Meta

DEVELOPER PRODUCTIVITY
ENGINEERING SUMMIT 2023

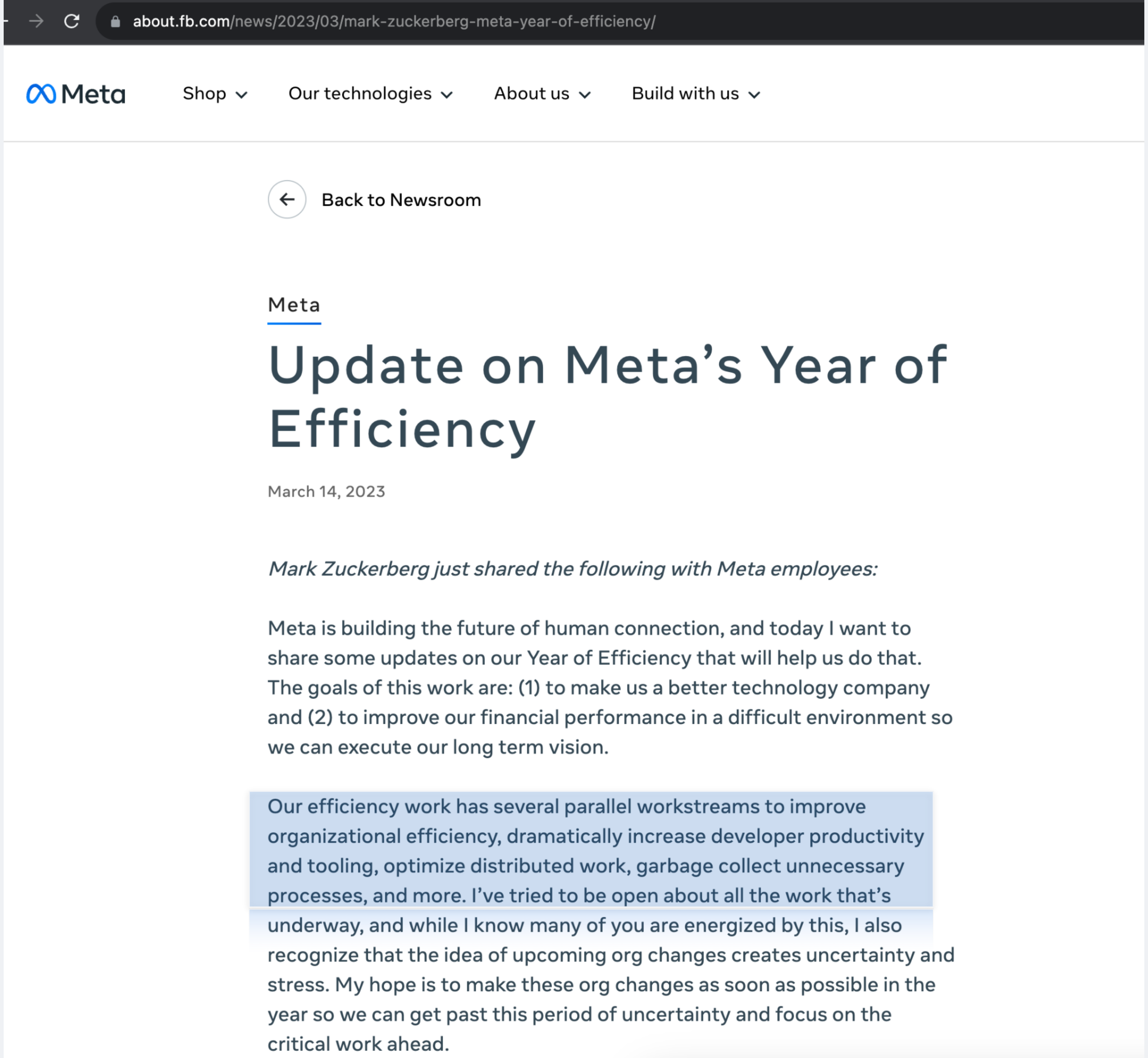
São Francisco I September 20-21
https://dpesummit.com/

Moving Fast While Delivering High Quality Code

Rui Abreu
Staff Software Engineer
𝕏@rmaranhao

Meta

One *(out of many)* trigger(s)


about.fb.com/news/2023/03/mark-zuckerberg-meta-year-of-efficiency/
Meta
Shop
Our technologies
About us
Build with us
Back to Newsroom

Meta

Update on Meta's Year of Efficiency

March 14, 2023

Mark Zuckerberg just shared the following with Meta employees:

Meta is building the future of human connection, and today I want to share some updates on our Year of Efficiency that will help us do that. The goals of this work are: (1) to make us a better technology company and (2) to improve our financial performance in a difficult environment so we can execute our long term vision.

Our efficiency work has several parallel workstreams to improve organizational efficiency, dramatically increase developer productivity and tooling, optimize distributed work, garbage collect unnecessary processes, and more. I've tried to be open about all the work that's underway, and while I know many of you are energized by this, I also recognize that the idea of upcoming org changes creates uncertainty and stress. My hope is to make these org changes as soon as possible in the year so we can get past this period of uncertainty and focus on the critical work ahead.

MOVING FAST? HIGH QUALITY? PRODUCTIVITY?

Agenda

1. Measuring Productivity

2. Drivers (developer POV)
   a. Authoring Velocity
   b. Reliability (incl. Quality)
   c. Knowledge (incl. Readability)

3. What's Next?

4. Q&A

HOW DO WE MEASURE PRODUCTIVITY?

HOW DO WE MEASURE PRODUCTIVITY?

Satisfaction

Number of code reviews

Code review time

Number of Meetings

Number of bug fixes

Burnout

Code Quality

Words per minute

Number of wikis published

Deployment Time

Number of commits

Service uptime

Coding Time

Time to fix bugs

Amount of code written

API Latency

Number of sev resolutions

Meeting Time

Test Runtime

Organization attrition

Time to resolve sevs

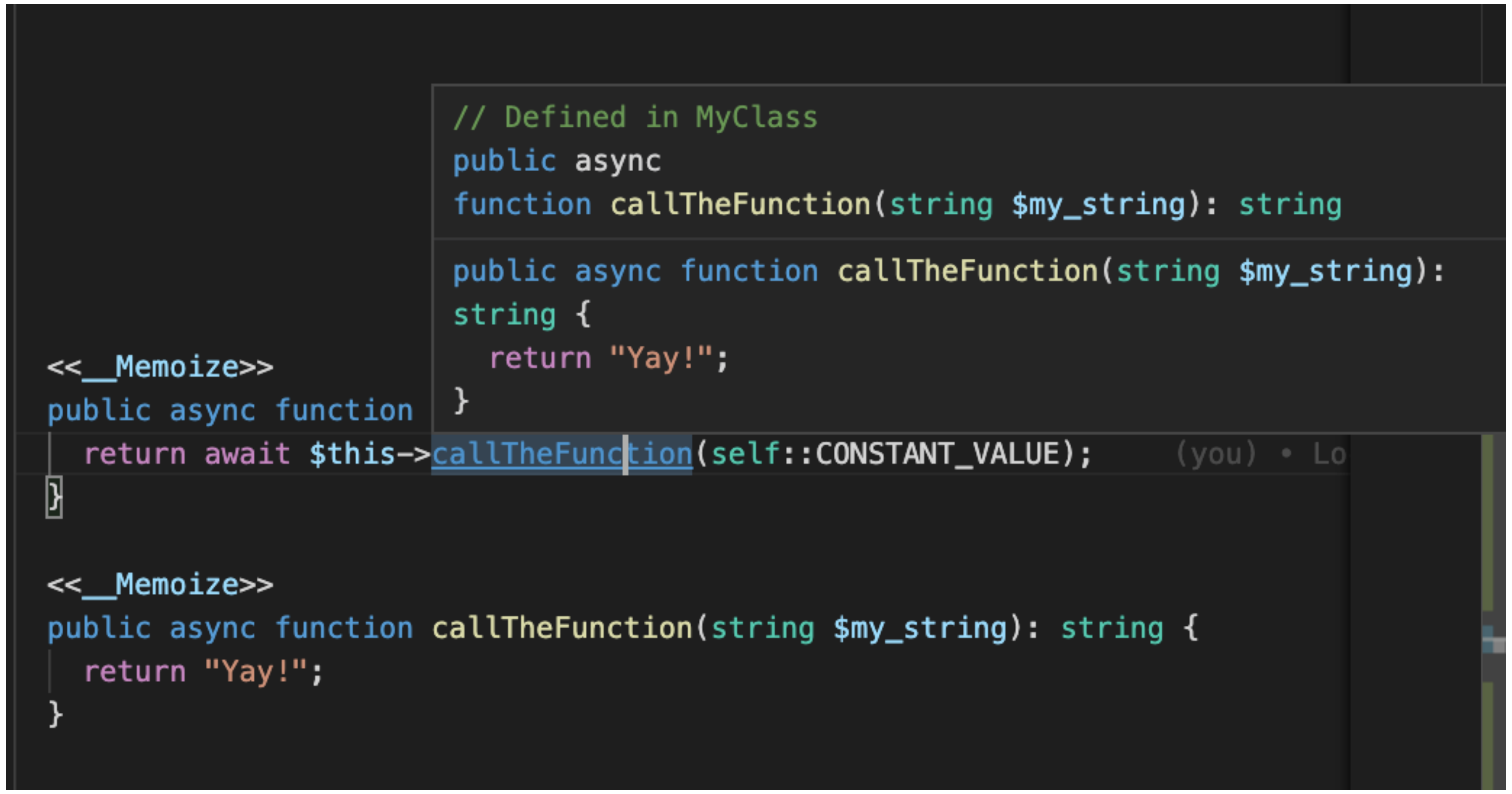
Build Runtime

Deployment frequency

Time to close ticket

Tickets closed

HOW DO WE MEASURE PRODUCTIVITY?

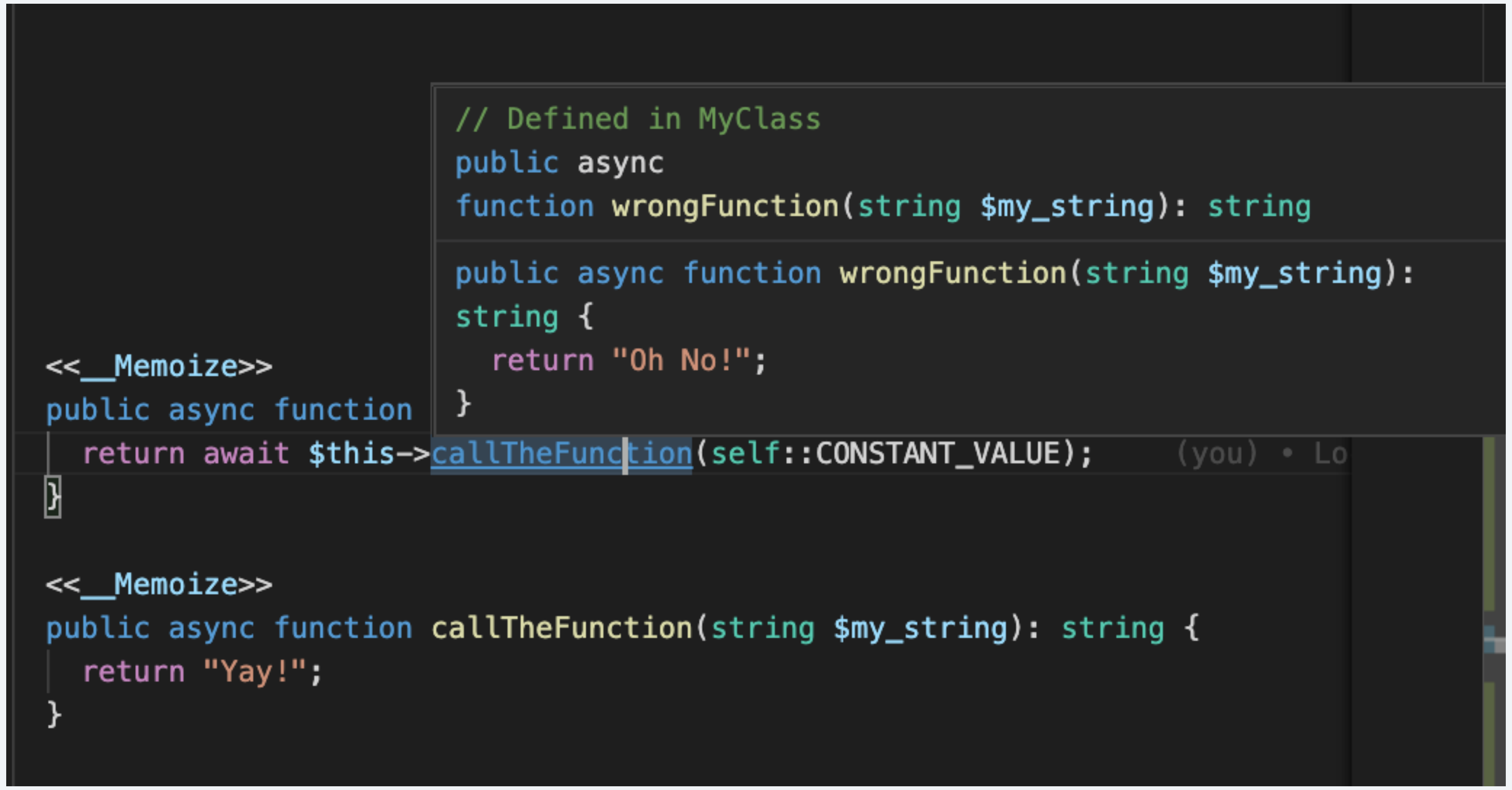


```
// Defined in MyClass
public async
function callTheFunction(string $my_string): string

public async function callTheFunction(string $my_string):
string {
  return "Yay!";
}
```

```
<<__Memoize>>
public async function
  return await $this->callTheFunction(self::CONSTANT_VALUE);    (you) • Lo
}

<<__Memoize>>
public async function callTheFunction(string $my_string): string {
  return "Yay!";
}
```



```
// Defined in MyClass
public async
function wrongFunction(string $my_string): string

public async function wrongFunction(string $my_string):
string {
  return "Oh No!";
}
```

```
<<__Memoize>>
public async function
  return await $this->callTheFunction(self::CONSTANT_VALUE);    (you) • Lo
}

<<__Memoize>>
public async function callTheFunction(string $my_string): string {
  return "Yay!";
}
```

𝕏@rmaranhao

PRODUCTIVITY FRAMEWORKS

DevOps Research and Assessment

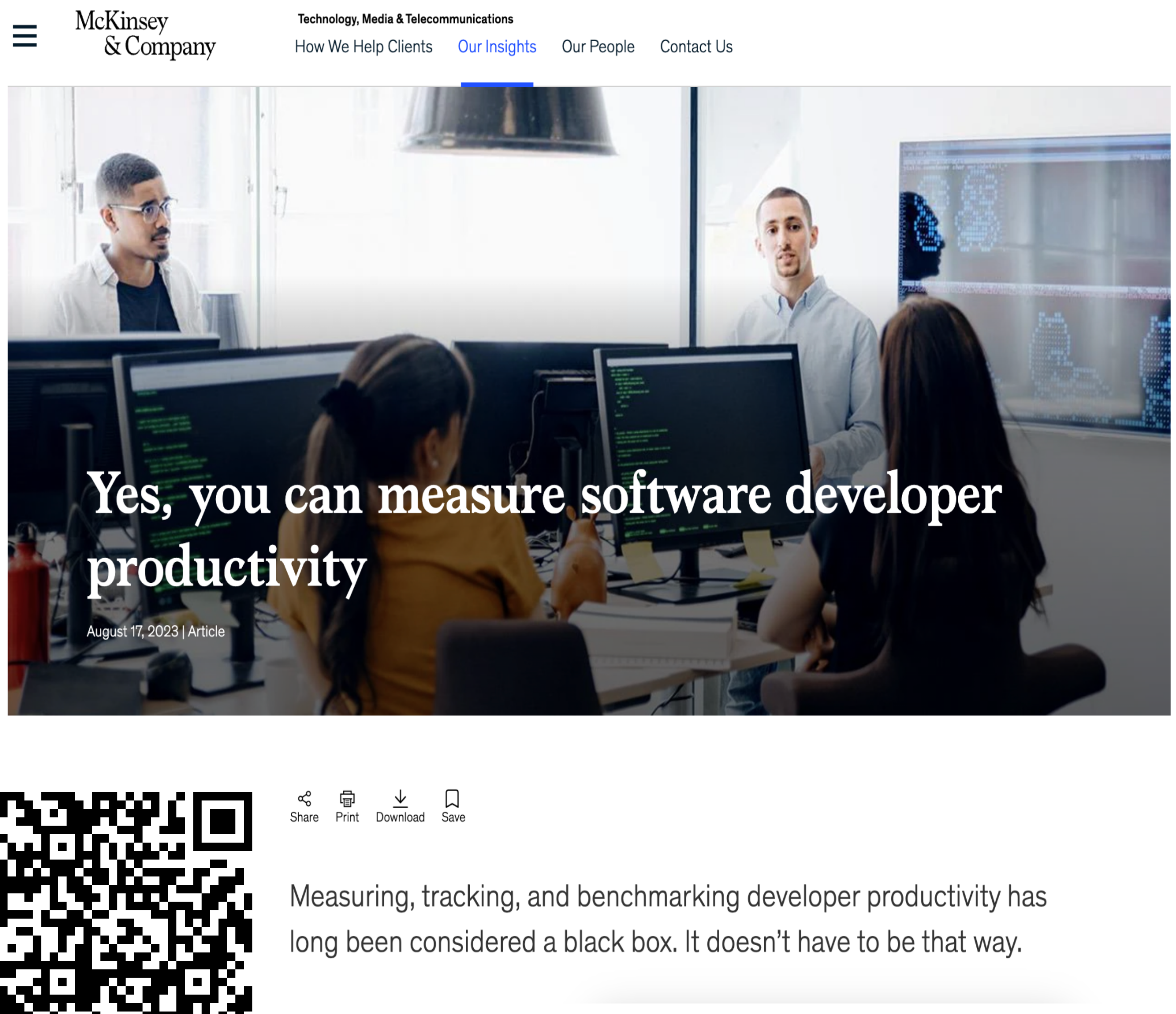
D	O	R	A
Deployment Frequency	Lead Time for Changes	Mean Time to Recovery (MTTR)	Change Failure Rate

PRODUCTIVITY FRAMEWORKS

S	P	A	C	E
Satisfaction and Well-Being	Performance	Activity	Communication and Collaboration	Efficiency and Flow

@rmaranhao

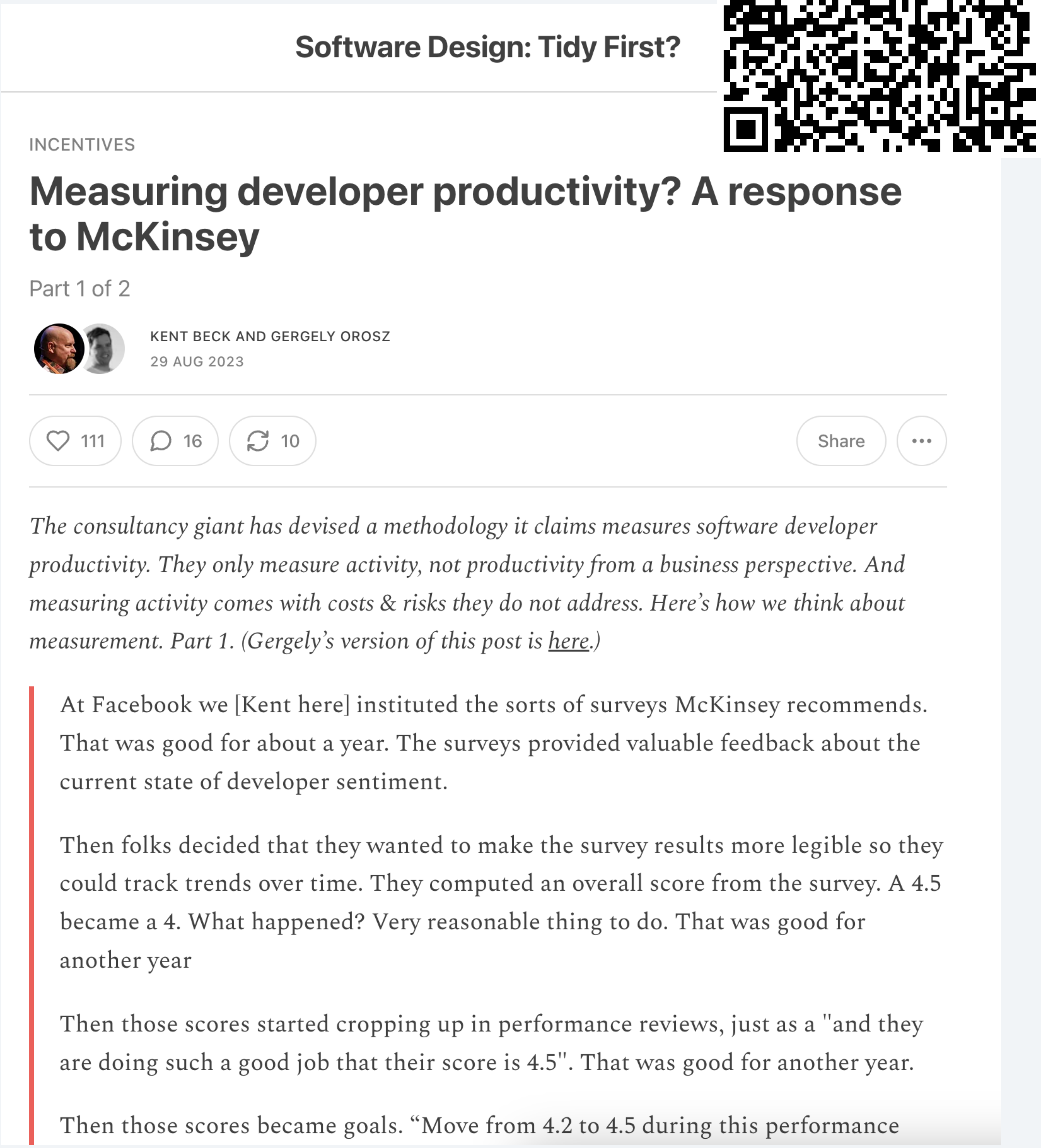
PRODUCTIVITY FRAMEWORKS



Yes, you can measure software developer productivity

August 17, 2023 | Article

Measuring, tracking, and benchmarking developer productivity has long been considered a black box. It doesn't have to be that way.



INCENTIVES

Measuring developer productivity? A response to McKinsey

Part 1 of 2

KENT BECK AND GERGELY OROSZ

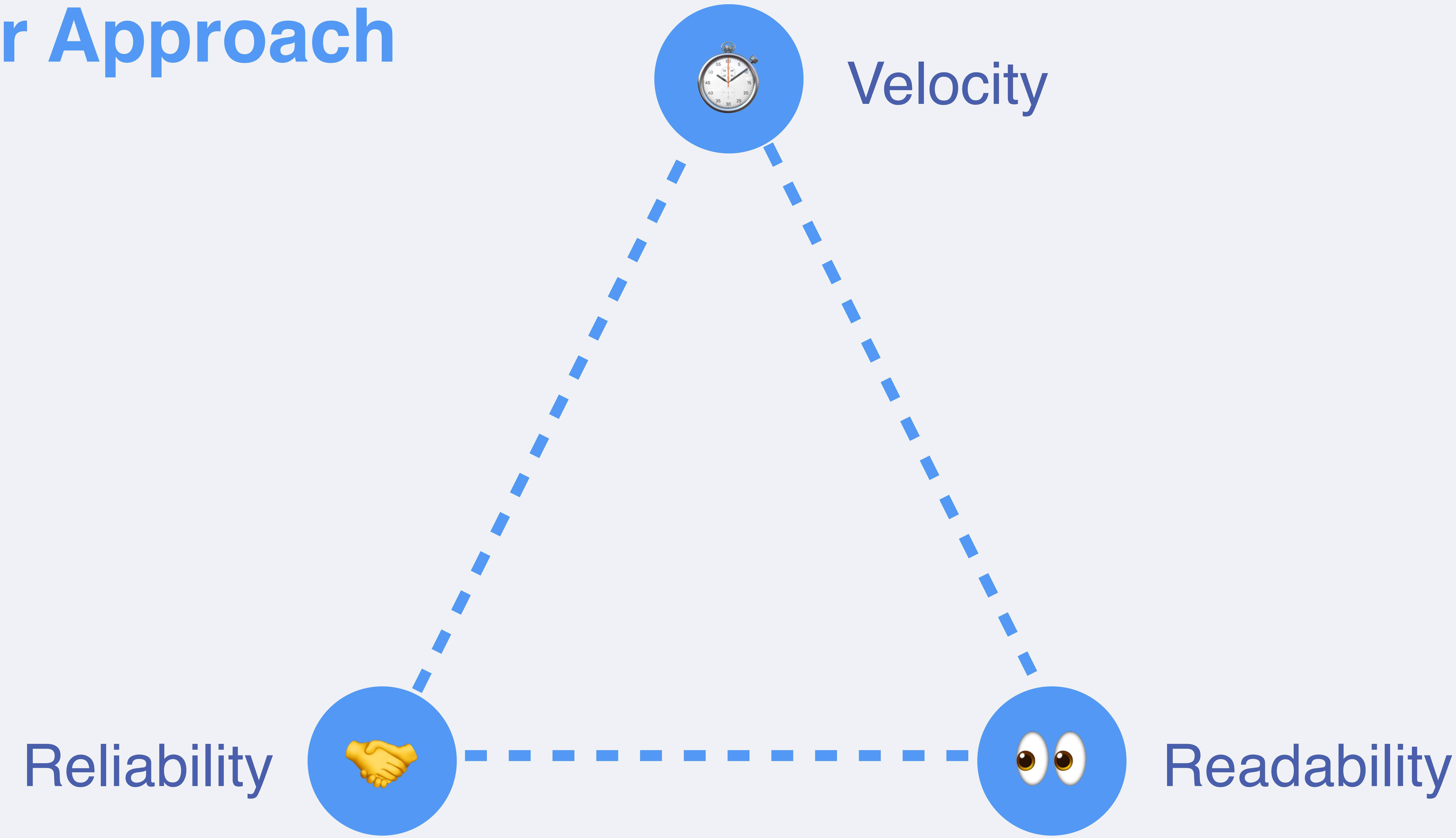
29 AUG 2023

The consultancy giant has devised a methodology it claims measures software developer productivity. They only measure activity, not productivity from a business perspective. And measuring activity comes with costs & risks they do not address. Here's how we think about measurement. Part 1. (Gergely's version of this post is here.)

> At Facebook we [Kent here] instituted the sorts of surveys McKinsey recommends. That was good for about a year. The surveys provided valuable feedback about the current state of developer sentiment.
>
> Then folks decided that they wanted to make the survey results more legible so they could track trends over time. They computed an overall score from the survey. A 4.5 became a 4. What happened? Very reasonable thing to do. That was good for another year
>
> Then those scores started cropping up in performance reviews, just as a "and they are doing such a good job that their score is 4.5". That was good for another year.
>
> Then those scores became goals. "Move from 4.2 to 4.5 during this performance

@rmaranhao

Our Approach



Drivers

1. Measuring Productivity

2. Drivers (developer POV)
    a. Authoring Velocity
    b. Reliability (incl. Quality)
    c. Knowledge (incl. Readability)

3. What’s Next?

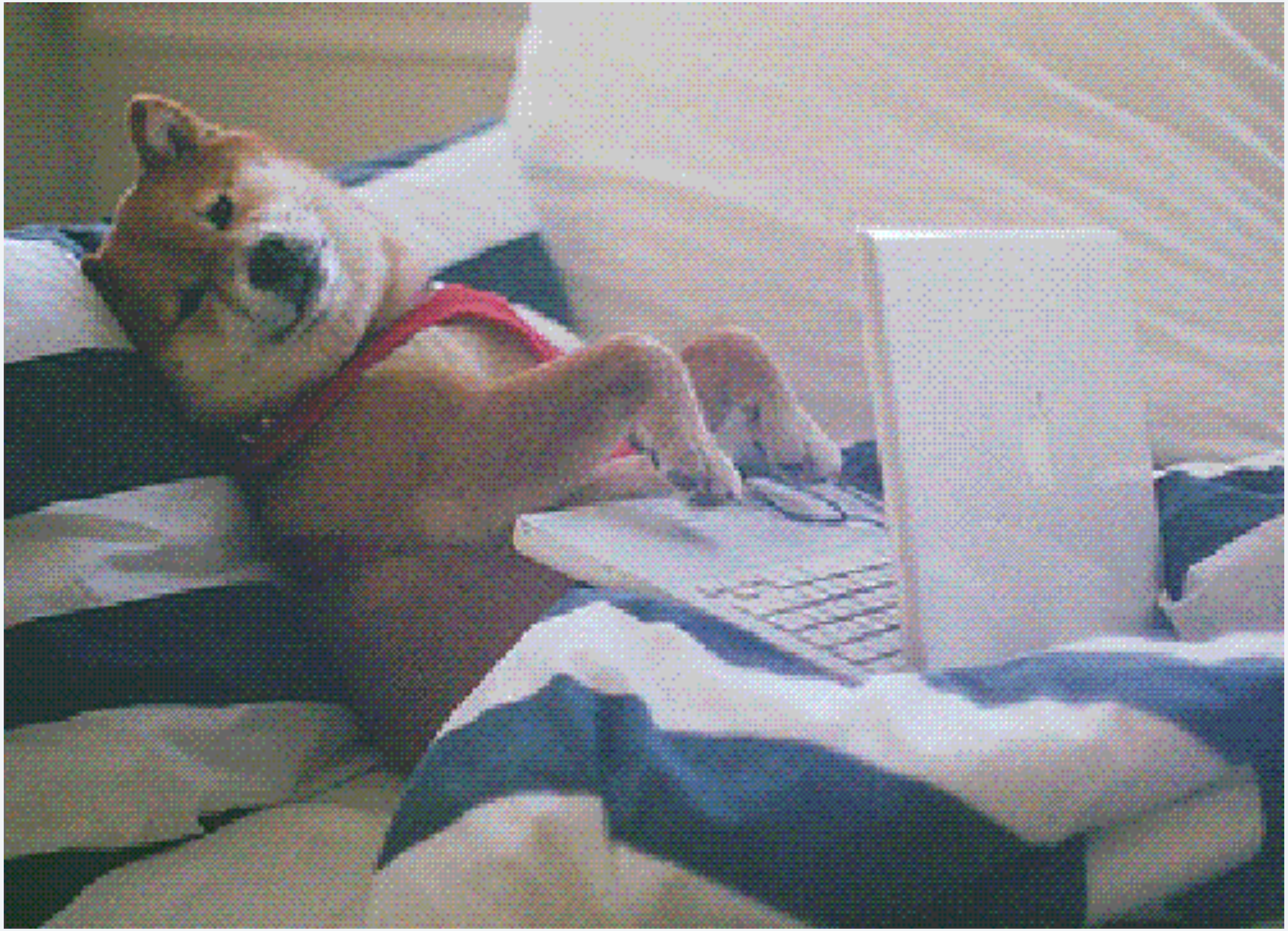
4. Q&A

Credits to DALL-E.

SWE’s CHORES

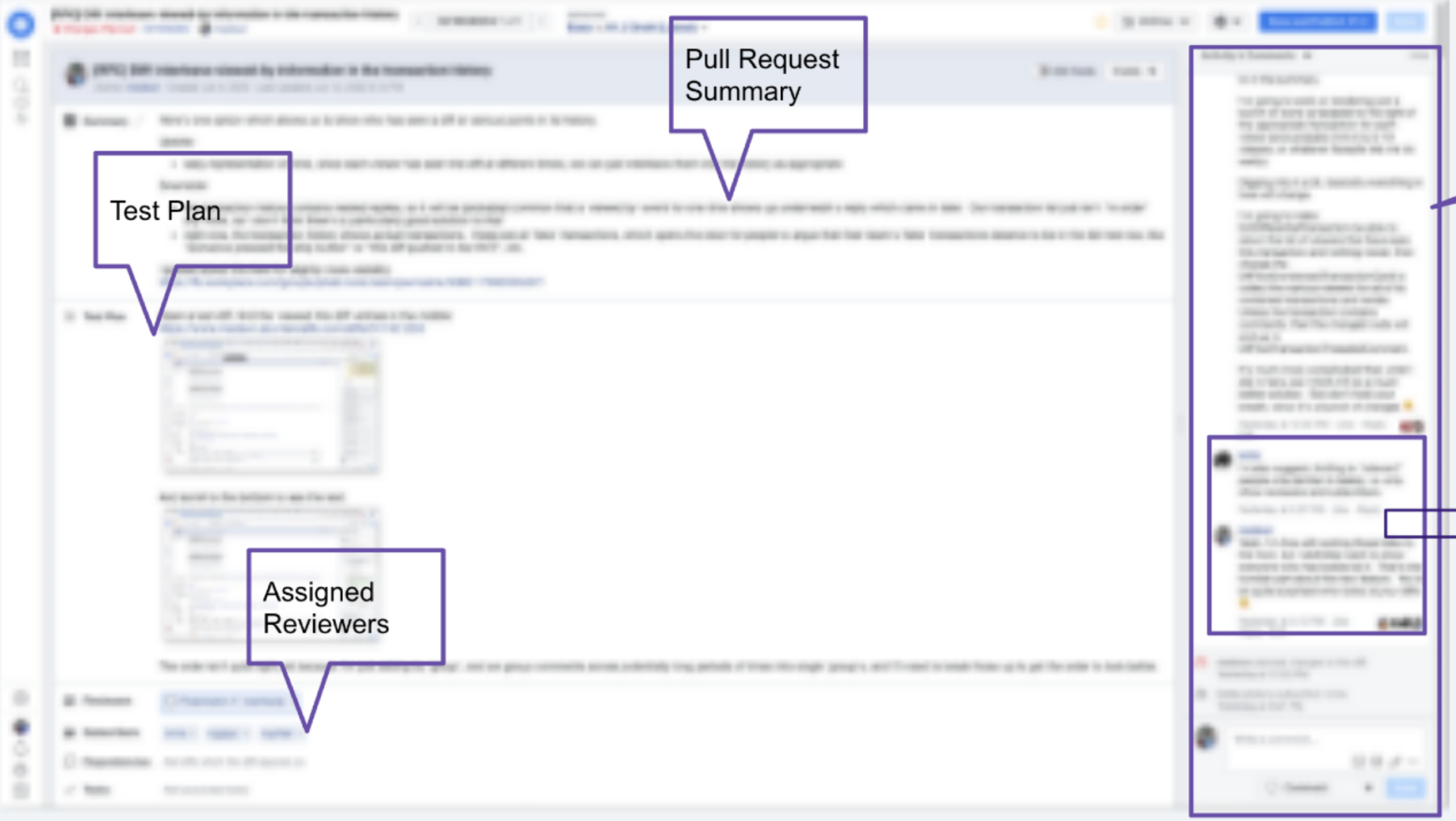
- Split mega-classes into component classes
- Fix architectural design flaws
- Remove unused variables
- Migrate to the latest frameworks
- Update your dependencies
- Upgrade tooling
- Write scripts to automate common work
- Improve documentation
- Fix bugs
- Fix papercuts
- …

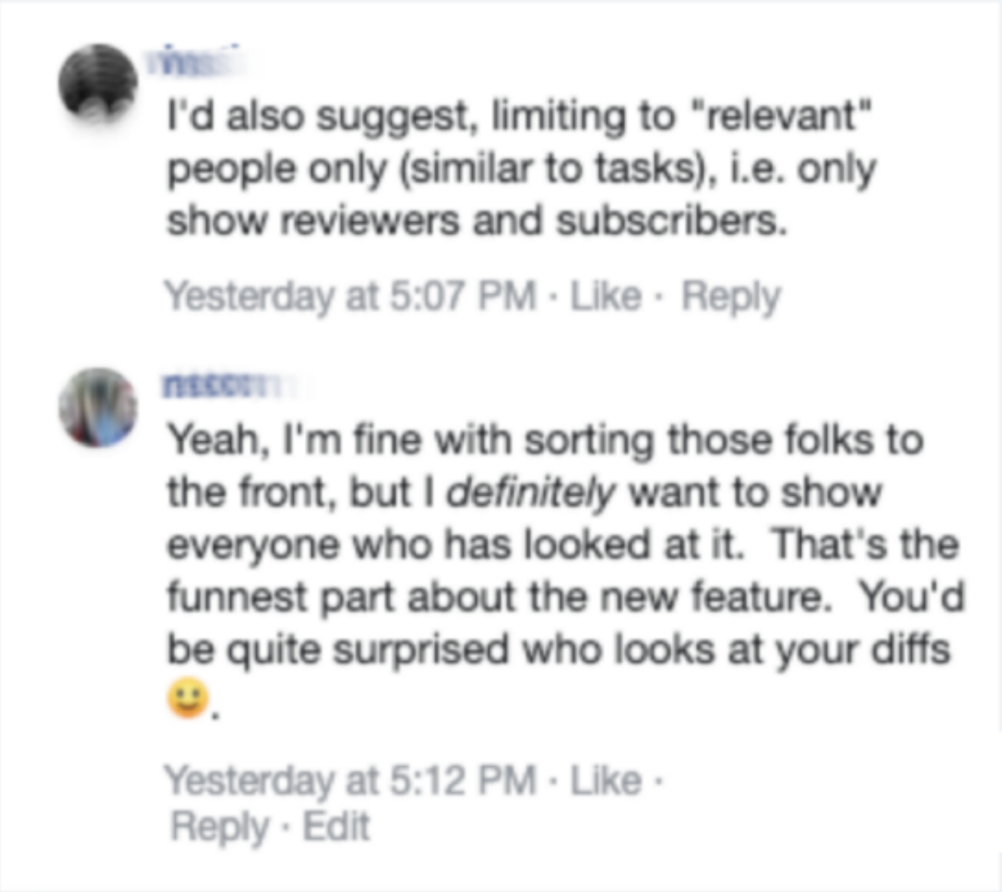
HOW DO WE MEASURE EASE OF AUTHORING CODE?



𝕏@rmaranhao

AUTHORING CODE’s WORKFLOW @ Meta





[redacted]

I'd also suggest, limiting to "relevant" people only (similar to tasks), i.e. only show reviewers and subscribers.

Yesterday at 5:07 PM · Like · Reply

[redacted]

Yeah, I'm fine with sorting those folks to the front, but I *definitely* want to show everyone who has looked at it. That's the funnest part about the new feature. You'd be quite surprised who looks at your diffs 🙂.

Yesterday at 5:12 PM · Like · Reply · Edit

Velocity

1. Measuring Productivity

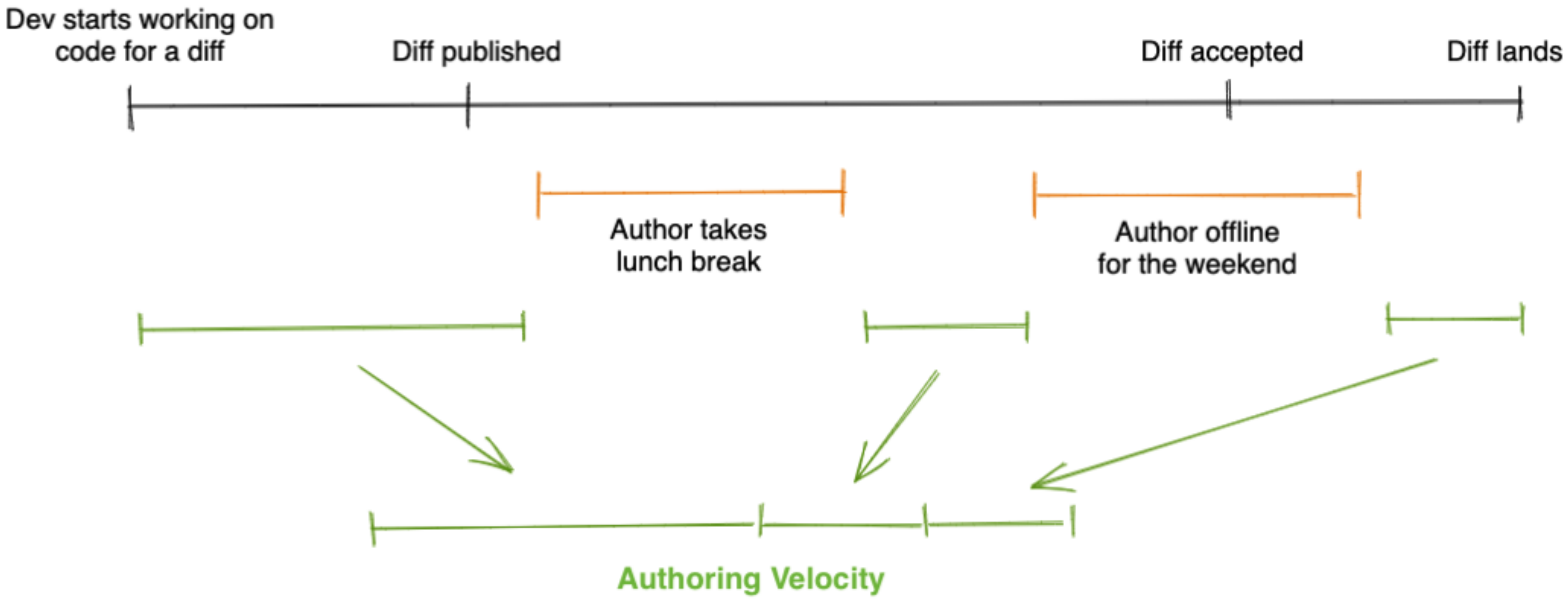
2. Drivers (developer POV)
    a. Authoring Velocity
    b. Reliability (incl. Quality)
    c. Knowledge (incl. Readability)

3. What’s Next?

4. Q&A

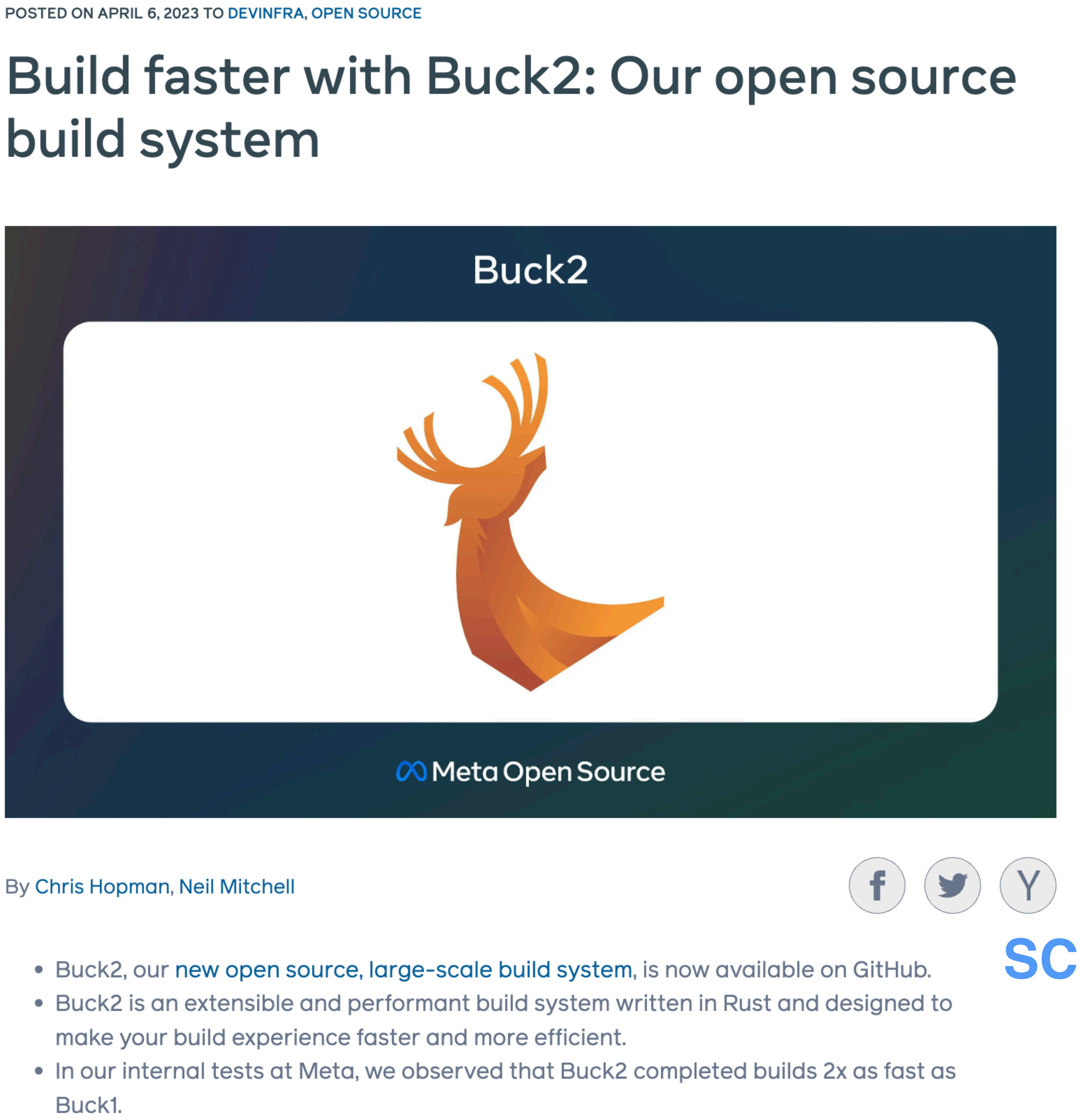
AUTHORING VELOCITY (AV)

How many working hours does it take to write and land a diff?



@rmaranhao

IMPACT OF TOOLING ON AUTHOR VELOCITY



POSTED ON APRIL 6, 2023 TO DEVINFRA, OPEN SOURCE

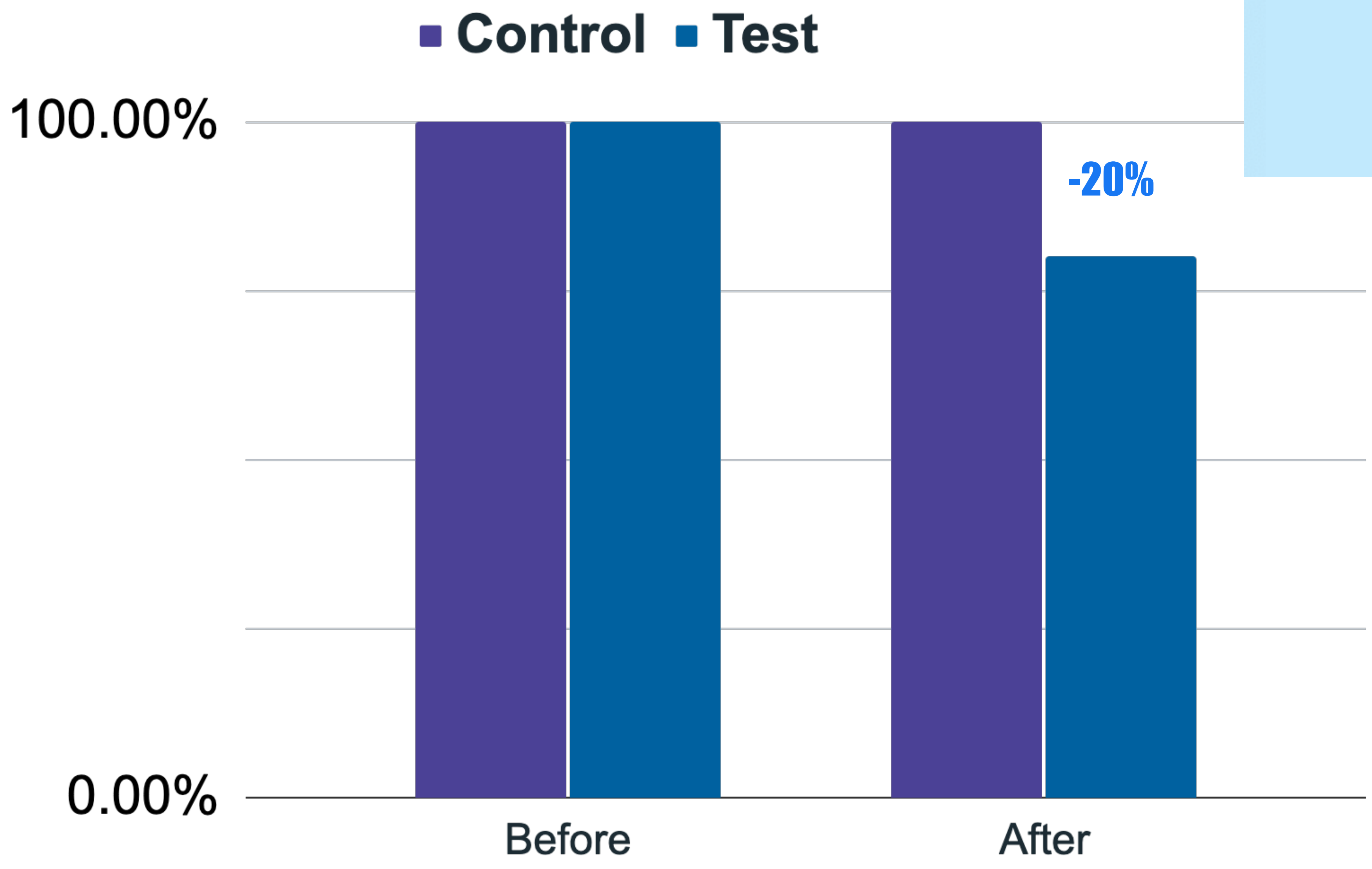
Build faster with Buck2: Our open source build system

By Chris Hopman, Neil Mitchell

- Buck2, our new open source, large-scale build system, is now available on GitHub.
- Buck2 is an extensible and performant build system written in Rust and designed to make your build experience faster and more efficient.
- In our internal tests at Meta, we observed that Buck2 completed builds 2x as fast as Buck1.

SCAN ME

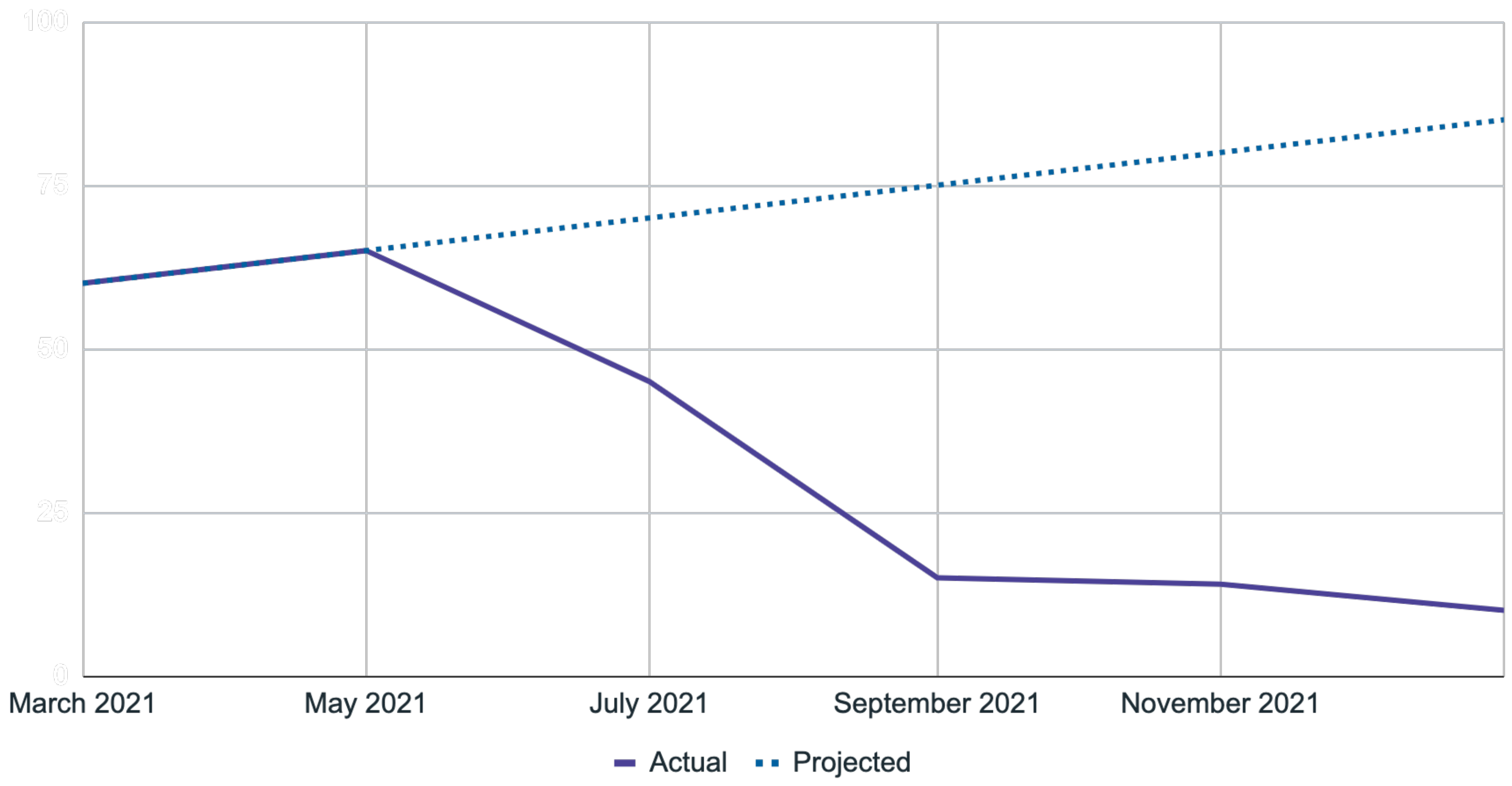
AUTHORING VELOCITY & WASABI



SCAN ME

AUTHORING VELOCITY & CODE COMPLEXITY

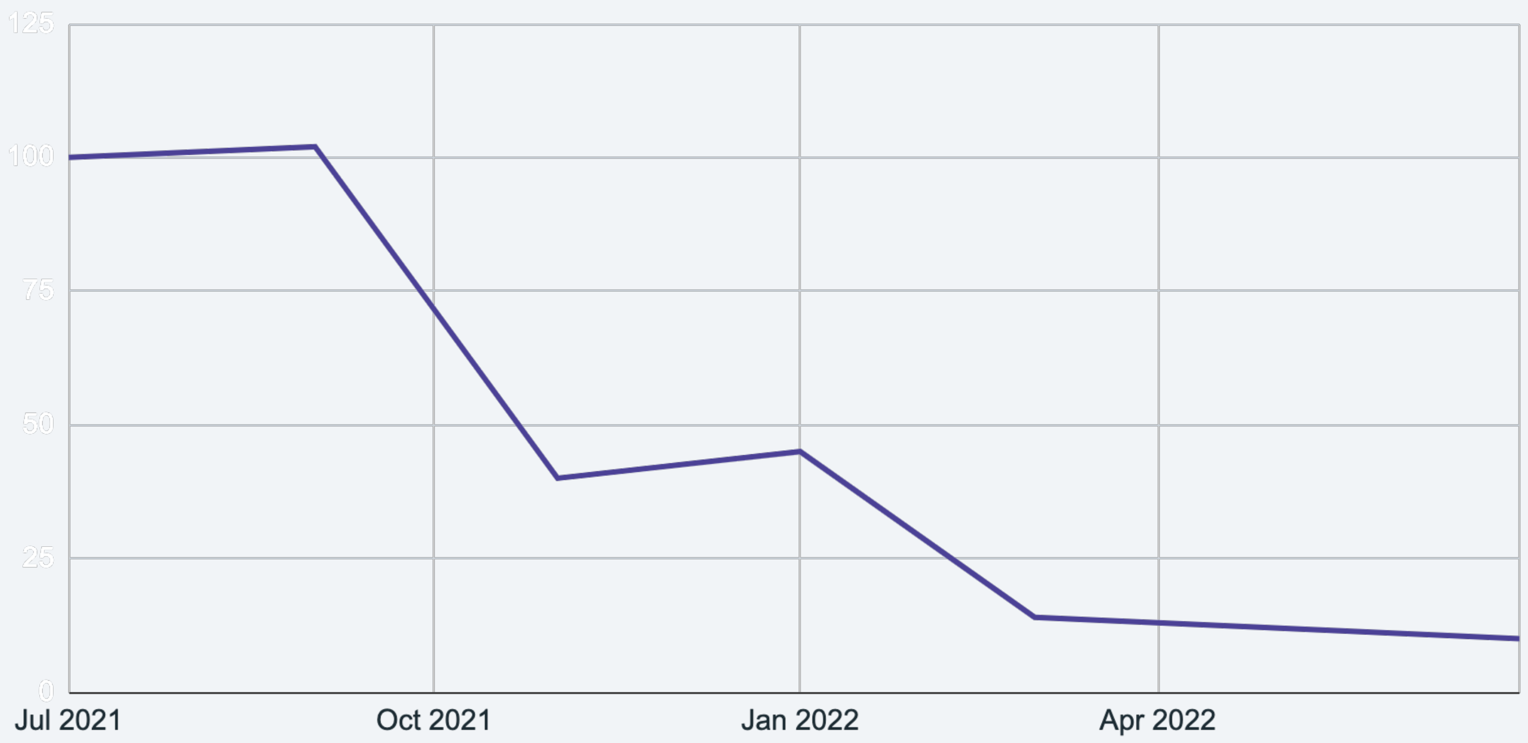
Code Complexity



60% reduction in complexity

42% reduction in Authoring Velocity

Author Velocity



Reliability

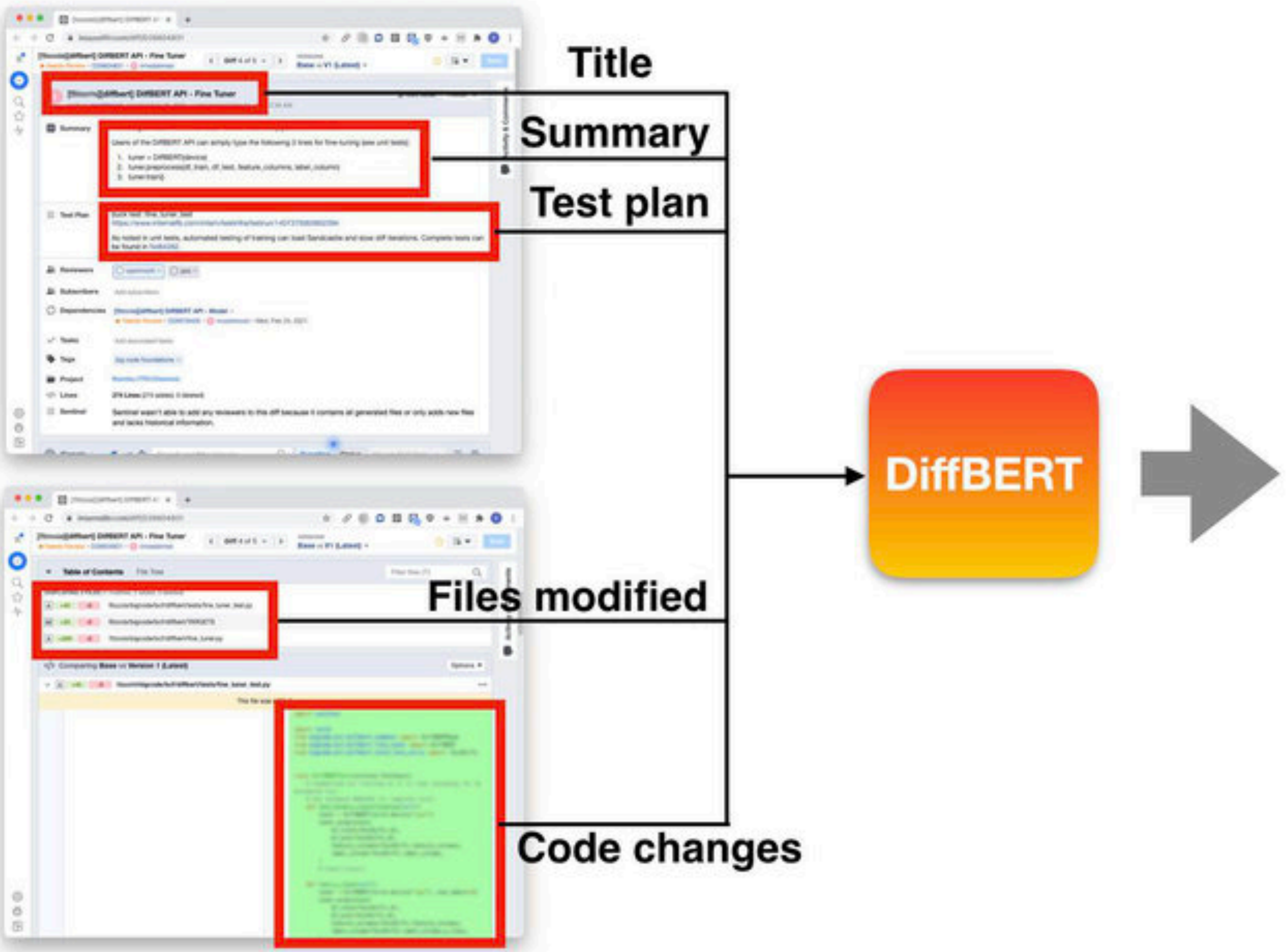
1. Measuring Productivity

2. Drivers (developer POV)
    a. Authoring Velocity
    b. Reliability (incl. Quality)
    c. Knowledge (incl. Readability)

3. What’s Next?

4. Q&A

DIFF INTENT WITH DIFFBERT/LLM



[illegible] DiffBERT API - Fine Tuner

Summary

Users of the DiffBERT API can simply type the following 3 lines for fine-tuning (see unit tests):

1. tuner = DiffBERT(device)
2. tuner.preprocess(df_train, df_test, feature_column, label_column)
3. tuner.train()

Test Plan

buck test [illegible] fine_tuner_test

[illegible]

As noted in unit tests, automated testing of training can load Sandcastle and slow diff iterations. Complete tests can be found in [illegible]

274 Lines (274 added, 0 deleted)

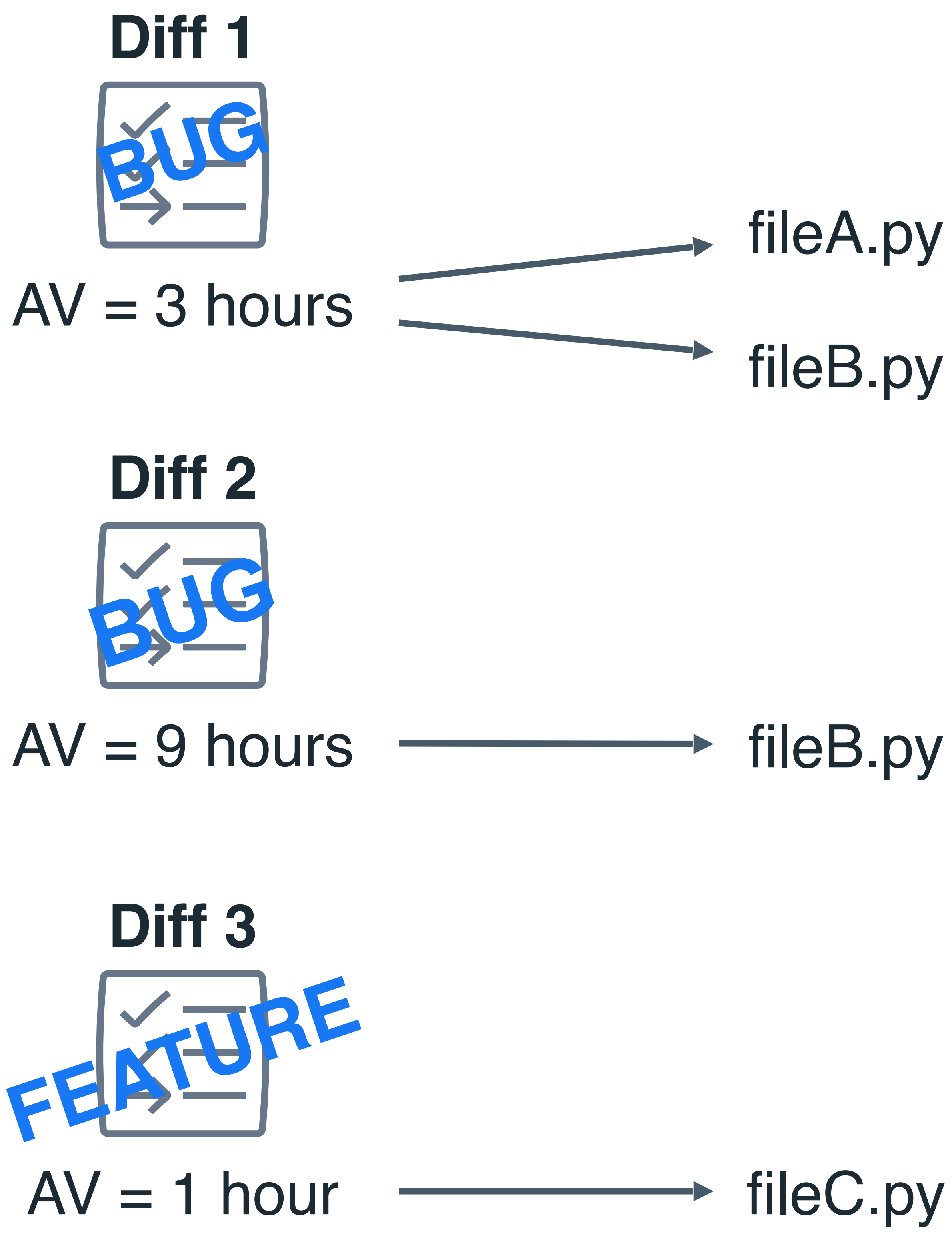
Sentinel wasn't able to add any reviewers to this diff because it contains all generated files or only adds new files and lacks historical information.

```
[-1.1451401710510254,
 1.3620548248291016,
 -2.7420852184295654,
 -2.5980613231658936,
 5.120920181274414,
 -3.0651018619537354,
 -0.4263494610786438,
 0.5120811462402344,
 1.0371060371398926,
```

Diff embedding

X @rmaranhao

DIFF INTENT vs. AV vs. RELIABILITY



Diff 1

BUG

AV = 3 hours

fileA.py

fileB.py

Diff 2

BUG

AV = 9 hours

fileB.py

Diff 3

FEATURE

AV = 1 hour

fileC.py

File	Bug Fixes	Features
fileB.py	2	0
fileA.py	1	0
fileC.py	0	1

TEST PLAN QUALITY vs. RELIABILITY

🎯 Prob

🧠 **Moc** ails in 📄).

👍 ***High quality test plans help improving review quality and engagement***

💯 ***NLP-based techniques are well suited for predicting test plan quality***

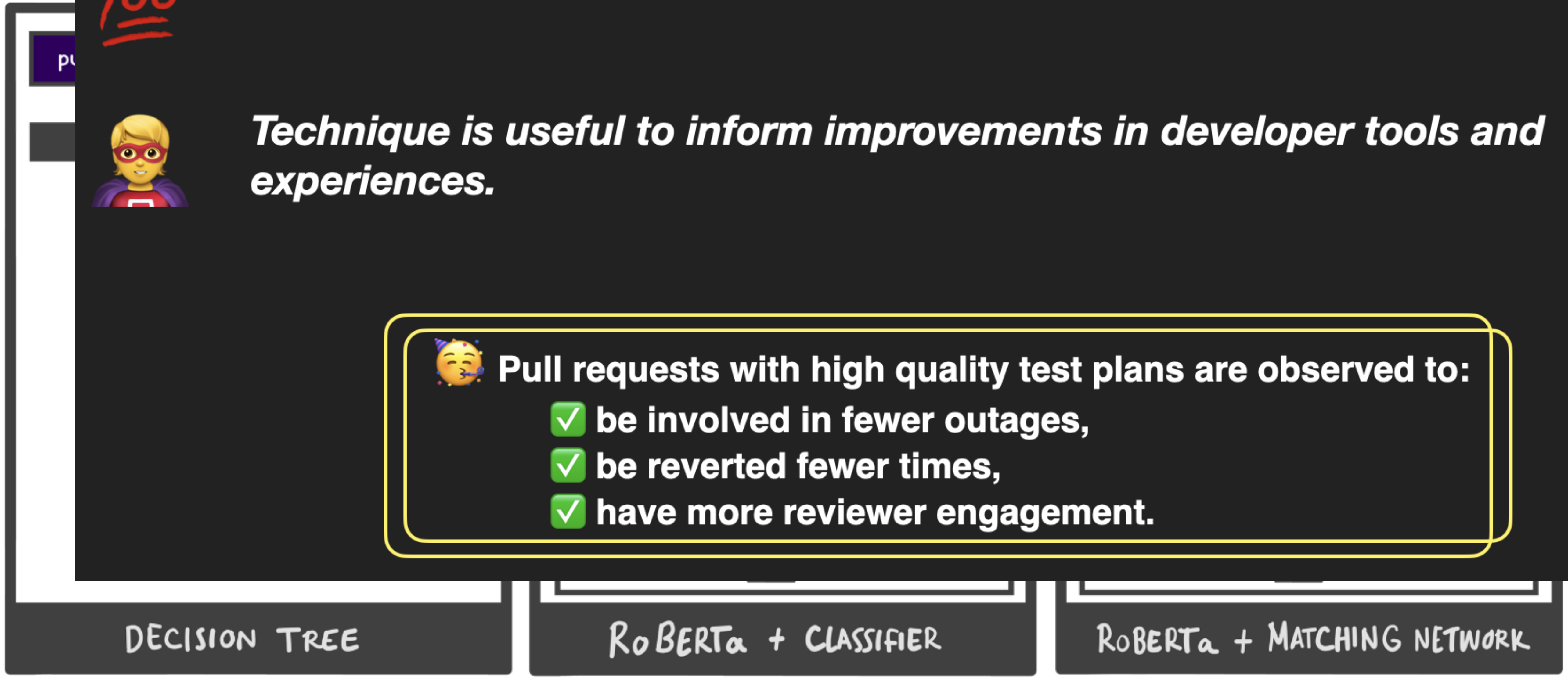
🦸 ***Technique is useful to inform improvements in developer tools and experiences.***

🥳 **Pull requests with high quality test plans are observed to:**

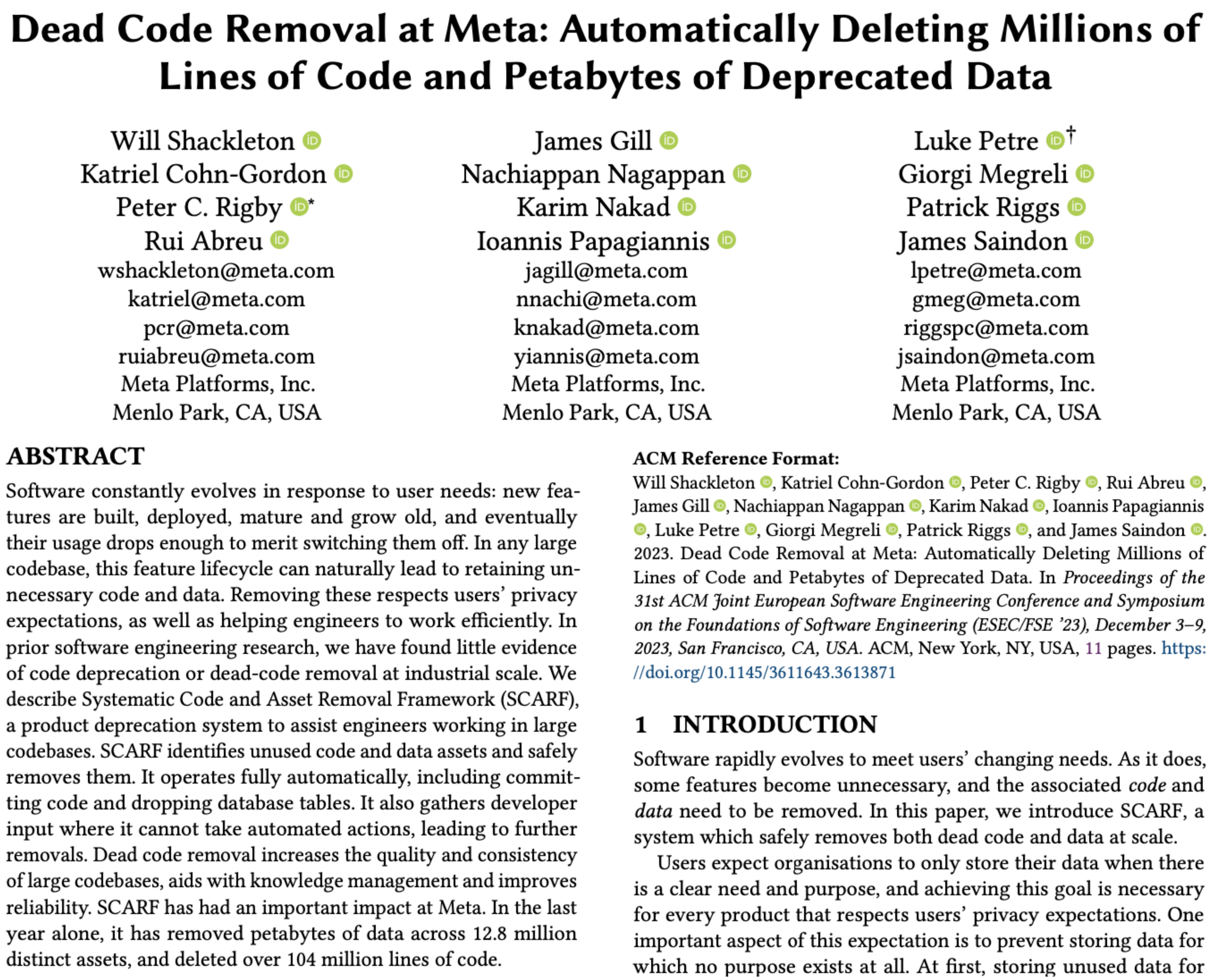
✅ **be involved in fewer outages,**

✅ **be reverted fewer times,**

✅ **have more reviewer engagement.**



DEAD CODE REMOVAL



Dead Code Removal at Meta: Automatically Deleting Millions of Lines of Code and Petabytes of Deprecated Data

Will Shackleton
Katriel Cohn-Gordon
Peter C. Rigby[*]
Rui Abreu
wshackleton@meta.com
katriel@meta.com
pcr@meta.com
ruiabreu@meta.com
Meta Platforms, Inc.
Menlo Park, CA, USA

James Gill
Nachiappan Nagappan
Karim Nakad
Ioannis Papagiannis
jagill@meta.com
nnachi@meta.com
knakad@meta.com
yiannis@meta.com
Meta Platforms, Inc.
Menlo Park, CA, USA

Luke Petre[†]
Giorgi Megreli
Patrick Riggs
James Saindon
lpetre@meta.com
gmeg@meta.com
riggspc@meta.com
jsaindon@meta.com
Meta Platforms, Inc.
Menlo Park, CA, USA

ABSTRACT

Software constantly evolves in response to user needs: new features are built, deployed, mature and grow old, and eventually their usage drops enough to merit switching them off. In any large codebase, this feature lifecycle can naturally lead to retaining unnecessary code and data. Removing these respects users' privacy expectations, as well as helping engineers to work efficiently. In prior software engineering research, we have found little evidence of code deprecation or dead-code removal at industrial scale. We describe Systematic Code and Asset Removal Framework (SCARF), a product deprecation system to assist engineers working in large codebases. SCARF identifies unused code and data assets and safely removes them. It operates fully automatically, including committing code and dropping database tables. It also gathers developer input where it cannot take automated actions, leading to further removals. Dead code removal increases the quality and consistency of large codebases, aids with knowledge management and improves reliability. SCARF has had an important impact at Meta. In the last year alone, it has removed petabytes of data across 12.8 million distinct assets, and deleted over 104 million lines of code.

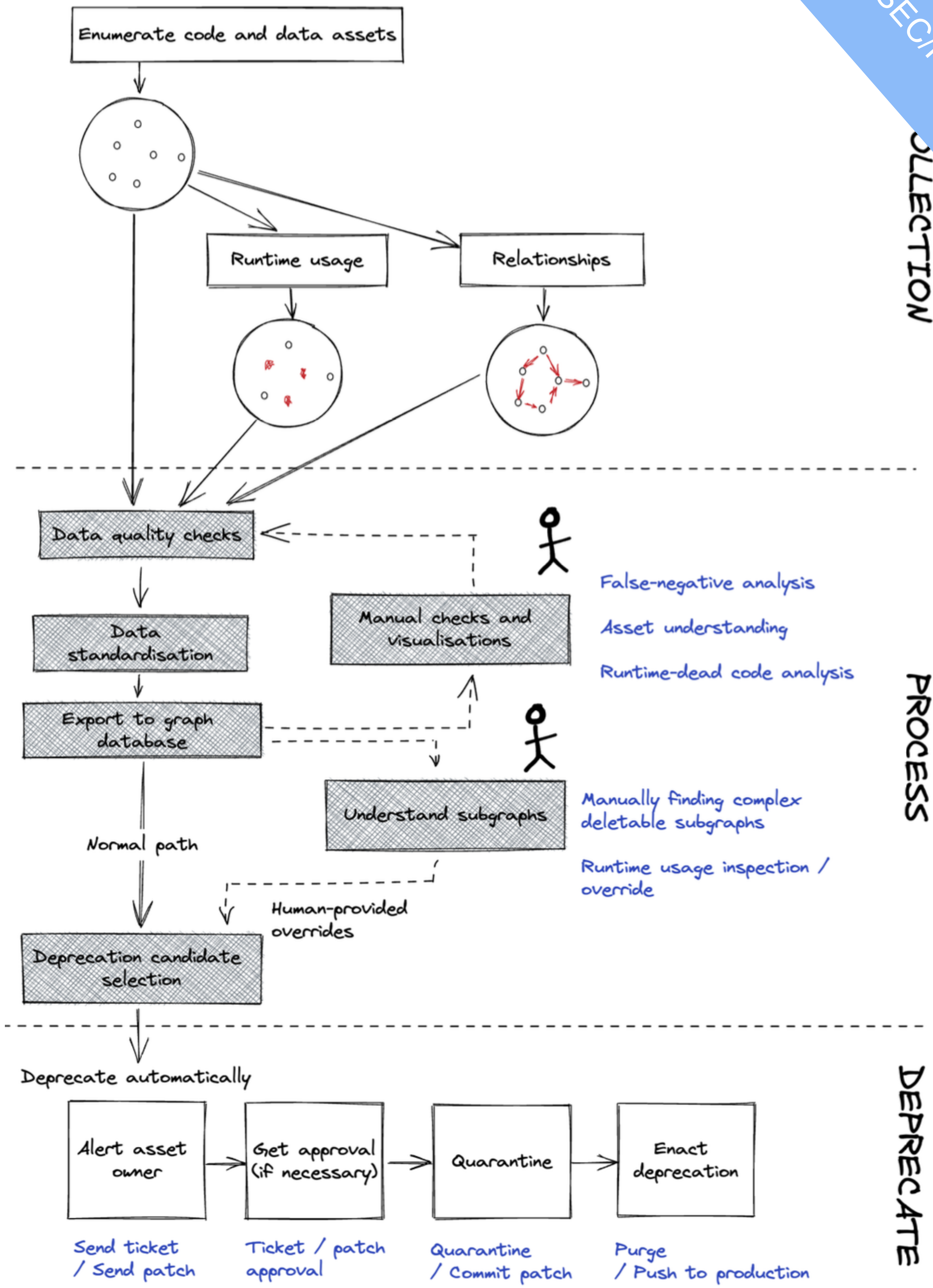
ACM Reference Format:
Will Shackleton, Katriel Cohn-Gordon, Peter C. Rigby, Rui Abreu, James Gill, Nachiappan Nagappan, Karim Nakad, Ioannis Papagiannis, Luke Petre, Giorgi Megreli, Patrick Riggs, and James Saindon. 2023. Dead Code Removal at Meta: Automatically Deleting Millions of Lines of Code and Petabytes of Deprecated Data. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '23), December 3–9, 2023, San Francisco, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3611643.3613871

1 INTRODUCTION

Software rapidly evolves to meet users' changing needs. As it does, some features become unnecessary, and the associated *code* and *data* need to be removed. In this paper, we introduce SCARF, a system which safely removes both dead code and data at scale.

Users expect organisations to only store their data when there is a clear need and purpose, and achieving this goal is necessary for every product that respects users' privacy expectations. One important aspect of this expectation is to prevent storing data for which no purpose exists at all. At first, storing unused data for

Dead Code Removal at Meta: Automatically Deleting Millions of Lines of Code and Petabytes of Deprecated Data, to appear at ESEC/FSE 2023



to appear at ESEC/FSE 2023

Knowledge

1. Measuring Productivity

2. Drivers (developer POV)
    a. Authoring Velocity
    b. Reliability (incl. Quality)
    c. Knowledge (Readability)

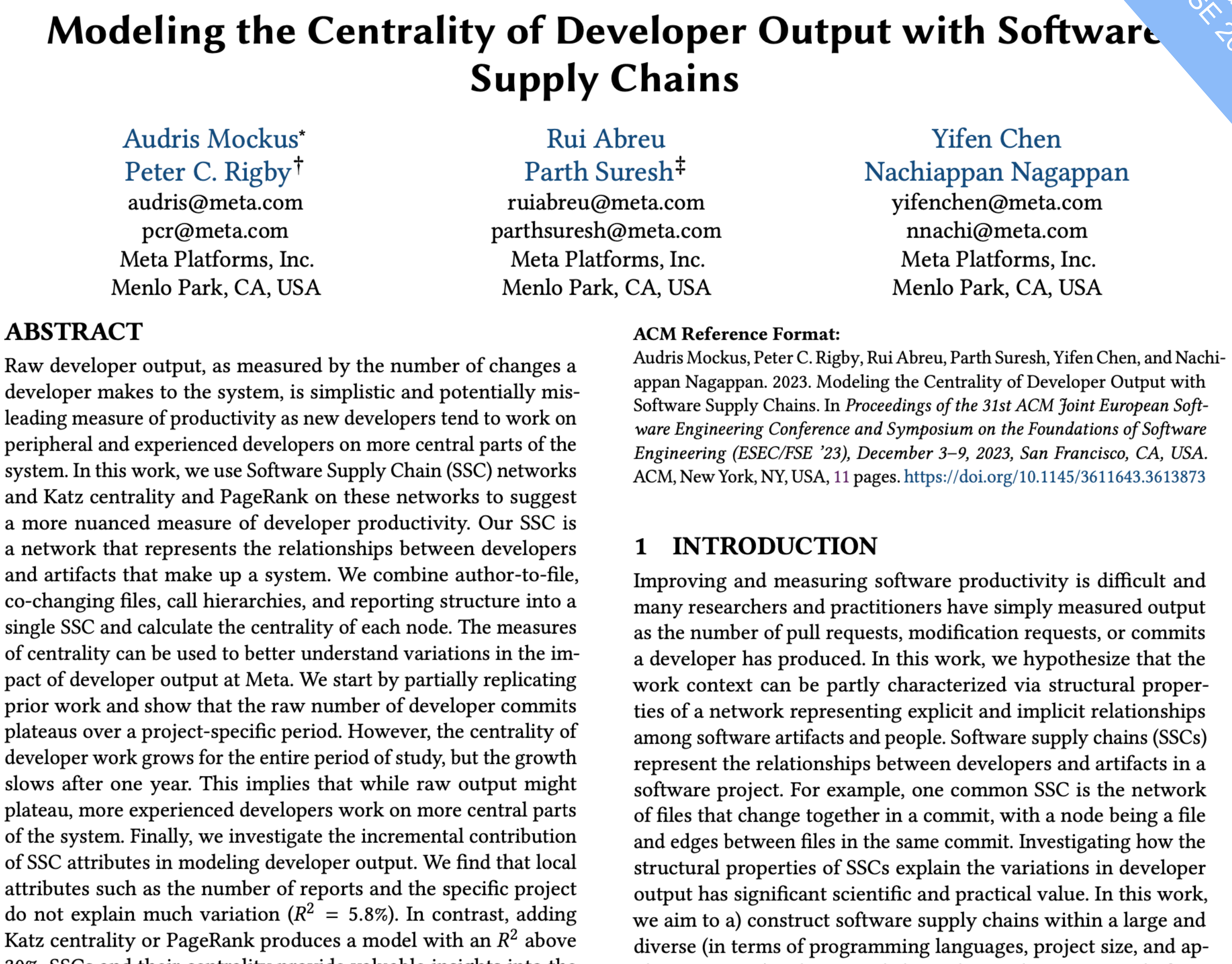
3. What’s Next?

4. Q&A

MODELLING CENTRALITY OF DEVELOPER OUTPUT

— knowlege —

to appear at ESEC/FSE 2023



Modeling the Centrality of Developer Output with Software Supply Chains

Audris Mockus*
Peter C. Rigby†
audris@meta.com
pcr@meta.com
Meta Platforms, Inc.
Menlo Park, CA, USA

Rui Abreu
Parth Suresh‡
ruiabreu@meta.com
parthsuresh@meta.com
Meta Platforms, Inc.
Menlo Park, CA, USA

Yifen Chen
Nachiappan Nagappan
yifenchen@meta.com
nnachi@meta.com
Meta Platforms, Inc.
Menlo Park, CA, USA

ABSTRACT

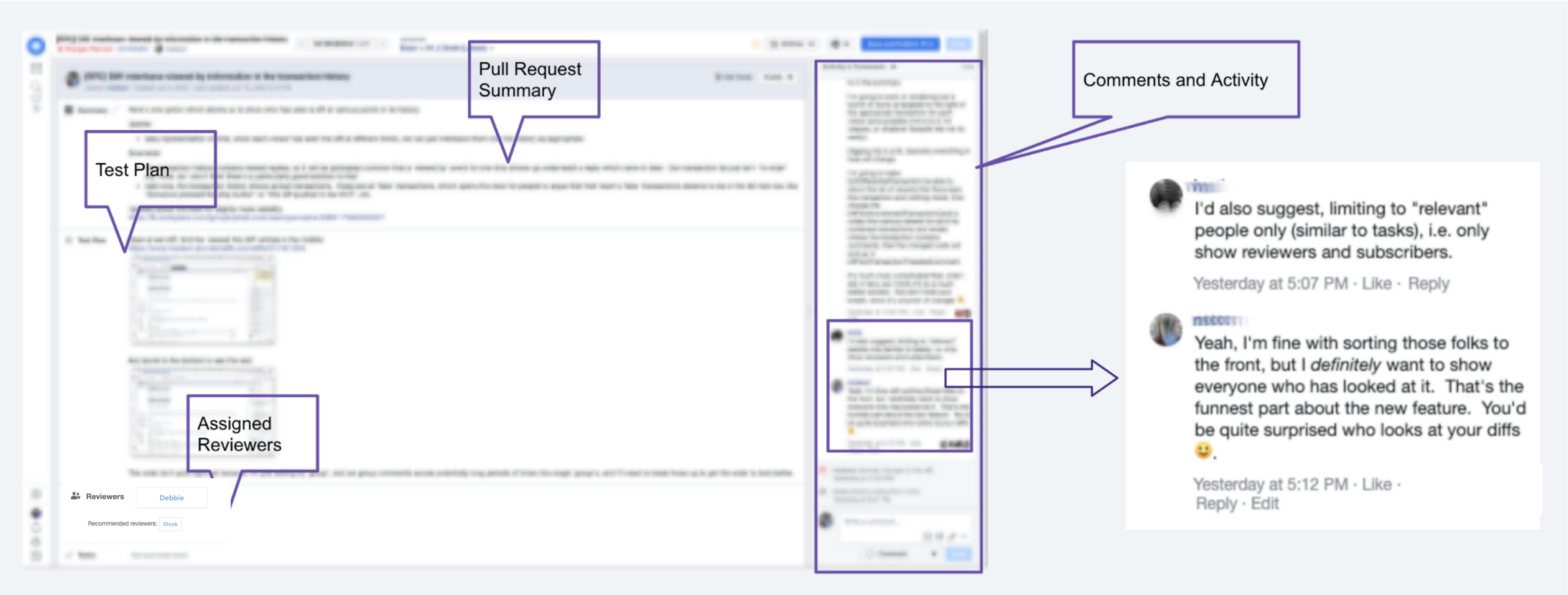
Raw developer output, as measured by the number of changes a developer makes to the system, is simplistic and potentially misleading measure of productivity as new developers tend to work on peripheral and experienced developers on more central parts of the system. In this work, we use Software Supply Chain (SSC) networks and Katz centrality and PageRank on these networks to suggest a more nuanced measure of developer productivity. Our SSC is a network that represents the relationships between developers and artifacts that make up a system. We combine author-to-file, co-changing files, call hierarchies, and reporting structure into a single SSC and calculate the centrality of each node. The measures of centrality can be used to better understand variations in the impact of developer output at Meta. We start by partially replicating prior work and show that the raw number of developer commits plateaus over a project-specific period. However, the centrality of developer work grows for the entire period of study, but the growth slows after one year. This implies that while raw output might plateau, more experienced developers work on more central parts of the system. Finally, we investigate the incremental contribution of SSC attributes in modeling developer output. We find that local attributes such as the number of reports and the specific project do not explain much variation ($R^2 = 5.8\%$). In contrast, adding Katz centrality or PageRank produces a model with an R^2 above 30%. SSCs and their centrality provide valuable insights into the

ACM Reference Format:
Audris Mockus, Peter C. Rigby, Rui Abreu, Parth Suresh, Yifen Chen, and Nachiappan Nagappan. 2023. Modeling the Centrality of Developer Output with Software Supply Chains. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '23), December 3–9, 2023, San Francisco, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3611643.3613873

1 INTRODUCTION

Improving and measuring software productivity is difficult and many researchers and practitioners have simply measured output as the number of pull requests, modification requests, or commits a developer has produced. In this work, we hypothesize that the work context can be partly characterized via structural properties of a network representing explicit and implicit relationships among software artifacts and people. Software supply chains (SSCs) represent the relationships between developers and artifacts in a software project. For example, one common SSC is the network of files that change together in a commit, with a node being a file and edges between files in the same commit. Investigating how the structural properties of SSCs explain the variations in developer output has significant scientific and practical value. In this work, we aim to a) construct software supply chains within a large and diverse (in terms of programming languages, project size, and ap-

WHAT TO DO WITH CENTRALITY/KNOWLEDGE AND INTENT?



I'd also suggest, limiting to "relevant" people only (similar to tasks), i.e. only show reviewers and subscribers.

Yesterday at 5:07 PM · Like · Reply

Yeah, I'm fine with sorting those folks to the front, but I *definitely* want to show everyone who has looked at it. That's the funnest part about the new feature. You'd be quite surprised who looks at your diffs 😀.

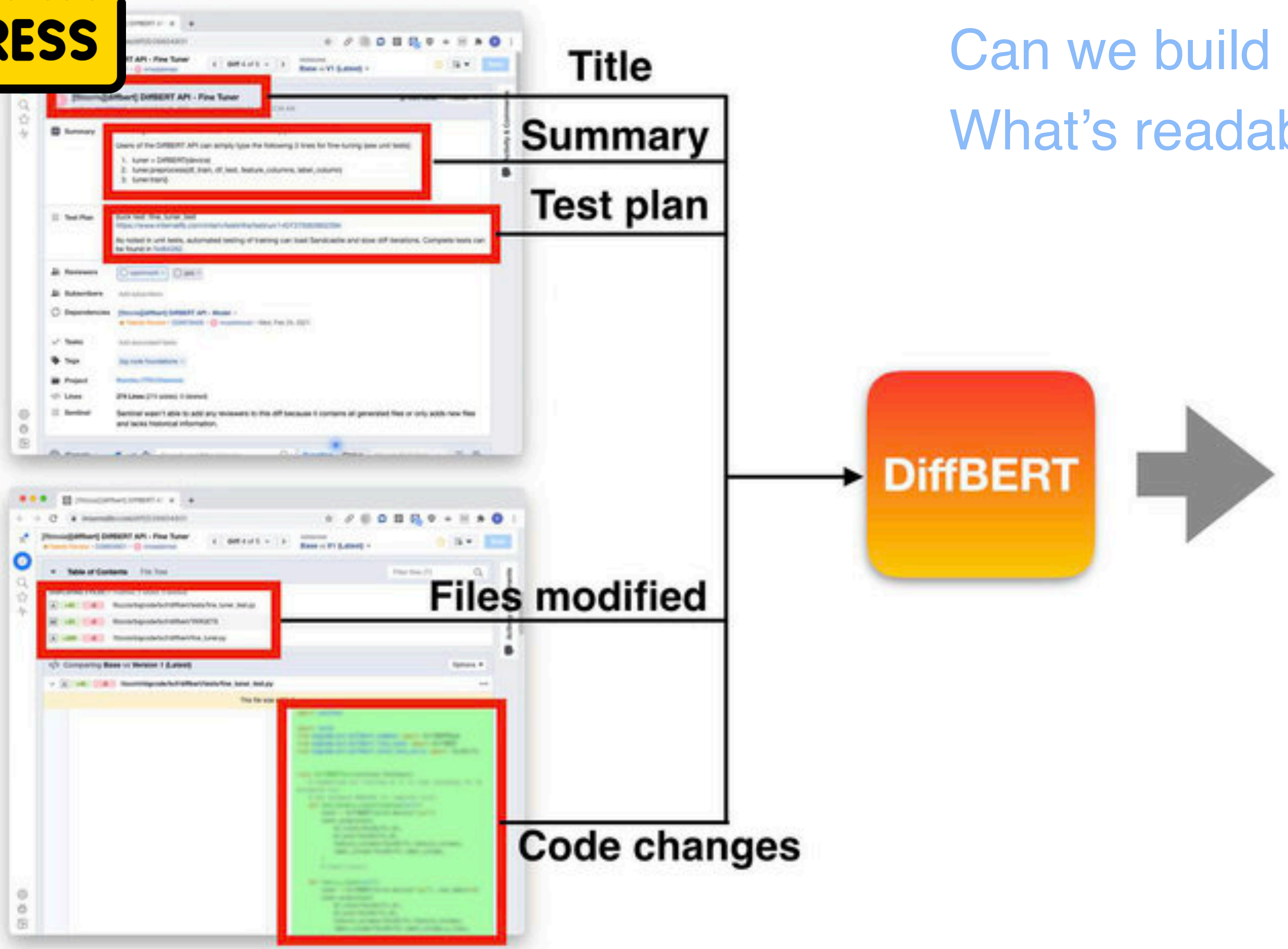
Yesterday at 5:12 PM · Like · Reply · Edit

WORK IN PROGRESS

CODE READABILITY

Can we build such a model?

What’s readable code?



```
[-1.1451401710510254,
 1.3620548248291016,
 -2.7420852184295654,
 -2.5980613231658936,
 5.120920181274414,
 -3.0651018619537354,
 -0.4263494610786438,
 0.5120811462402344,
 1.0371060371398926,
```

Diff embedding

@rmaranhao

What's Next?

1. Measuring Productivity

2. Debbie the developer
    a. Authoring Velocity
    b. Reliability (incl. Quality)
    c. Knowledge (incl. Reliability)

3. What's Next?

4. Q&A

WHAT’S NEXT?

Predictions

Integrations

Q&A

1. Measuring Productivity

2. Drivers (developer POV)
   a. Authoring Velocity
   b. Reliability (incl. Quality)
   c. Knowledge (incl. Readability)

3. What’s Next?

4. Q&A

Meta