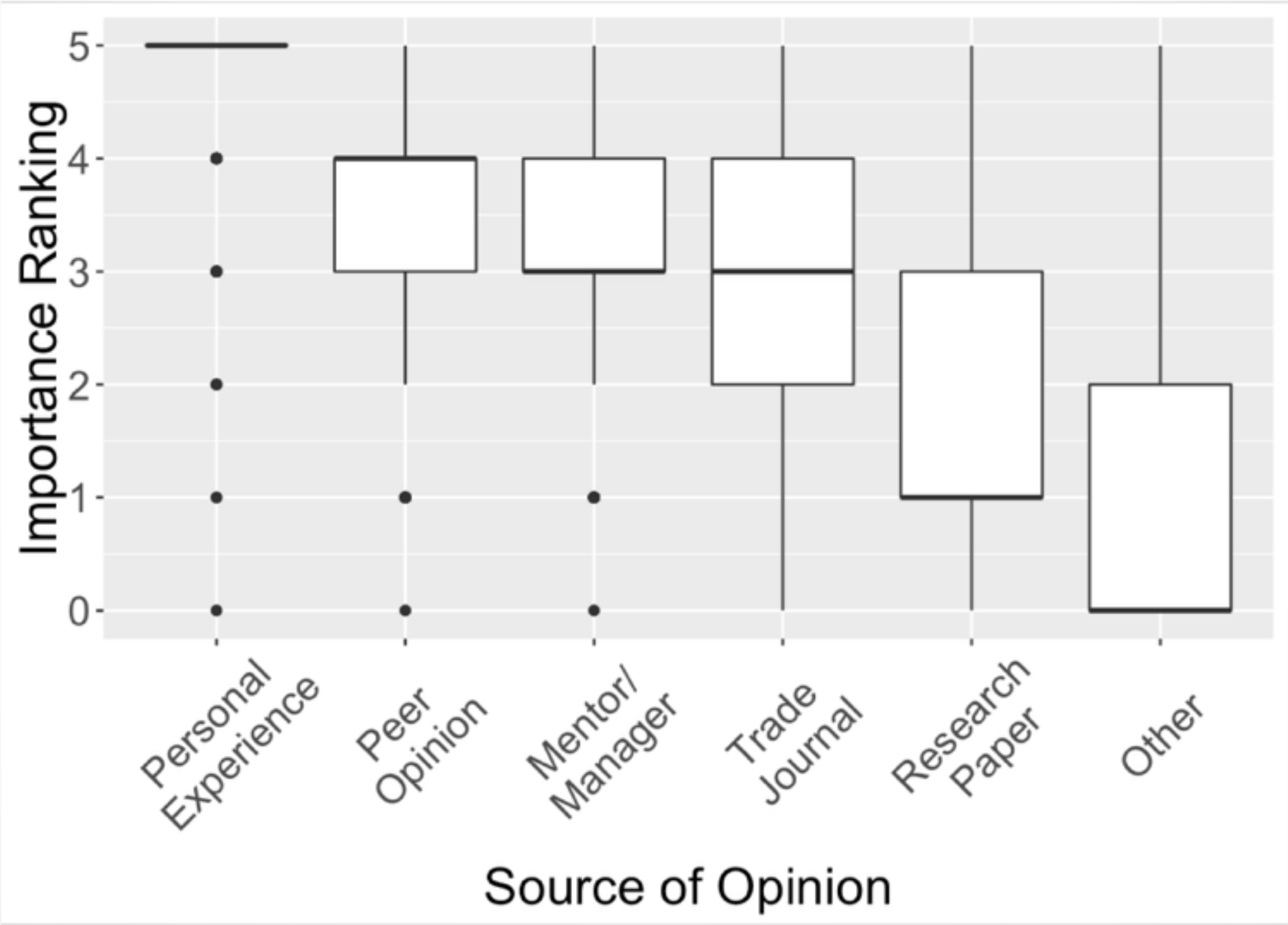
Opinion Source

Opinion Source

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

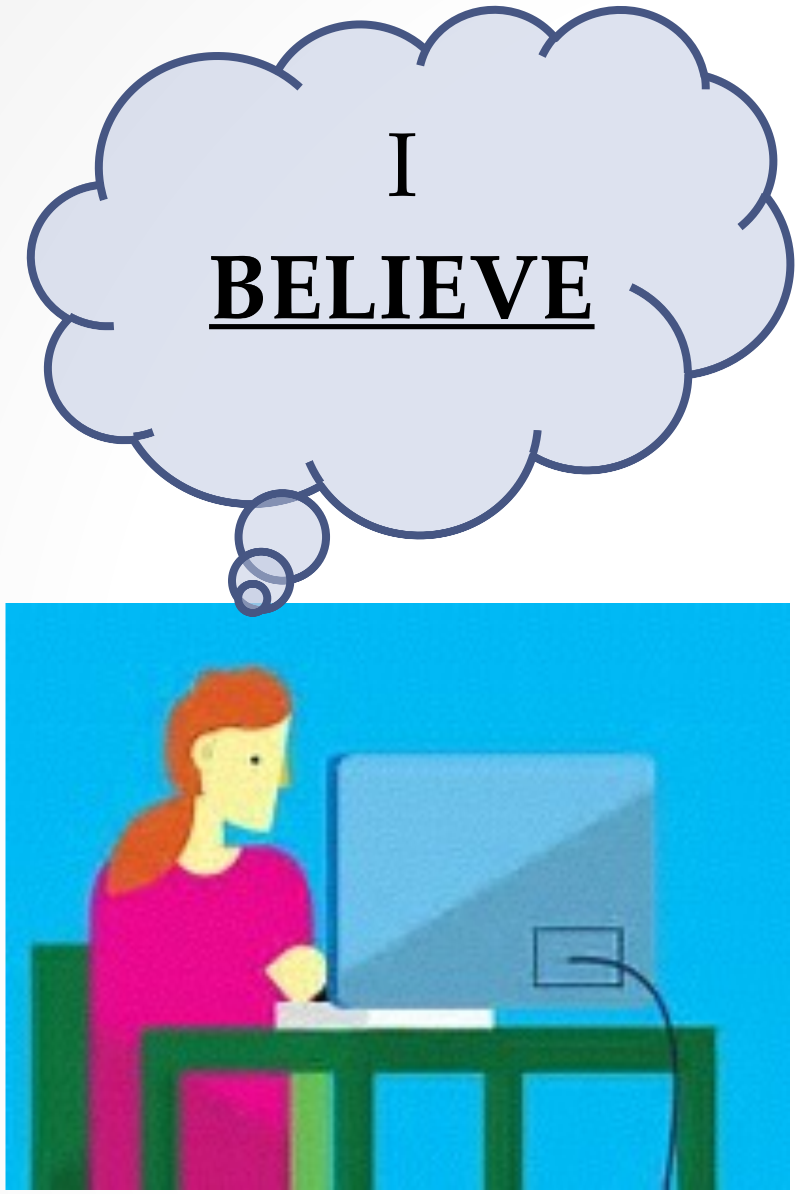
1. Research Papers
2. Articles in Industry Magazines
3. What I hear from my mentors/managers
4. What I hear from my peers
5. Personal Experience
6. Other

Opinion Formation



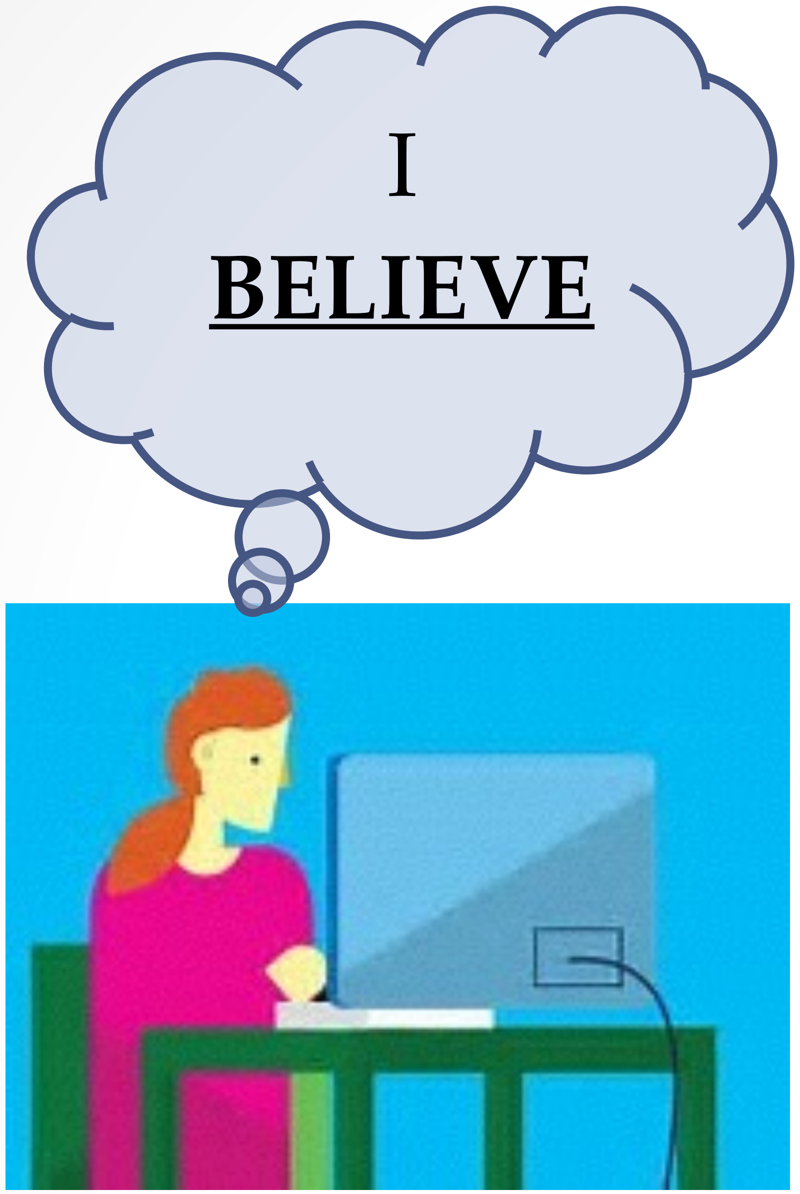
Belief

Evidence



Belief

Evidence



Belief

Evidence

I

<u>**BELIEVE**</u>

..but

<u>**EVIDENCE**</u>

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*.

A Tale of Two Projects

- *Project*-A: Operating System; 400,000 files, 150 Million SL

"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

Evidence?

- *Project* …, 150 Million SLOC, …
- *Project* … illion 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* … , 150 Million SLOC, …
- *Project* … illion SLOC, always distributed.
- Both … cities, regio…
- *Proje* … *ect-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.

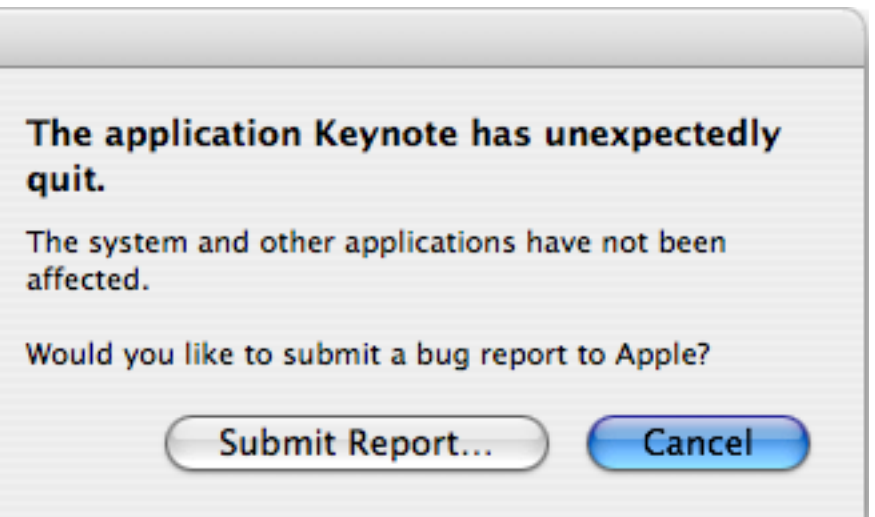
A few success stories

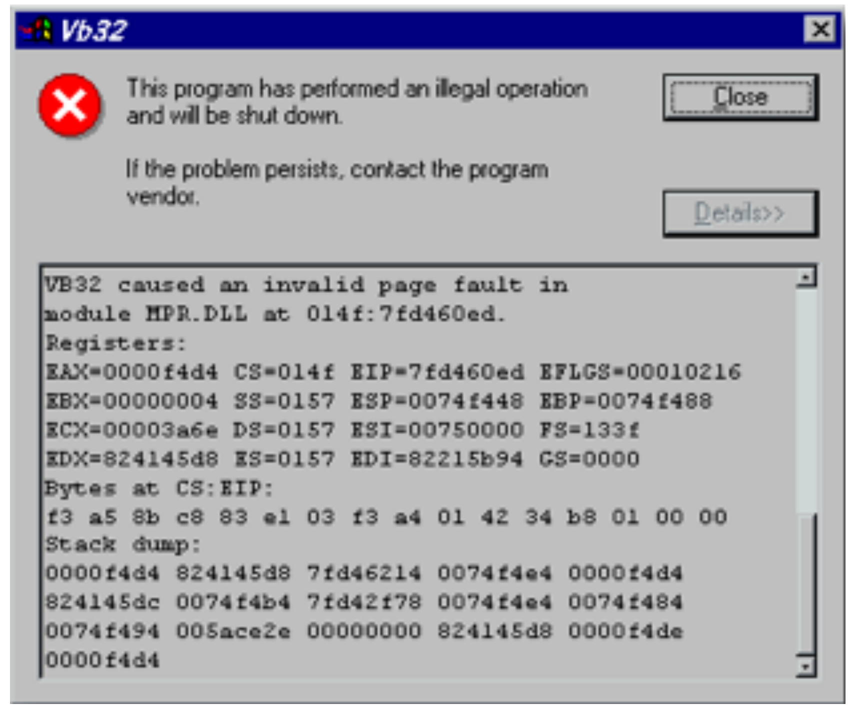
Microsoft Research

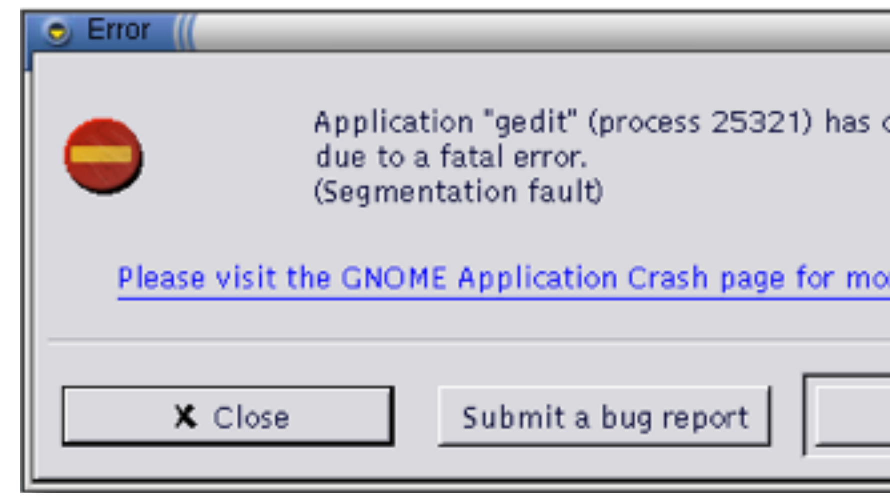
DEFECT PREDICTION

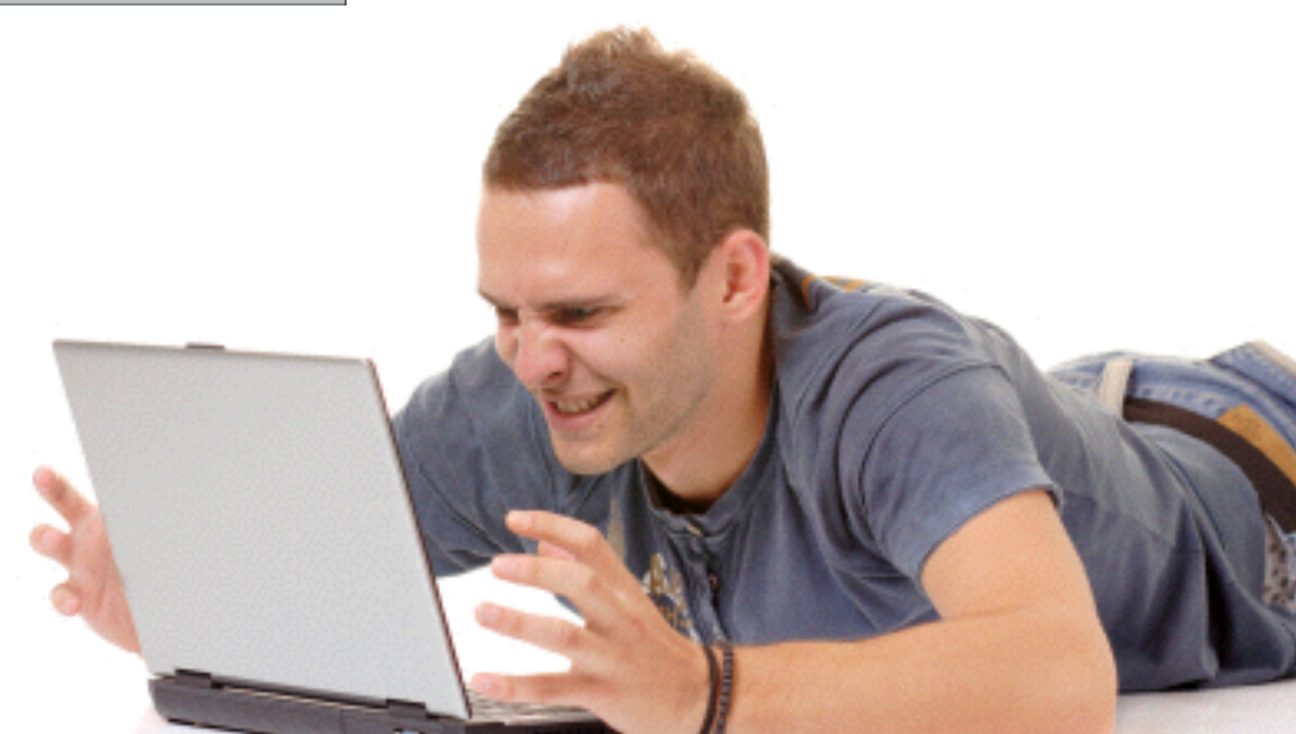
Bugs are everywhere


Bugs are everywhere









Microsoft Research

Quality assurance is limited...

...by time...

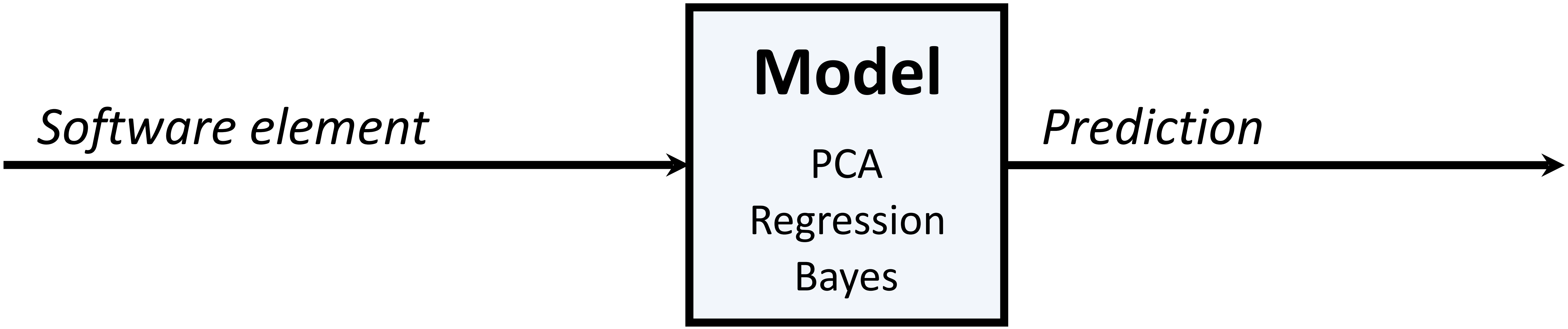
Quality assurance is limited...

...by time...

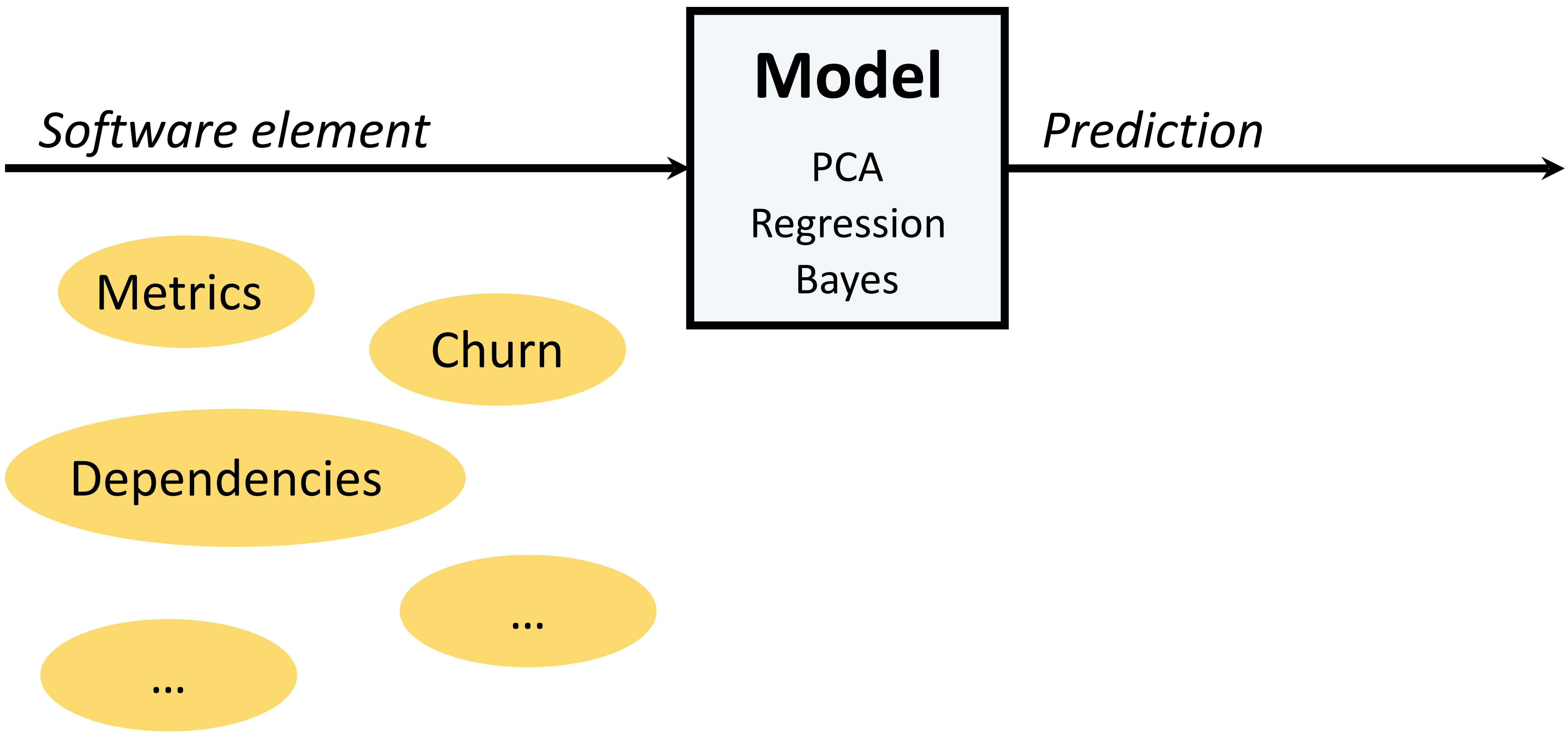
...and by money.

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

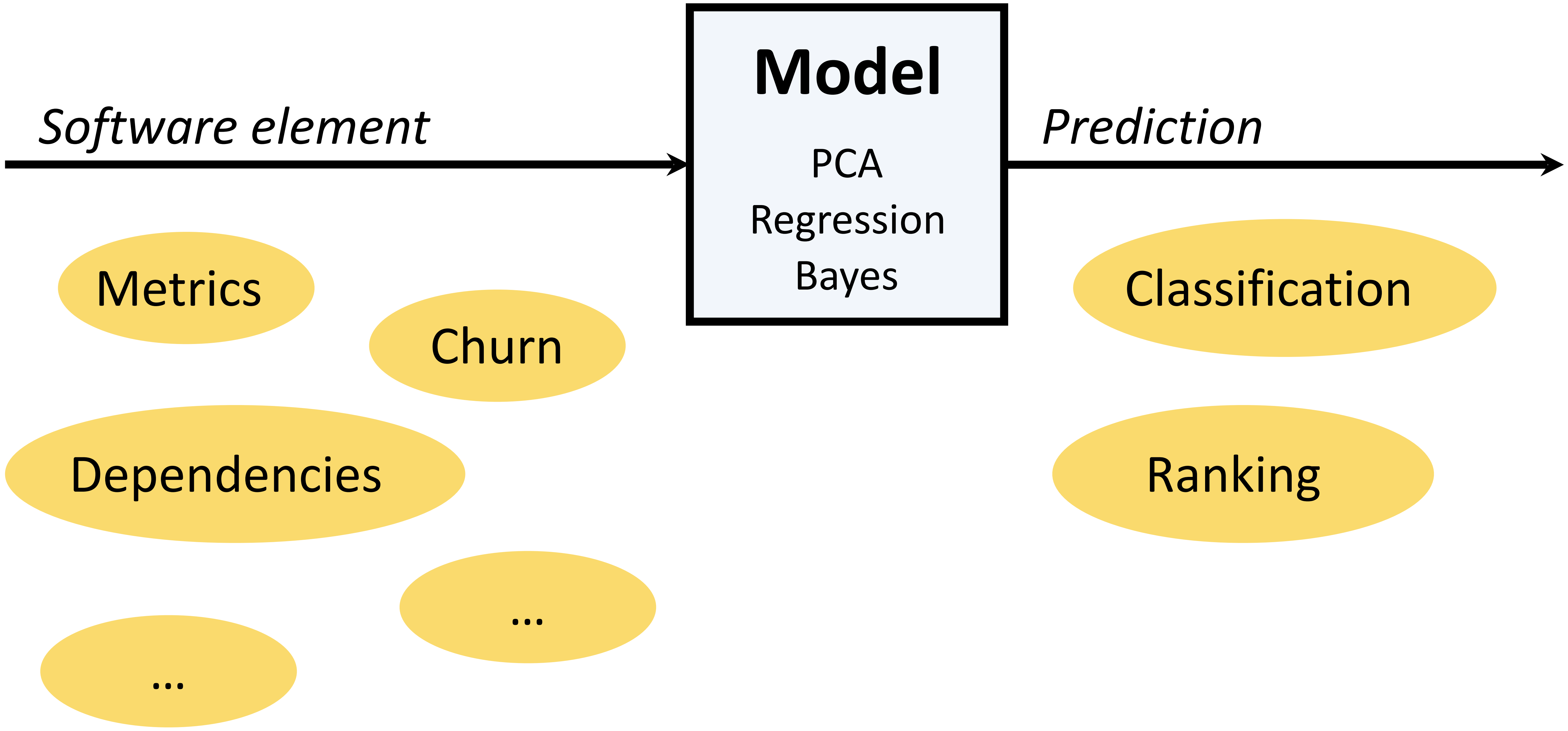
Defect prediction



Defect prediction



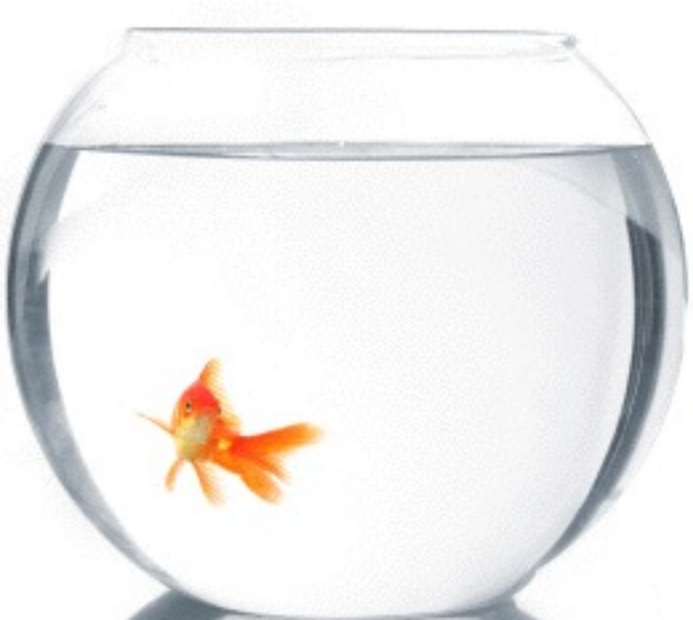
Defect prediction



Classification

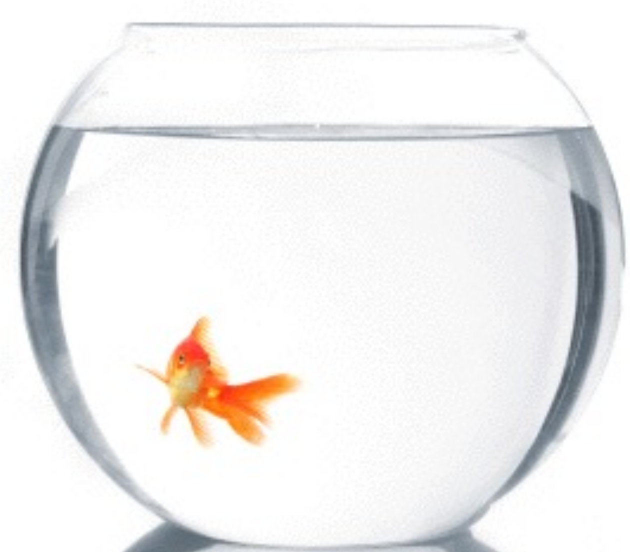
Has a binary a defect or not?

or



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%.

Predictor	Precision	Recall
Pre-Release Bugs	73.80%	62.90%
Test Coverage	83.80%	54.40%
Dependencies	74.40%	69.90%
Code Complexity	79.30%	66.00%
Code Churn	78.60%	79.90%
Org. Structure	86.20%	84.00%

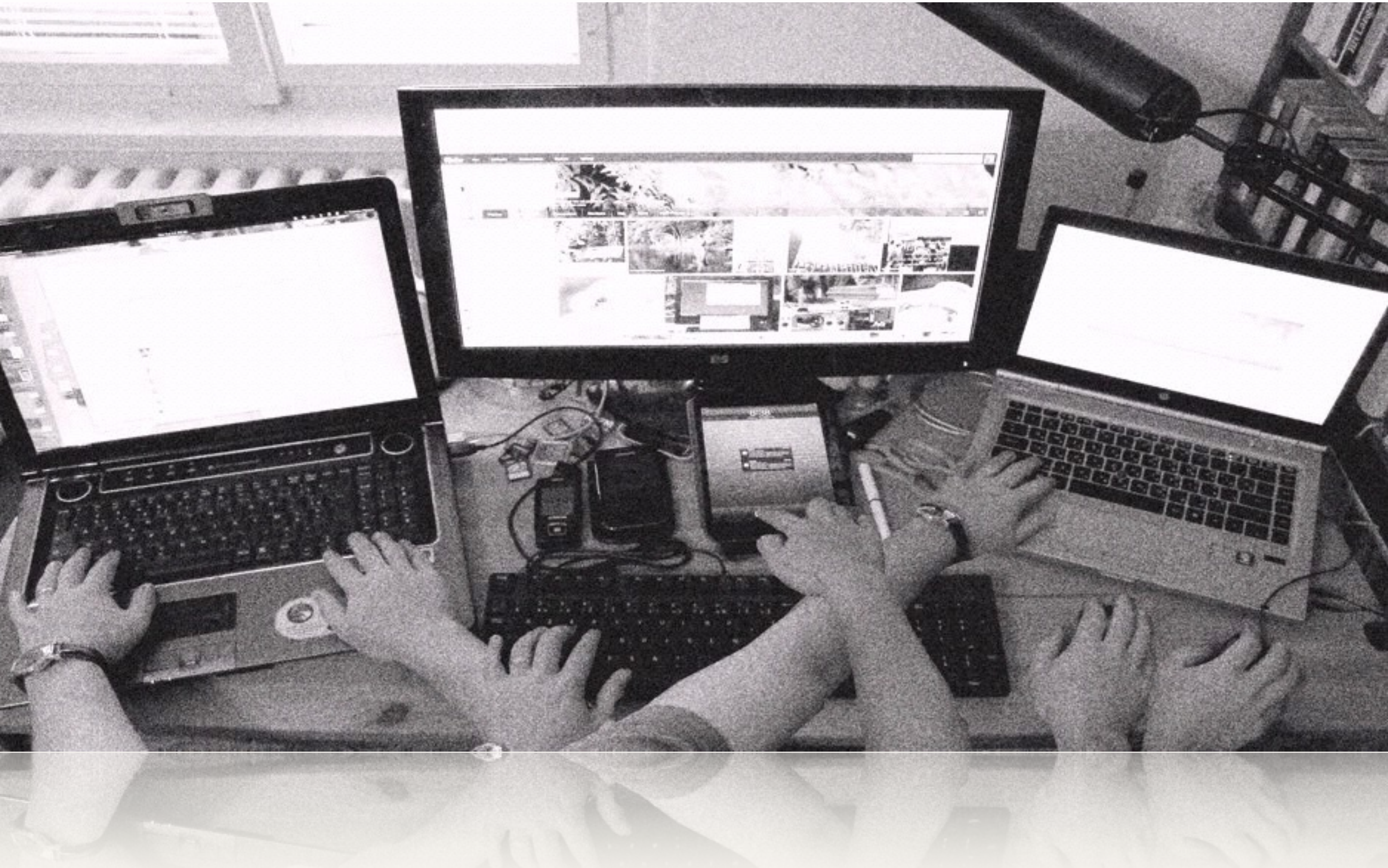
From: N. Nagappan, B. Murphy, and V. Basili. The influence of organizational structure on software quality. ICSE 2008.

How many projects to work on at the same time?

Octocat, here and elsewhere, by GitHub https://octodex.github.com

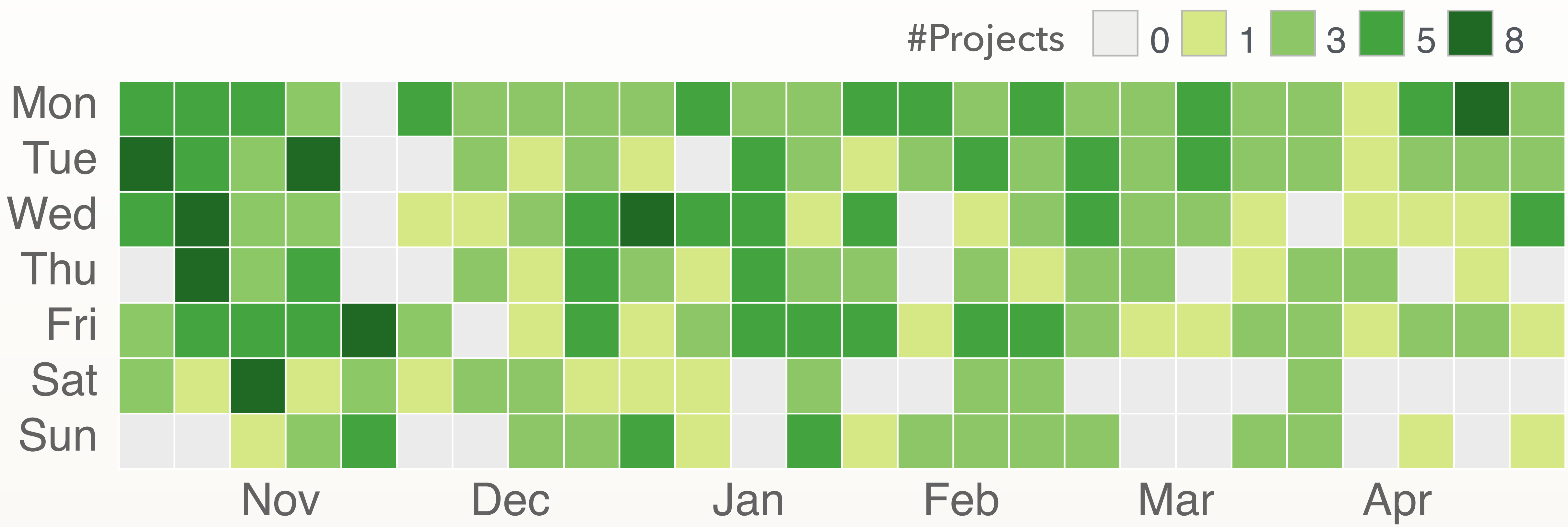
Vasilescu, B., Blincoe, K., Xuan, Q., Casalnuovo, C., Damian, D., Devanbu, P., & Filkov, V. (2016, May). The sky is not the limit: multitasking across GitHub projects. In *Proceedings of the 38th International Conference on Software Engineering* (pp. 994-1005). ACM.

Multitasking is common



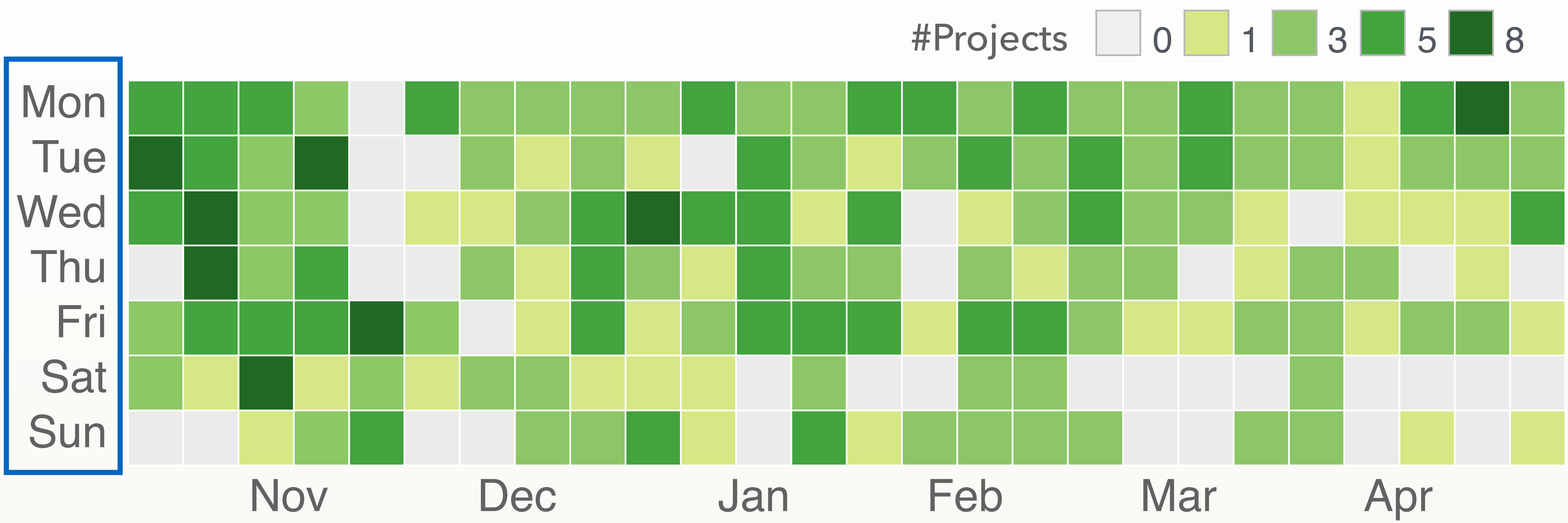
Software developers multitask too

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



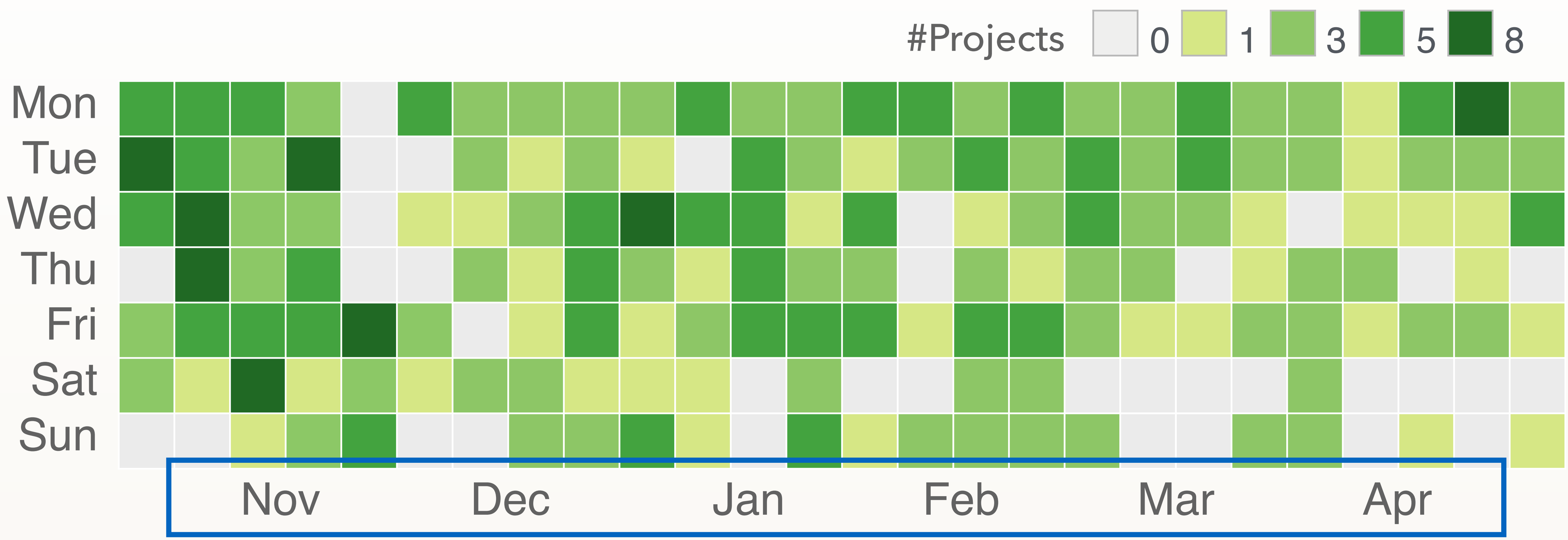
Software developers multitask too

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



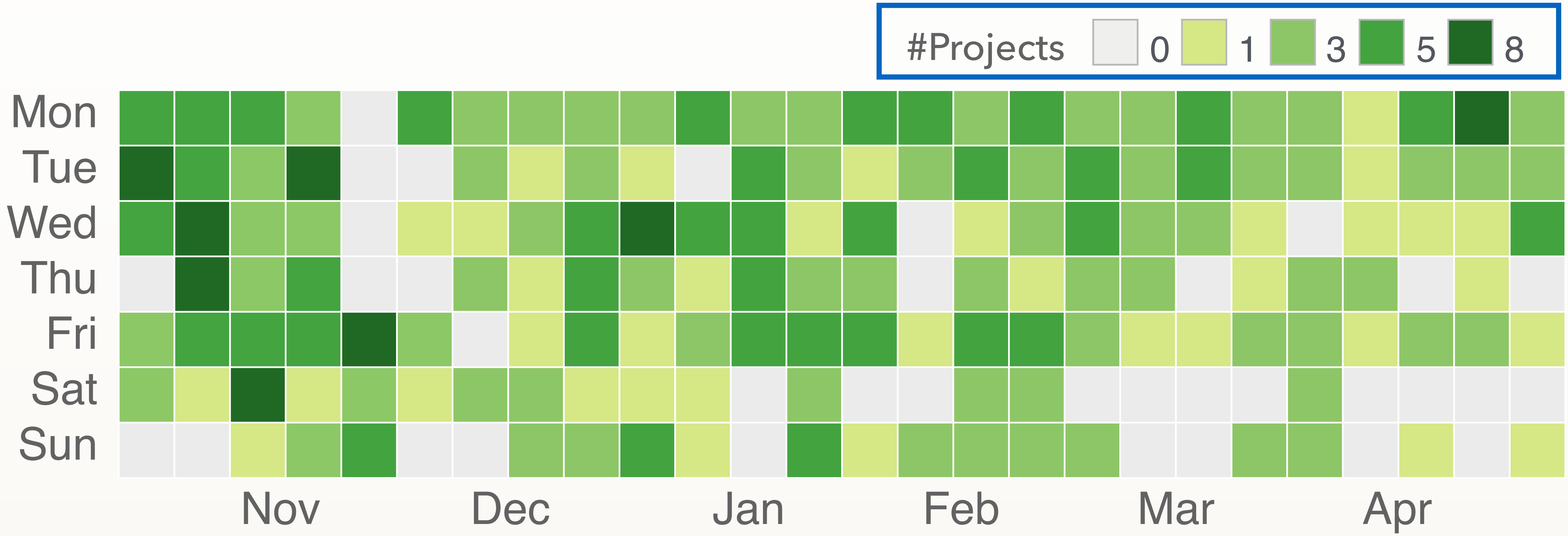
Software developers multitask too

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



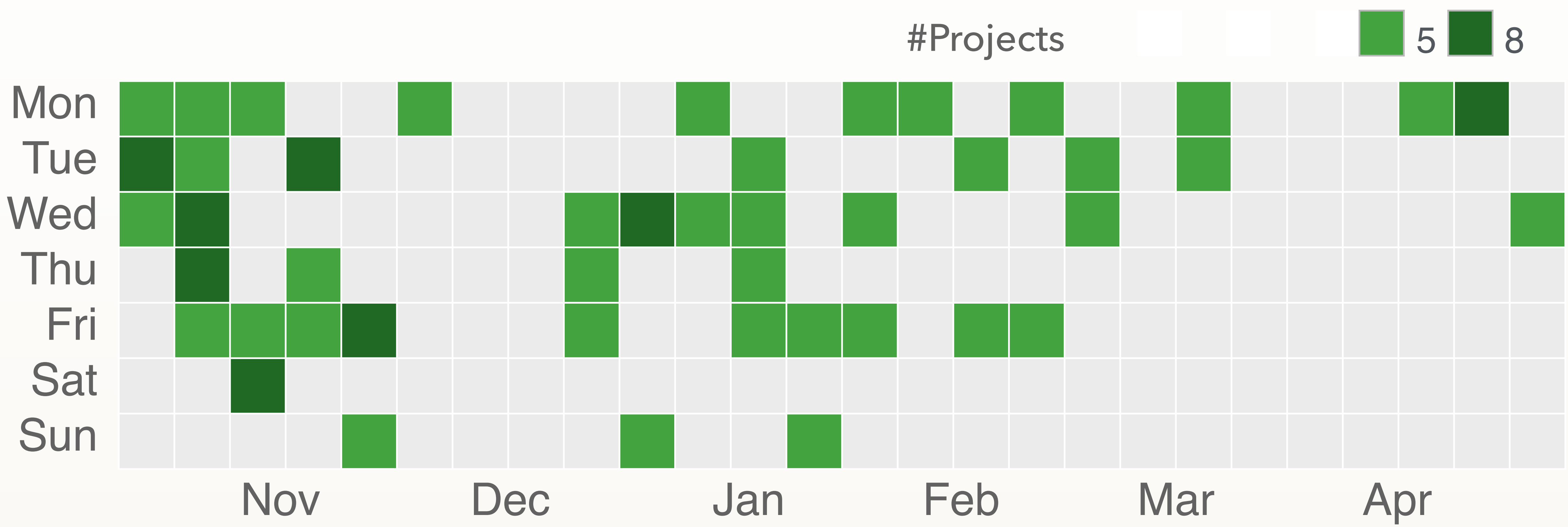
Software developers multitask too

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



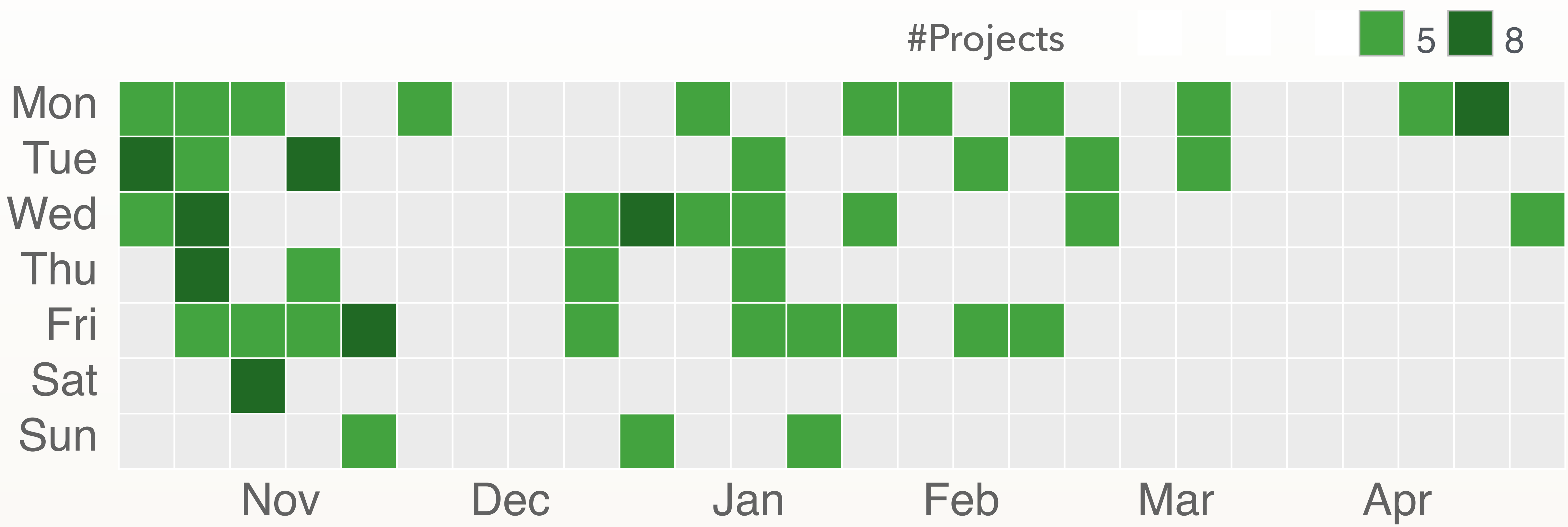
Software developers multitask too

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



Software developers multitask too

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

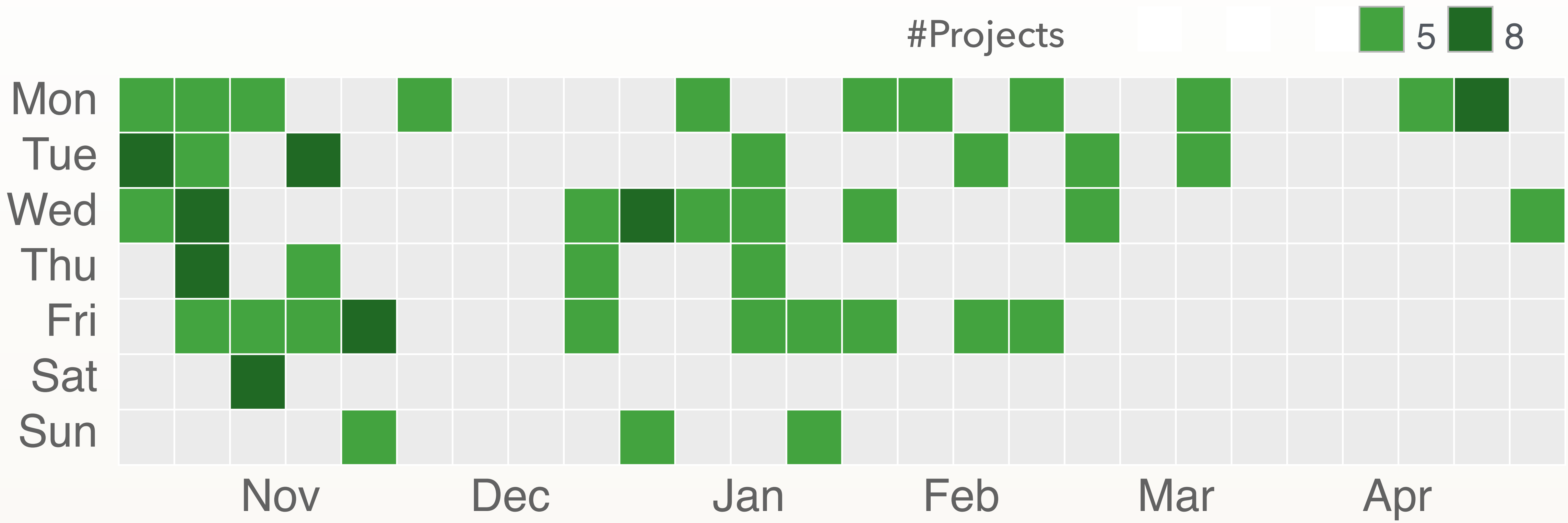


WHY?

- Request from other dev's / management

Software developers multitask too

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

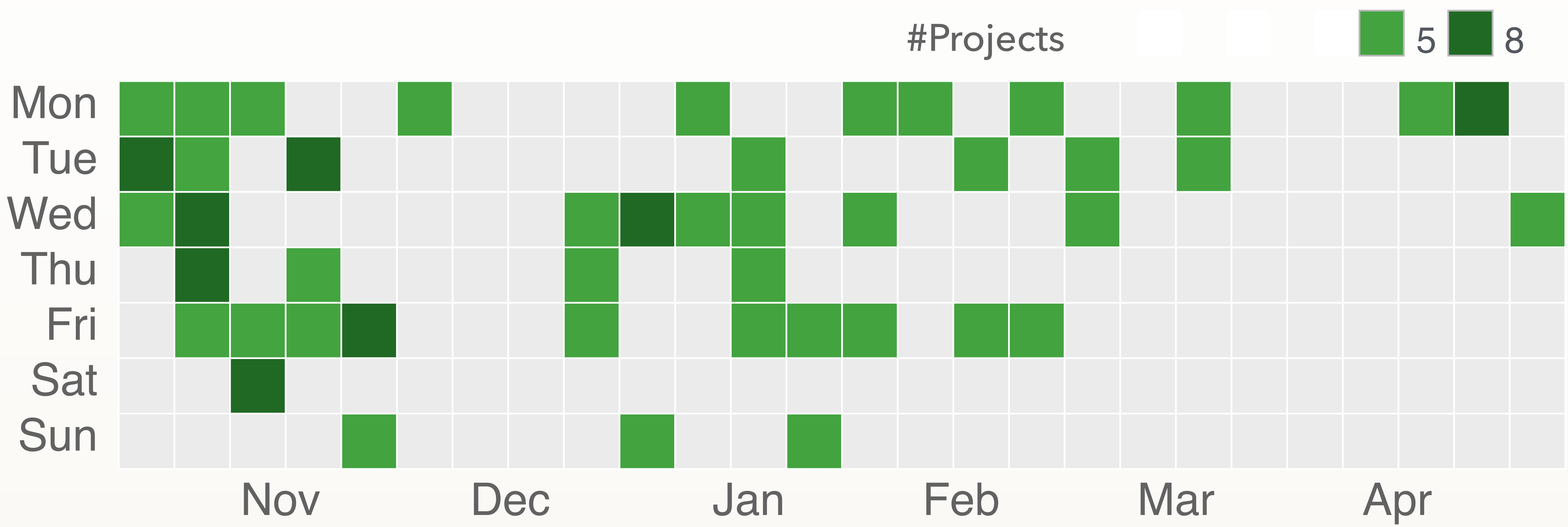


WHY?

- Request from other dev's / management
- Dependencies

Software developers multitask too

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

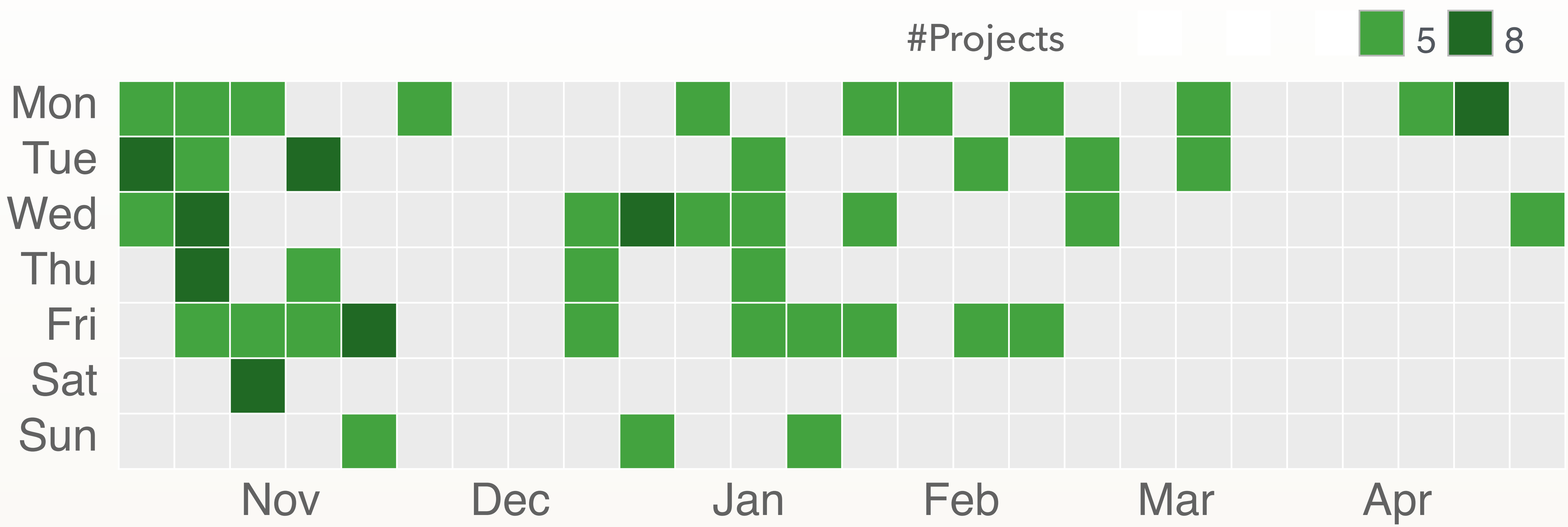


WHY?

- Request from other dev's / management
- Dependencies
- Being "stuck"
- Downtime

Software developers multitask too

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

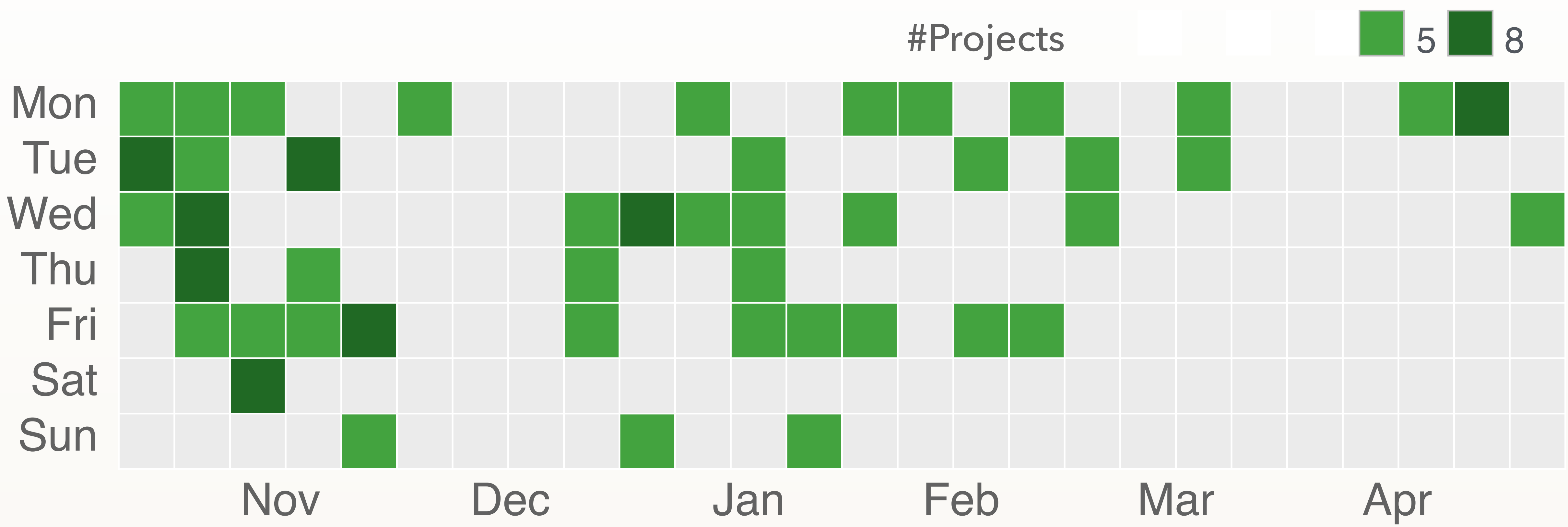


WHY?

- Request from other dev's / management
- Dependencies
- Being "stuck"
- Downtime
- Personal interest

Software developers multitask too

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

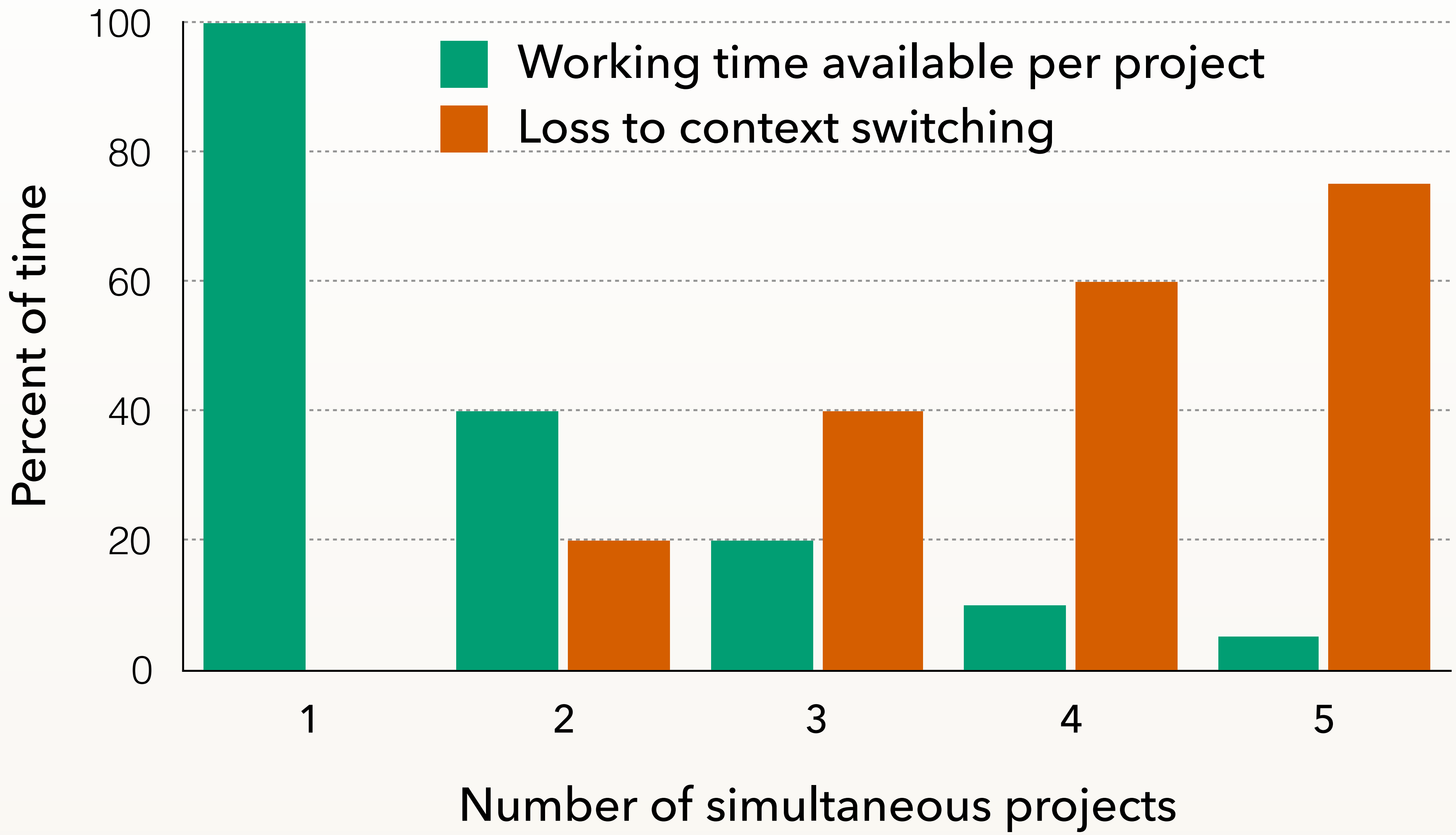


WHY?

- Request from other dev's / management
- Dependencies
- Being "stuck"
- Downtime
- Personal interest
- Signaling

Hardly any empirical evidence

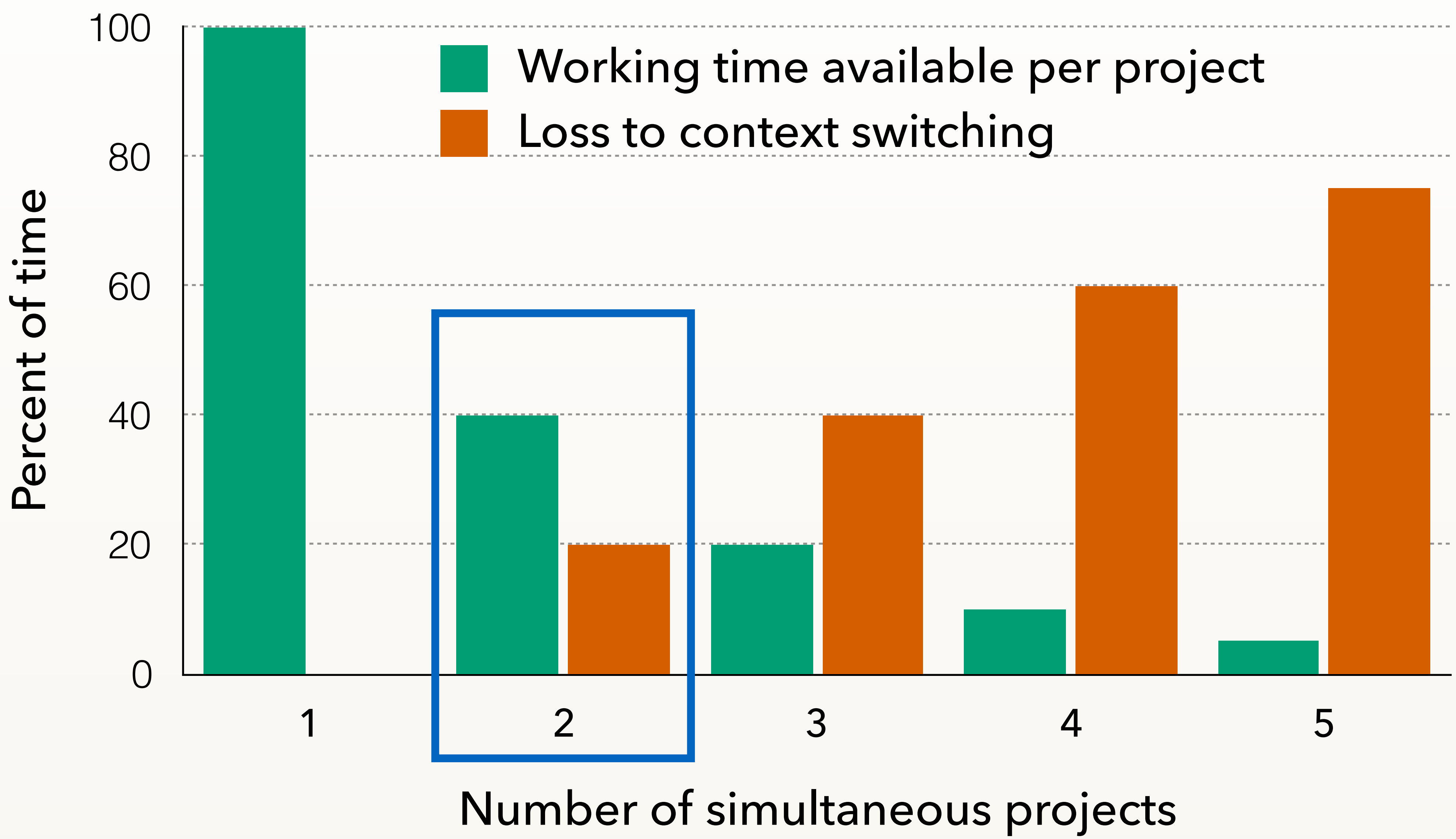
Rule of thumb (Weinberg, 1992) - <u>not</u> based on data



From: http://blog.codinghorror.com/the-multi-tasking-myth/

Hardly any empirical evidence

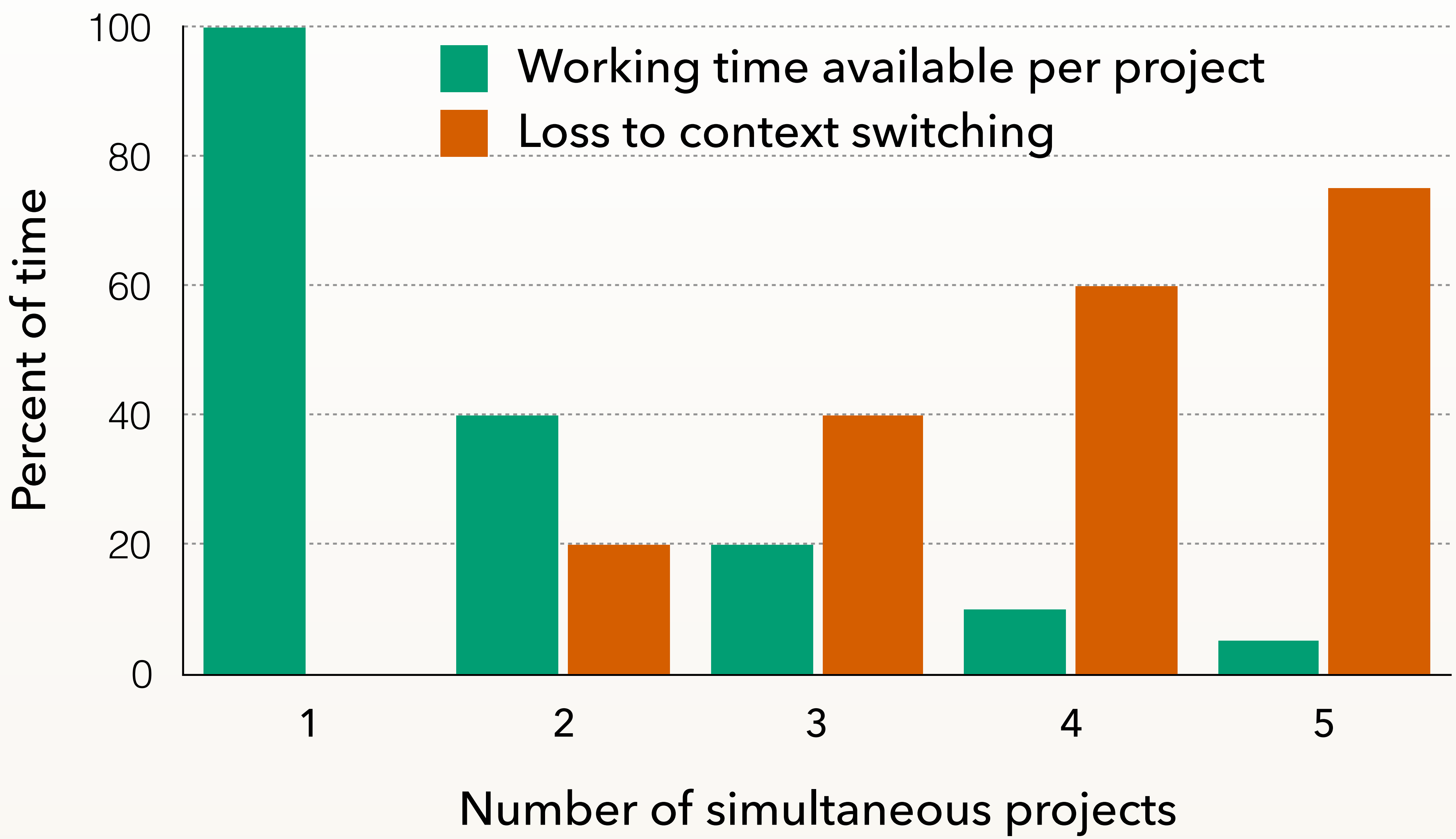
Rule of thumb (Weinberg, 1992) - <u>not</u> based on data



From: http://blog.codinghorror.com/the-multi-tasking-myth/

Hardly any empirical evidence

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

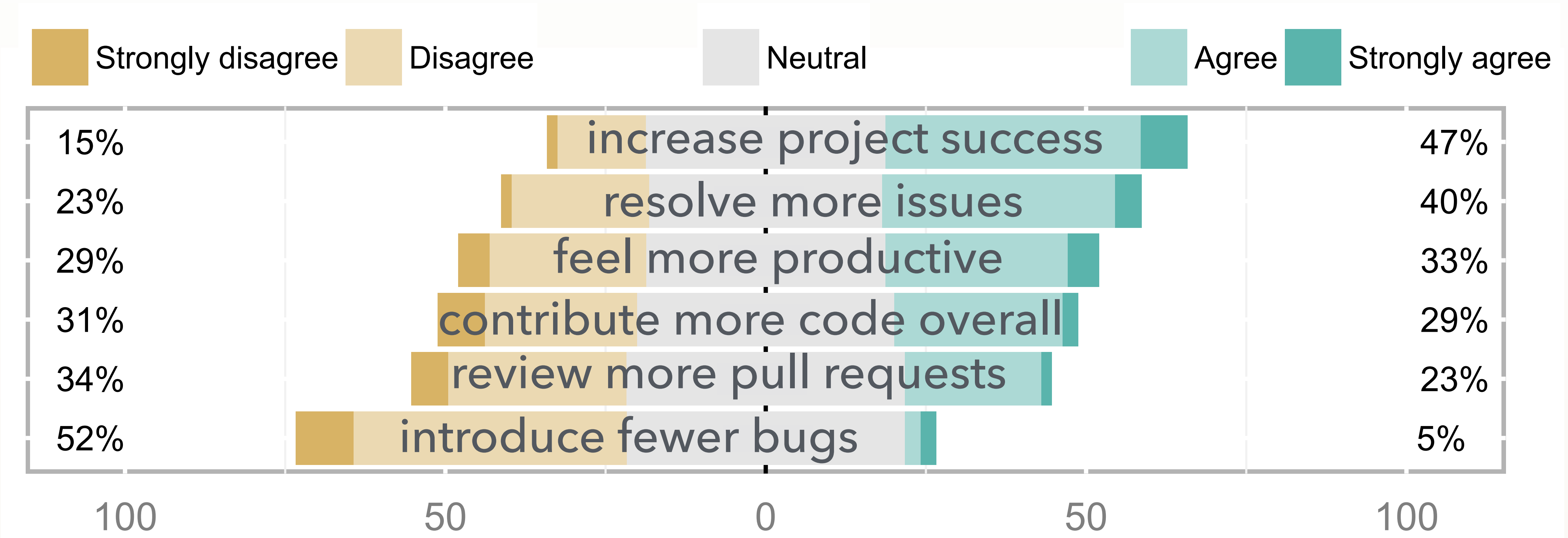


From: http://blog.codinghorror.com/the-multi-tasking-myth/

Effects: perception vs. data

PERCEPTION

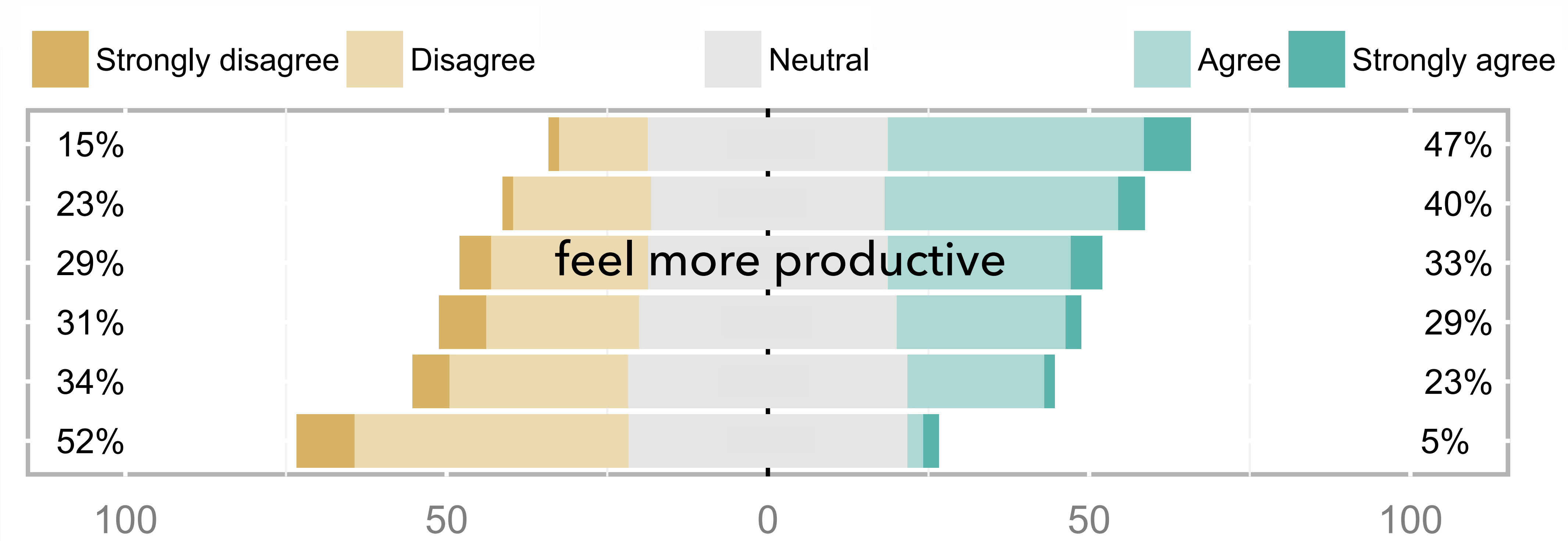
"When contributing to multiple projects in parallel, I:"



Effects: perception vs. data

PERCEPTION

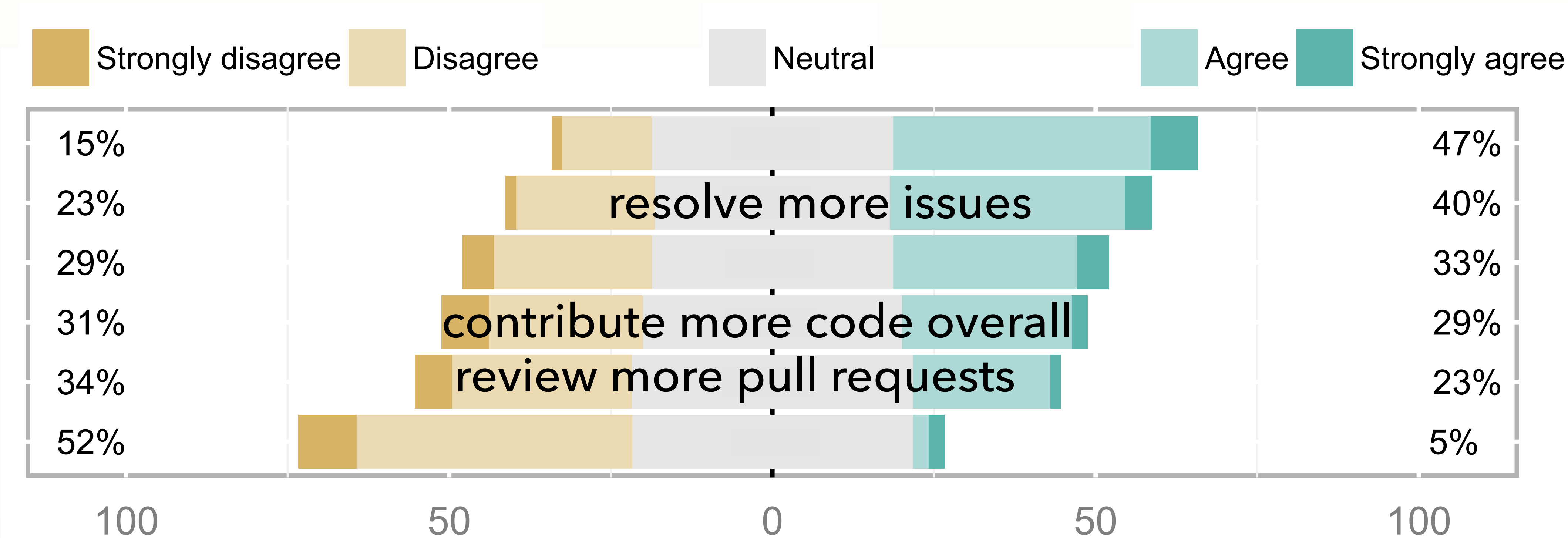
“When contributing to multiple projects in parallel, I:”



Effects: perception vs. data

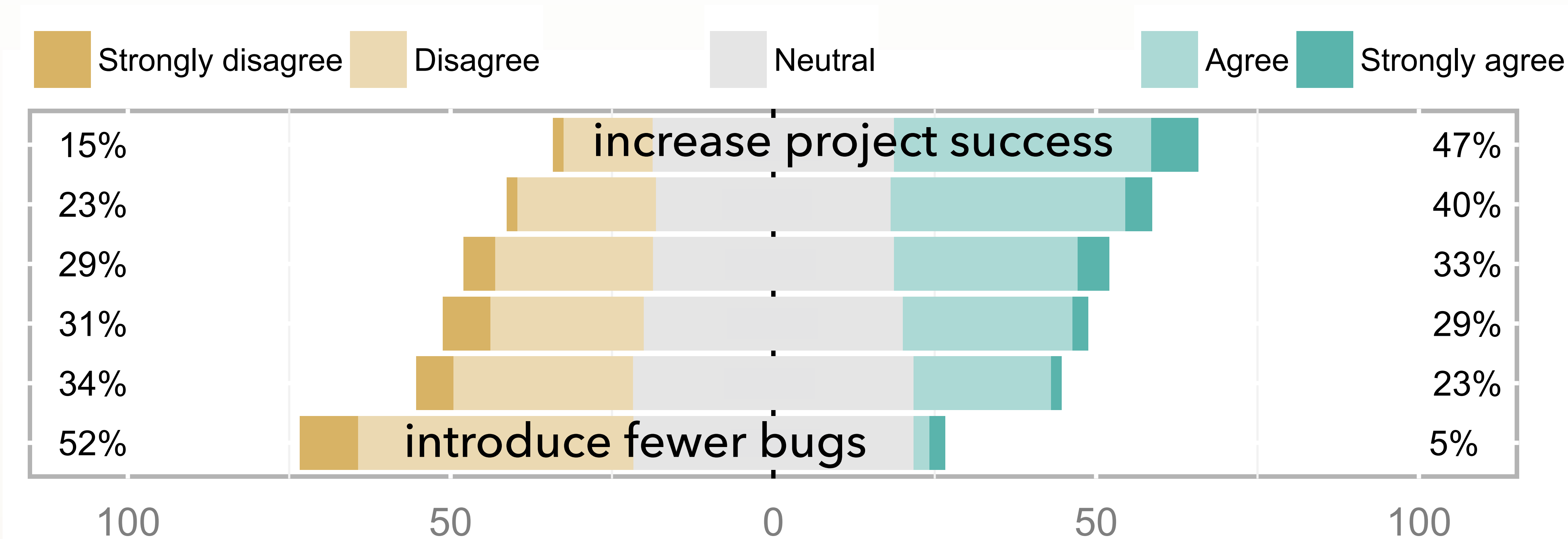
PERCEPTION

"When contributing to multiple projects in parallel, I:"



Effects: perception vs. data

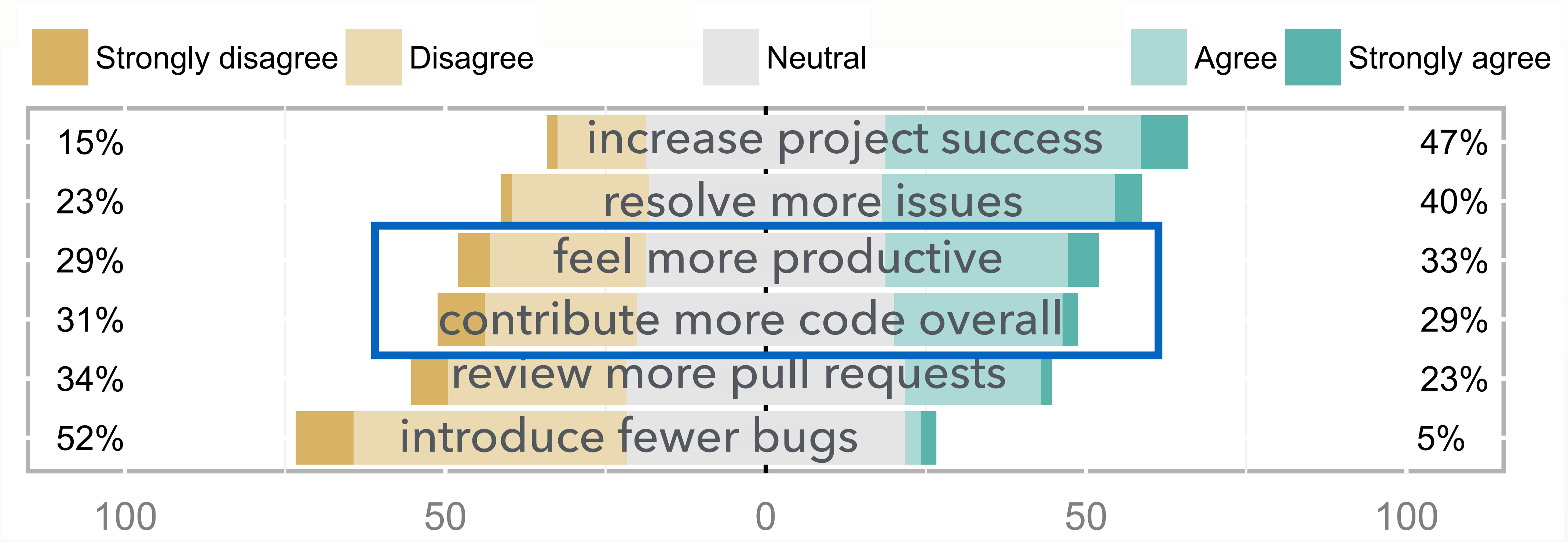
PERCEPTION *"When contributing to multiple projects in parallel, I:"*



Effects: perception vs. data

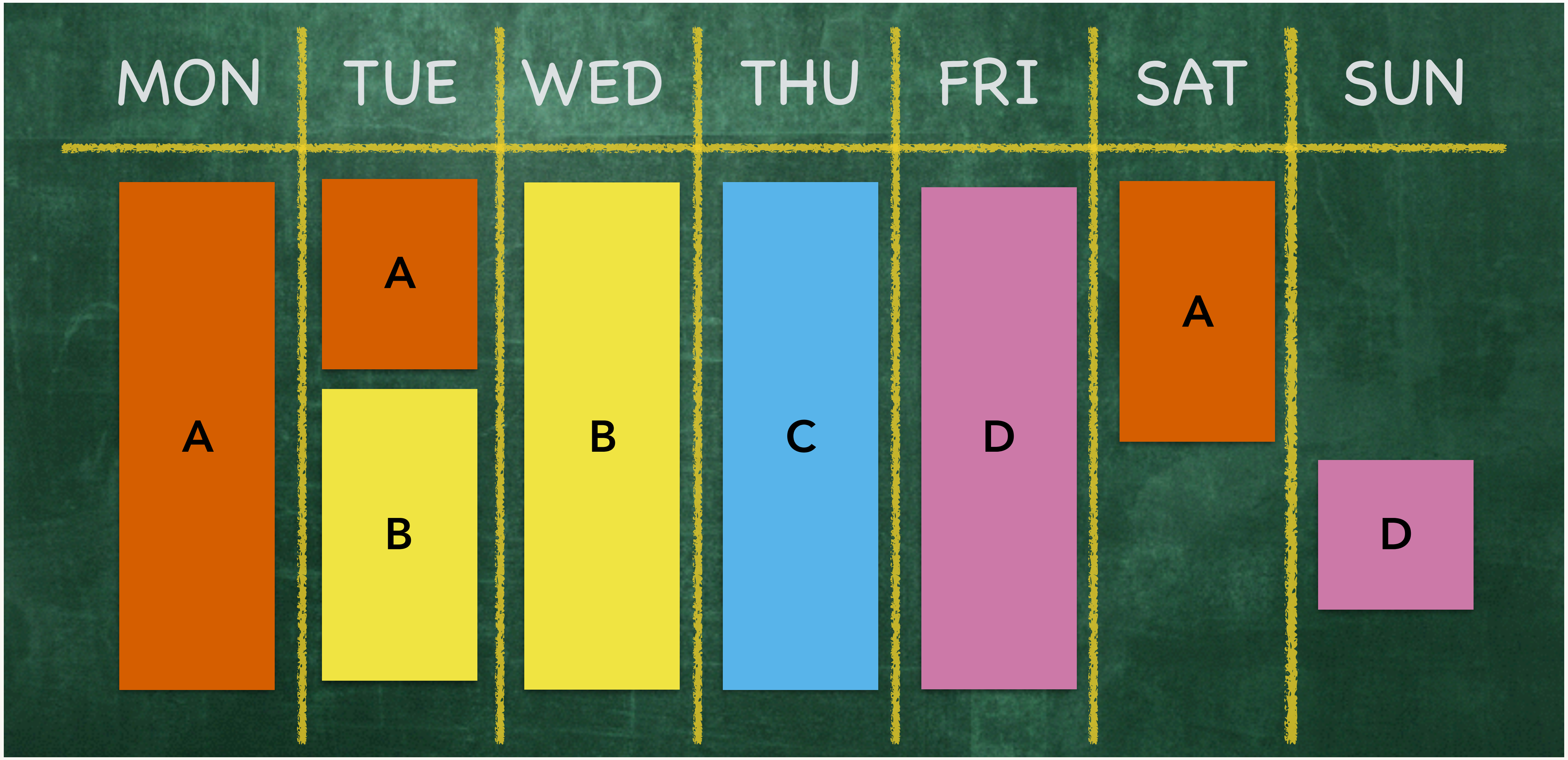
PERCEPTION

"When contributing to multiple projects in parallel, I:"



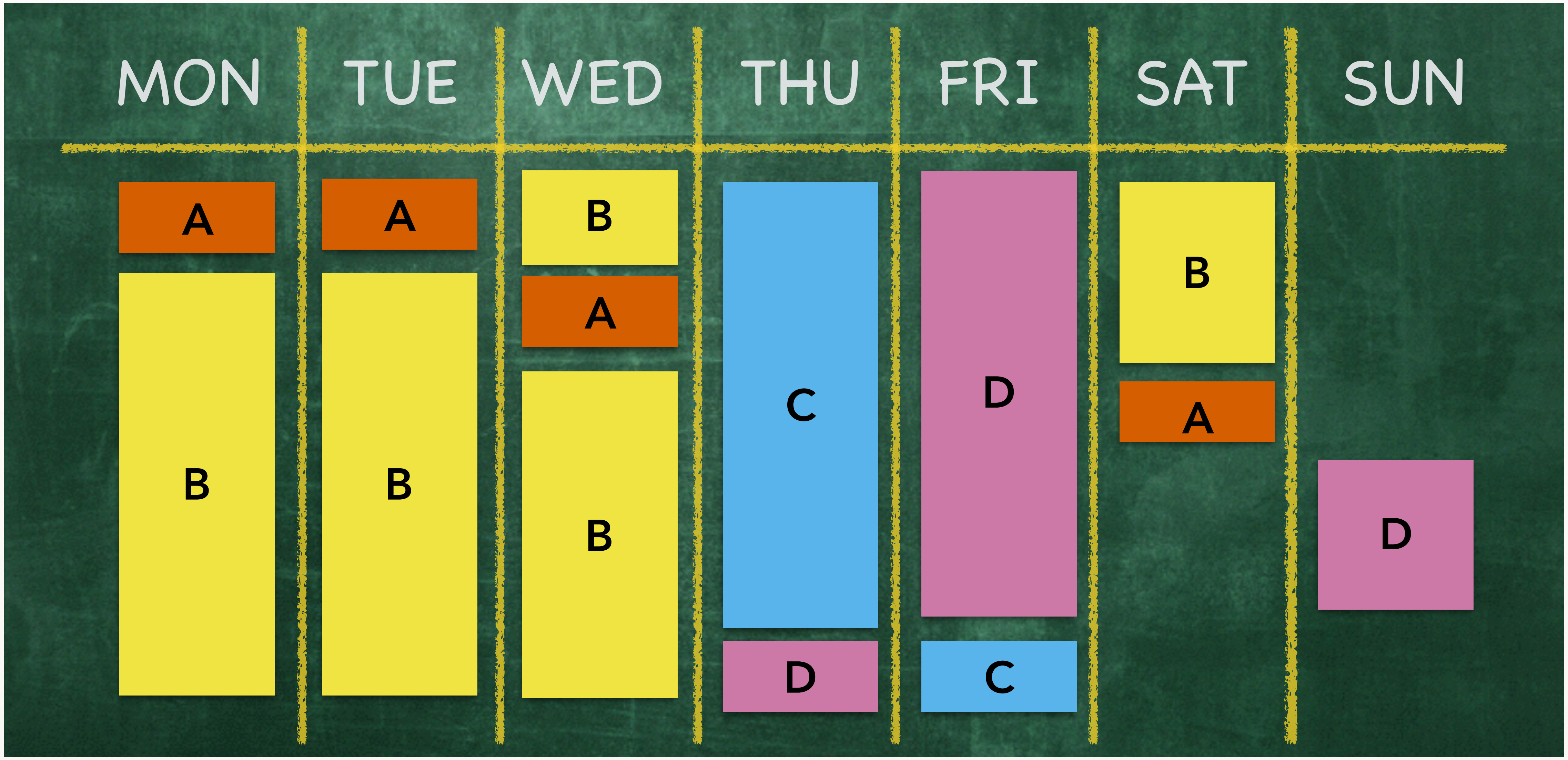
Modeling multitasking

- Period matters



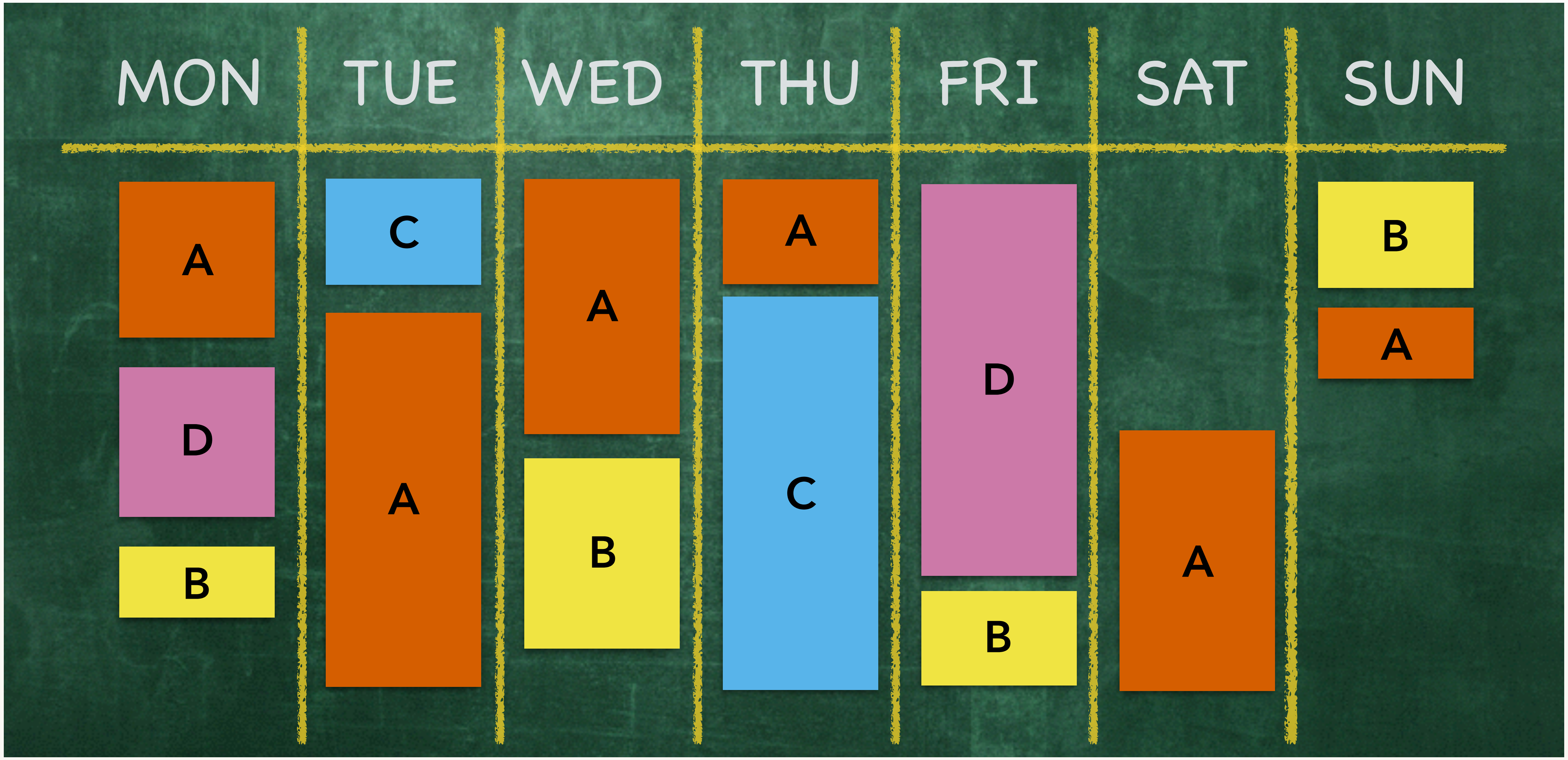
Modeling multitasking

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



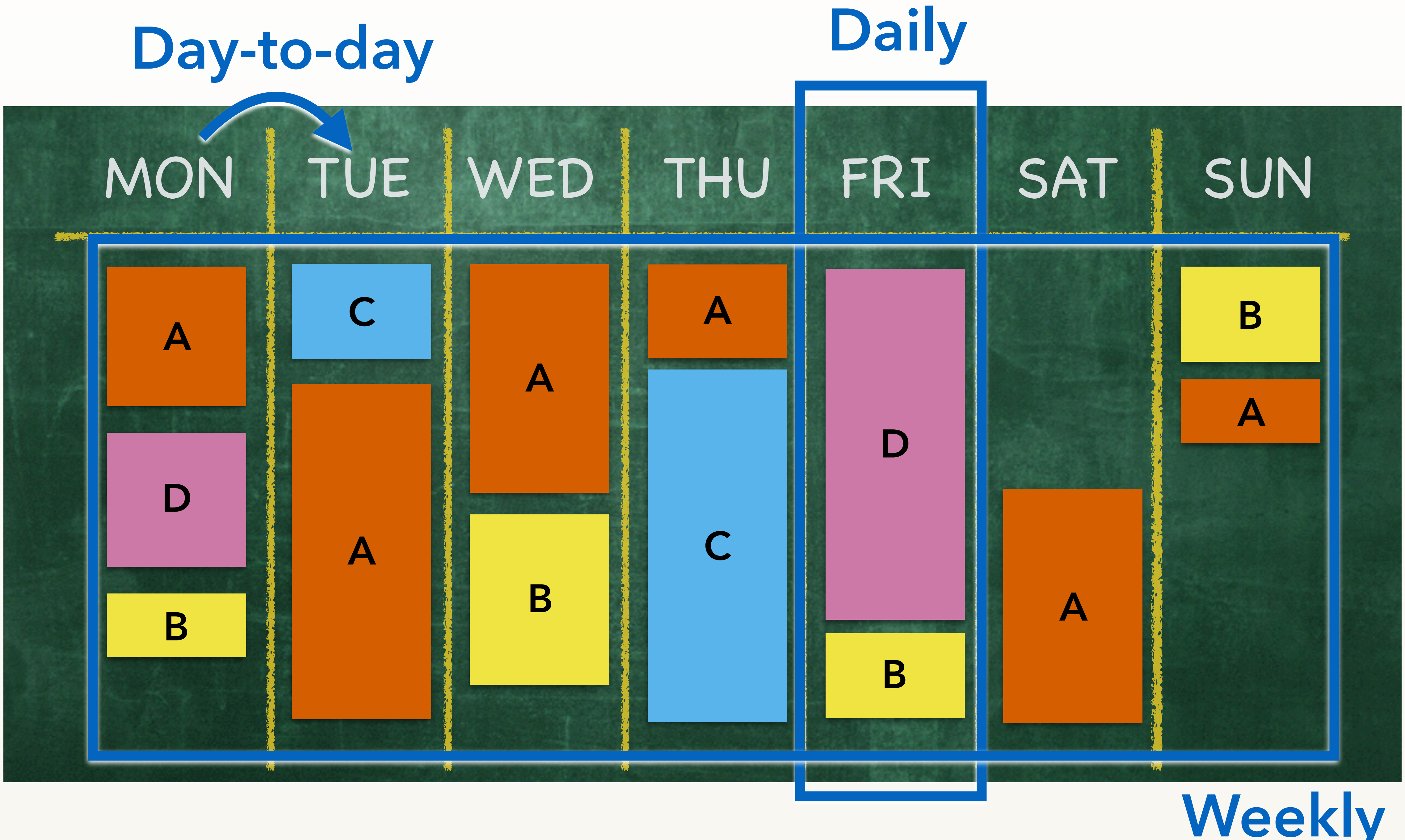
Modeling multitasking

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



Modeling multitasking

- Period matters
- Effort matters
- Break matters
- ...



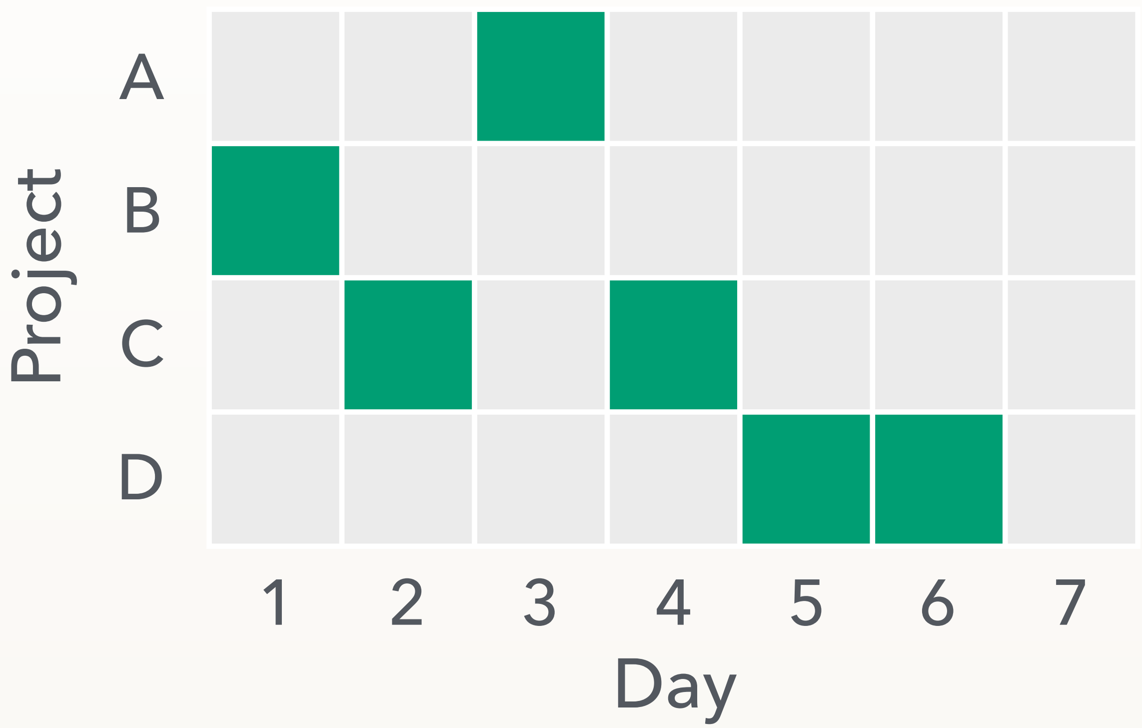
WE MODELED:

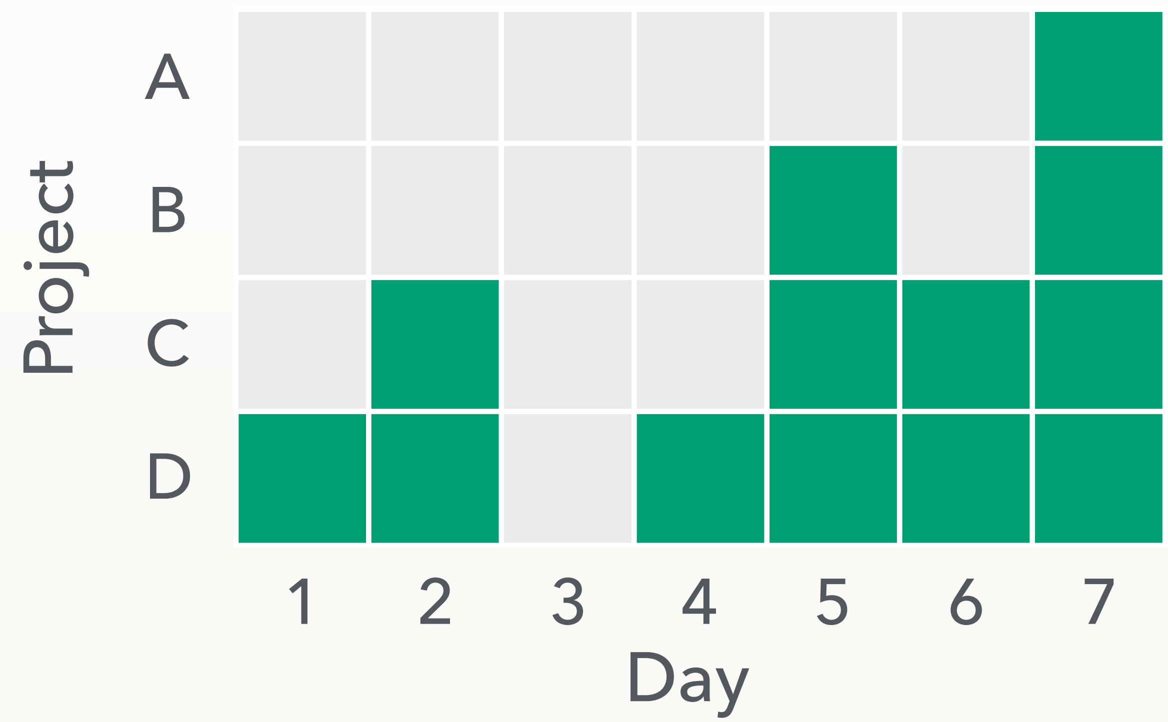
- One-week panels
- Three dimensions

Multitasking dimensions

1. PROJECTS PER DAY

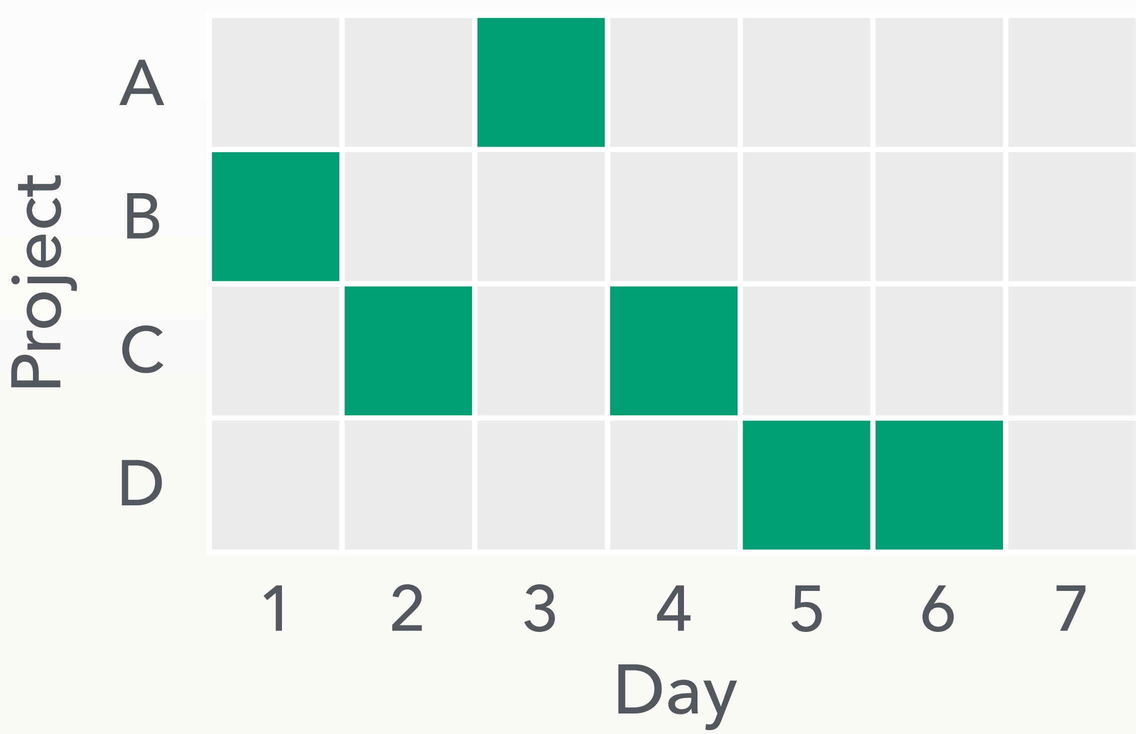
Working sequentially vs. Within-day multitasking

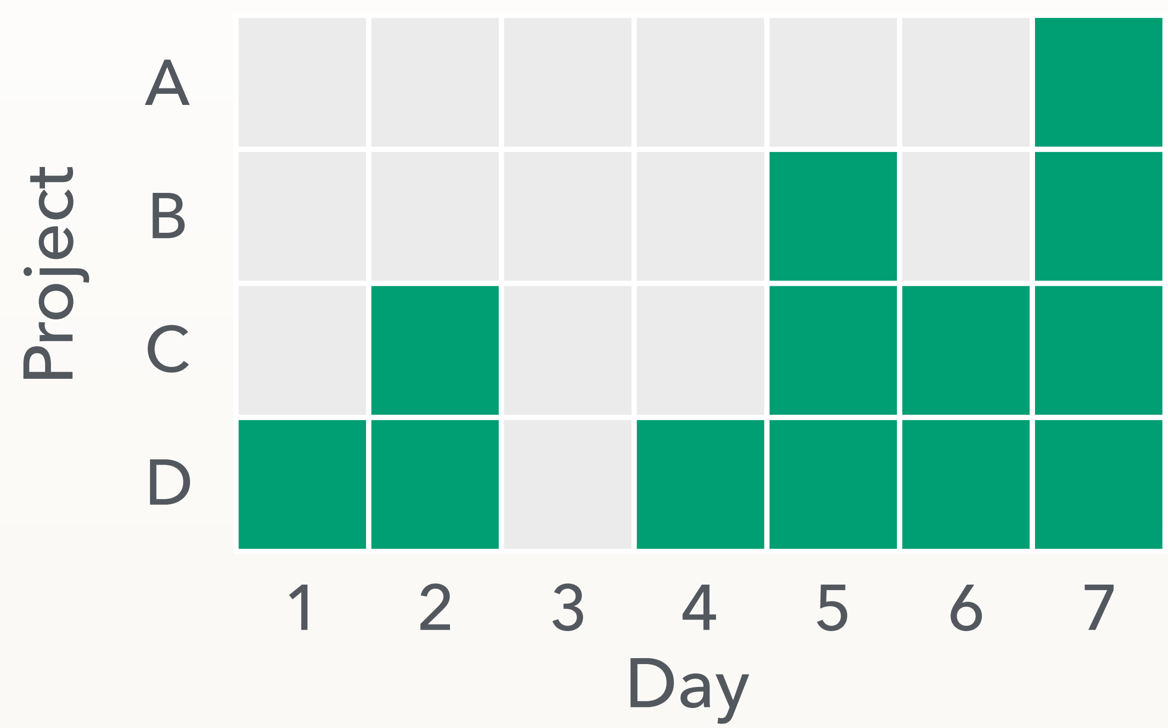




Multitasking dimensions

Working sequentially vs. Within-day multitasking



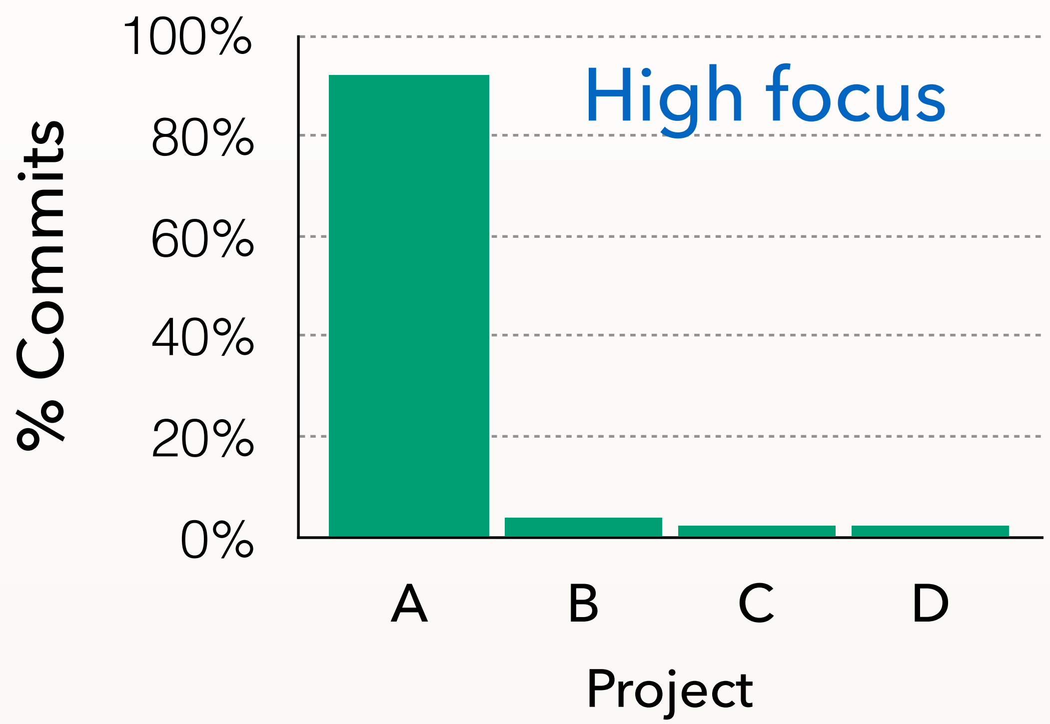


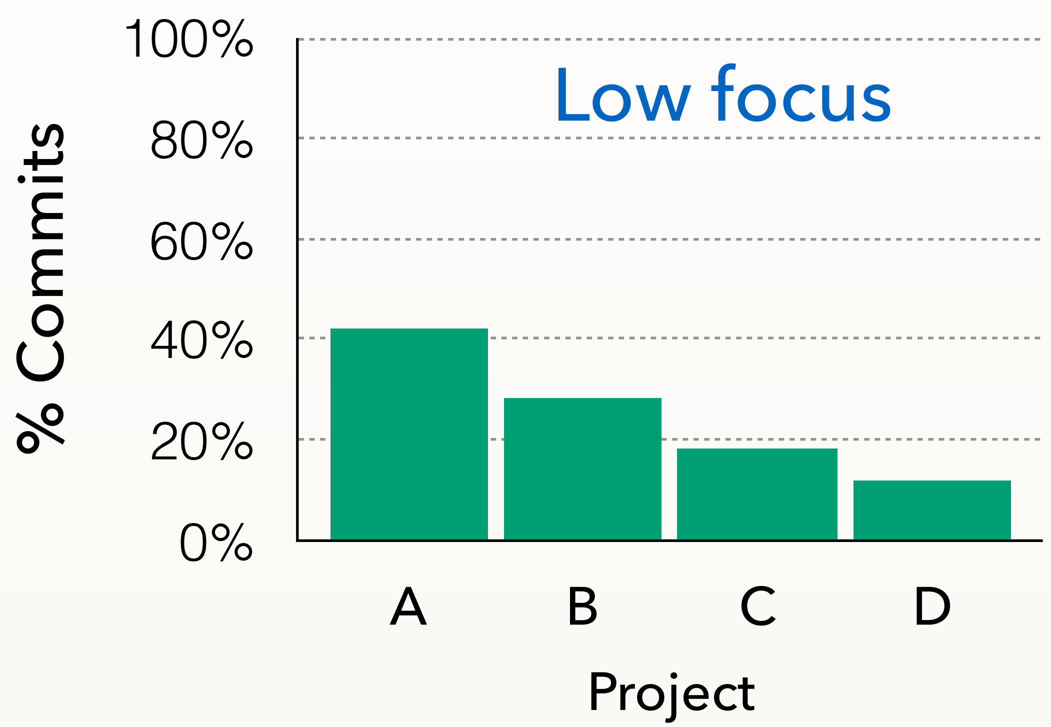
$\text{AvgProjectsPerDay} = 1$

$\text{AvgProjectsPerDay} = 2.2$

Multitasking dimensions

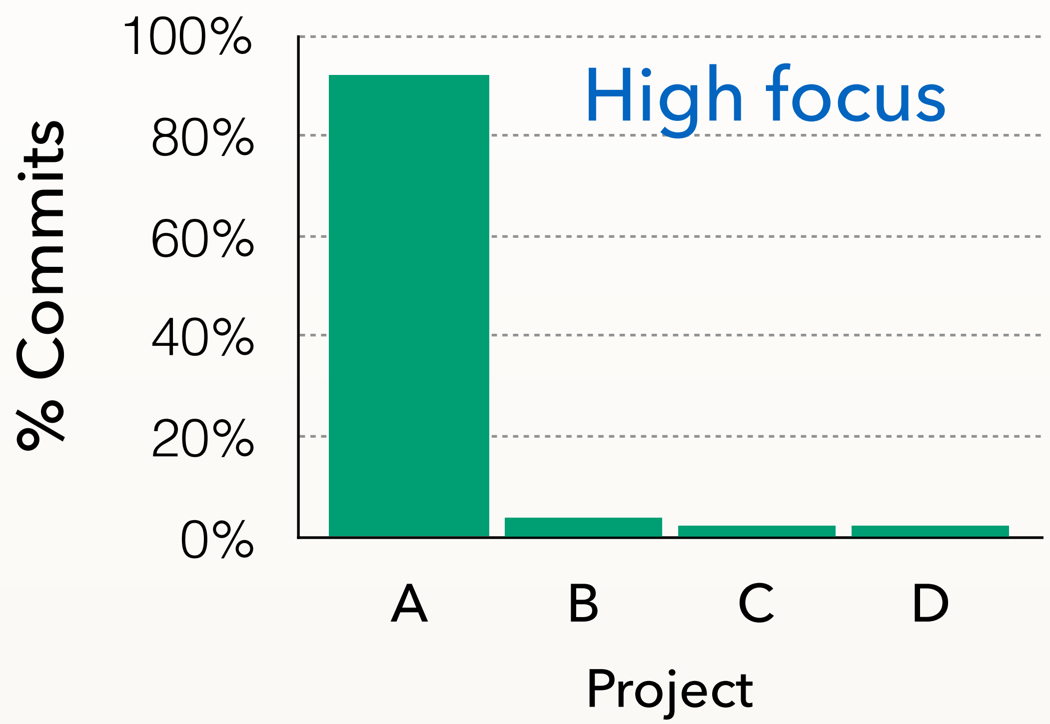
Focusing on one project vs. Contributing evenly to all



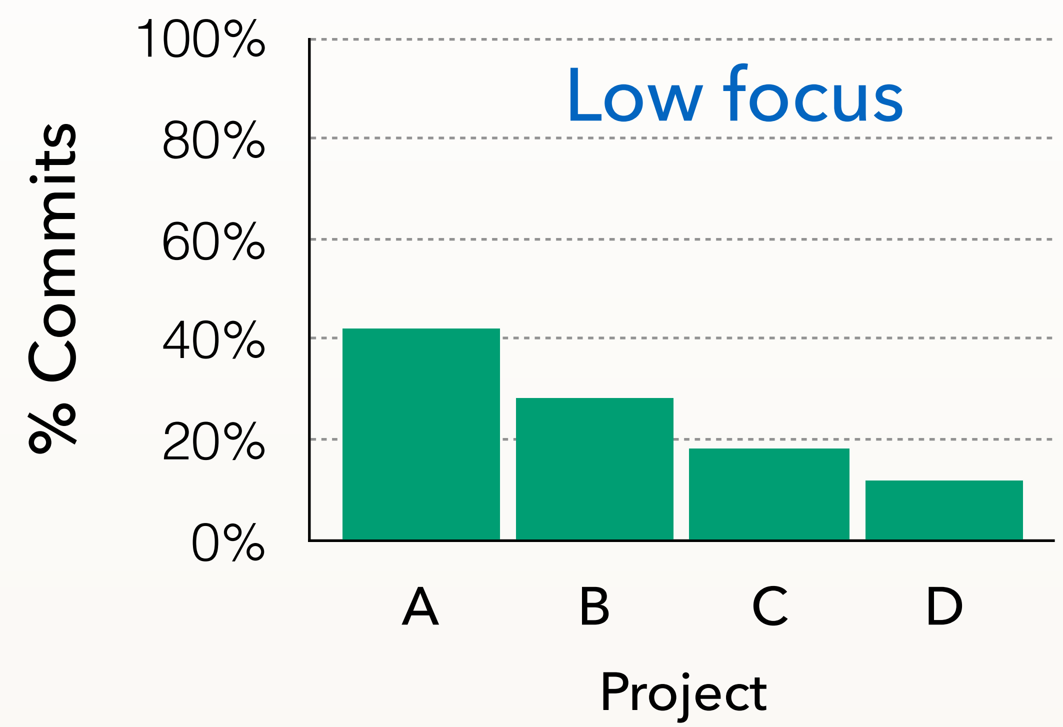


Multitasking dimensions

Focusing on one project vs. Contributing evenly to all



$$S_{\text{Focus}} = 0.25$$



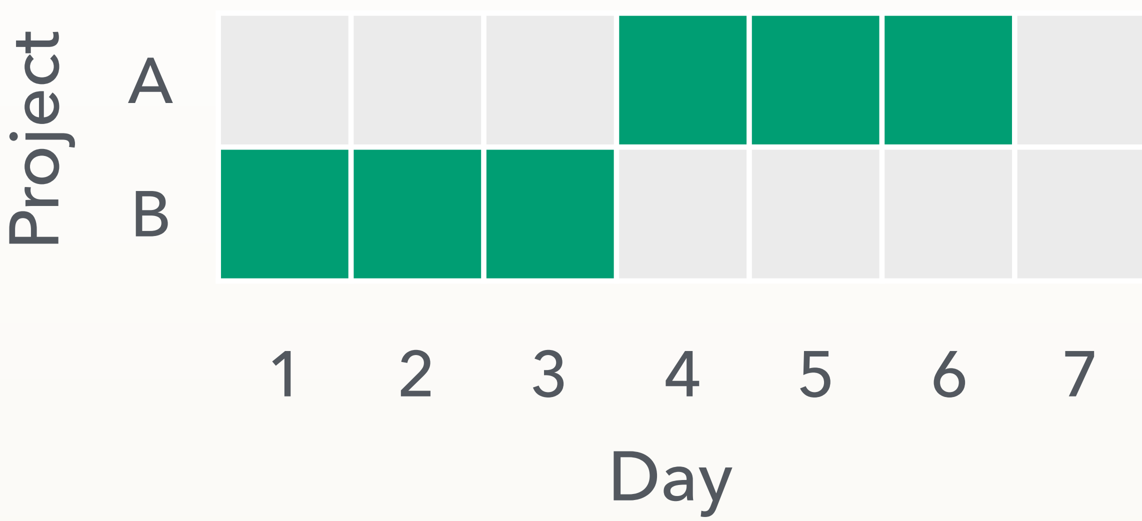
$$S_{\text{Focus}} = 1.85$$

Shannon entropy: $$S_{\text{Focus}} = -\sum_{i=1}^{N} p_i \log_2 p_i$$

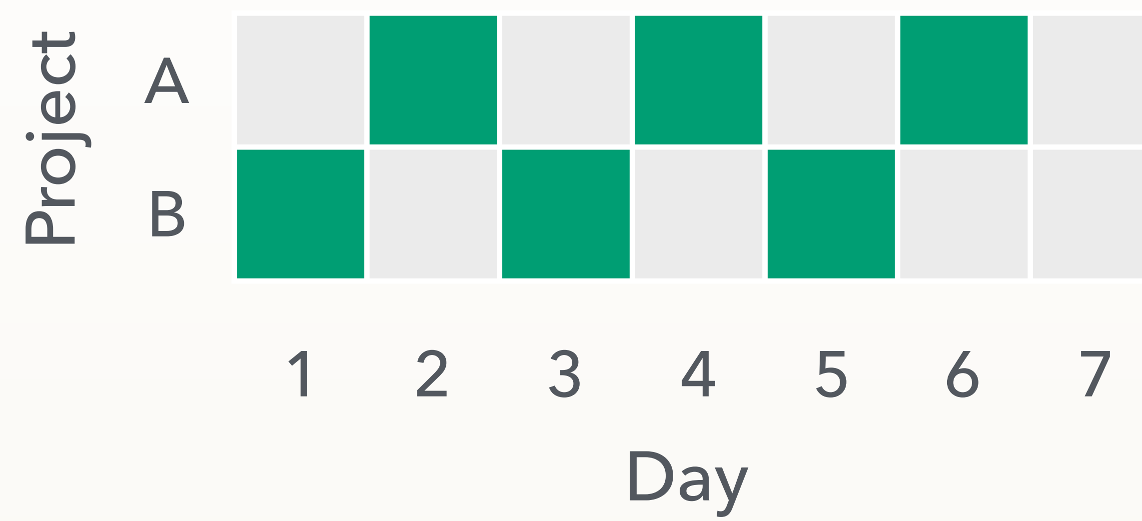
Fraction commits in project i

Multitasking dimensions

Repetitive day-to-day vs. Switching focus



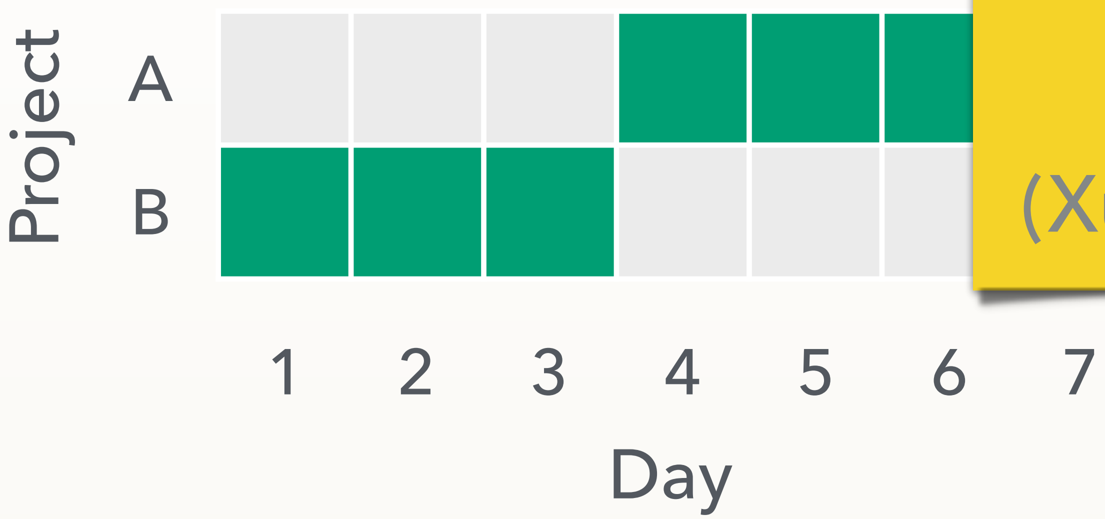
$$\text{AvgProjectsPerDay} = 1$$
$$S_{\text{Focus}} = 1$$



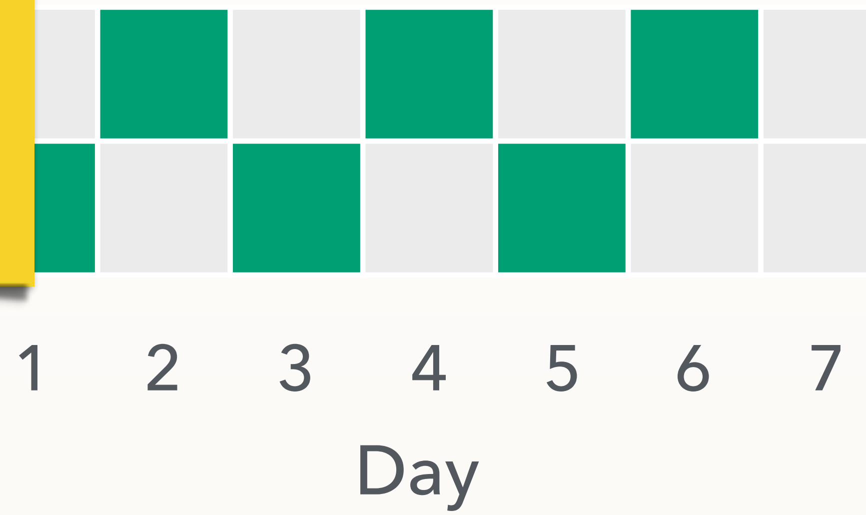
$$\text{AvgProjectsPerDay} = 1$$
$$S_{\text{Focus}} = 1$$

Multitasking dimensions

Repetitive day-to-day vs. Switching focus



Focus shifting networks
(Xuan et al, 2014)

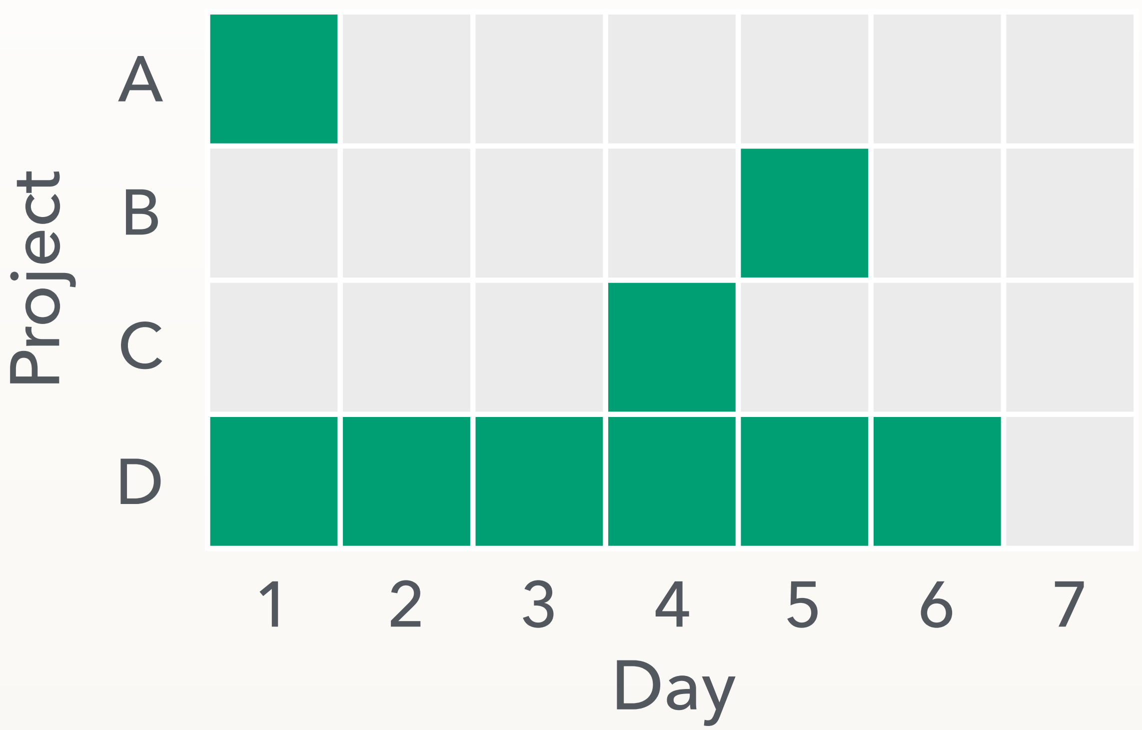


$$\mathrm{AvgProjectsPerDay} = 1$$
$$S_{\mathrm{Focus}} = 1$$

$$\mathrm{AvgProjectsPerDay} = 1$$
$$S_{\mathrm{Focus}} = 1$$

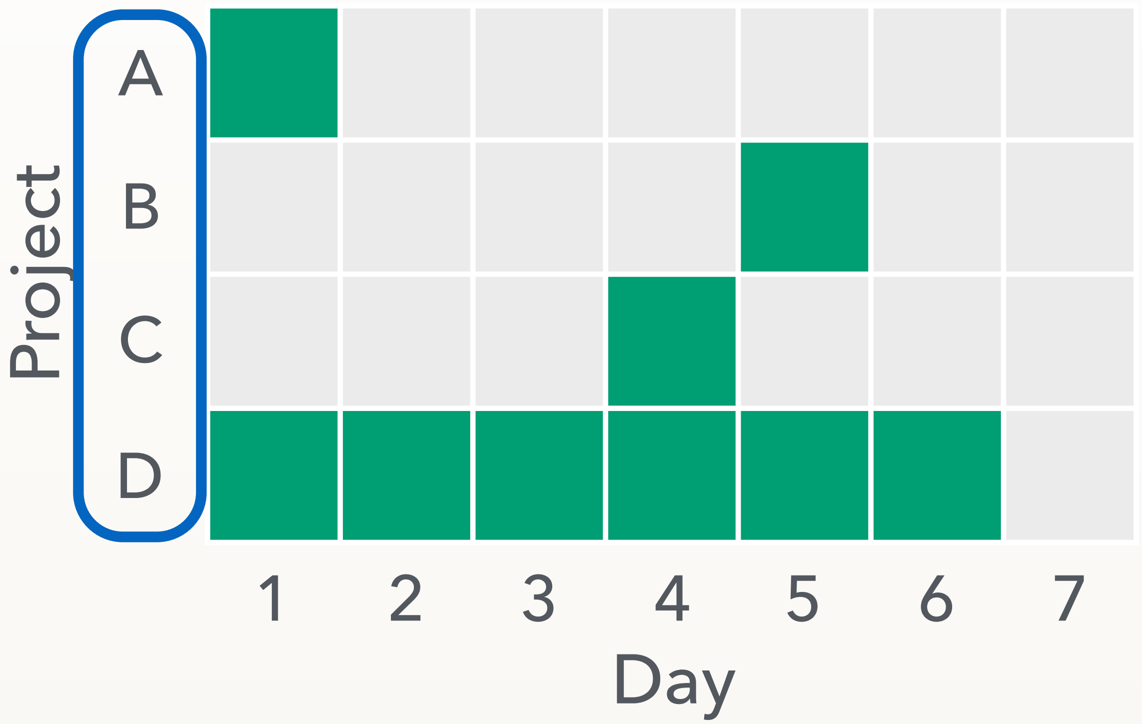
Multitasking dimensions

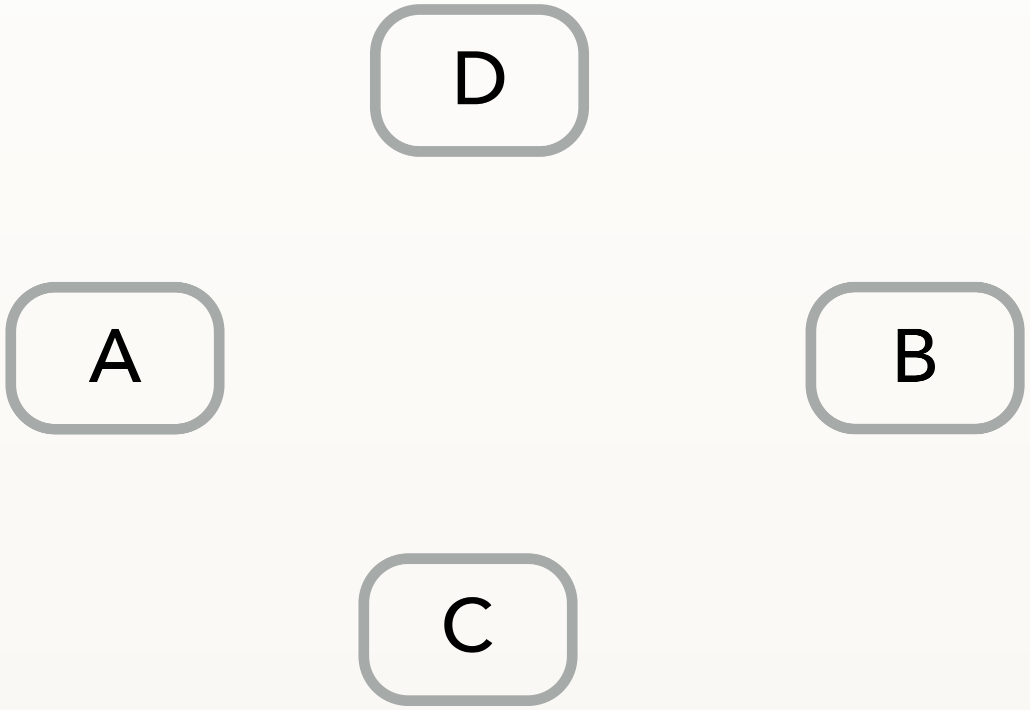
Repetitive day-to-day vs. Switching focus



Multitasking dimensions

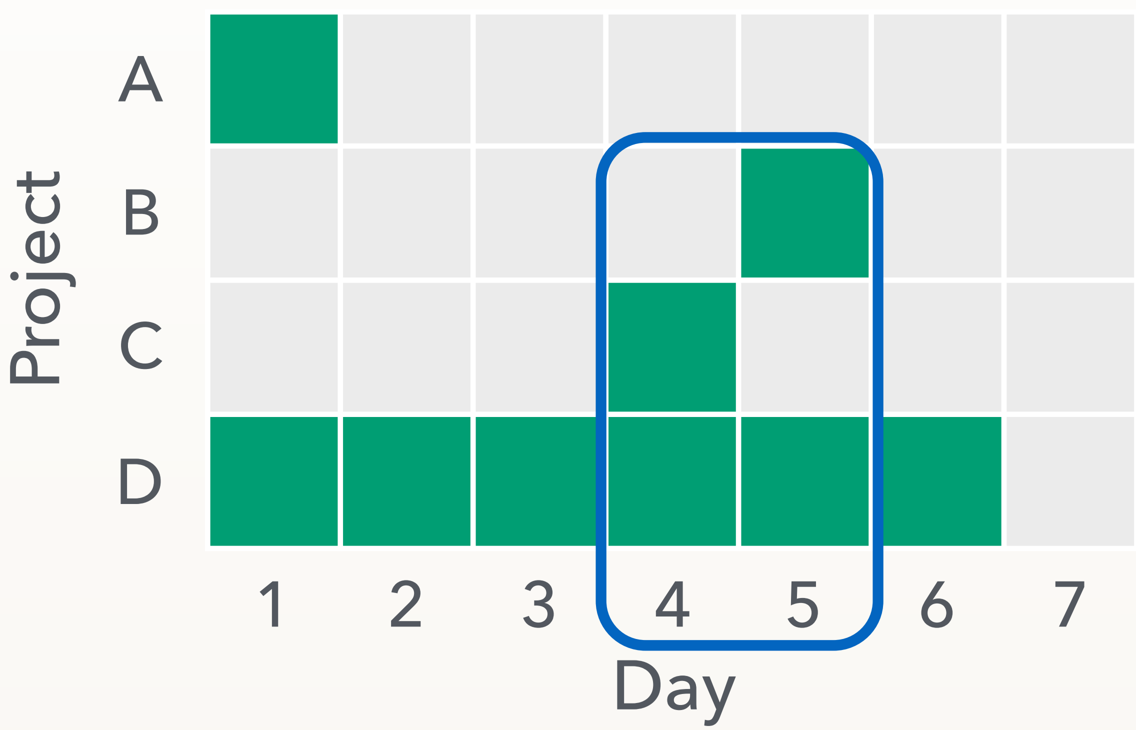
Repetitive day-to-day vs. Switching focus

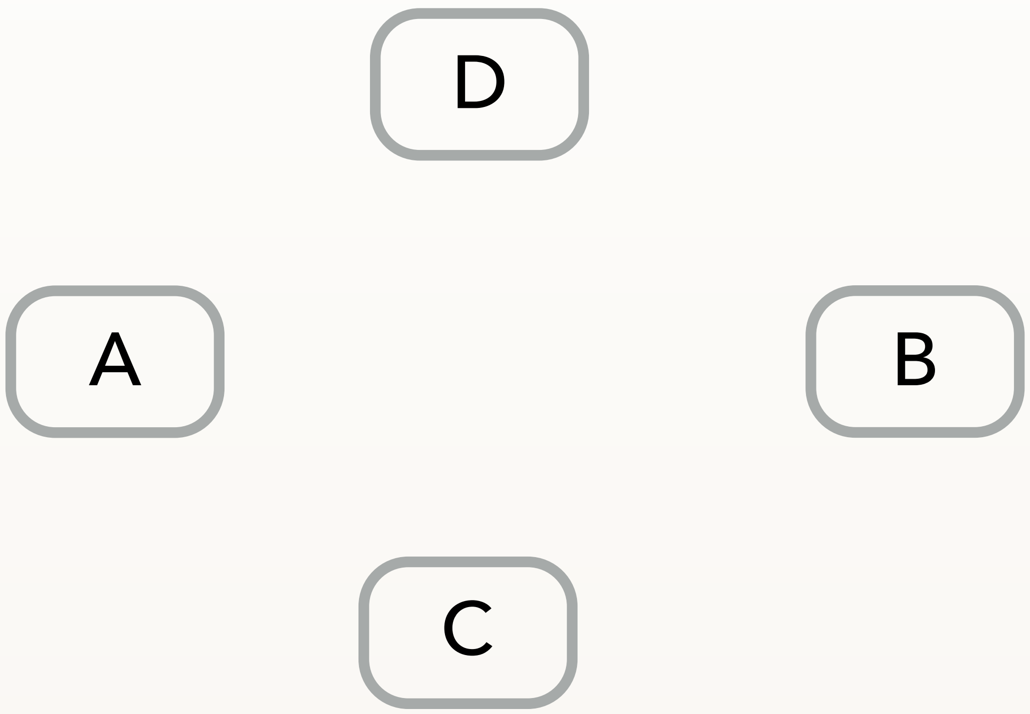




Multitasking dimensions

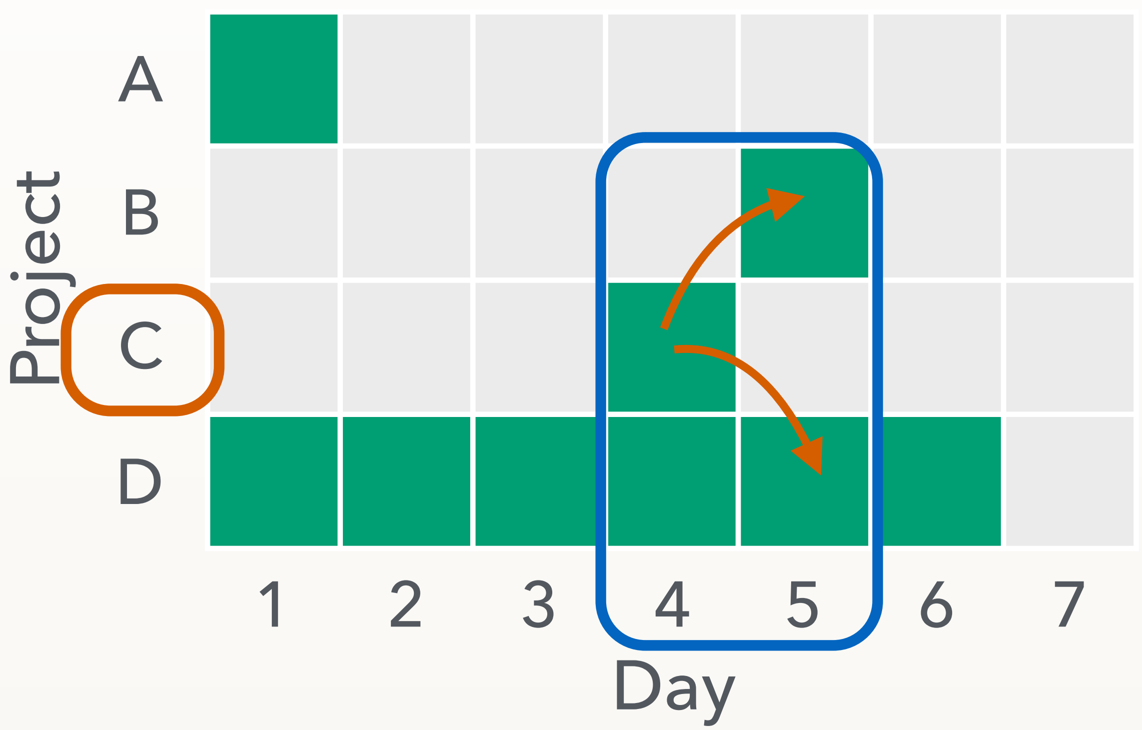
Repetitive day-to-day vs. Switching focus

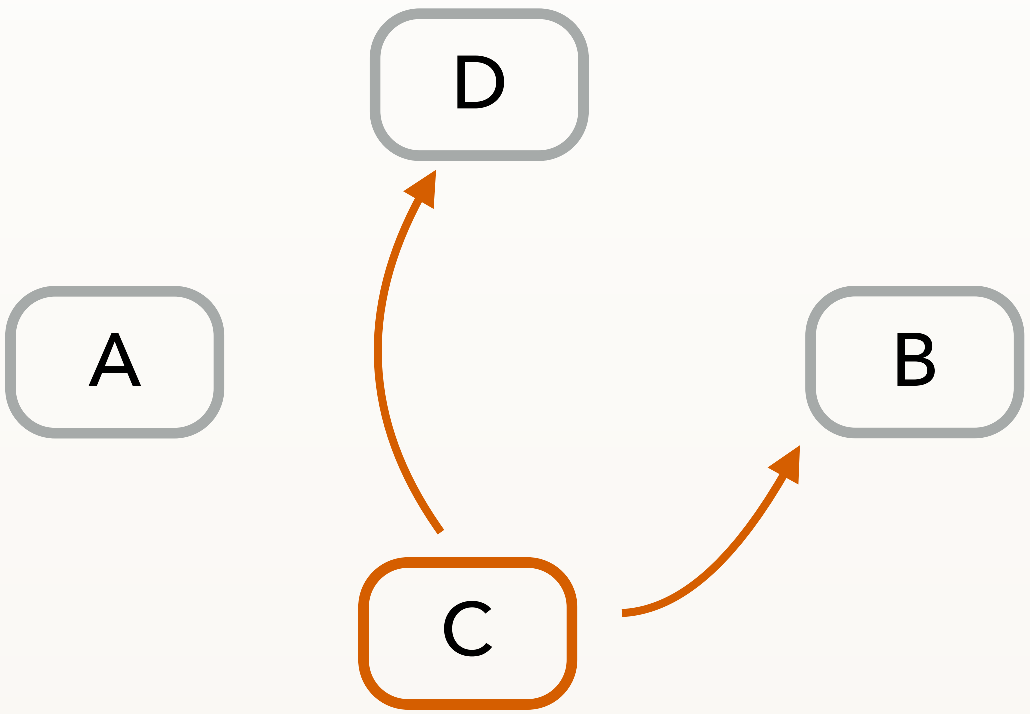




Multitasking dimensions

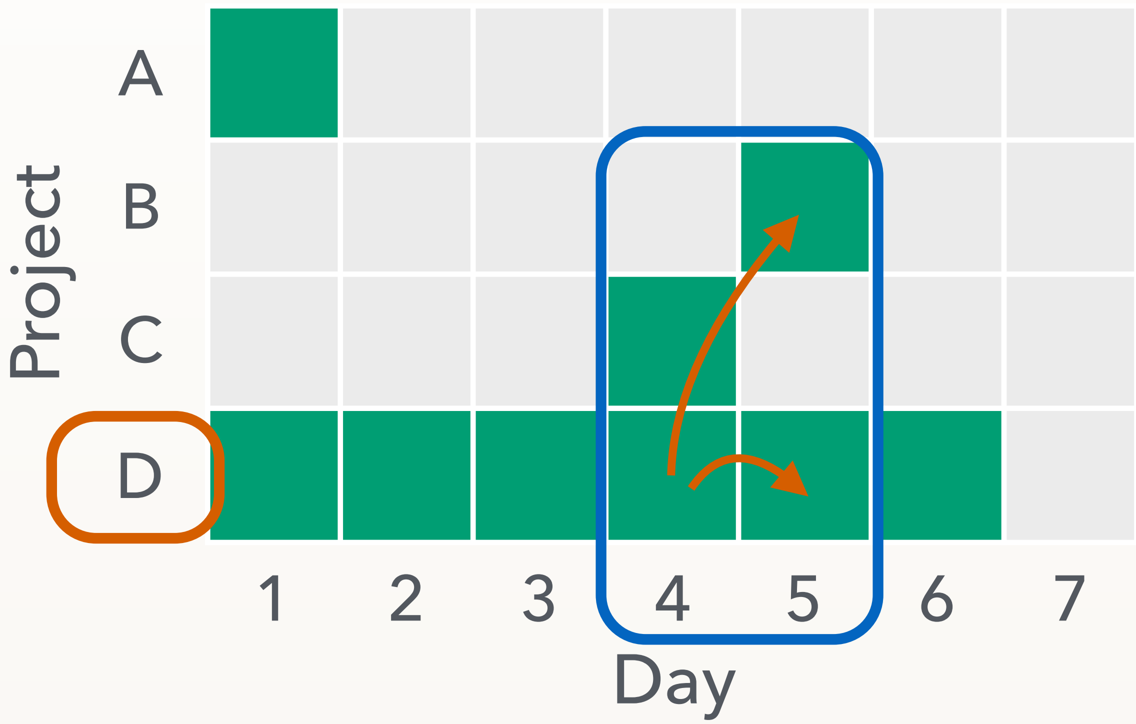
Repetitive day-to-day vs. Switching focus

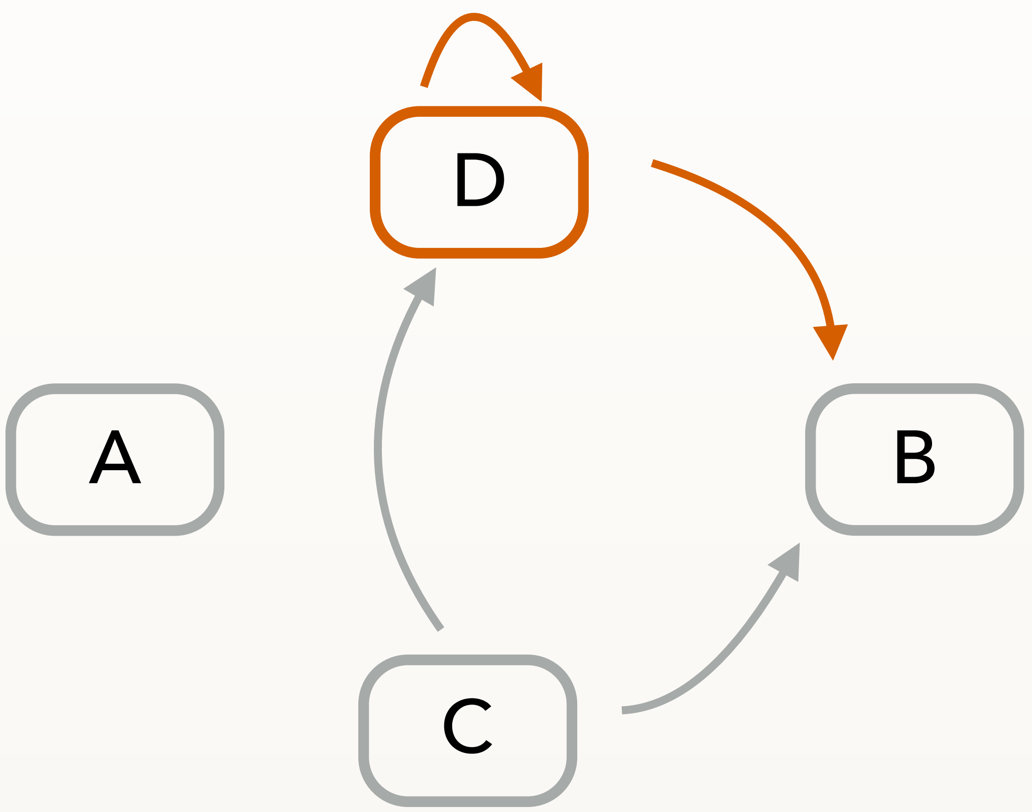




Multitasking dimensions

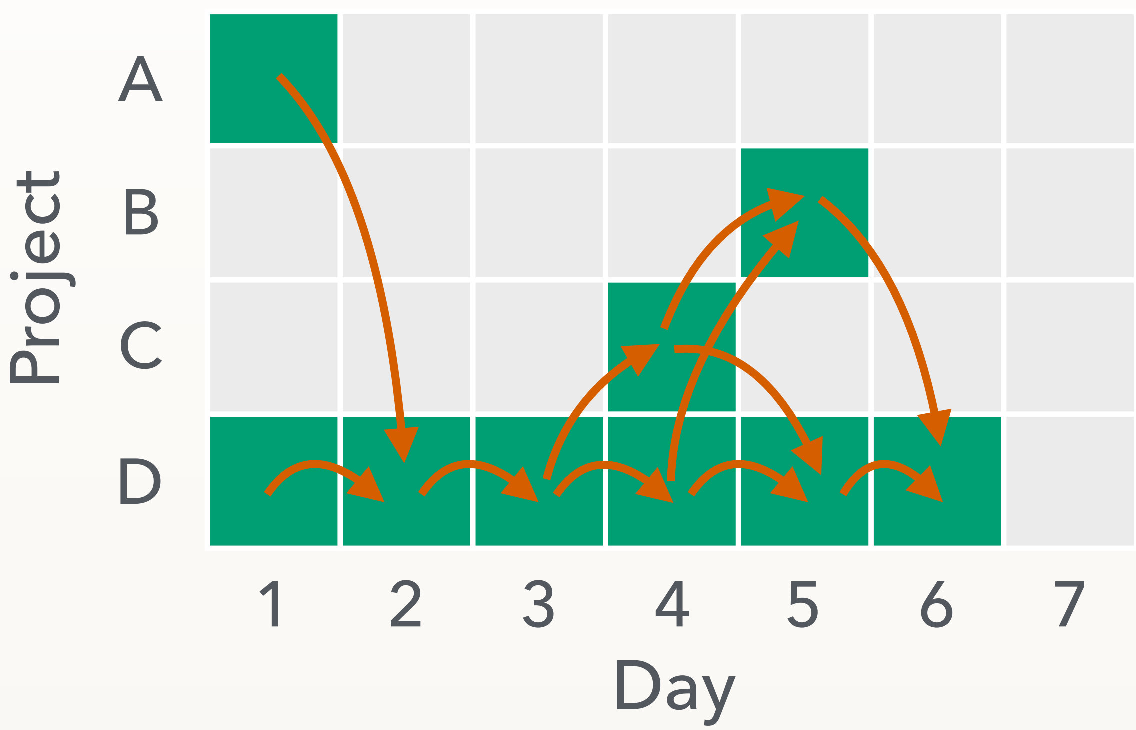
Repetitive day-to-day vs. Switching focus

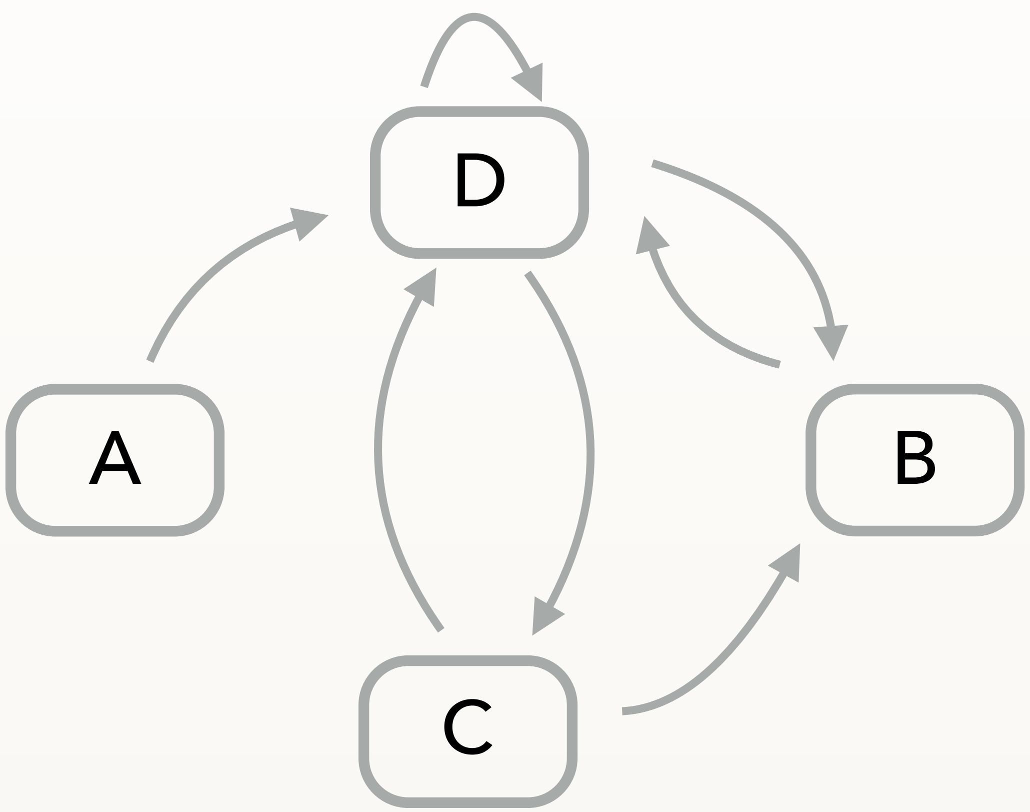




Multitasking dimensions

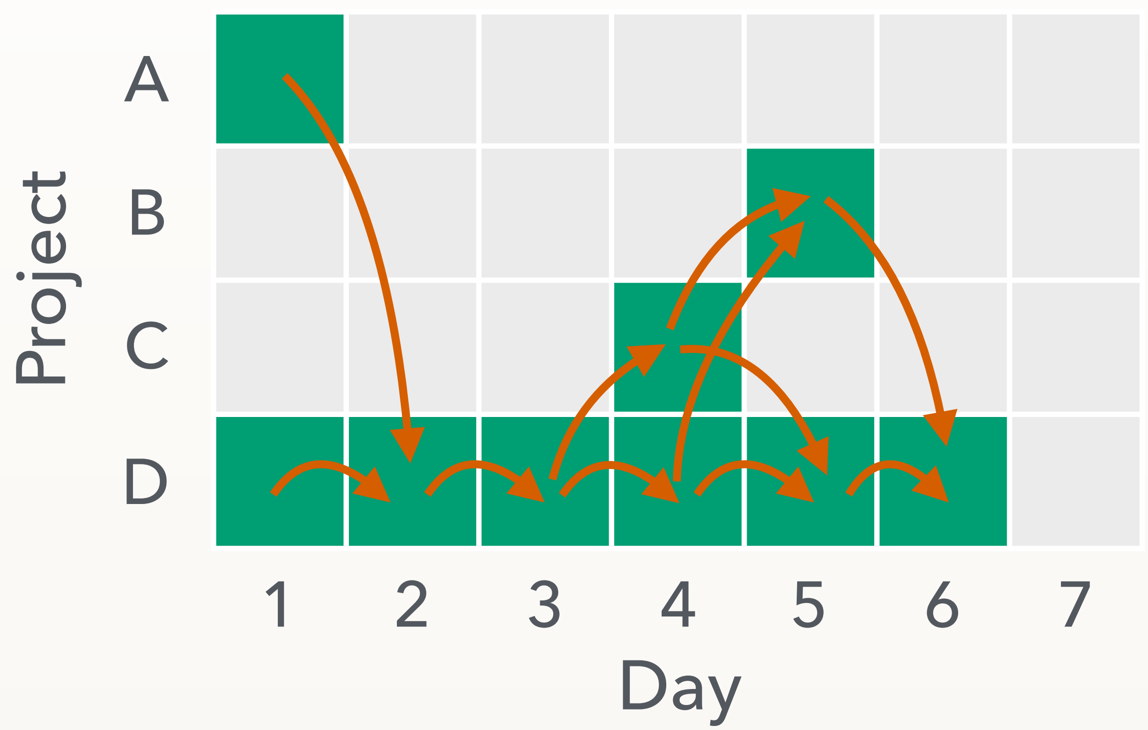
Repetitive day-to-day vs. Switching focus

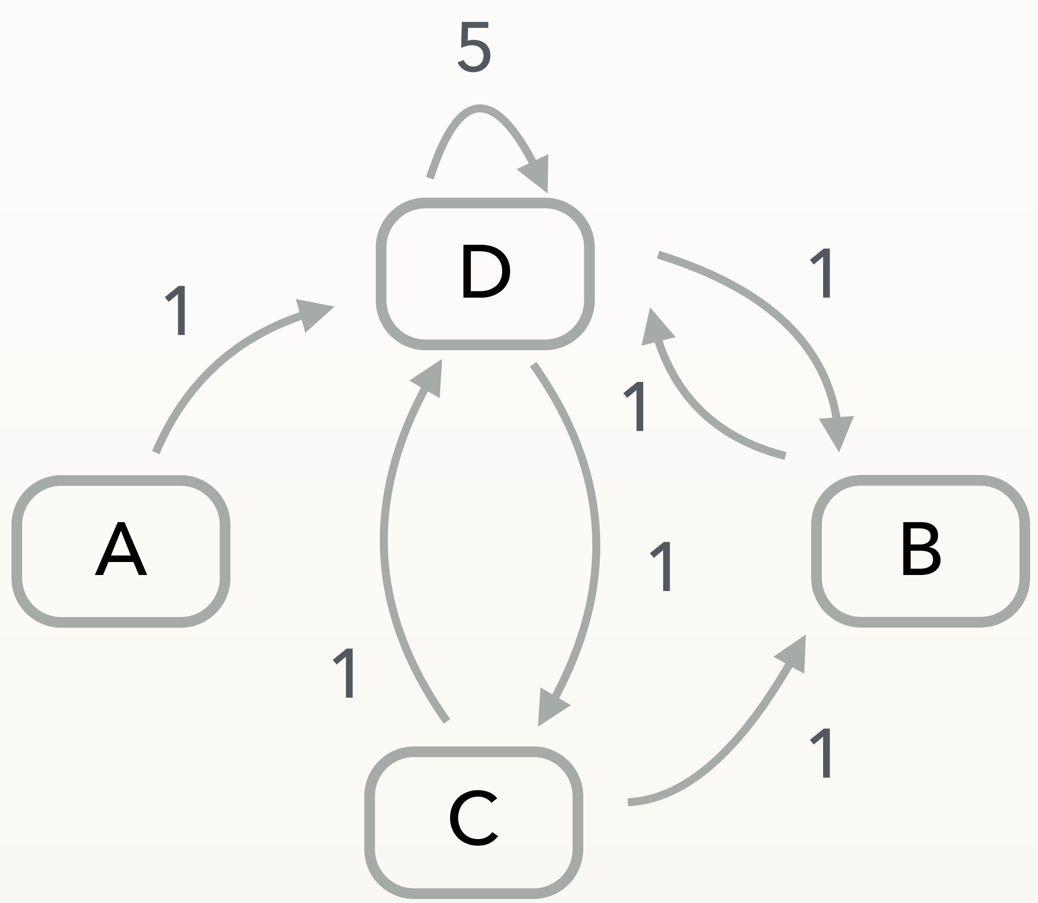




Multitasking dimensions

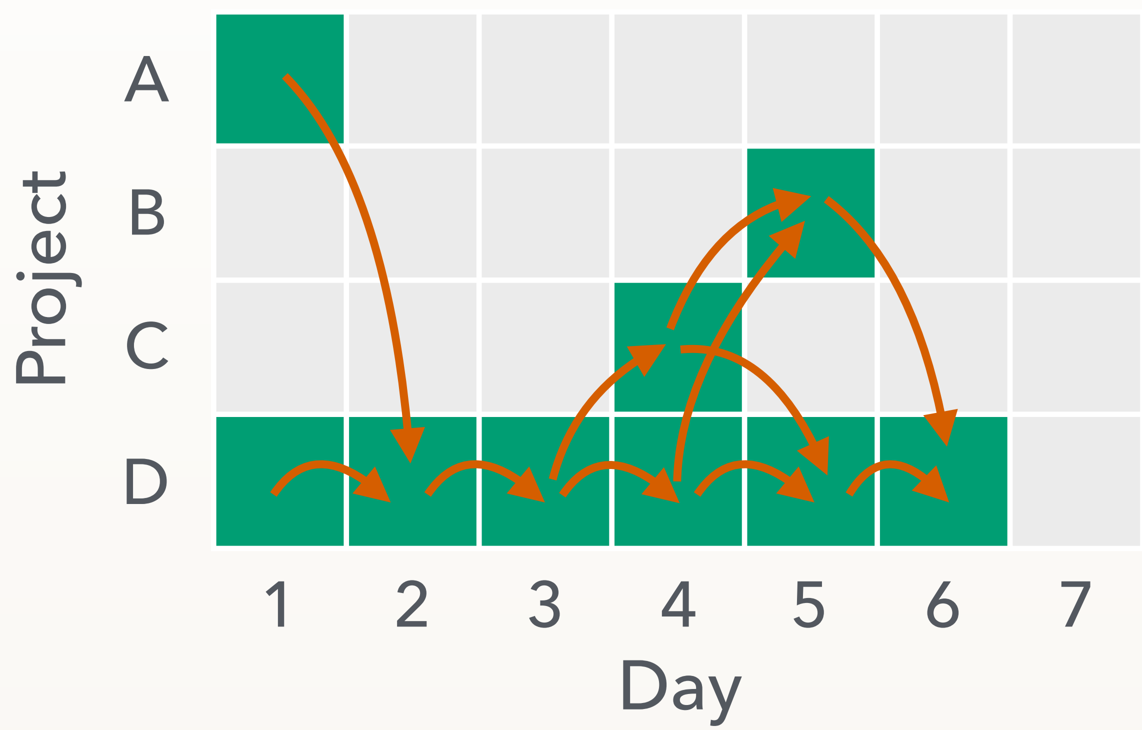
Repetitive day-to-day vs. Switching focus

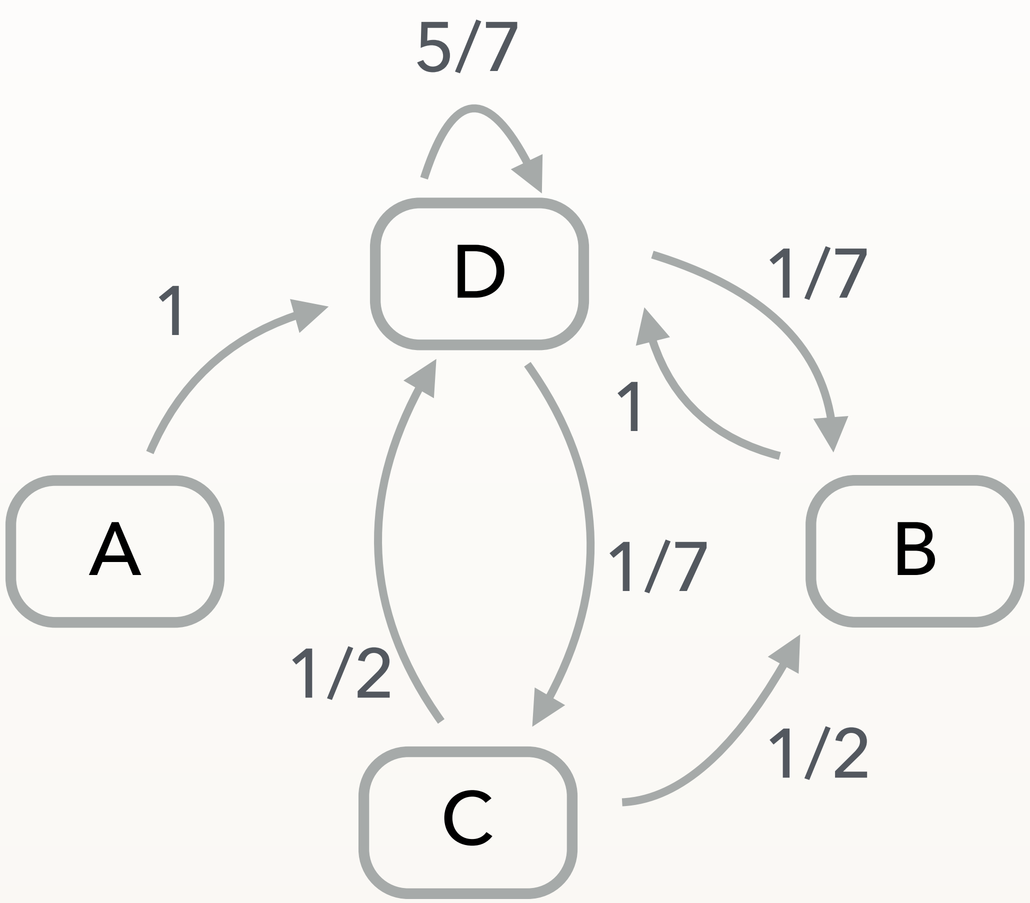




Multitasking dimensions

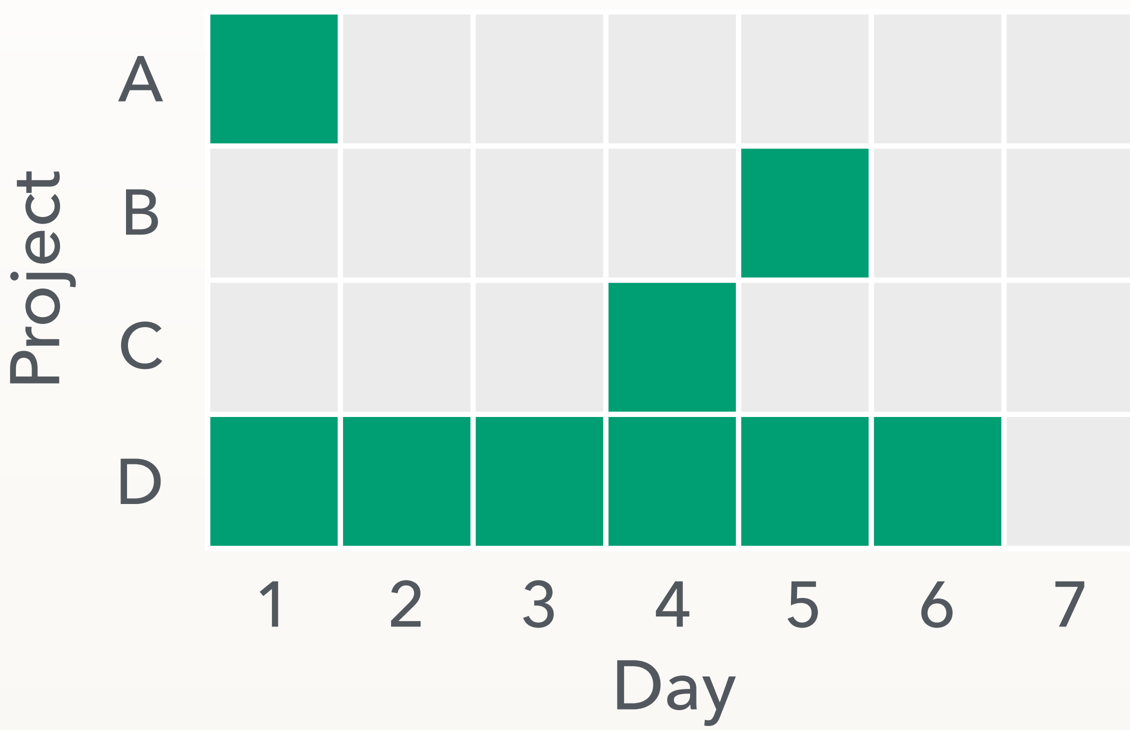
Repetitive day-to-day vs. Switching focus

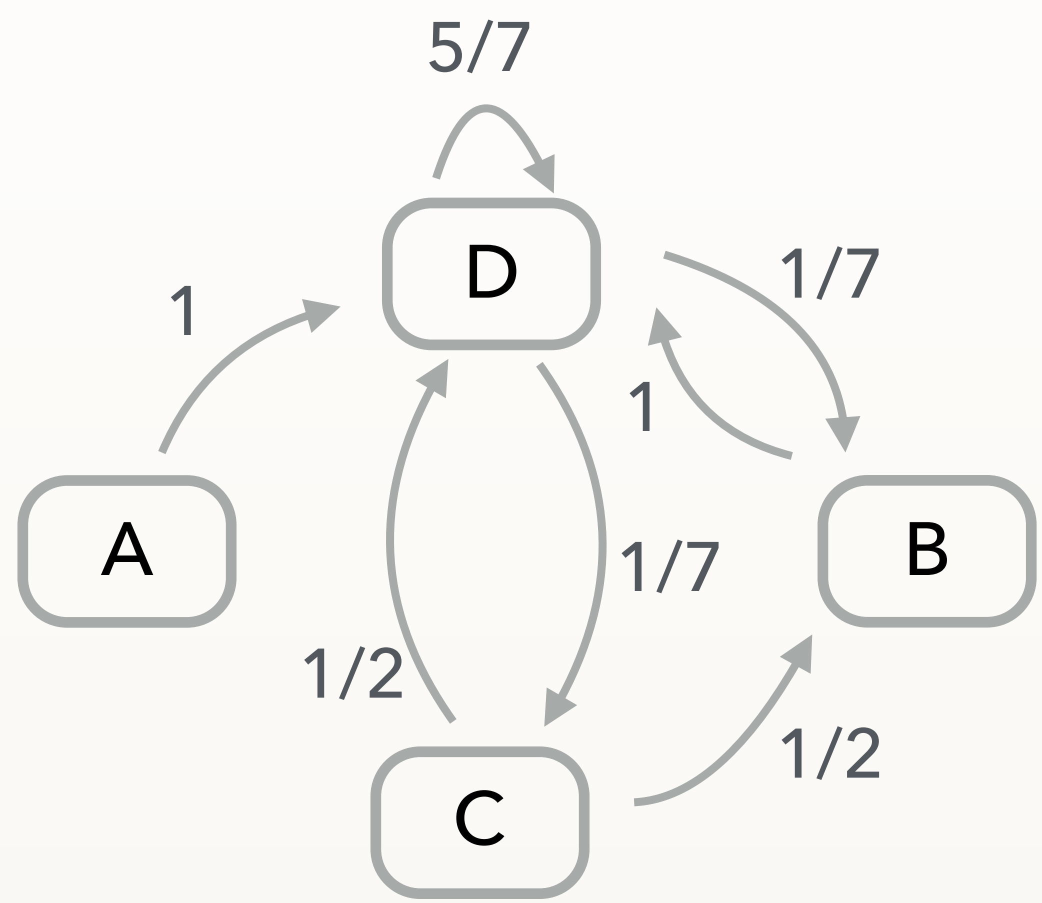




Multitasking dimensions

Repetitive day-to-day vs. Switching focus





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

How predictable is my focus tomorrow if today I work on project j?

Linear mixed-effects regression

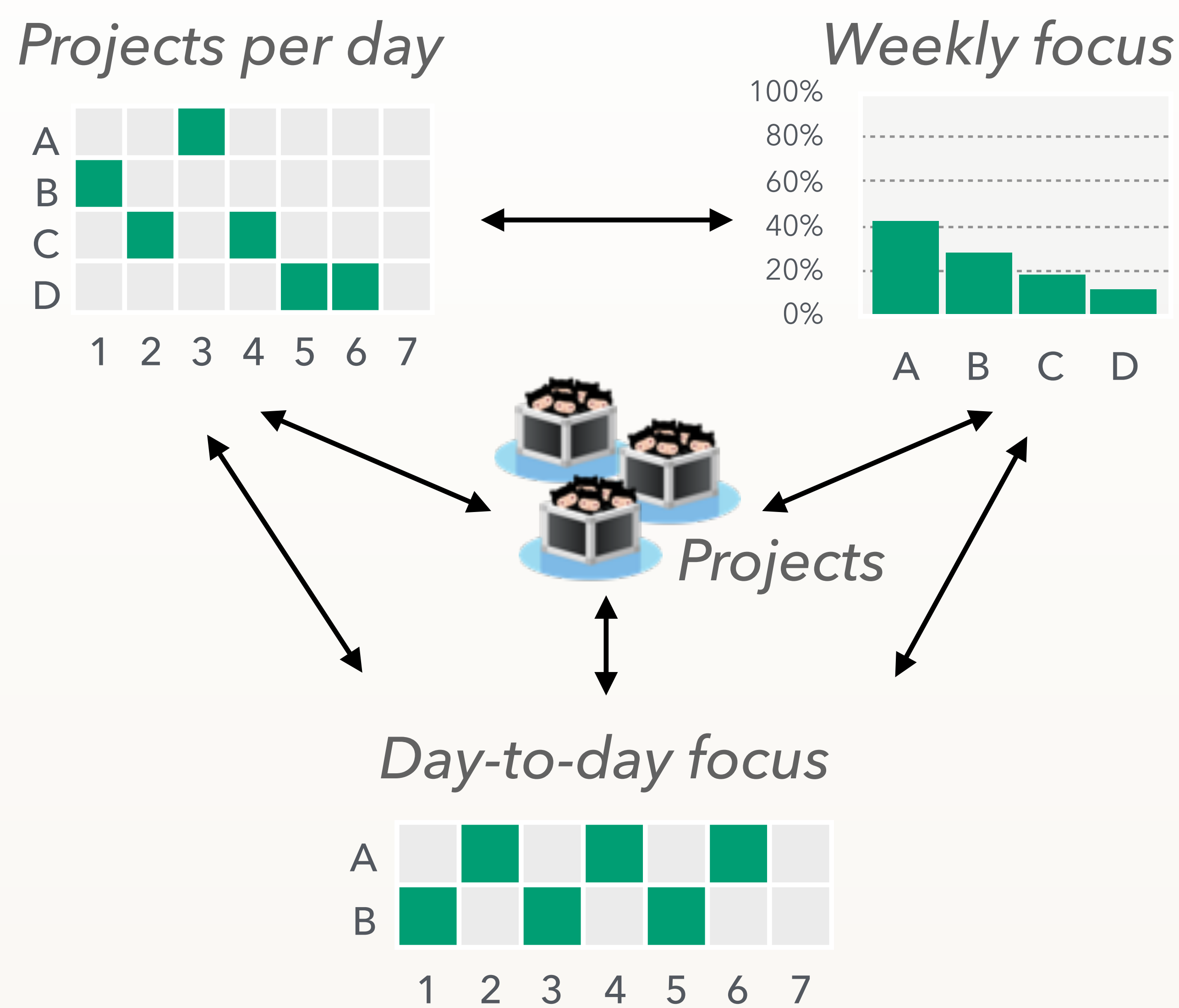
Response:

LOC added / week

Controls:

- time
- total projects
- programming languages

Predictors:



Longitudinal data

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

Random effect: developer

- developer-to-developer variability in the response

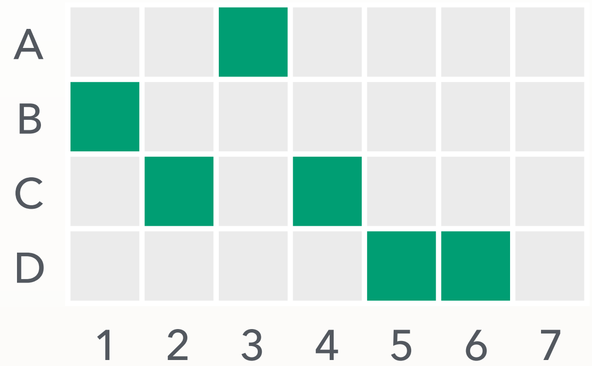
Random slope: time | developer

- developers more productive initially may be less strongly affected by time passing

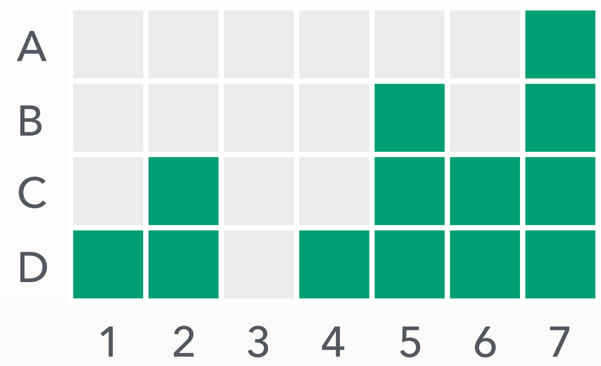
Multitaskers do more; scheduling matters

Higher LOC added

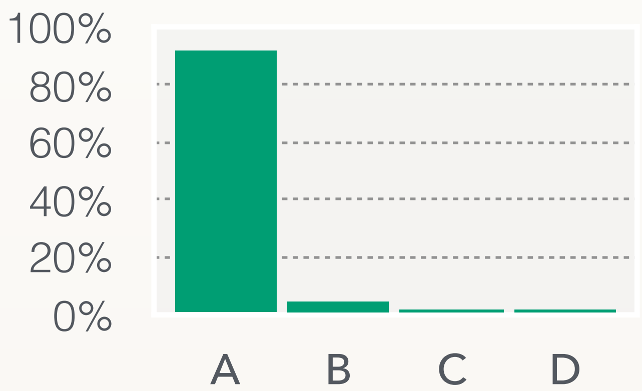
Projects per day



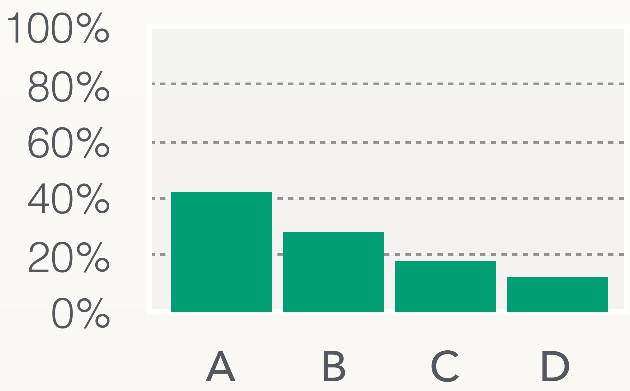
vs.



Weekly focus

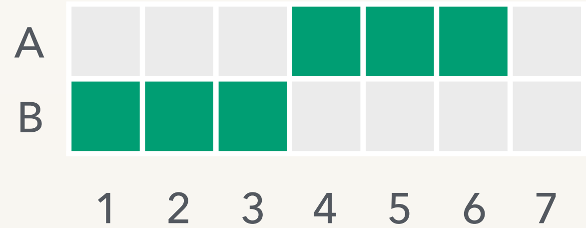


vs.



Day-to-day focus (repeatability)

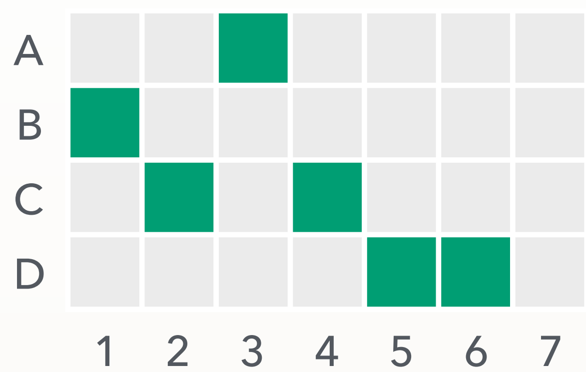
vs.



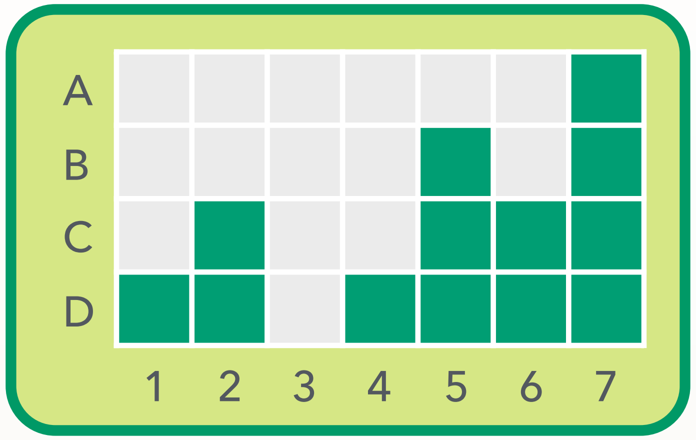
Multitaskers do more; scheduling matters

Higher LOC added

Projects per day

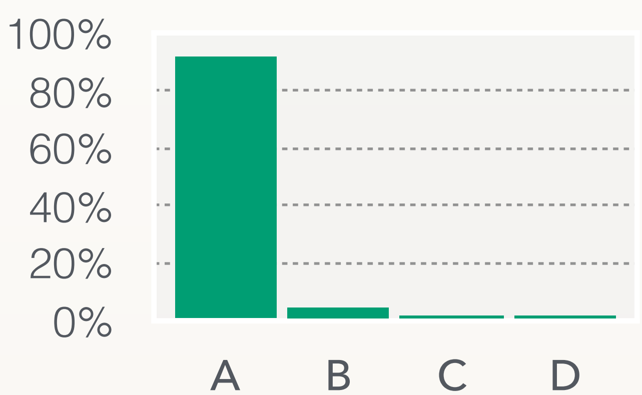


vs.



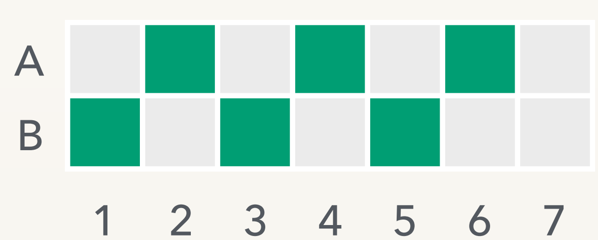
More within-day multitasking

Weekly focus

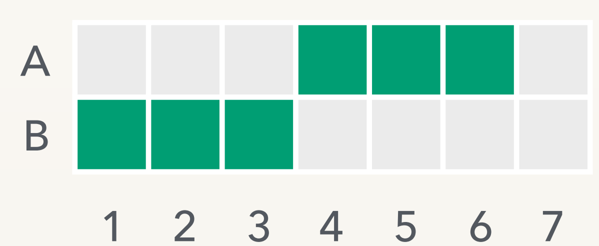


vs.

Day-to-day focus (repeatability)



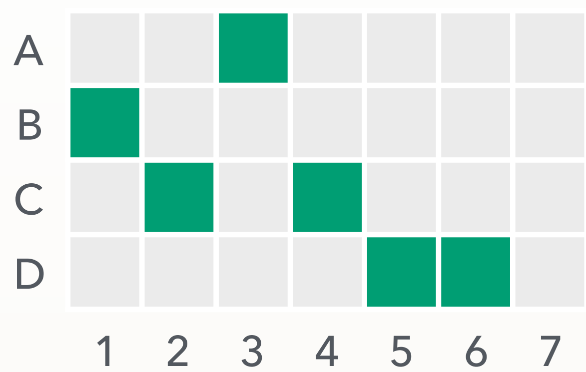
vs.



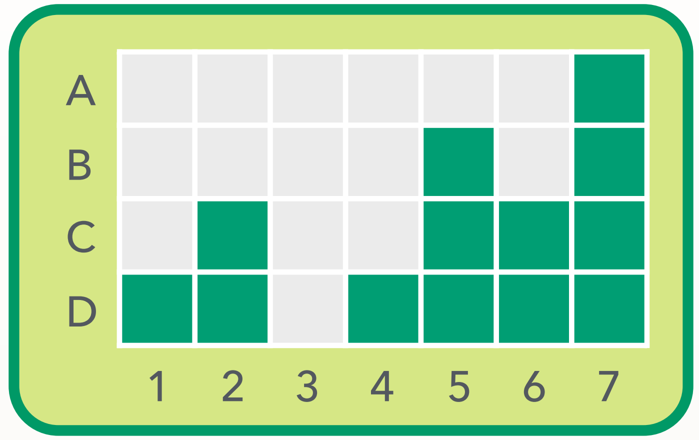
Multitaskers do more; scheduling matters

Higher LOC added

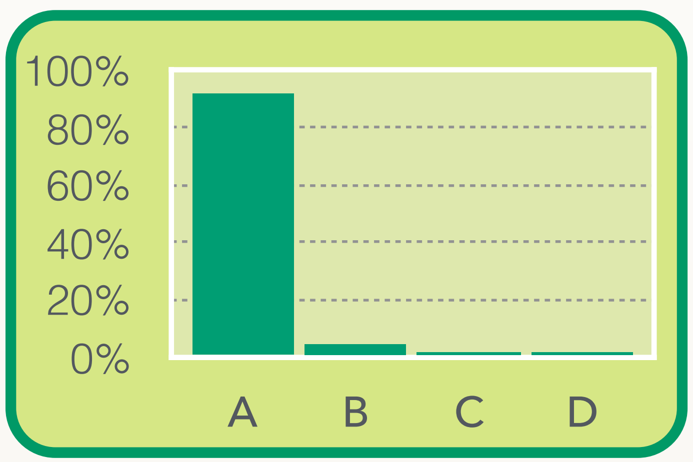
Projects per day



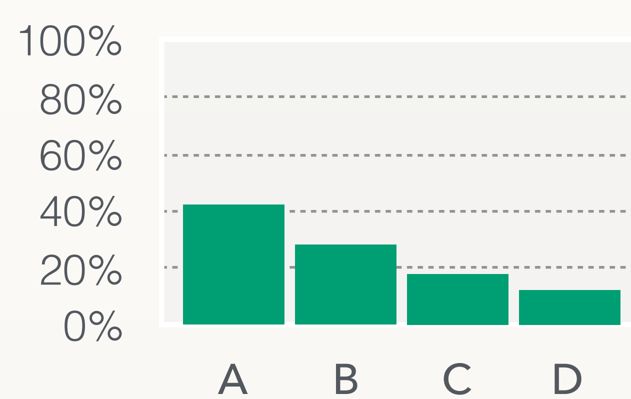
vs.



Weekly focus

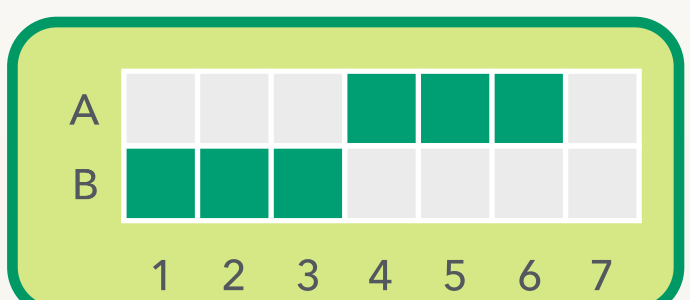


vs.



Day-to-day focus (repeatability)

vs.



More within-day multitasking

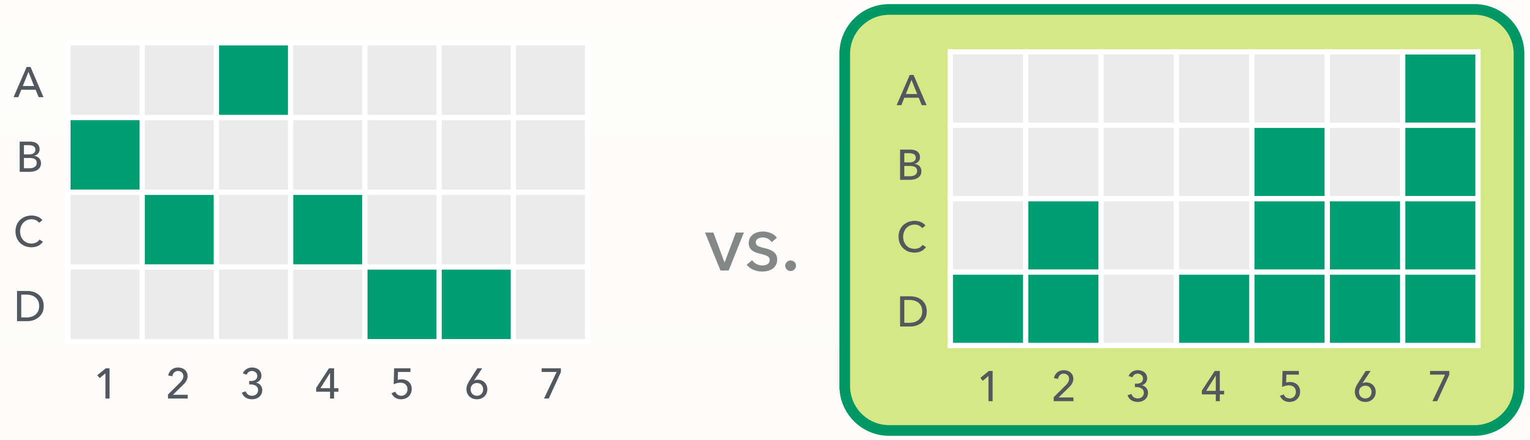
Higher focus

More repetitive day-to-day work

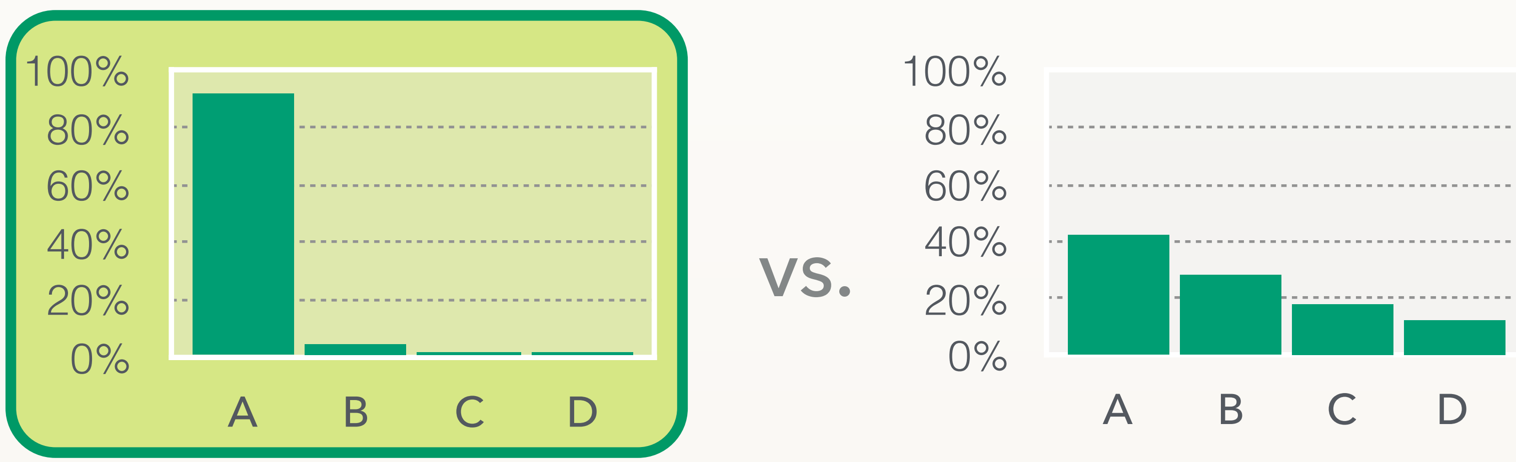
Multitaskers do more; scheduling matters

Higher LOC added

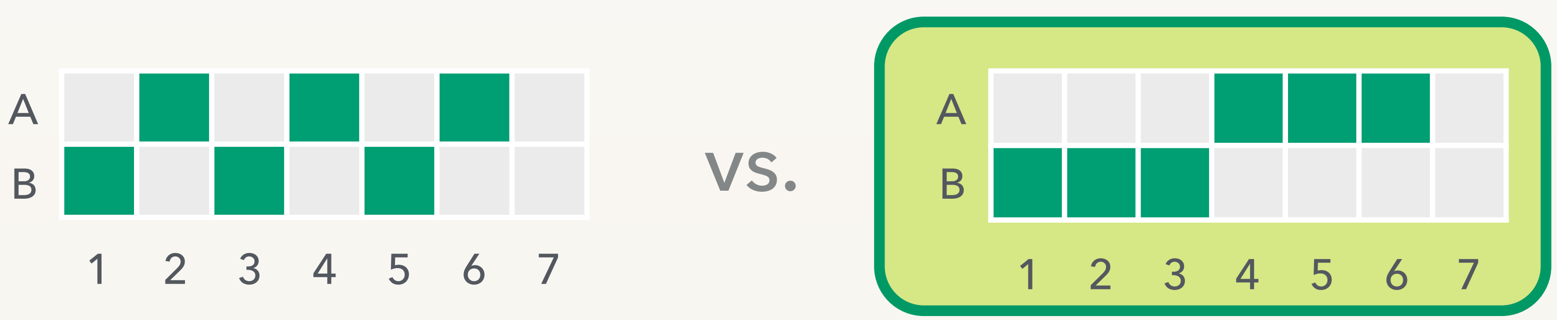
Projects per day



Weekly focus



Day-to-day focus (repeatability)



More within-day multitasking

Higher focus

More repetitive day-to-day work

<u>Interaction effects</u>:

No scheduling is productive over 5 projects/week

How long will my pull request take?

Yu, Y., Wang, H., Filkov, V., Devanbu, P., & Vasilescu, B. (2015, May). Wait for it: determinants of pull request evaluation latency on GitHub. In *Mining software repositories (MSR), 2015 IEEE/ACM 12th working conference on* (pp. 367-371). IEEE.

The pull-based model

... traditionally

Octocats from https://octodex.github.com/

The pull-based model

… traditionally

Octocats from https://octodex.github.com/

The pull-based model

... traditionally

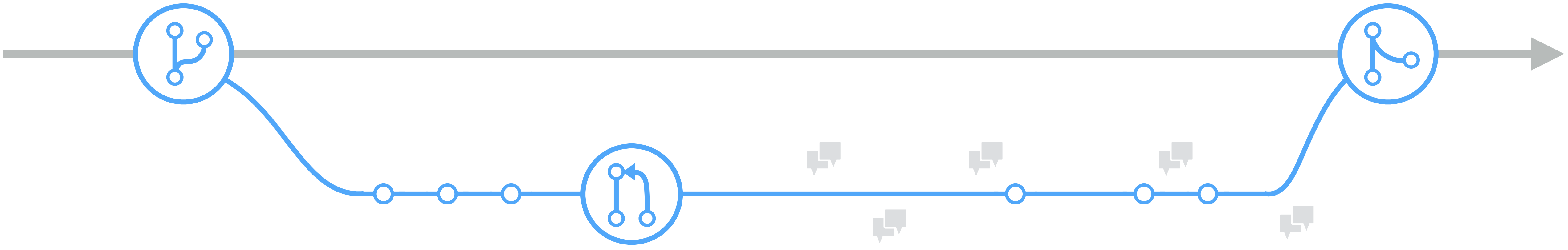
Octocats from https://octodex.github.com/

The pull-based model

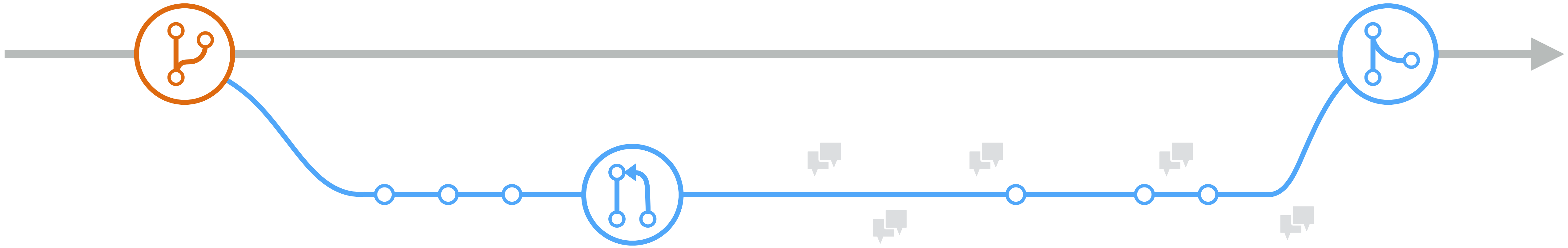
... traditionally

Octocats from https://octodex.github.com/

The Pull Request process

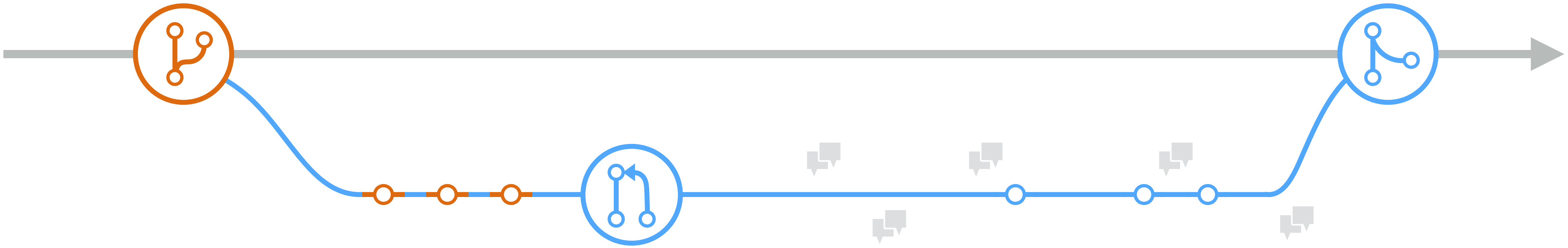


The Pull Request process



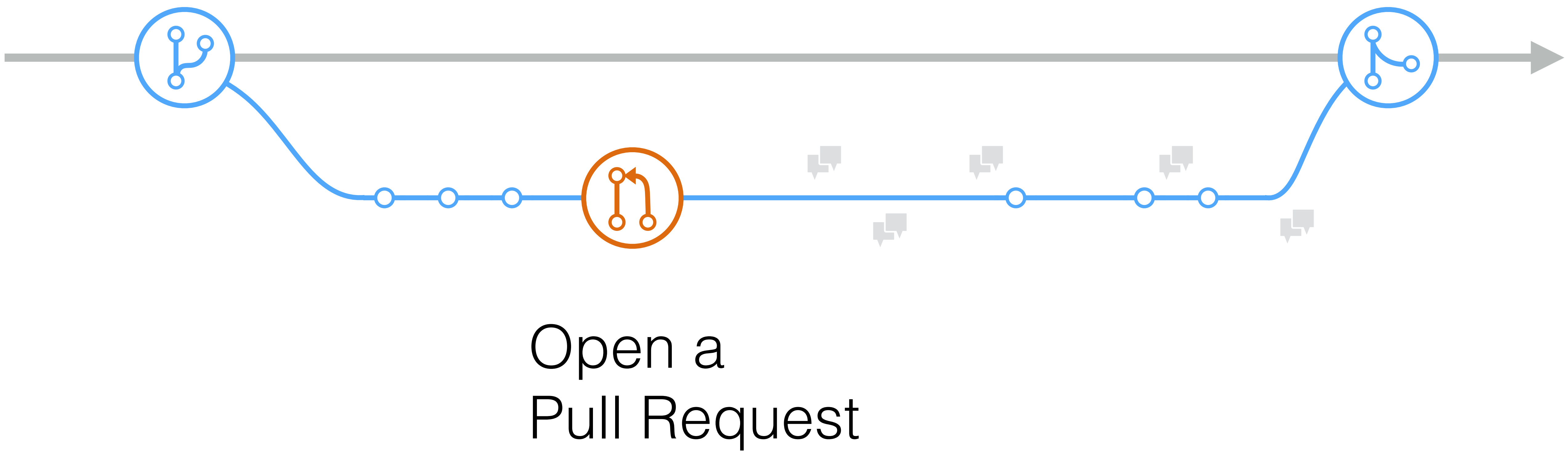
Create
a branch

The Pull Request process

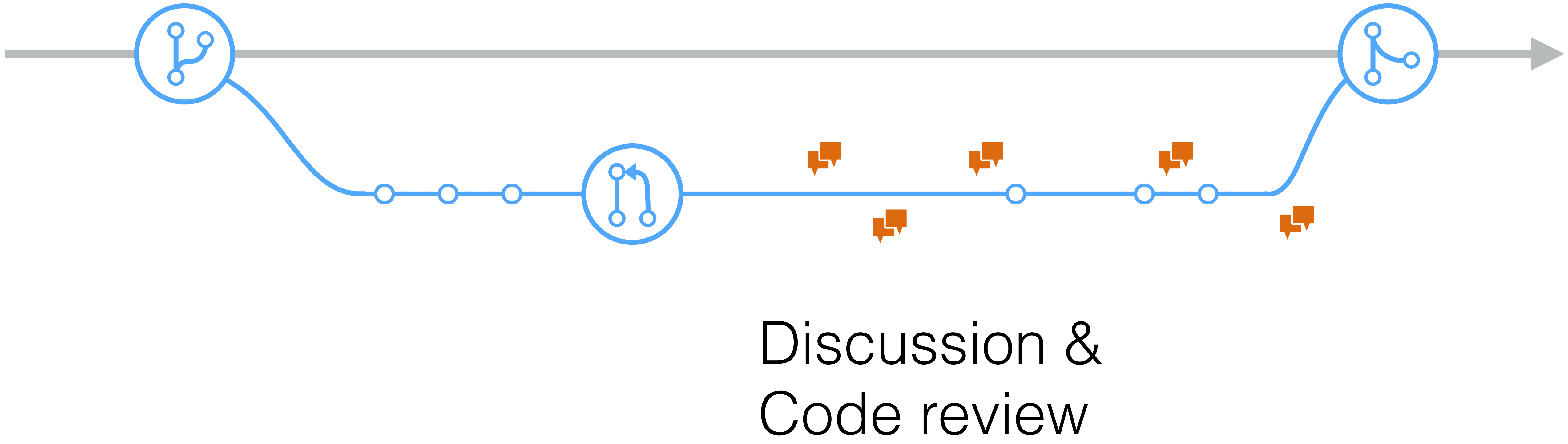


Add
commits

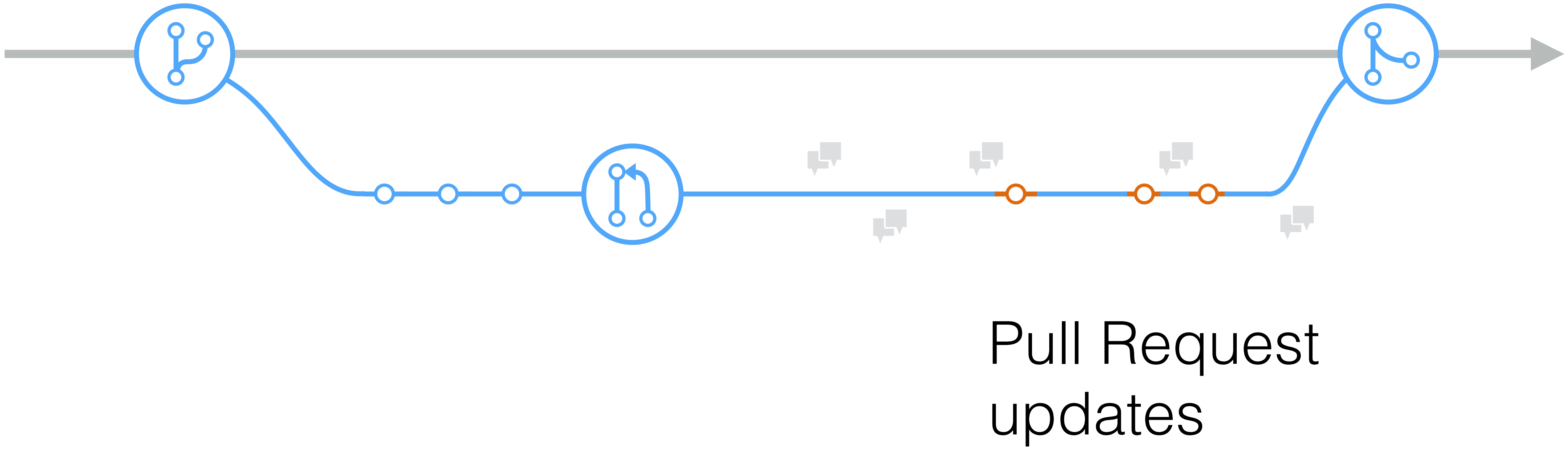
The Pull Request process



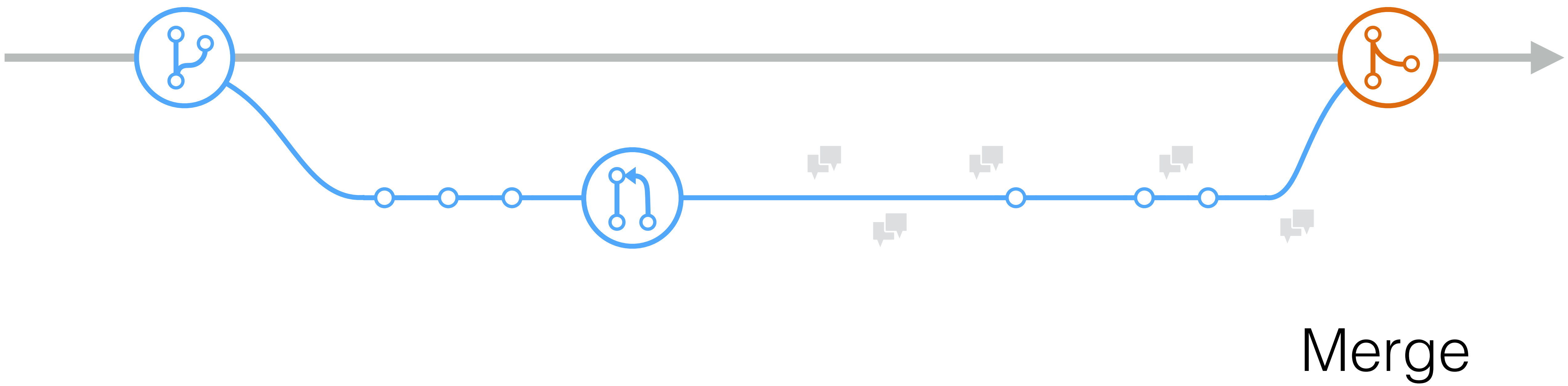
The Pull Request process



The Pull Request process



The Pull Request process



The pull-based model

... modernly

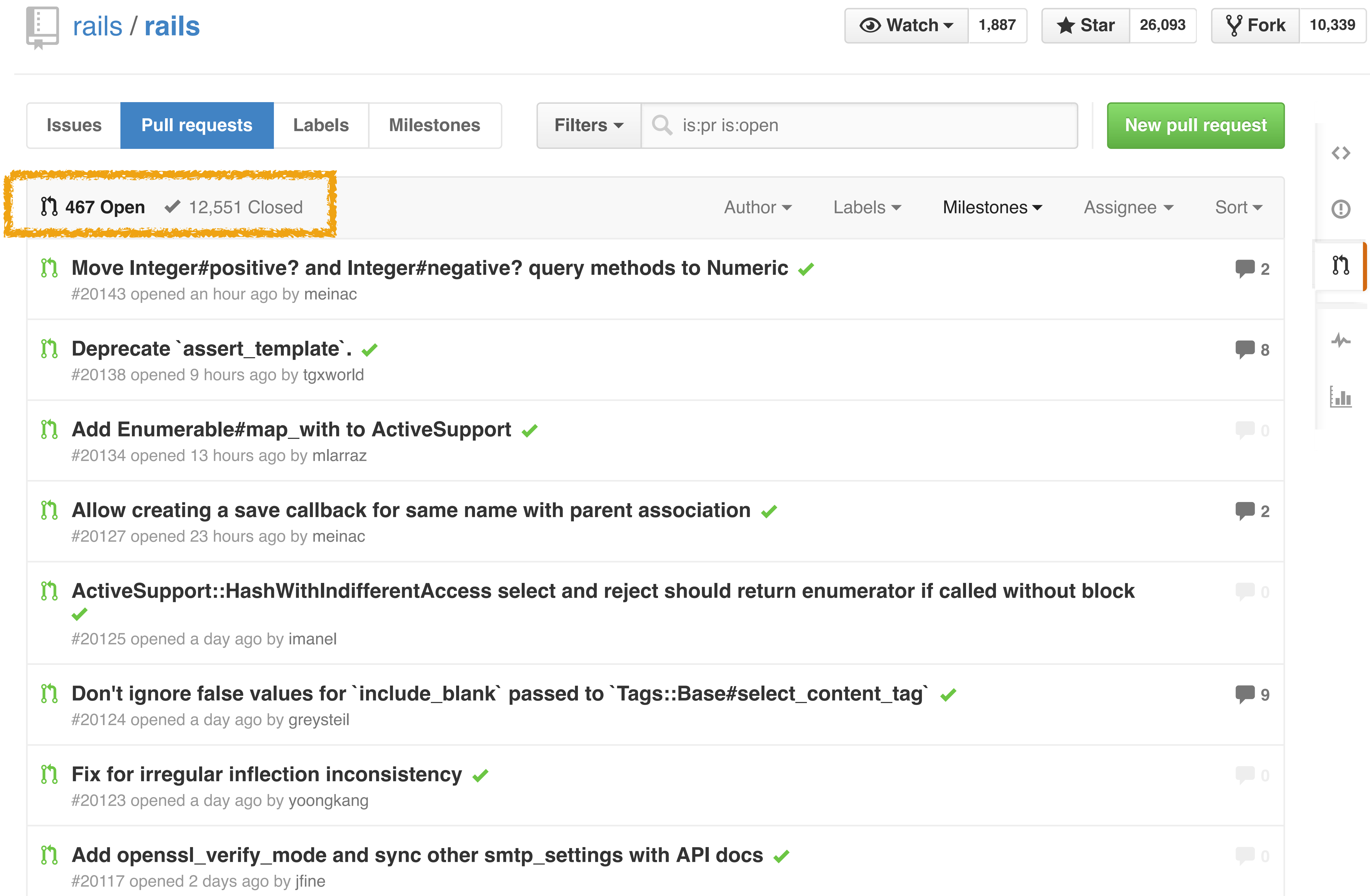
The pull-based model

... modernly

... because code review

- 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*

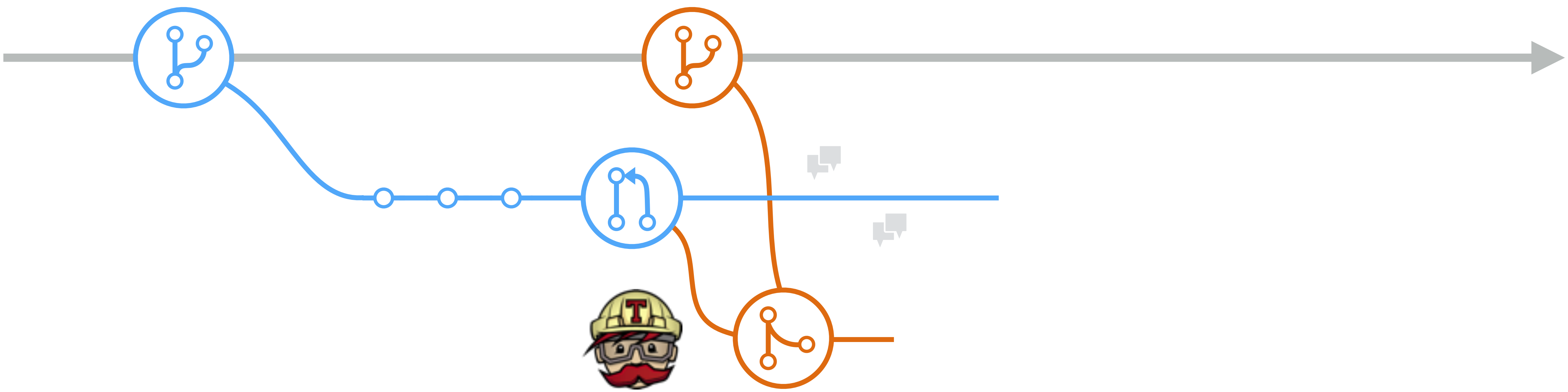
Considerable review load



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*

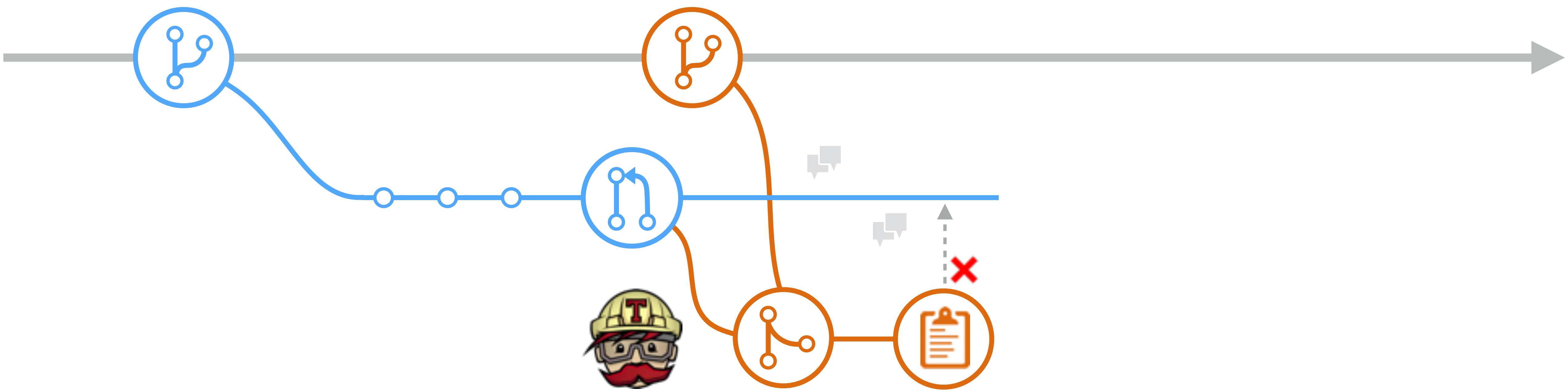
The Pull Request process
… with Travis-CI



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*

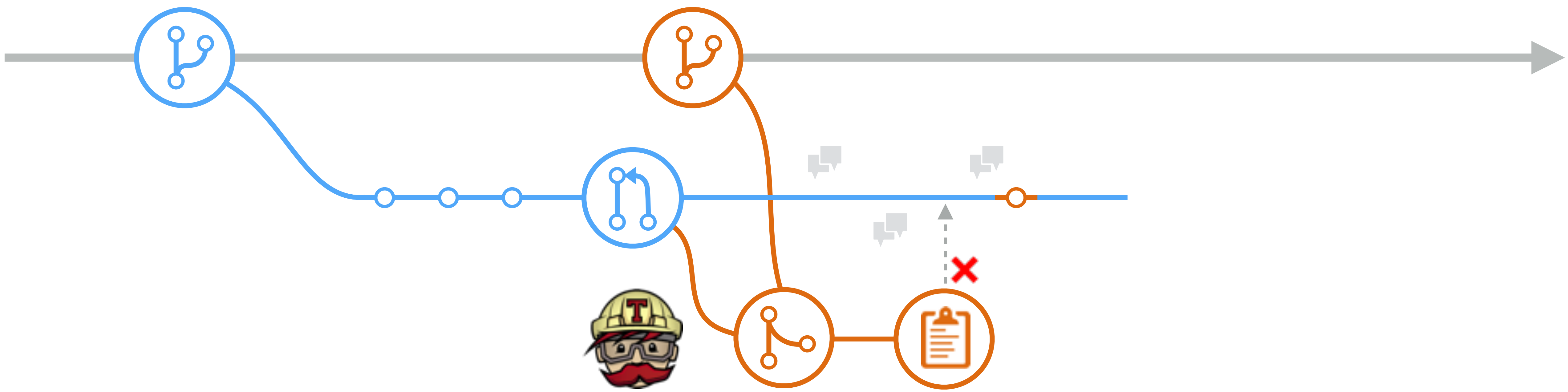
The Pull Request process
… with Travis-CI



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*

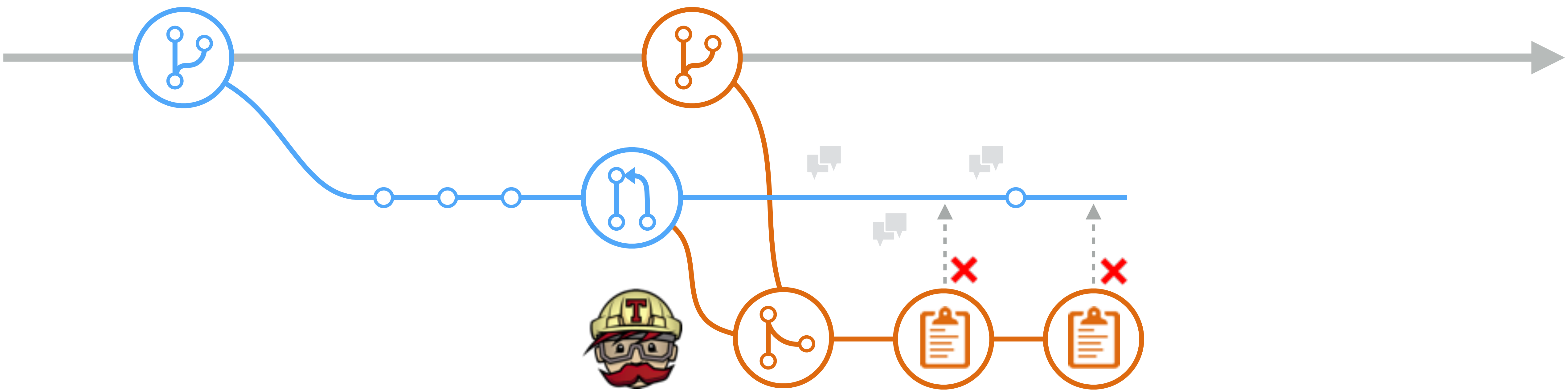
The Pull Request process
… with Travis-CI



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*

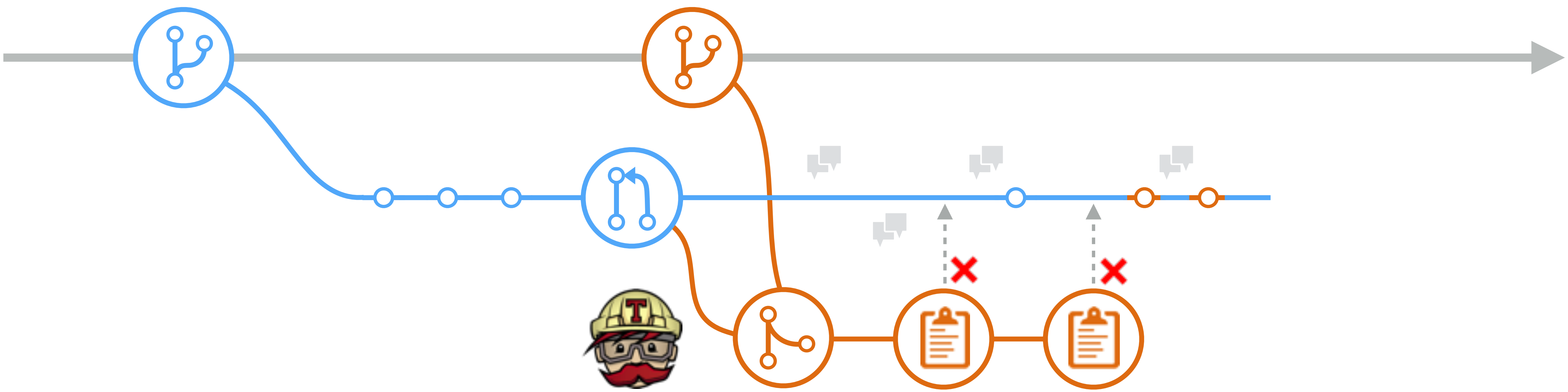
The Pull Request process
… with Travis-CI



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*

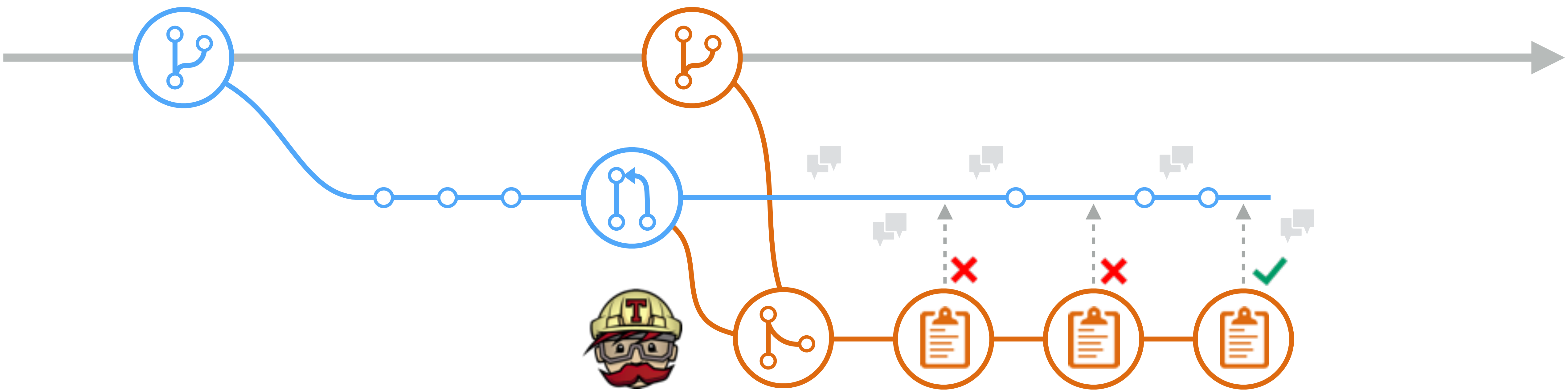
The Pull Request process
… with Travis-CI



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*

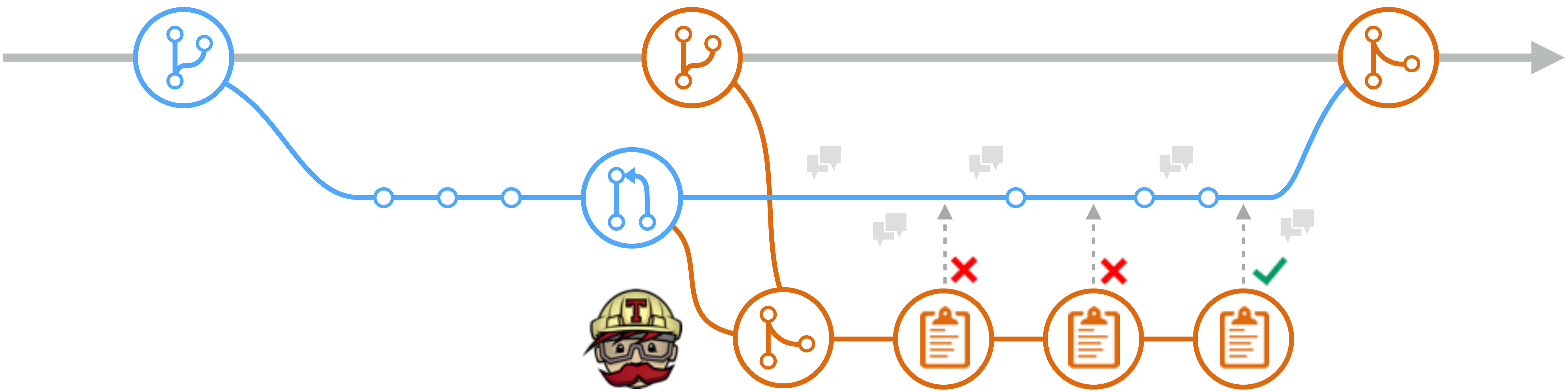
The Pull Request process
… with Travis-CI



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*

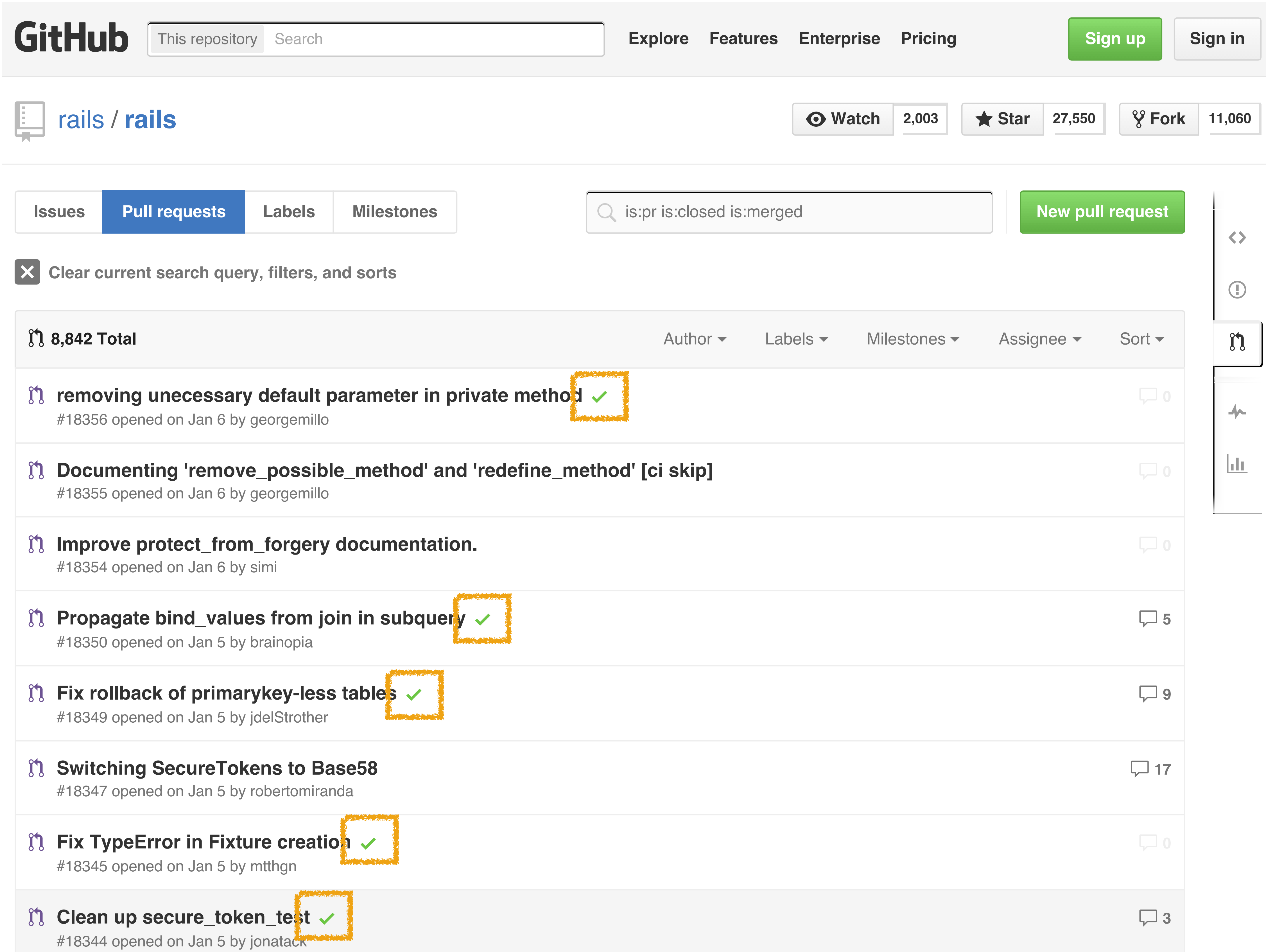
The Pull Request process
… with Travis-CI



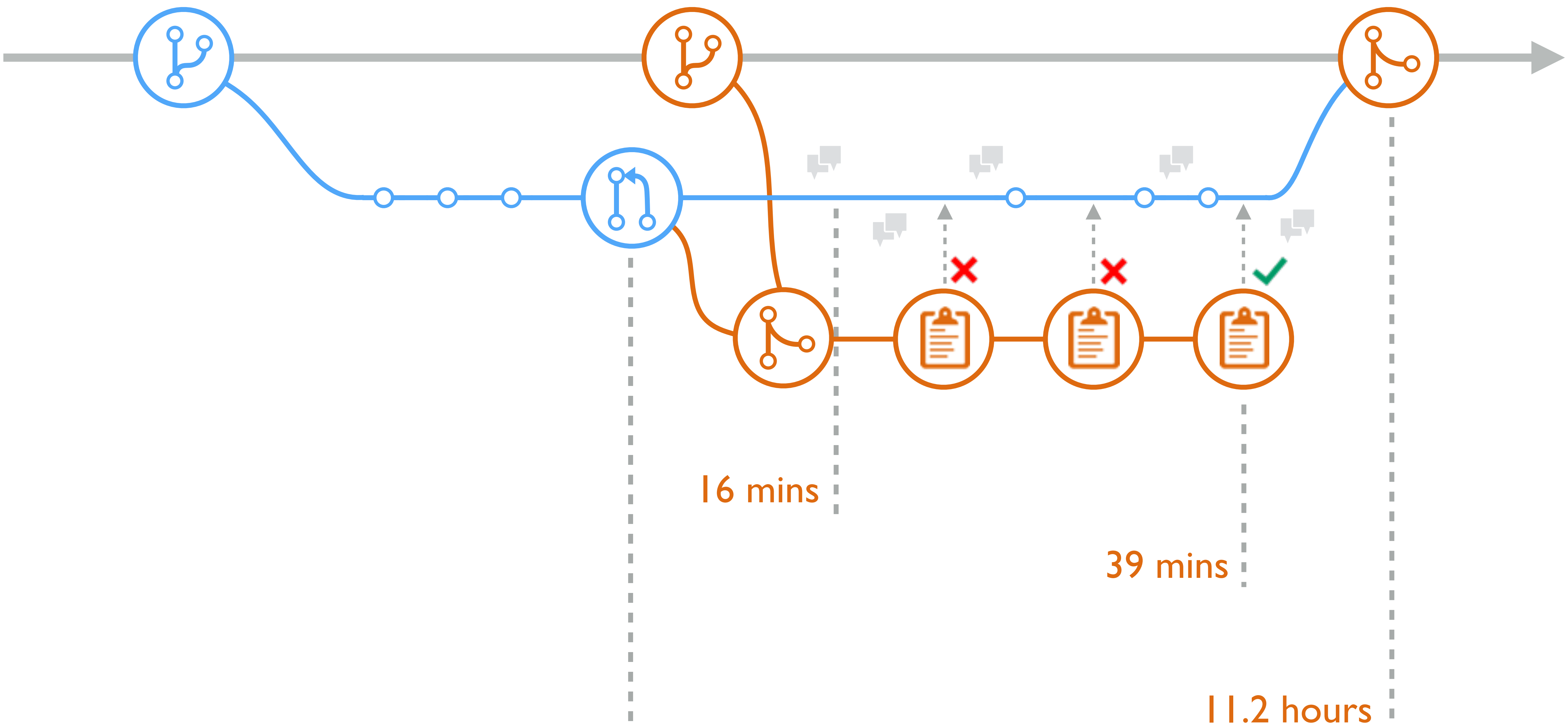
Merge

- 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

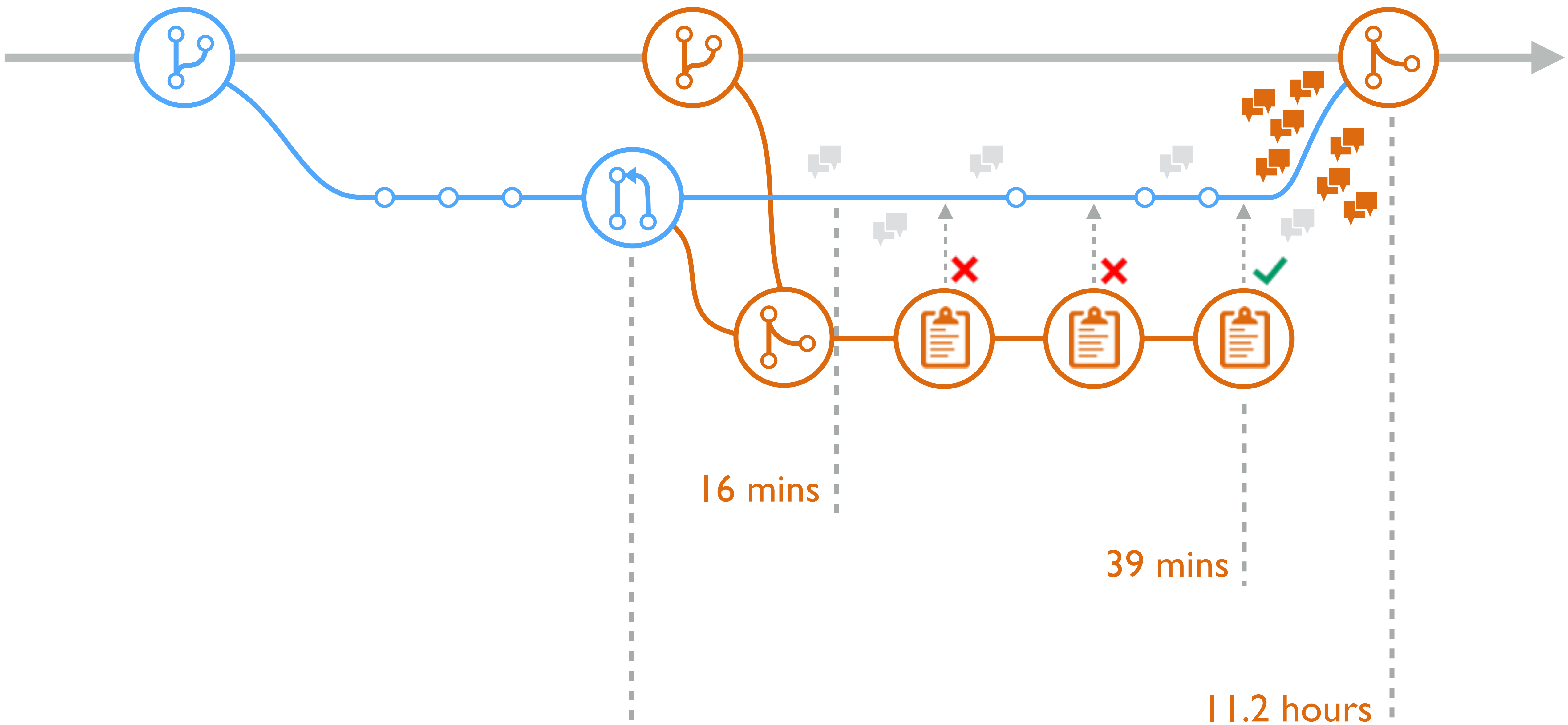


~~Merge~~ after CI tests pass
Code review



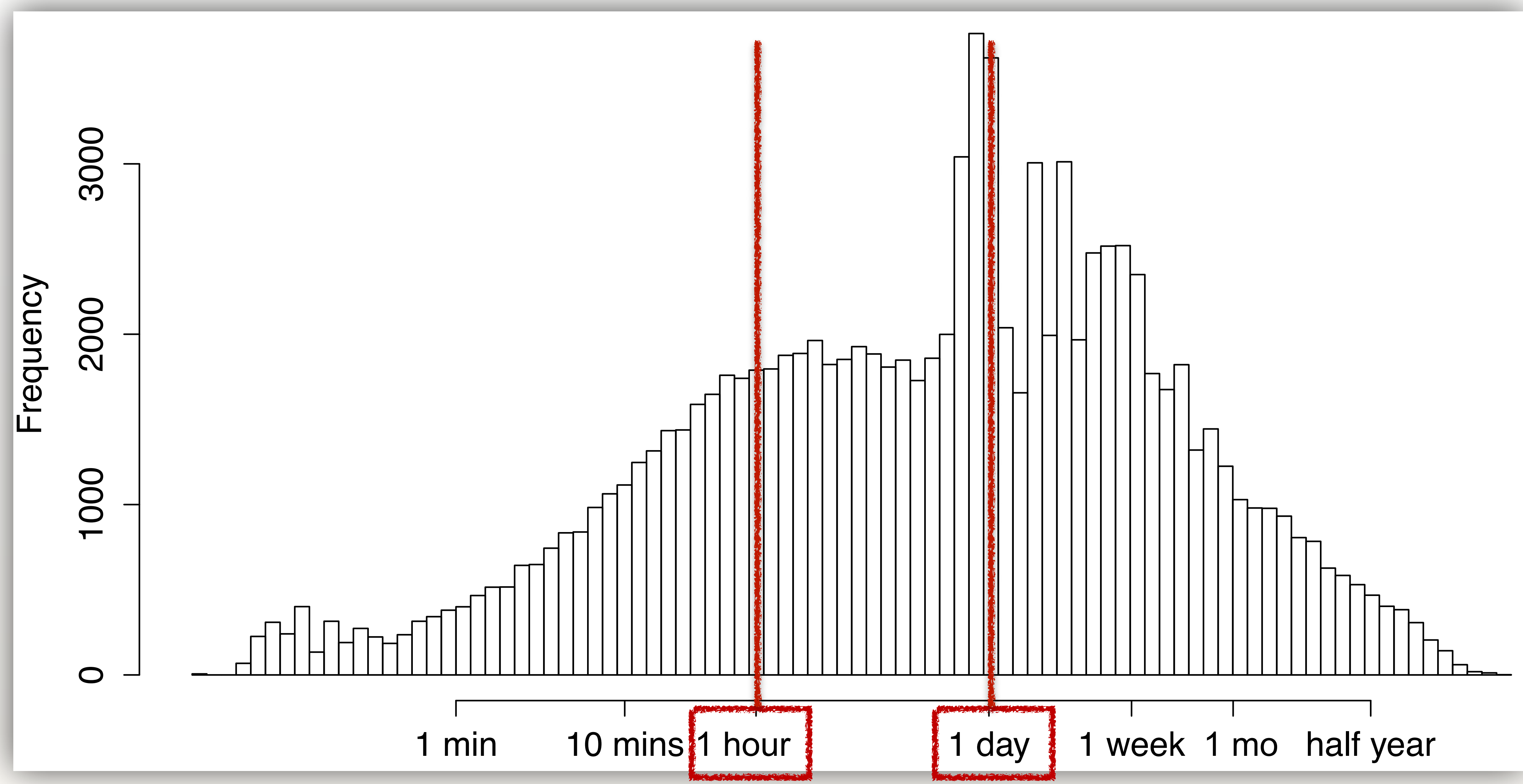
- Wait for it: Determinants of pull request evaluation latency on GitHub
  Y Yu, H Wang, V Filkov, P Devanbu, B Vasilescu. *MSR 2015*

~~Merge~~ after CI tests pass
Code review

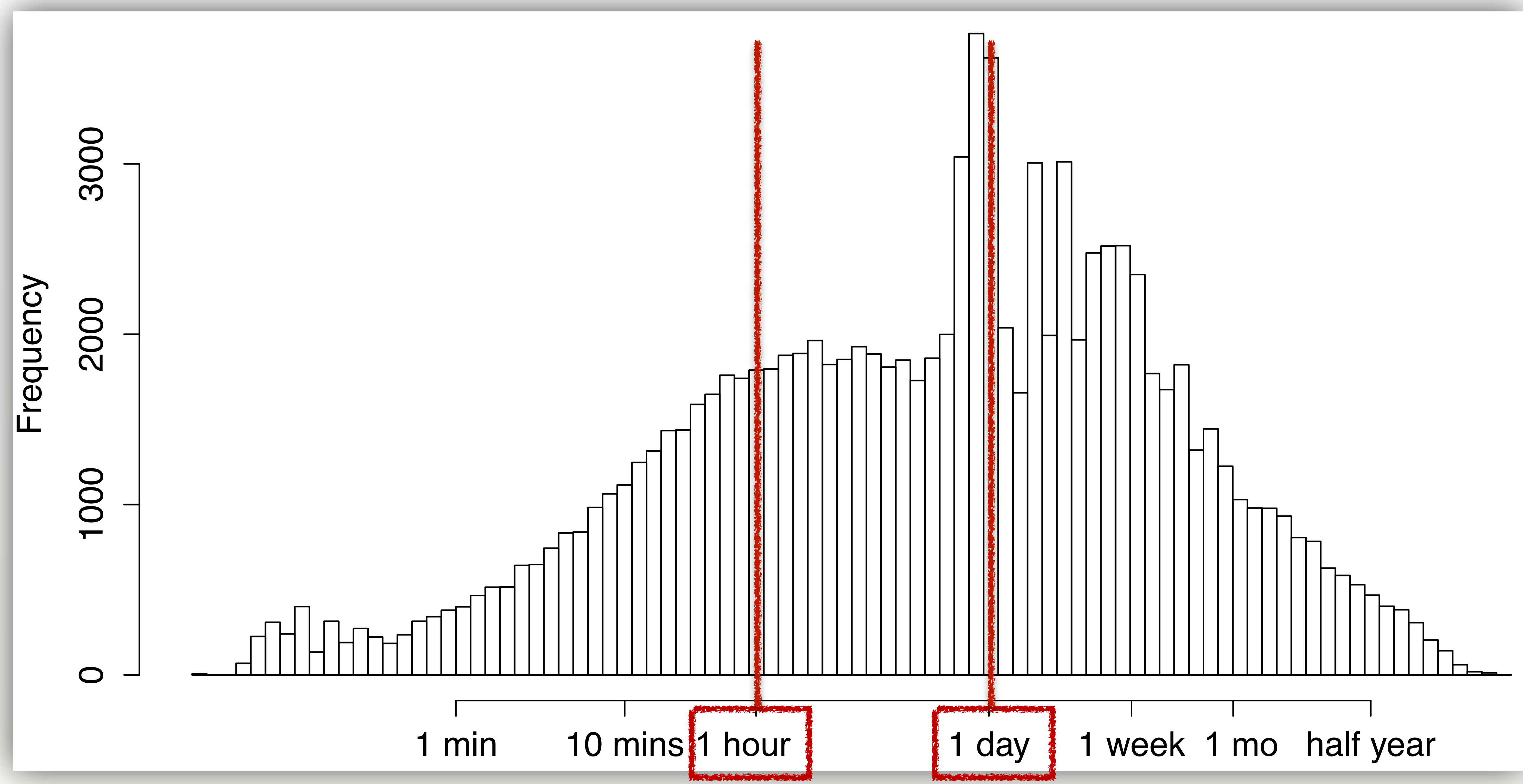


- Wait for it: Determinants of pull request evaluation latency on GitHub
  Y Yu, H Wang, V Filkov, P Devanbu, B Vasilescu. *MSR 2015*

PULL REQUEST EVALUATION TIME



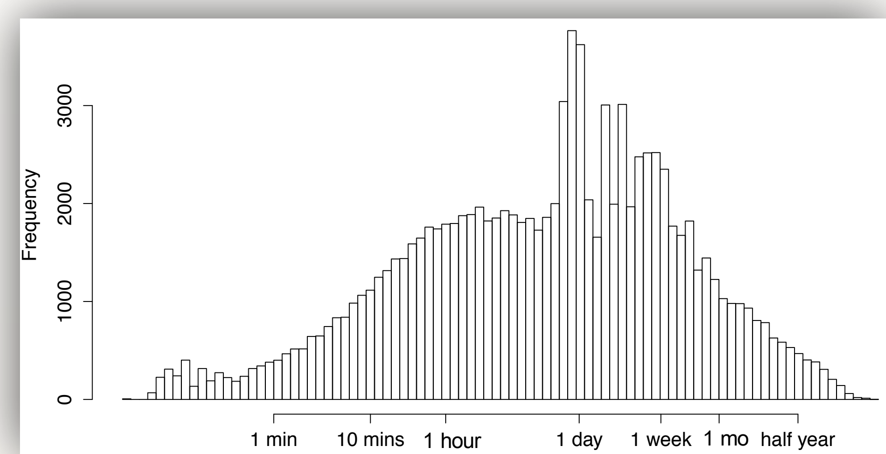
PULL REQUEST EVALUATION TIME

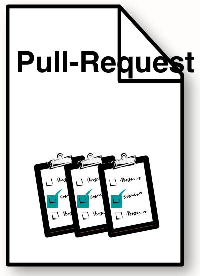


Hypothesis:

Technical attributes dominate: Size, Complexity, Having Tests

PULL REQUEST EVALUATION TIME



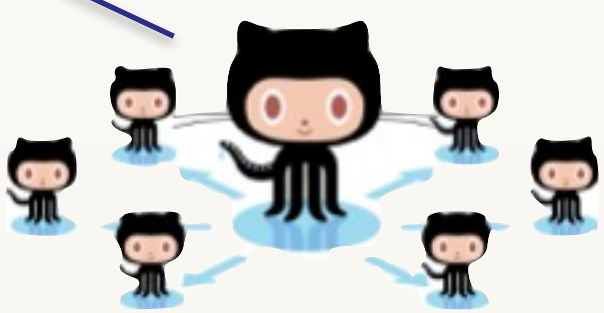


Size

- n_additions
- n_commits

Review

- n_comments



Experience & Social Connections

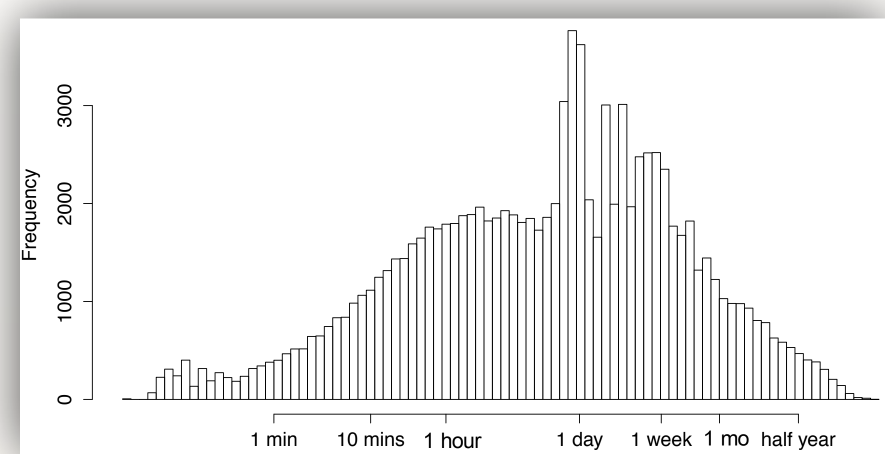
- merge_rate
- connection_strength
- n_followers

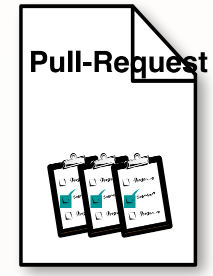
[Gousios et al, ICSE'14, ICSE'15]
[Tsay et al, ICSE'14, FSE'14]

PULL REQUEST EVALUATION TIME

M1: Previously-identified factors

✓ $R^2 = 36.2\%$



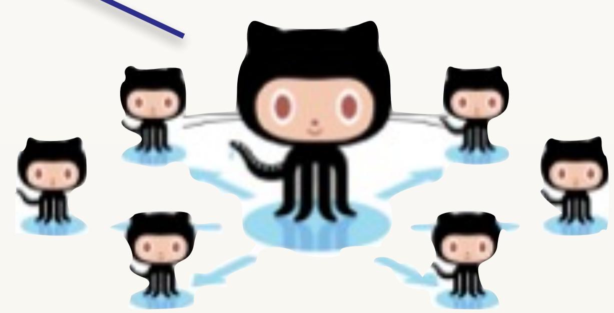


Size

- n_additions
- n_commits

Review

- n_comments



Experience & Social Connections

- merge_rate
- connection_strength
- n_followers

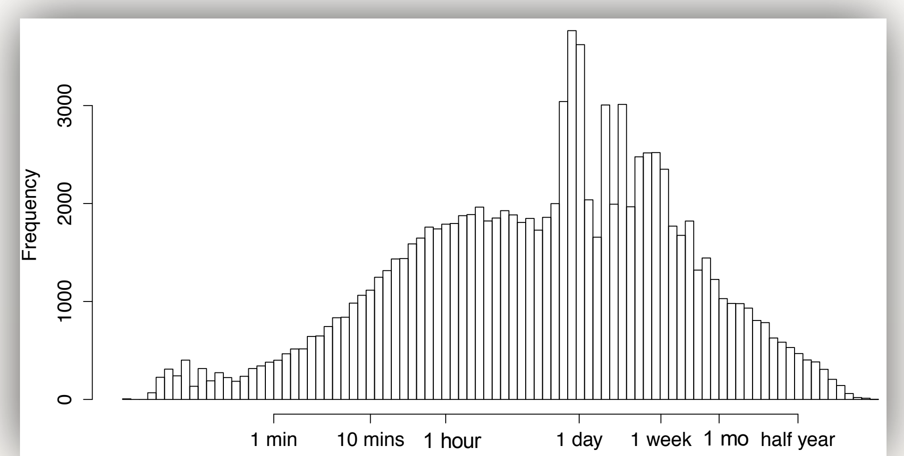
[Gousios et al, ICSE'14, ICSE'15]
[Tsay et al, ICSE'14, FSE'14]

PULL REQUEST EVALUATION TIME

M2: M1 + process-related factors + continuous integration

Title & description

- n_tokens

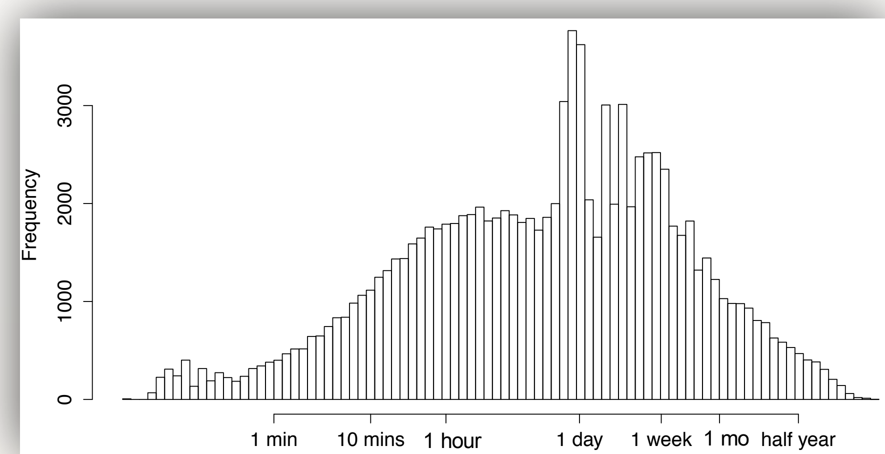


PULL REQUEST EVALUATION TIME

M2: M1 + process-related factors + continuous integration

Title & description

- n_tokens



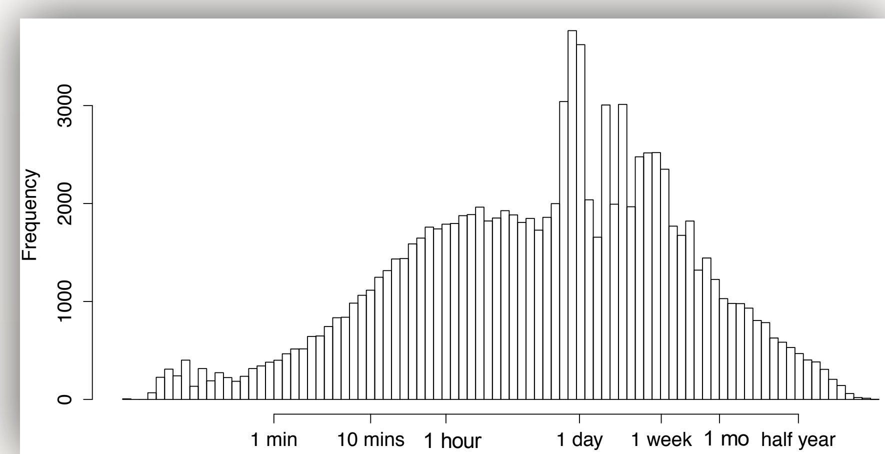
Management

- workload
- availability

[MSR 2015]

PULL REQUEST EVALUATION TIME

M2: M1 + process-related factors + continuous integration



Title & description

- n_tokens

Priority

- time_to_first_response

Continuous Integration

- response time

Management

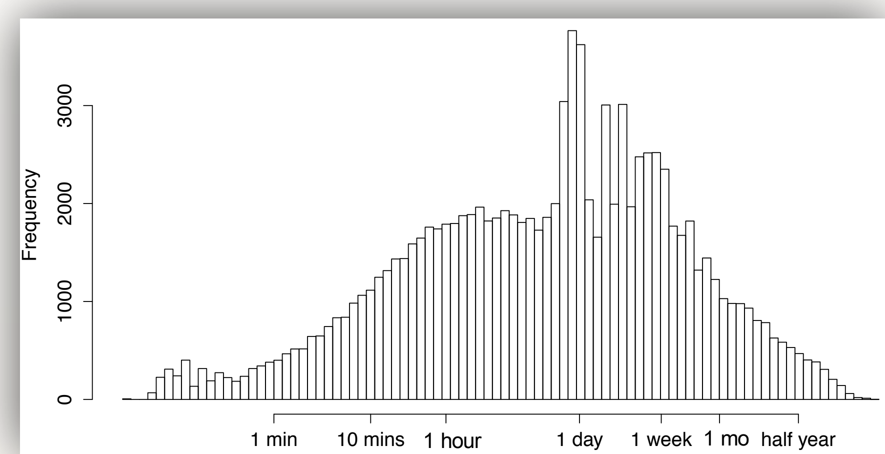
- workload
- availability

[MSR 2015]

PULL REQUEST EVALUATION TIME

M2: M1 + process-related factors + continuous integration

✓ $R^2 = 58.7\%$



Title & description
- n_tokens

Priority
- time_to_first_response

ravis

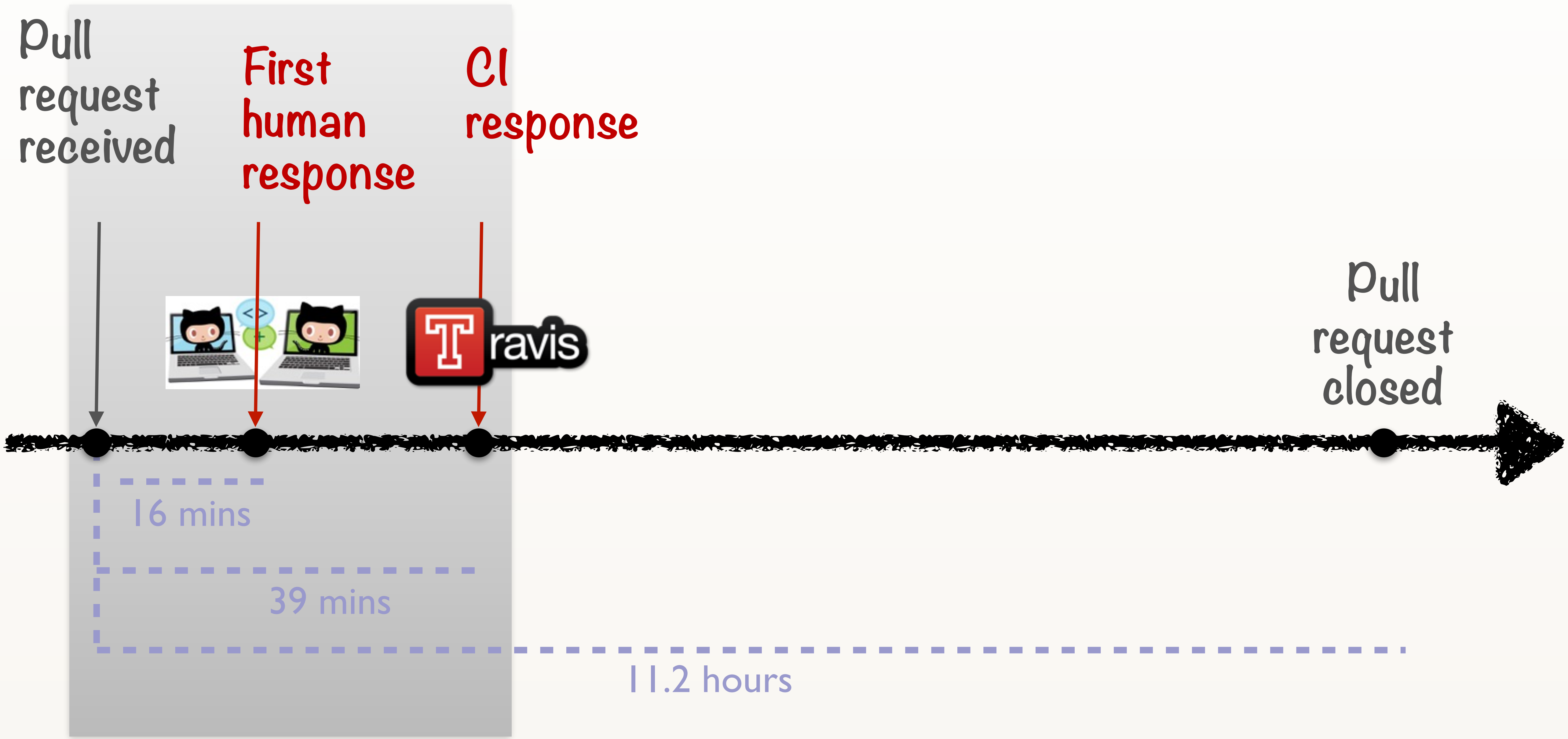
Continuous Integration
- response time

Management
- workload
- availability

Social tagging
- @mention
- #issue

[MSR 2015]

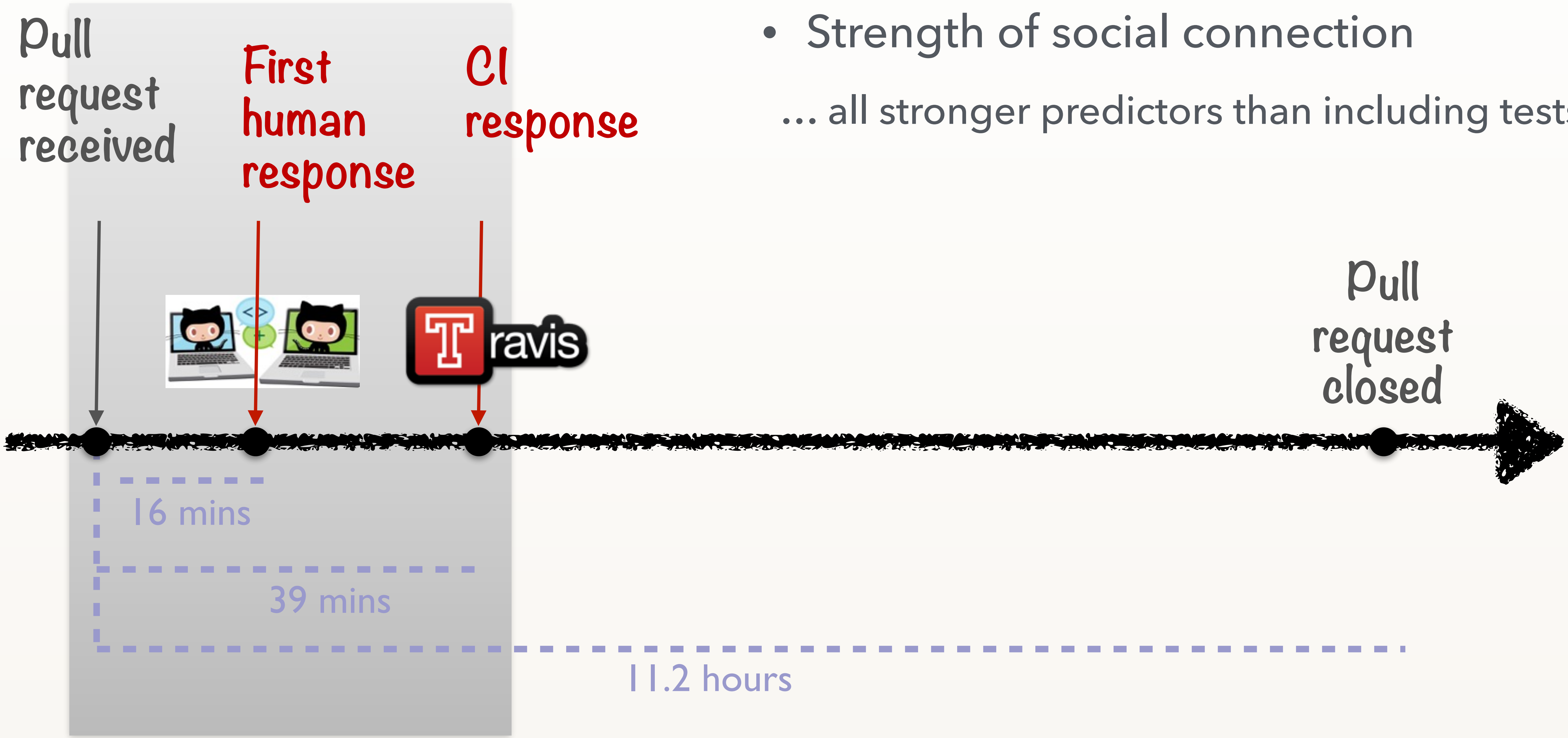
PULL REQUEST EVALUATION TIME IS PREDICTABLE



PULL REQUEST EVALUATION TIME

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

… all stronger predictors than including tests



Science is hard to get right

Sobel, A. E. K., & Clarkson, M. R. (2002). Formal methods application: An empirical tale of software development. *IEEE Transactions on Software Engineering*, *28*(3), 308-320.

- 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)

Sobel, A. E. K., & Clarkson, M. R. (2002). Formal methods application: An empirical tale of software development. *IEEE Transactions on Software Engineering*, *28*(3), 308-320.

- 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"

Berry, D. M., & Tichy, W. F. (2003). Comments on" Formal methods application: an empirical tale of software development". *IEEE Transactions on Software Engineering*, *29*(6), 567-571.

- "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."

Berry, D. M., & Tichy, W. F. (2003). Comments on" Formal methods application: an empirical tale of software development". *IEEE Transactions on Software Engineering*, *29*(6), 567-571.

- 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
- …

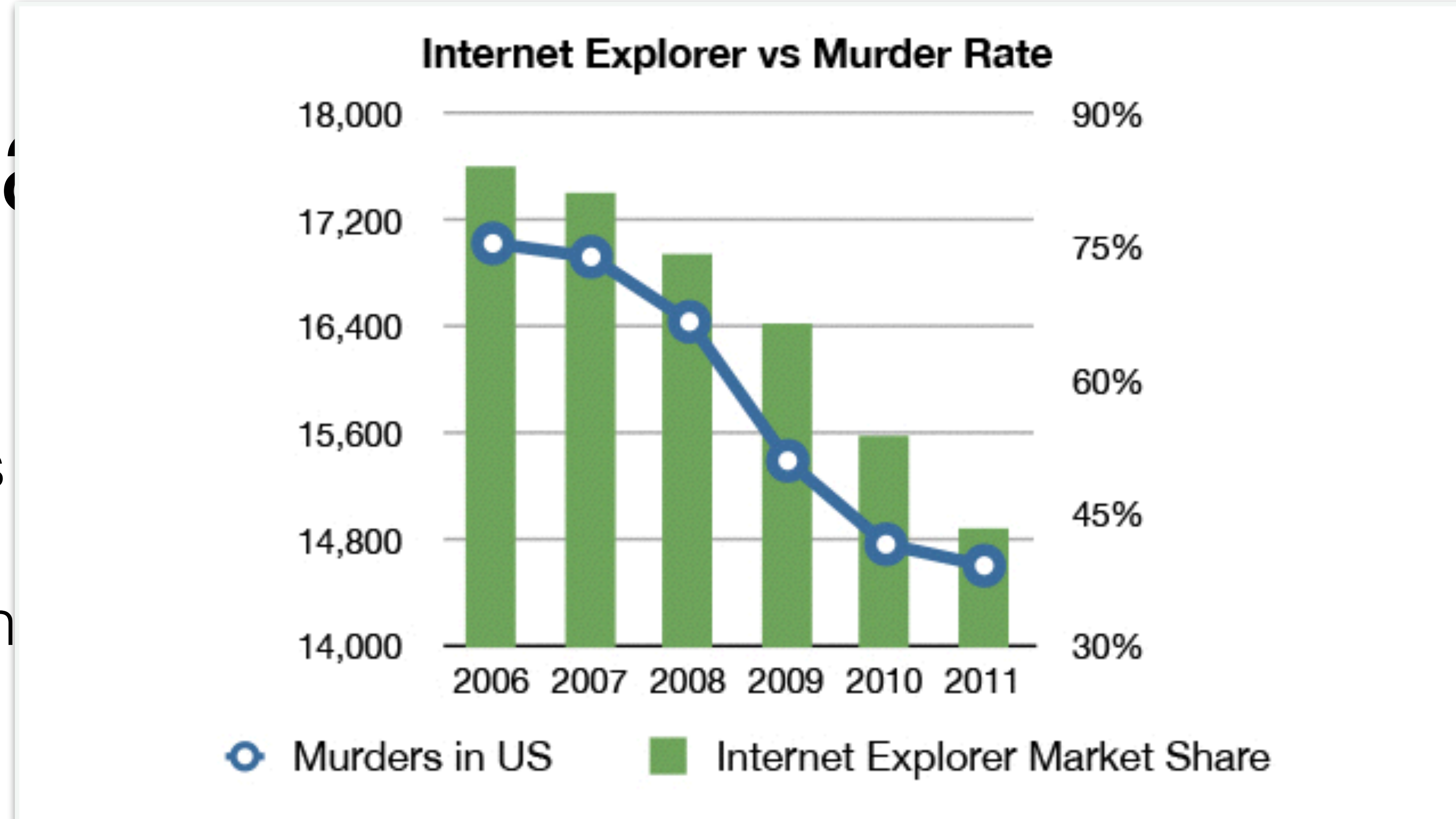
Why big data needs thick data

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

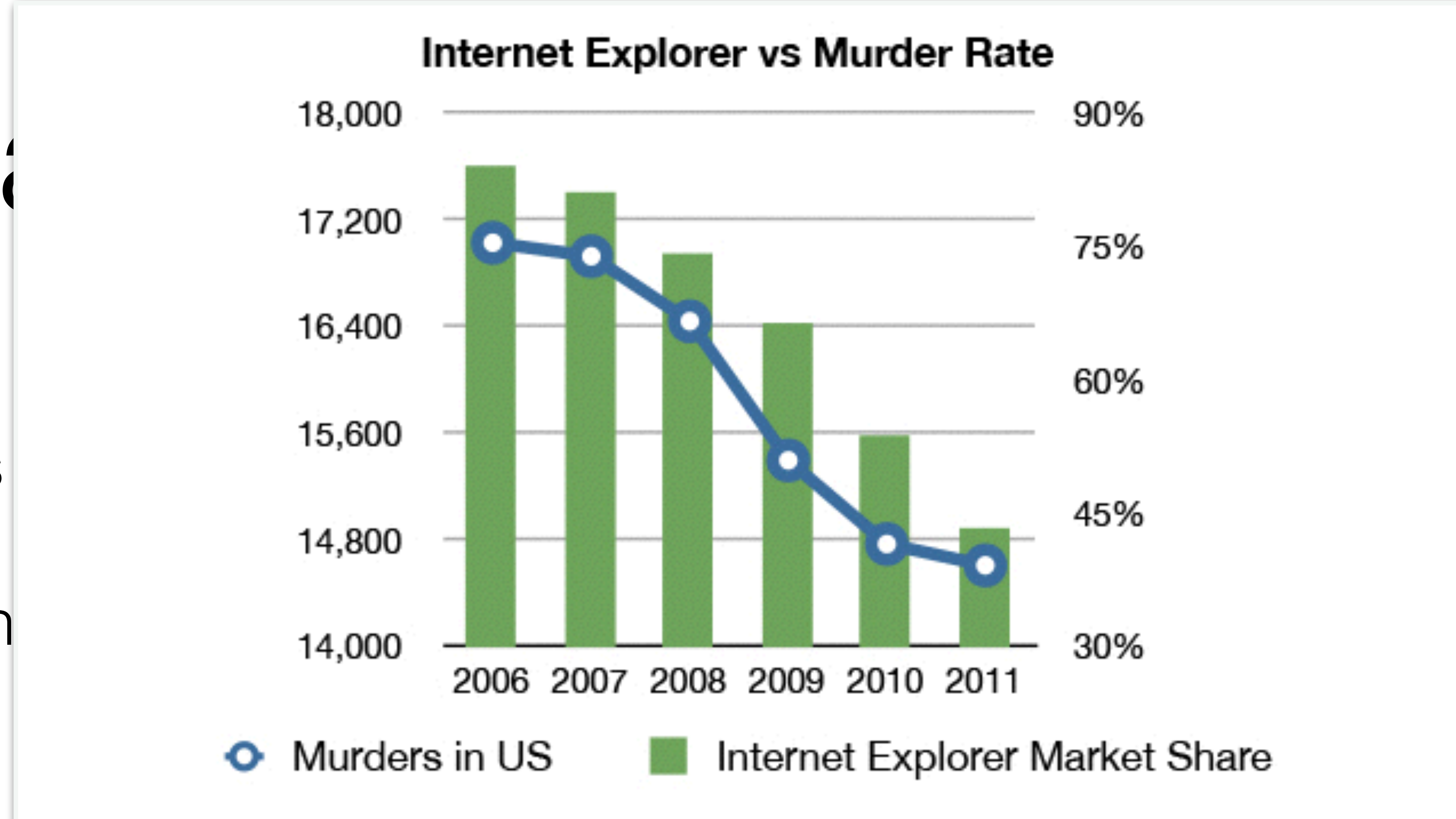
Why big da



- Looking for answers
    - A/B testing doesn 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 da

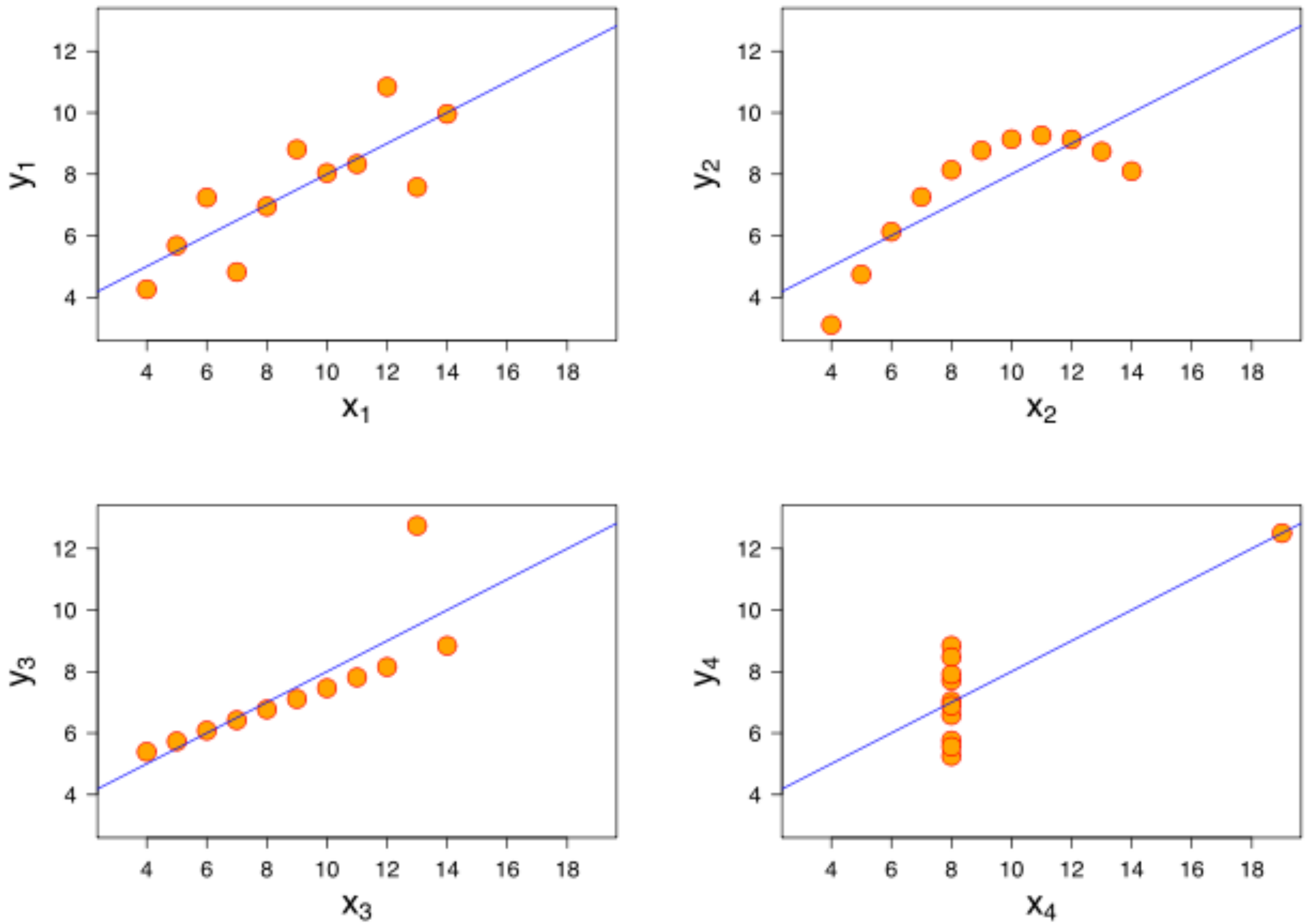
- Looking for answers
  - A/B testing doesn
    certain feature



- 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”

Carnegie Mellon University

Anscombe's quartet



stitute for
OFTWARE
ESEARCH

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?

Microsoft's 10 Most Unwise Questions

	Unwise
Which individual measures correlate with employee productivity (e.g. employee age, tenure, engineering skills, education, promotion velocity, IQ)?	25.5%
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)?	22.0%
What metrics can use used to compare employees?	21.3%
How can we measure the productivity of a Microsoft employee?	20.9%
Is the number of bugs a good measure of developer effectiveness?	17.2%
Can I generate 100% test coverage?	14.4%
Who should be in charge of creating and maintaining a consistent company-wide software process and tool chain?	12.3%
What are the benefits of a consistent, company-wide software process and tool chain?	10.4%
When are code comments worth the effort to write them?	9.6%
How much time and money does it cost to add customer input into your design?	8.3%

Percentage of women in top 100 Google image search results for CEO: 11%
Percentage of U.S. CEOs who are women: 27%

Percentage of women in the top 100 Google image search results for telemarketers: 64%
Percentage of U.S. telemarketers who are women: 50%

Kay, M., Matuszek, C., & Munson, S. A. (2015, April). Unequal representation and gender stereotypes in image search results for occupations. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (pp. 3819-3828). ACM.

Turkish - detected	English
o bir aşçı	she is a cook
o bir mühendis	he is an engineer
o bir doktor	he is a doctor
o bir hemşire	she is a nurse
o bir temizlikçi	he is a cleaner
o bir polis	He-she is a police
o bir asker	he is a soldier
o bir öğretmen	She's a teacher
o bir sekreter	he is a secretary
o bir arkadaş	he is a friend
o bir sevgili	she is a lover
onu sevmiyor	she does not like her
onu seviyor	she loves him
onu görüyor	she sees it
onu göremiyor	he can not see him
o onu kucaklıyor	she is embracing her
o onu kucaklamıyor	he does not embrace it
o evli	she is married
o bekar	he is single
o mutlu	he's happy
o mutsuz	she is unhappy
o çalışkan	he is hard working
o tembel	she is lazy