# We’re Measuring Productivity Wrong

Abi Noda

> “Knowing how to measure productivity or even define developer productivity has remained elusive.”

Abi Noda, Nicole Forsgren, et al.

“Quantifying our impact is an existential challenge."

Chad Sanderson, Head of Platform at Convoy

**Part 1: Measuring productivity is hard**

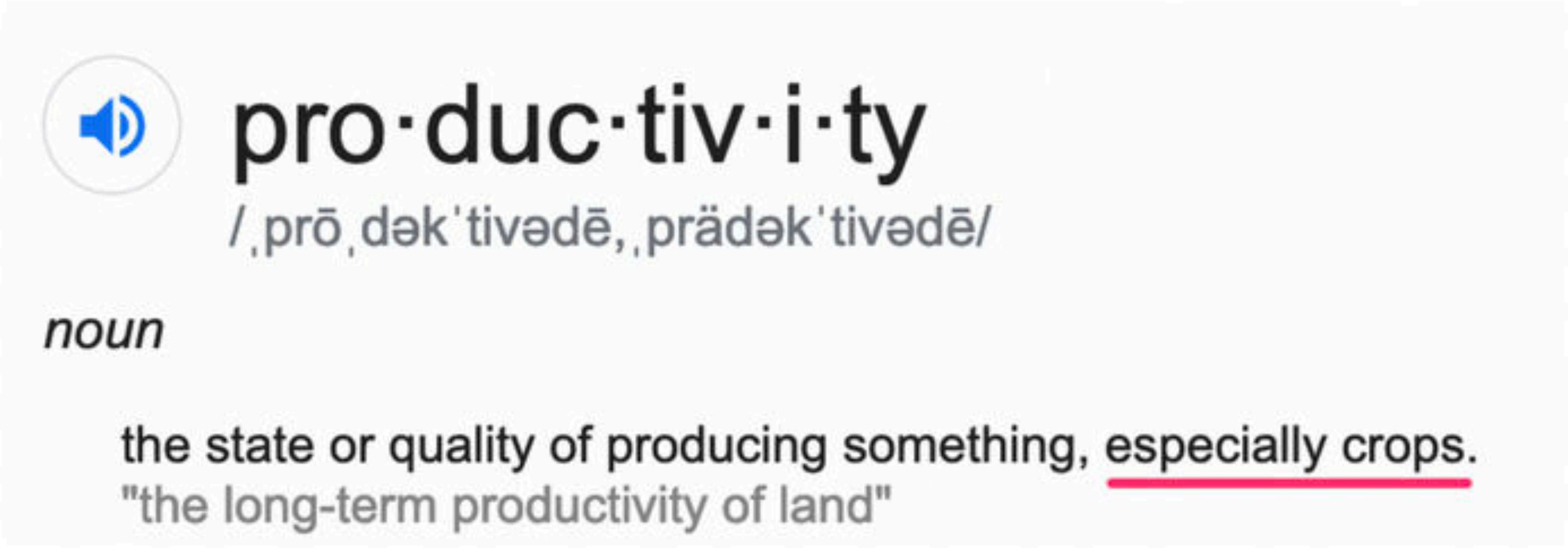
Part 2: Why basic metrics aren’t enough

Part 3: A better way to measure

> “Defining productivity has been a challenge facing both researchers and practitioners.”

Caitlin Sadowski, Google

$$Productivity = \frac{Revenue}{Developers}$$



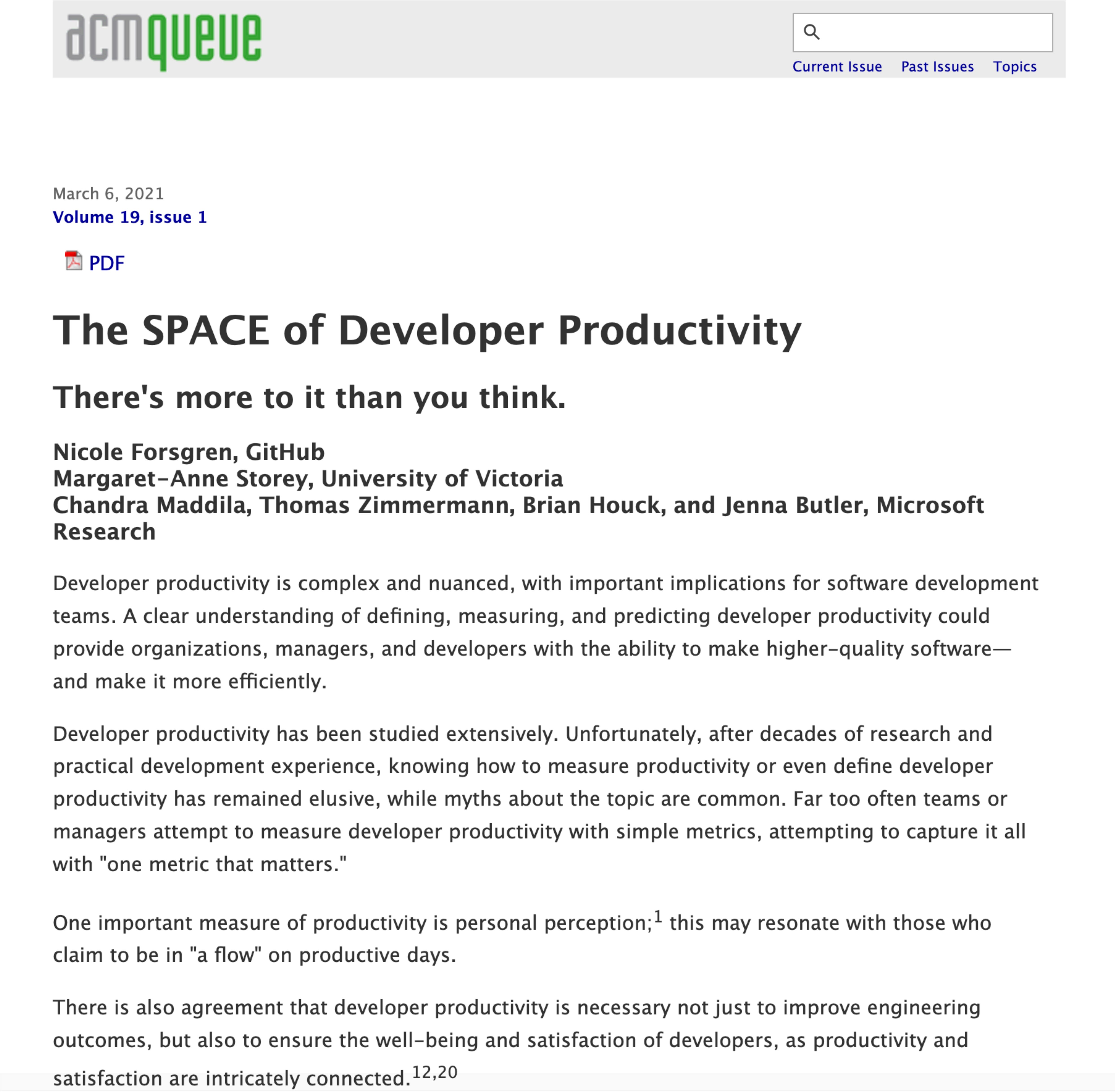
pro·duc·tiv·i·ty

/ˌprōˌdəkˈtivədē,ˌprädəkˈtivədē/

*noun*

the state or quality of producing something, especially crops.

"the long-term productivity of land"



March 6, 2021
Volume 19, issue 1

# The SPACE of Developer Productivity

## There's more to it than you think.

**Nicole Forsgren, GitHub**
**Margaret-Anne Storey, University of Victoria**
**Chandra Maddila, Thomas Zimmermann, Brian Houck, and Jenna Butler, Microsoft Research**

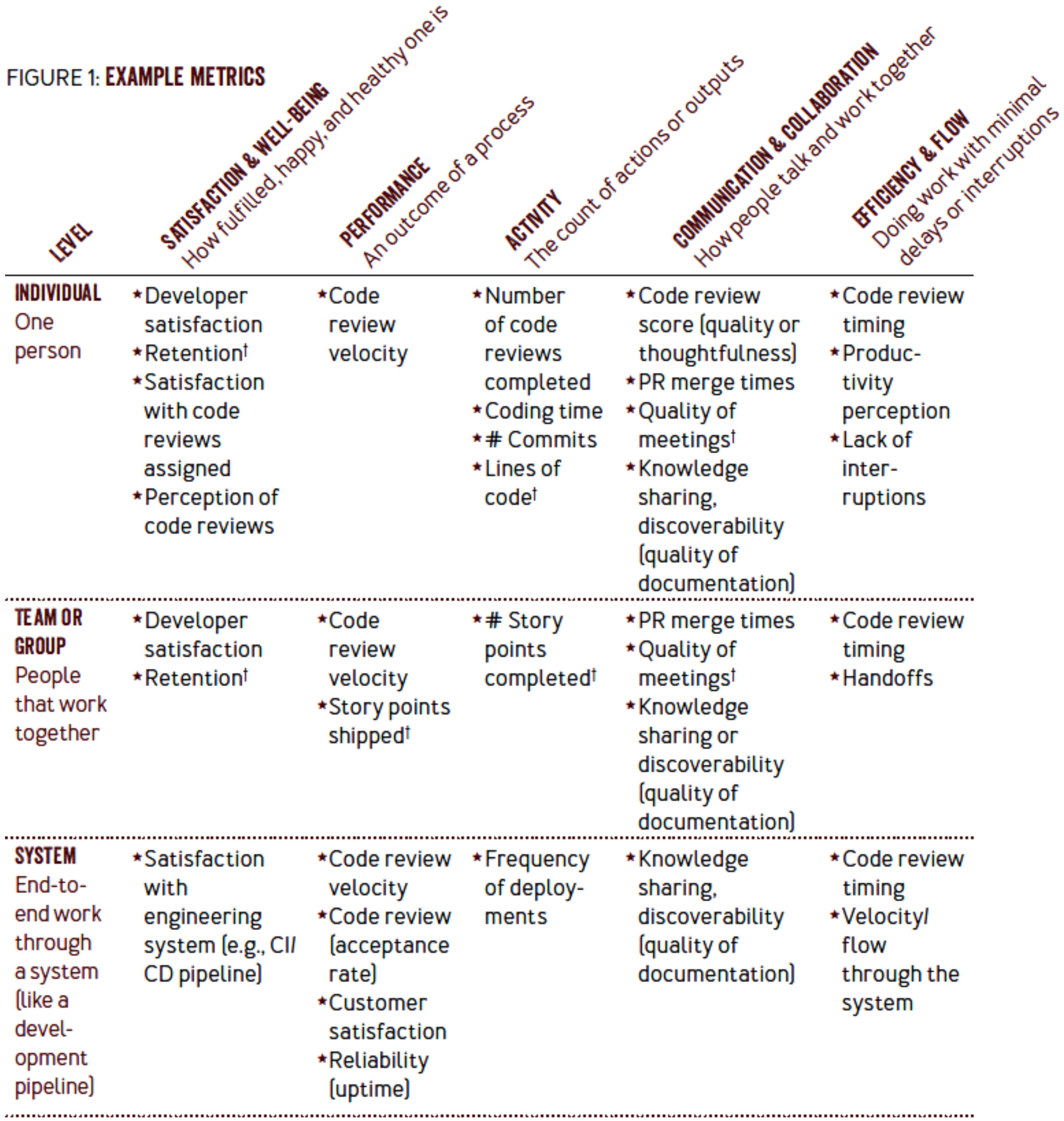
Developer productivity is complex and nuanced, with important implications for software development teams. A clear understanding of defining, measuring, and predicting developer productivity could provide organizations, managers, and developers with the ability to make higher-quality software—and make it more efficiently.

Developer productivity has been studied extensively. Unfortunately, after decades of research and practical development experience, knowing how to measure productivity or even define developer productivity has remained elusive, while myths about the topic are common. Far too often teams or managers attempt to measure developer productivity with simple metrics, attempting to capture it all with "one metric that matters."

One important measure of productivity is personal perception;[1] this may resonate with those who claim to be in "a flow" on productive days.

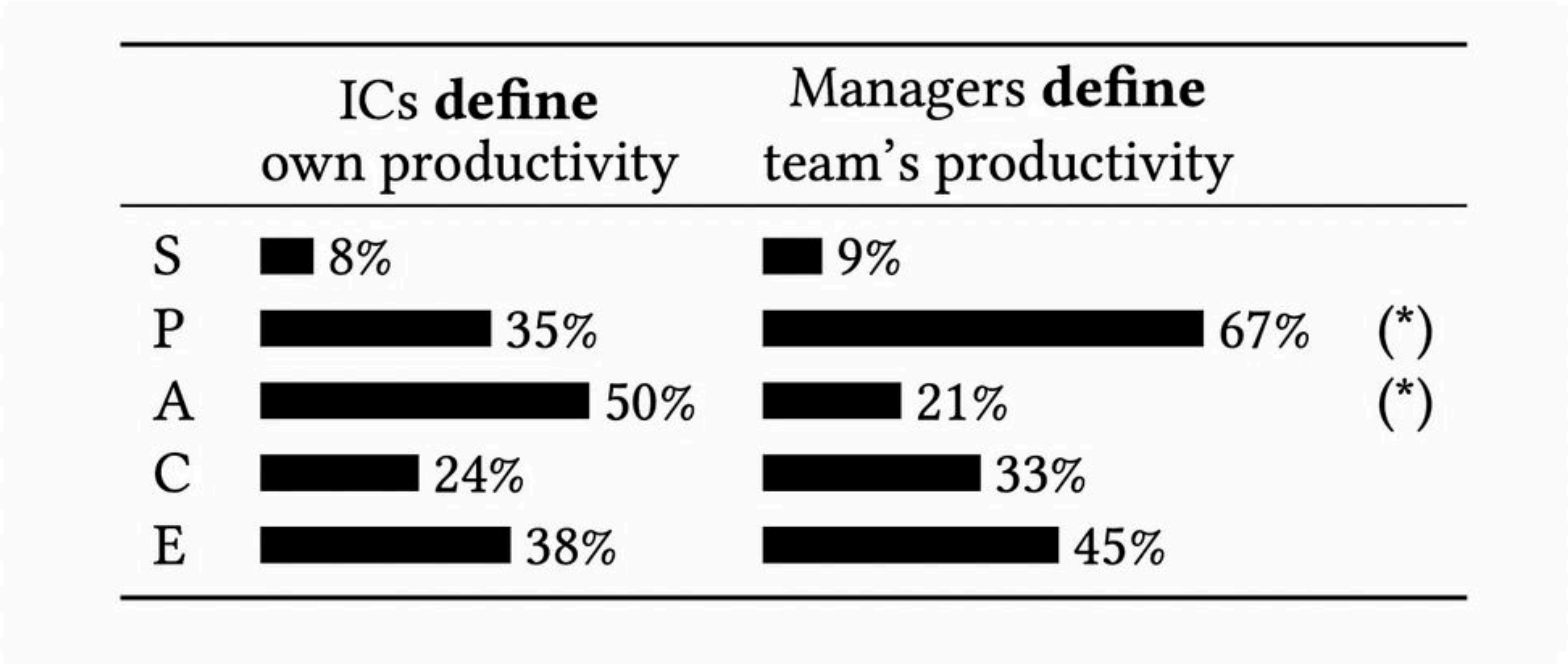
There is also agreement that developer productivity is necessary not just to improve engineering outcomes, but also to ensure the well-being and satisfaction of developers, as productivity and satisfaction are intricately connected.[12,20]

FIGURE 1: **EXAMPLE METRICS**

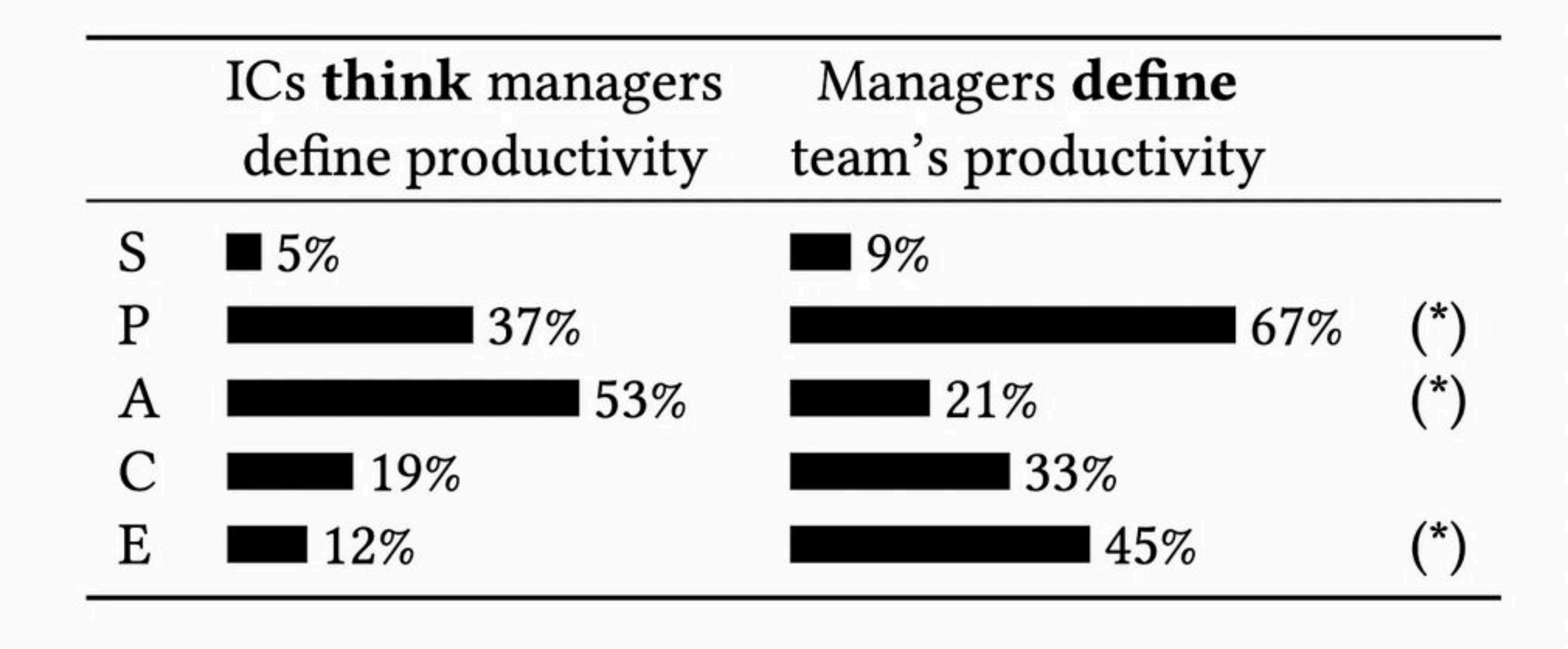


| **LEVEL** | **SATISFACTION & WELL-BEING** How fulfilled, happy, and healthy one is | **PERFORMANCE** An outcome of a process | **ACTIVITY** The count of actions or outputs | **COMMUNICATION & COLLABORATION** How people talk and work together | **EFFICIENCY & FLOW** Doing work with minimal delays or interruptions |
|---|---|---|---|---|---|
| **INDIVIDUAL** One person | * Developer satisfaction<br>* Retention†<br>* Satisfaction with code reviews assigned<br>* Perception of code reviews | * Code review velocity | * Number of code reviews completed<br>* Coding time<br>* # Commits<br>* Lines of code† | * Code review score (quality or thoughtfulness)<br>* PR merge times<br>* Quality of meetings†<br>* Knowledge sharing, discoverability (quality of documentation) | * Code review timing<br>* Productivity perception<br>* Lack of interruptions |
| **TEAM OR GROUP** People that work together | * Developer satisfaction<br>* Retention† | * Code review velocity<br>* Story points shipped† | * # Story points completed† | * PR merge times<br>* Quality of meetings†<br>* Knowledge sharing or discoverability (quality of documentation) | * Code review timing<br>* Handoffs |
| **SYSTEM** End-to-end work through a system (like a development pipeline) | * Satisfaction with engineering system (e.g., CI/CD pipeline) | * Code review velocity<br>* Code review (acceptance rate)<br>* Customer satisfaction<br>* Reliability (uptime) | * Frequency of deployments | * Knowledge sharing, discoverability (quality of documentation) | * Code review timing<br>* Velocity/flow through the system |

† *Use these metrics with (even more) caution — they can proxy more things.*



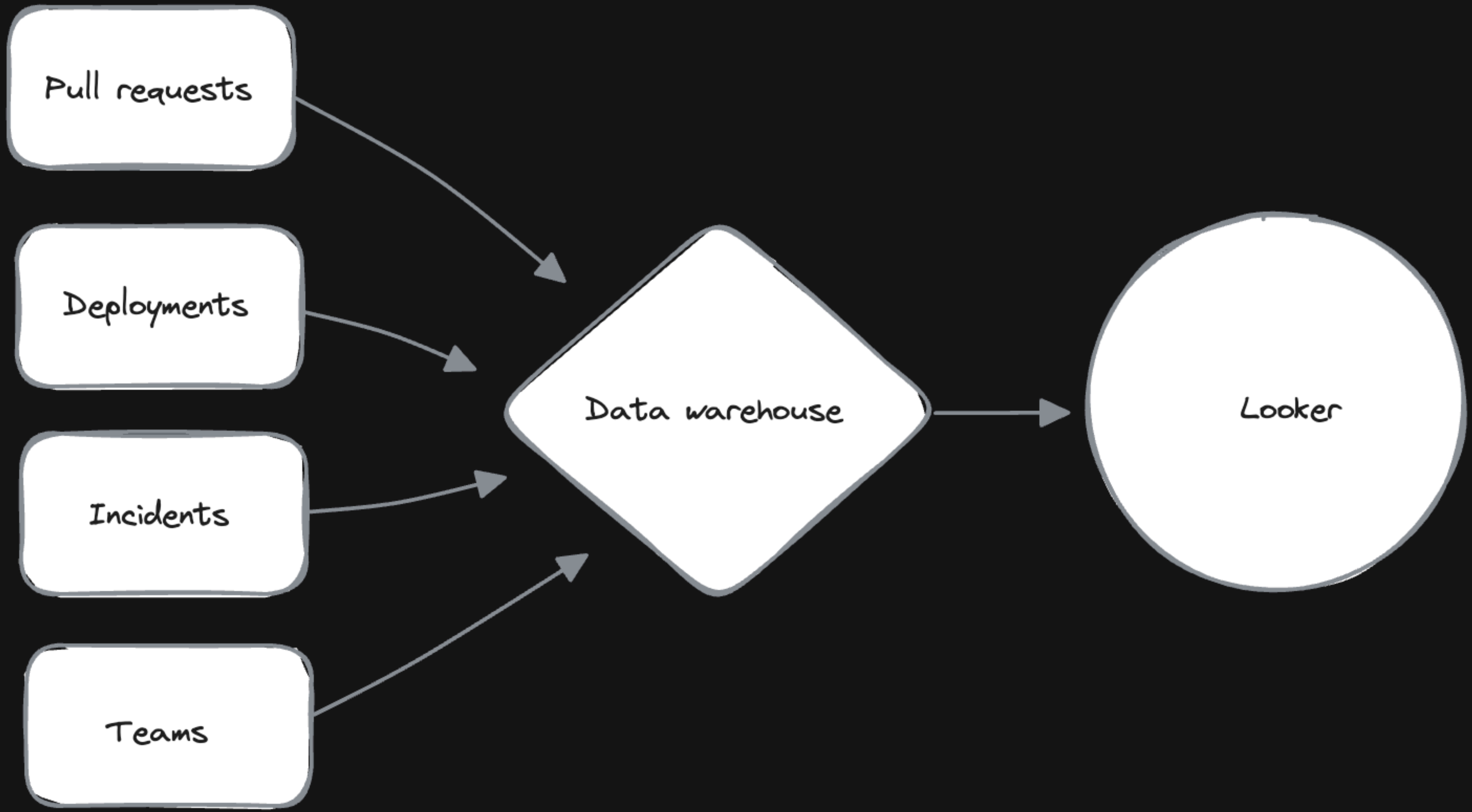
| | ICs **define** own productivity | Managers **define** team's productivity | |
|---|---|---|---|
| S | 8% | 9% | |
| P | 35% | 67% | (*) |
| A | 50% | 21% | (*) |
| C | 24% | 33% | |
| E | 38% | 45% | |

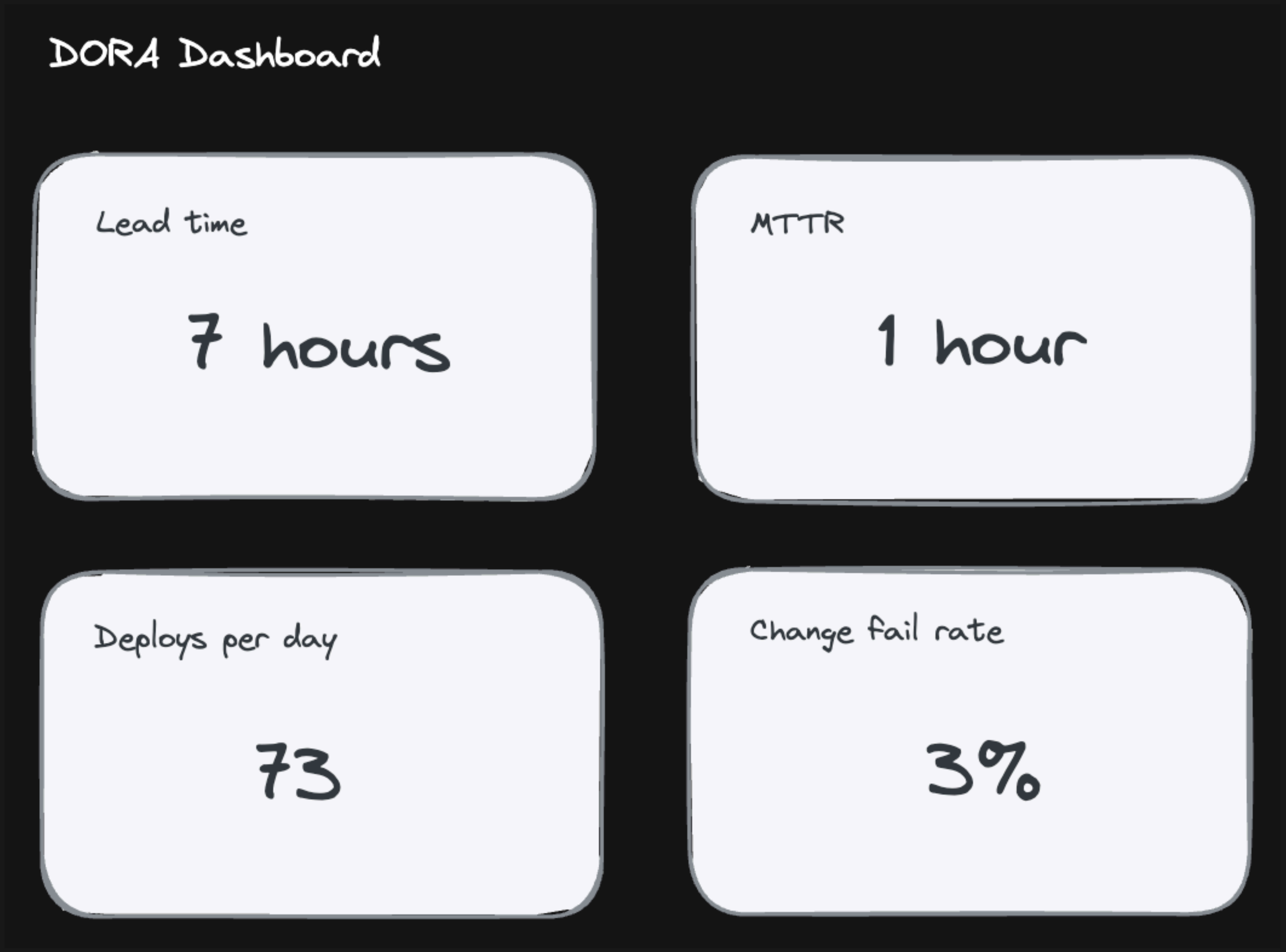


|  | ICs **think** managers define productivity | Managers **define** team's productivity |  |
|---|---|---|---|
| S | 5% | 9% |  |
| P | 37% | 67% | (*) |
| A | 53% | 21% | (*) |
| C | 19% | 33% |  |
| E | 12% | 45% | (*) |

“One failure mode I've seen is a leader comes in and says, ‘DORA metrics across the board.’ Because it's an easy button.”

Laura Tacho, Engineering Leadership Coach


Pull requests
Deployments
Incidents
Teams
Data warehouse
Looker



DORA Dashboard

Lead time

7 hours

MTTR

1 hour

Deploys per day

73

Change fail rate

3%

“Too many organizations spend effort building beautiful DORA dashboards that nobody looks at.”

Nathen Harvey, DORA

“Every conference I go to feels like it's full of people talking about not just the DORA metrics but their shortcomings.”

Brian Guthrie, VPE at Meetup

> “We didn't intend to claim that these are the metrics that you should use.”

Dr. Margaret-Anne Storey, Co-Author of SPACE

**Part 1:** Measuring productivity is hard

**Part 2: Why basic metrics aren’t enough**

Part 3: A better way to measure

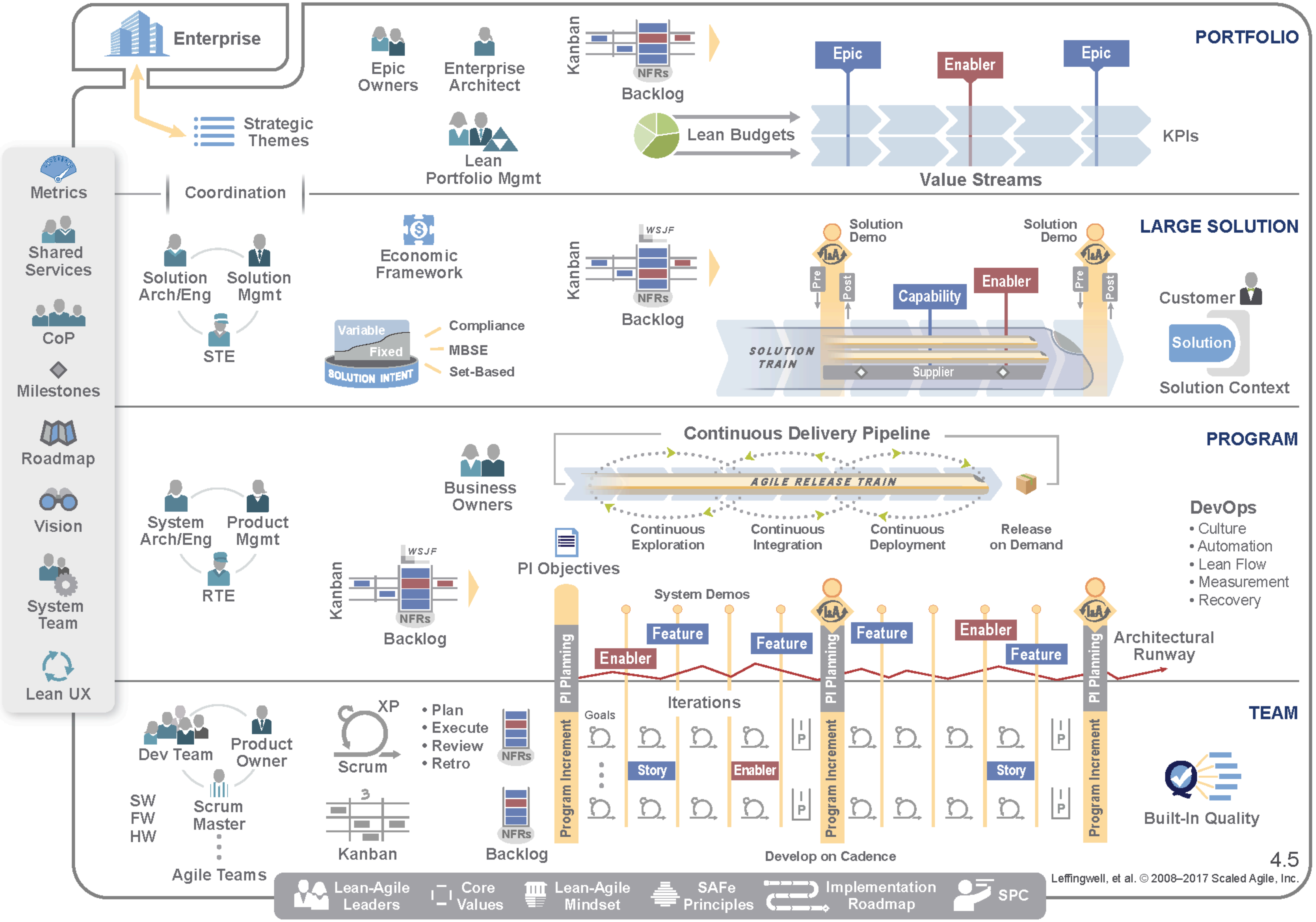
## Common engineering metrics

Lead time
Issue cycle time
WIPs
Deployment frequency
Pull request throughput
Pull request cycle time
Story points
Change failure rate
MTTR

CAPACITY 150 KGF
TOYOTA

# SAFe® for Lean Enterprises

**Full** Configuration



Enterprise

Strategic Themes

Epic Owners

Enterprise Architect

Lean Portfolio Mgmt

Kanban

NFRs

Backlog

Lean Budgets

Epic

Enabler

Epic

KPIs

Value Streams

PORTFOLIO

Metrics

Shared Services

CoP

Milestones

Roadmap

Vision

System Team

Lean UX

Coordination

Solution Arch/Eng

Solution Mgmt

STE

Economic Framework

Variable

Fixed

SOLUTION INTENT

Compliance

MBSE

Set-Based

Kanban

WSJF

NFRs

Backlog

Solution Demo

Solution Demo

Pre

Post

Pre

Post

Capability

Enabler

SOLUTION TRAIN

Supplier

LARGE SOLUTION

Customer

Solution

Solution Context

Continuous Delivery Pipeline

AGILE RELEASE TRAIN

Continuous Exploration

Continuous Integration

Continuous Deployment

Release on Demand

PROGRAM

DevOps

- Culture
- Automation
- Lean Flow
- Measurement
- Recovery

System Arch/Eng

Product Mgmt

RTE

Business Owners

Kanban

WSJF

NFRs

Backlog

PI Objectives

System Demos

PI Planning

Enabler

Feature

Feature

PI Planning

Feature

Enabler

Feature

PI Planning

Architectural Runway

Dev Team

Product Owner

Scrum Master

SW
FW
HW

Agile Teams

XP

Scrum

- Plan
- Execute
- Review
- Retro

NFRs

Kanban

NFRs

Backlog

Program Increment

Goals

Iterations

Story

Enabler

Program Increment

Story

Program Increment

P

P

P

P

Develop on Cadence

TEAM

Built-In Quality

4.5

Lean-Agile Leaders

Core Values

Lean-Agile Mindset

SAFe Principles

Implementation Roadmap

SPC

## Common engineering metrics

Lead time
Issue cycle time
WIPs
Deployment frequency
Pull request throughput
Pull request cycle time
Story points
Change failure rate
MTTR

## Common engineering metrics

Lead time
Issue cycle time
WIPs
Deployment frequency
Pull request throughput
Pull request cycle time
Story points
Change failure rate
MTTR

## Manufacturing metrics

Lead time
Total Cycle Time
WIP Inventory/Turns
On-Time Delivery to Commit
Throughput
Yield
Capacity Utilization
Reportable Incidents
Schedule or Production Attainment
Engineering Change Order Cycle Time

Source:
*Manufacturing Enterprise Solutions Association*

## Common engineering metrics

Lead time
Issue cycle time
WIPs
Deployment frequency
Pull request throughput
Pull request cycle time
Story points
Change failure rate
MTTR

## Manufacturing metrics

Lead time
Total Cycle Time
WIP Inventory/Turns
On-Time Delivery to Commit
Throughput
Yield
Capacity Utilization
Reportable Incidents
Schedule or Production Attainment
Engineering Change Order Cycle Time

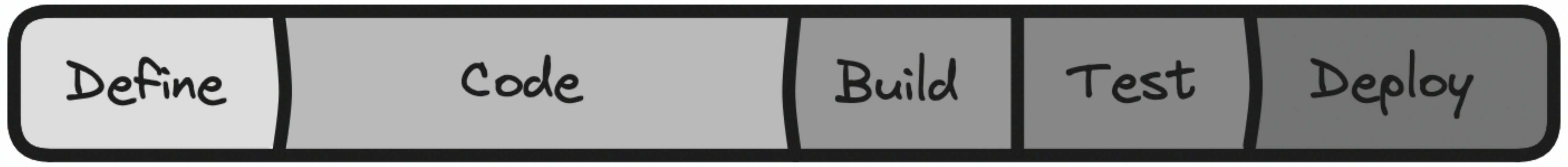
Source:
*Manufacturing Enterprise Solutions Association*

Pounds of coal shoveled tells you which shovelers are the best; lines of code will not tell you which software developers are the best.

Collin Green & Ciera Jaspan, Google

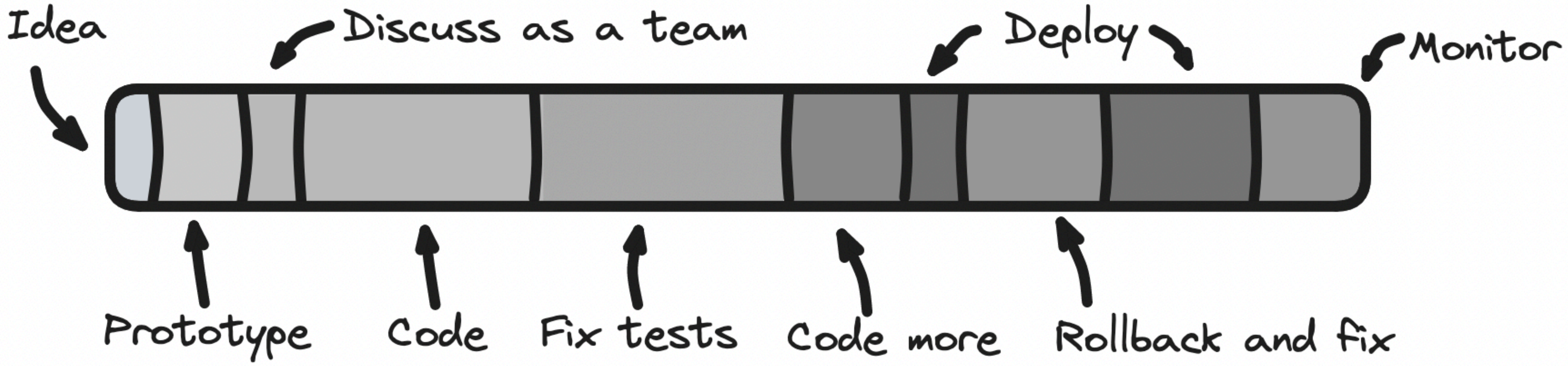
CAPACITY 150 KGF
2.5-5
TOYOTA

# DREAM



Define Code Build Test Deploy

# REALITY



Idea

Discuss as a team

Deploy

Monitor

Prototype

Code

Fix tests

Code more

Rollback and fix

“Engineers tell me: ‘I get it, the book *Accelerate* is great, but that’s not the world I live in.’”

Max Pugliese, Director of Platform Engineering at Apple

# Hard metrics don’t tell you the full story

> “Hard metrics tell you what developers are doing, but they don’t tell you why.”

Ciera Jaspan, Engineering Productivity Research at Google

Hard metrics don’t tell you where to focus

Part 1: Measuring productivity is hard

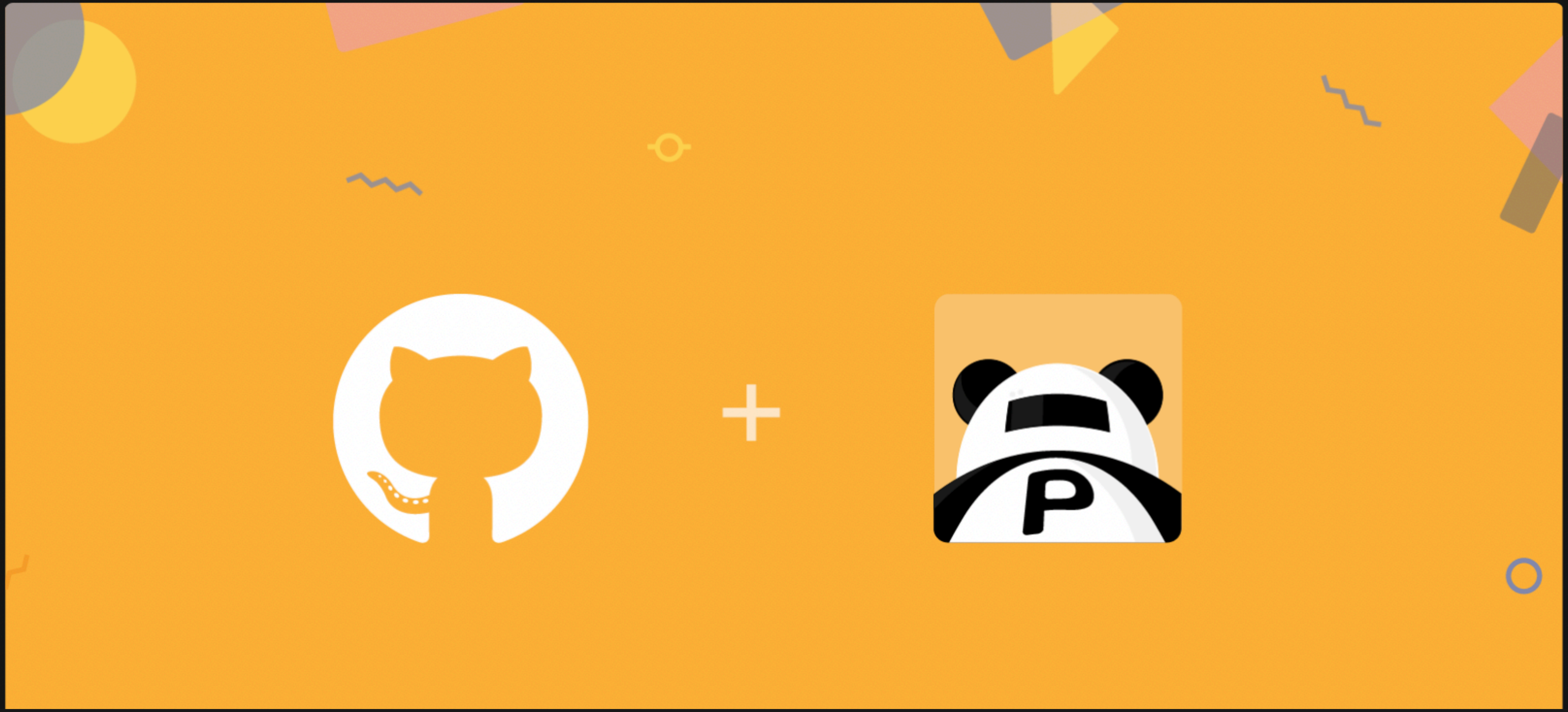
Part 2: Why basic metrics aren’t enough

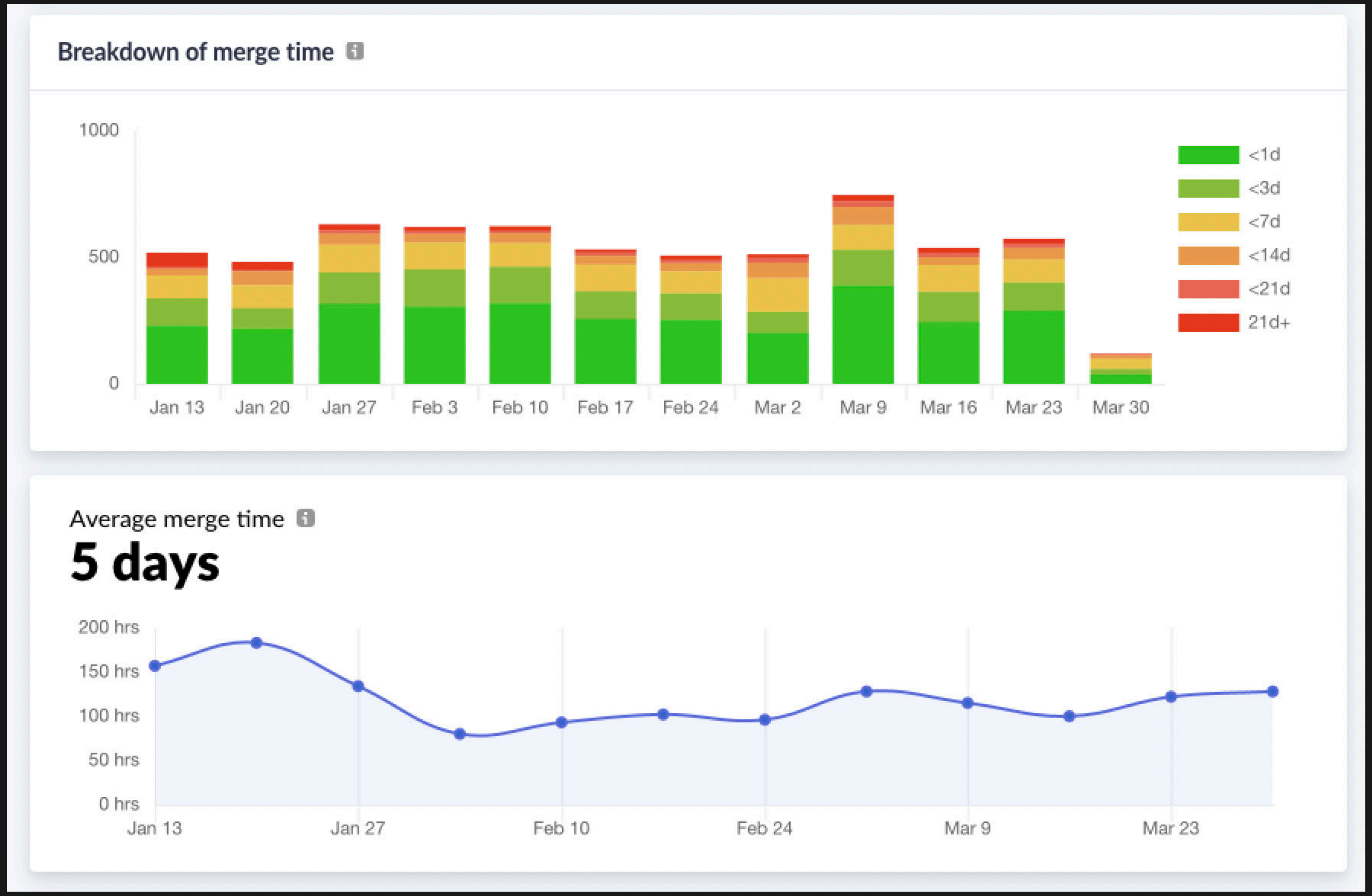
**Part 3: A better way to measure**

Company

# GitHub acquires Pull Panda—a better way to collaborate on code reviews

We've acquired Pull Panda to help teams create more efficient and effective code review workflows on GitHub.

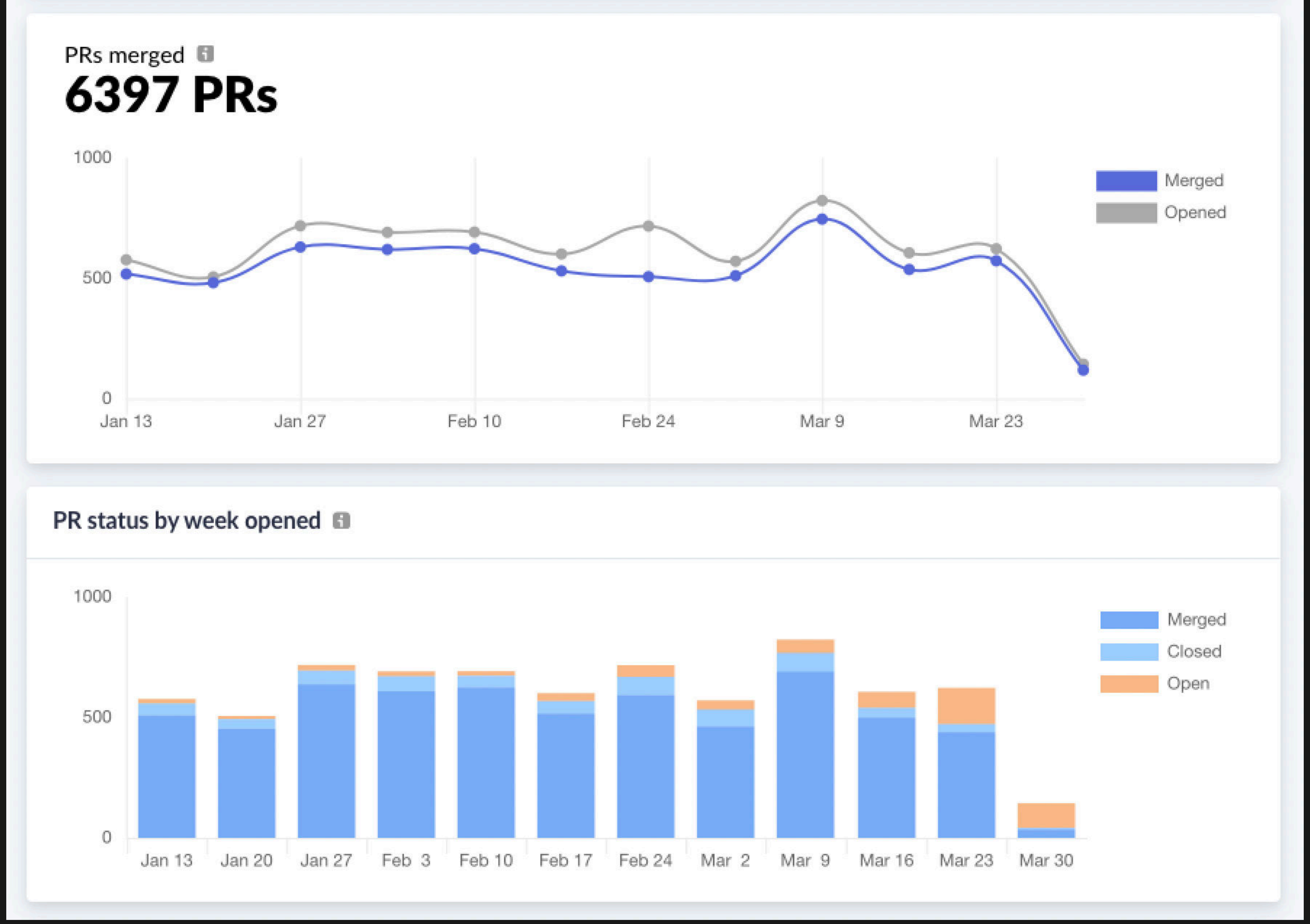




Breakdown of merge time

Average merge time

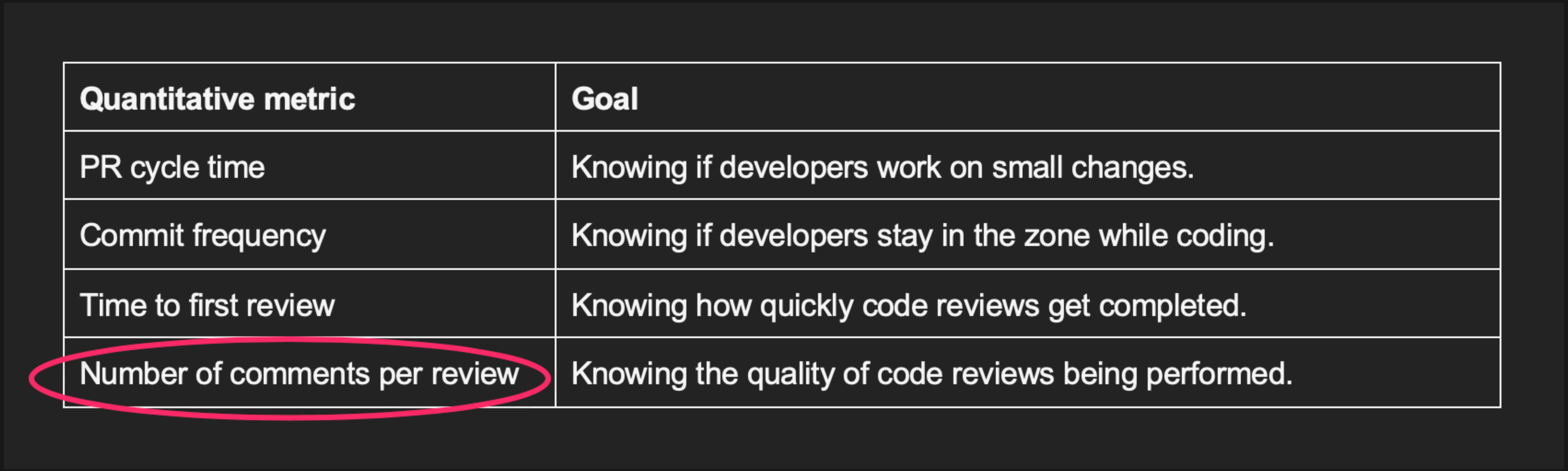
**5 days**



PRs merged

**6397 PRs**

PR status by week opened

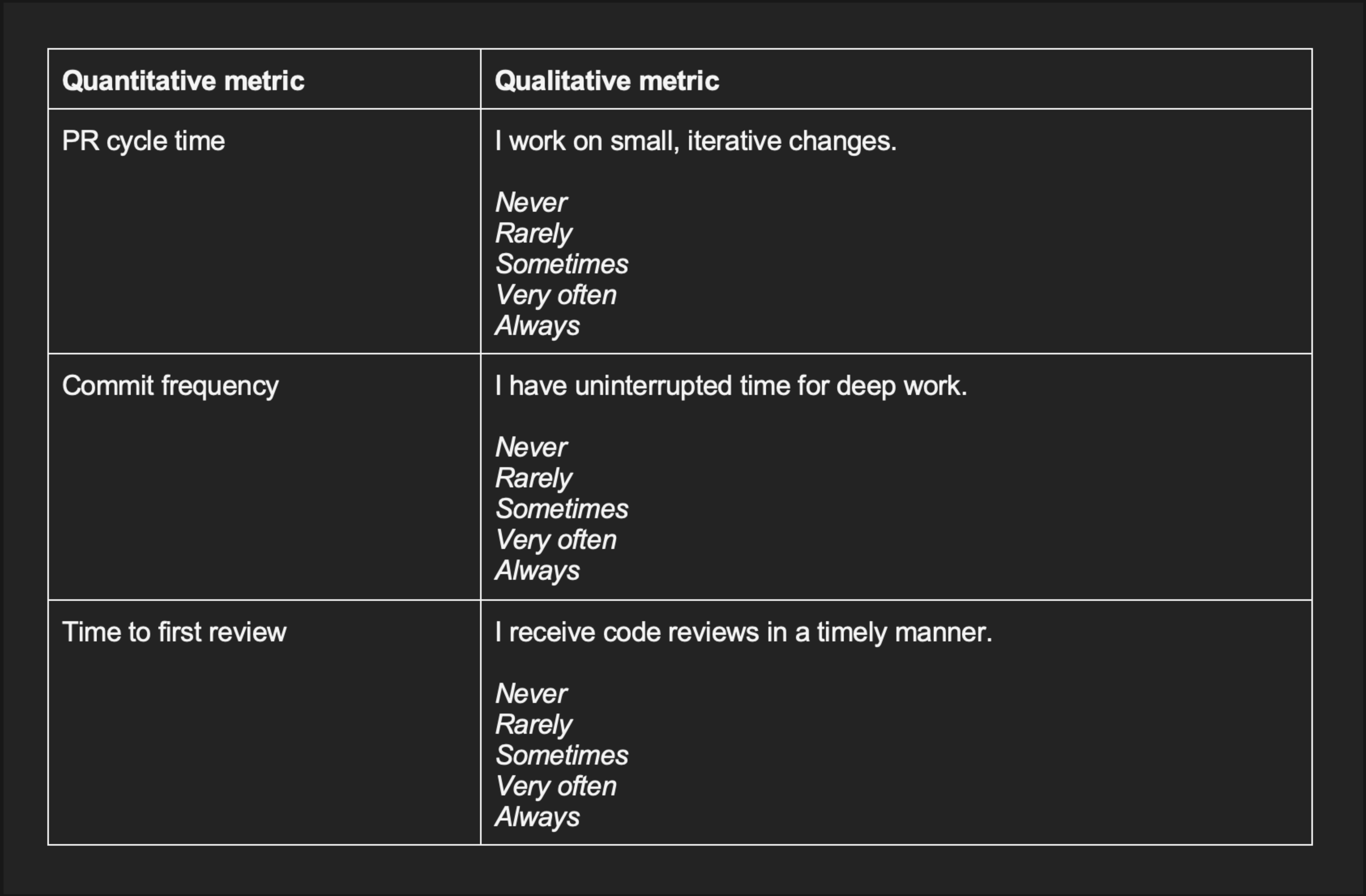


| Quantitative metric | Goal |
|---|---|
| PR cycle time | Knowing if developers work on small changes. |
| Commit frequency | Knowing if developers stay in the zone while coding. |
| Time to first review | Knowing how quickly code reviews get completed. |
| Number of comments per review | Knowing the quality of code reviews being performed. |

What if we just asked developers…

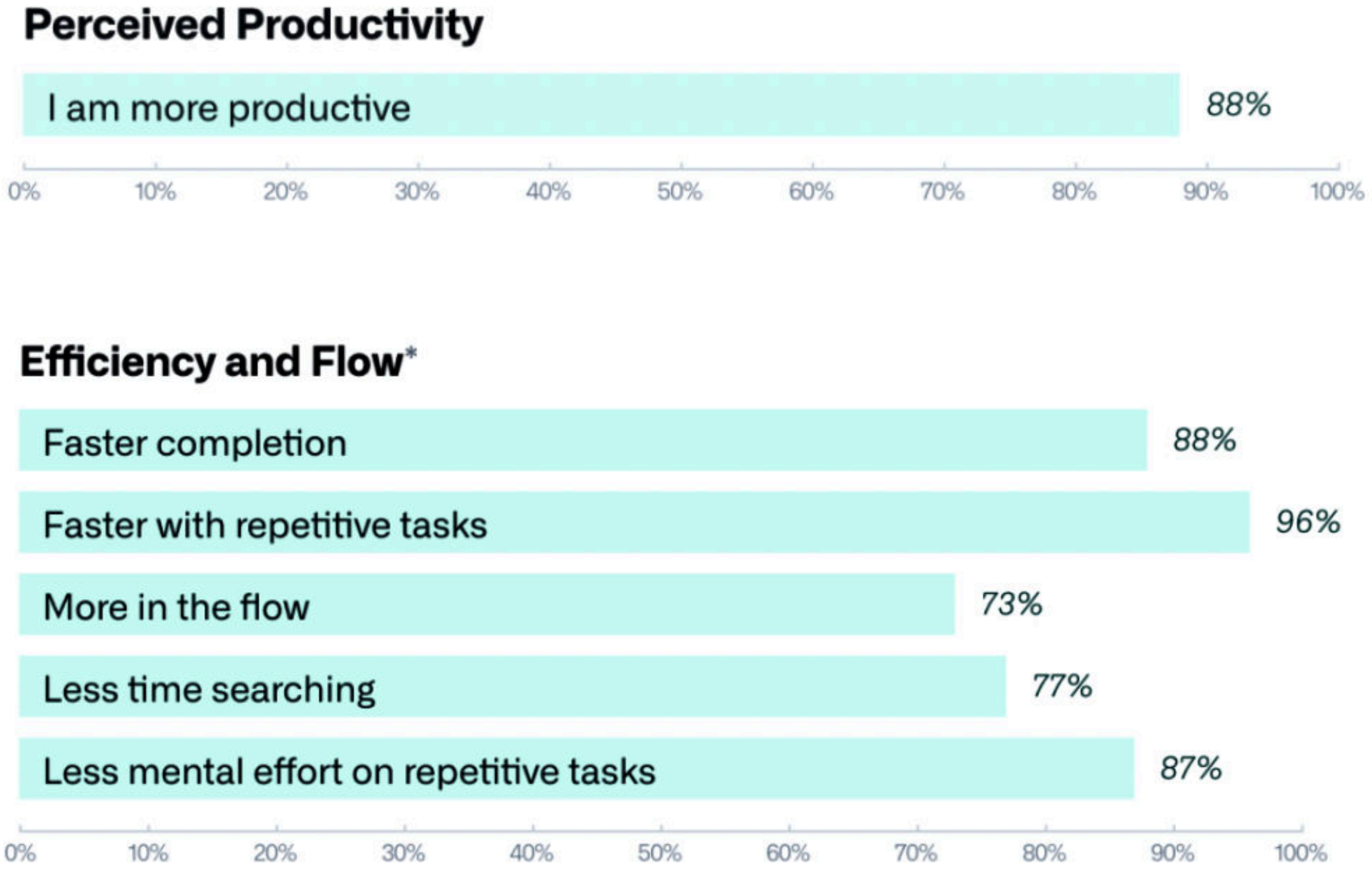
What if we just asked developers…

a.k.a. qualitative metrics



| **Quantitative metric** | **Qualitative metric** |
|---|---|
| PR cycle time | I work on small, iterative changes.<br>*Never*<br>*Rarely*<br>*Sometimes*<br>*Very often*<br>*Always* |
| Commit frequency | I have uninterrupted time for deep work.<br>*Never*<br>*Rarely*<br>*Sometimes*<br>*Very often*<br>*Always* |
| Time to first review | I receive code reviews in a timely manner.<br>*Never*<br>*Rarely*<br>*Sometimes*<br>*Very often*<br>*Always* |

# When using GitHub Copilot...



Perceived Productivity

Efficiency and Flow*

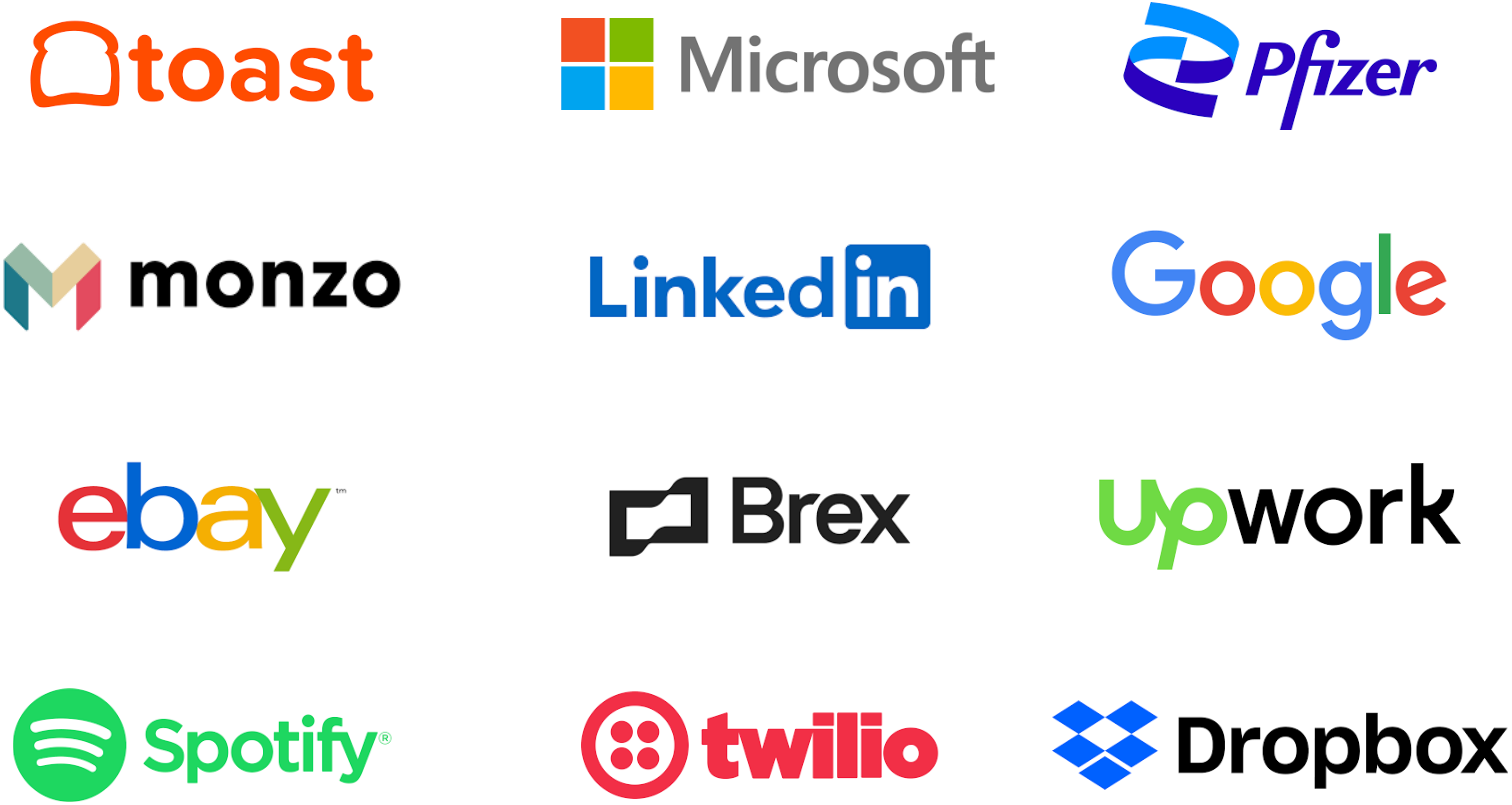
CHAPTER 14

# WHY USE A SURVEY

Now that we know our survey data can be trusted—that is, we have a reasonable assurance that data from our well-designed and well-tested psychometric survey constructs is telling us what we think it's telling us—why would we use a survey? And why should anyone else use a survey? Teams wanting to understand the performance of their software delivery process often begin by instrumenting their delivery process and toolchain to obtain data (we call data gathered in this way "system data" throughout this book). Indeed, several tools on the market now offer analysis on items such as lead time. Why would someone want to collect data from surveys and not just from your toolchain?

There are several reasons to use survey data. We'll briefly present some of these in this chapter.

1. Surveys allow you to collect and analyze data quickly.
2. Measuring the full stack with system data is difficult.
3. Measuring completely with system data is difficult.
4. You can trust survey data.
5. Some things can only be measured through surveys.


toast
Microsoft
Pfizer
monzo
Linked in
Google
ebay tm
Brex
upwork
Spotify®
twilio
Dropbox

“Surveys help you measure things that are in principle not measurable objectively.”

Ciera Jaspan, Engineering Productivity Research at Google

“Qualitative metrics are your highest coverage information.”

Max Kanat-Alexander, Principal Engineer at LinkedIn

> “The human mind has remarkable advantages over mechanical measurements for assessing complex and ambiguous situations.”

Douglas W. Hubbard, *How to Measure Anything*

8. **The Psychology of Survey Response**

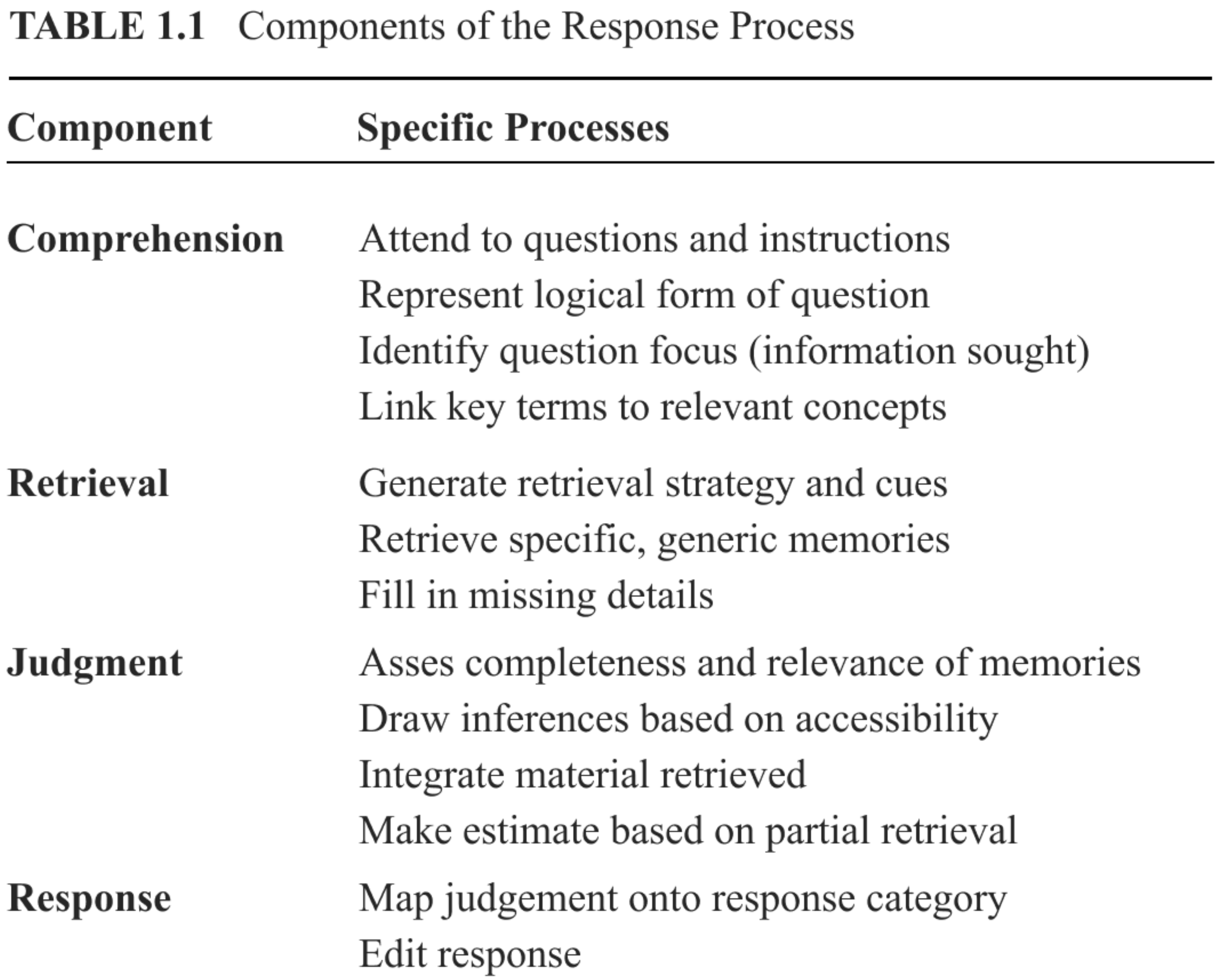


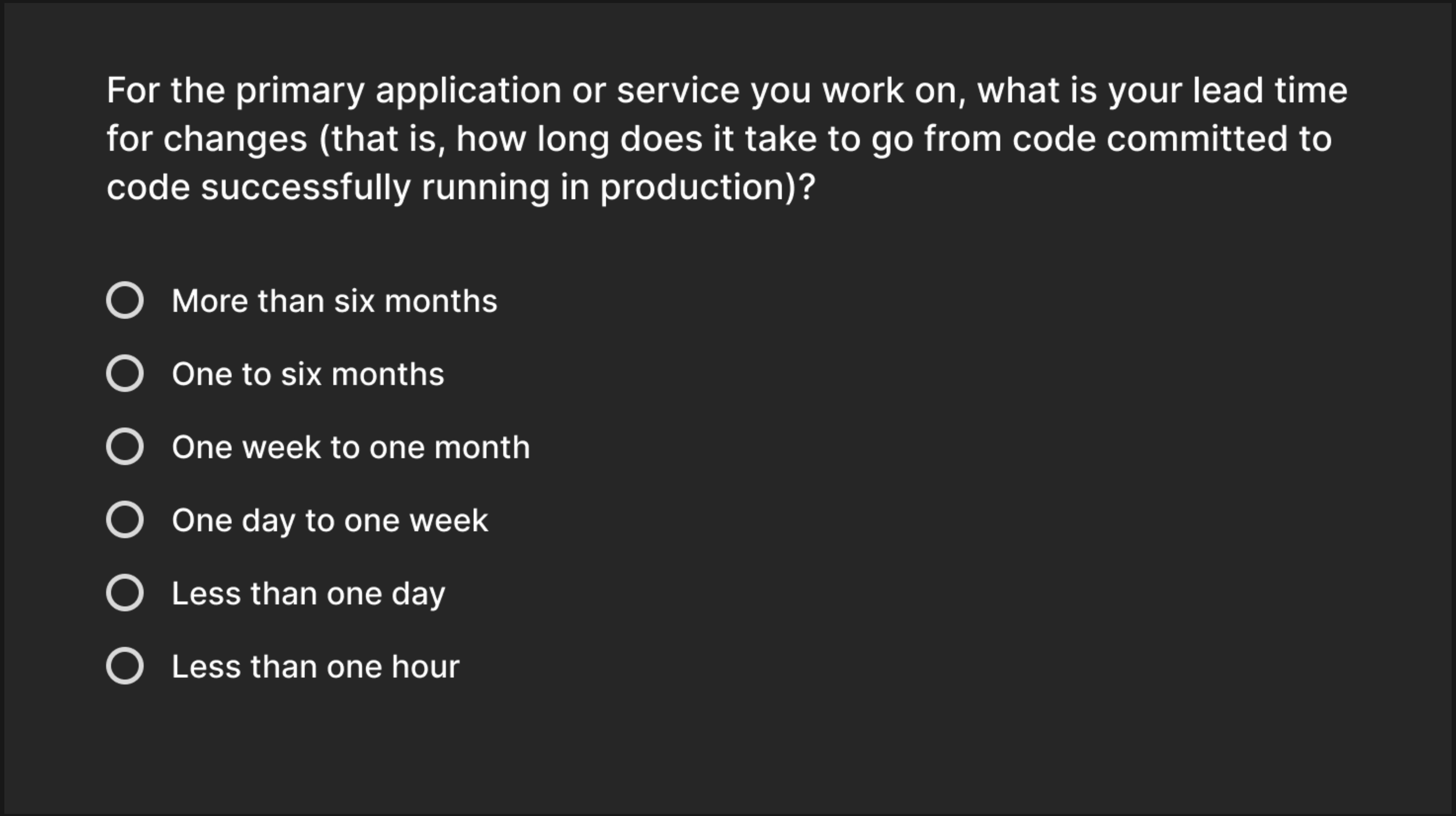
**TABLE 1.1** Components of the Response Process

| Component | Specific Processes |
|---|---|
| **Comprehension** | Attend to questions and instructions<br>Represent logical form of question<br>Identify question focus (information sought)<br>Link key terms to relevant concepts |
| **Retrieval** | Generate retrieval strategy and cues<br>Retrieve specific, generic memories<br>Fill in missing details |
| **Judgment** | Asses completeness and relevance of memories<br>Draw inferences based on accessibility<br>Integrate material retrieved<br>Make estimate based on partial retrieval |
| **Response** | Map judgement onto response category<br>Edit response |

“When we first started our survey, there was a lot of selling to execs like, ‘this isn't just people's opinions, this is actually valuable data.’”

Collin Green, Engineering Productivity Research at Google

# Myth: Survey data is purely subjective



For the primary application or service you work on, what is your lead time for changes (that is, how long does it take to go from code committed to code successfully running in production)?

○ More than six months

○ One to six months

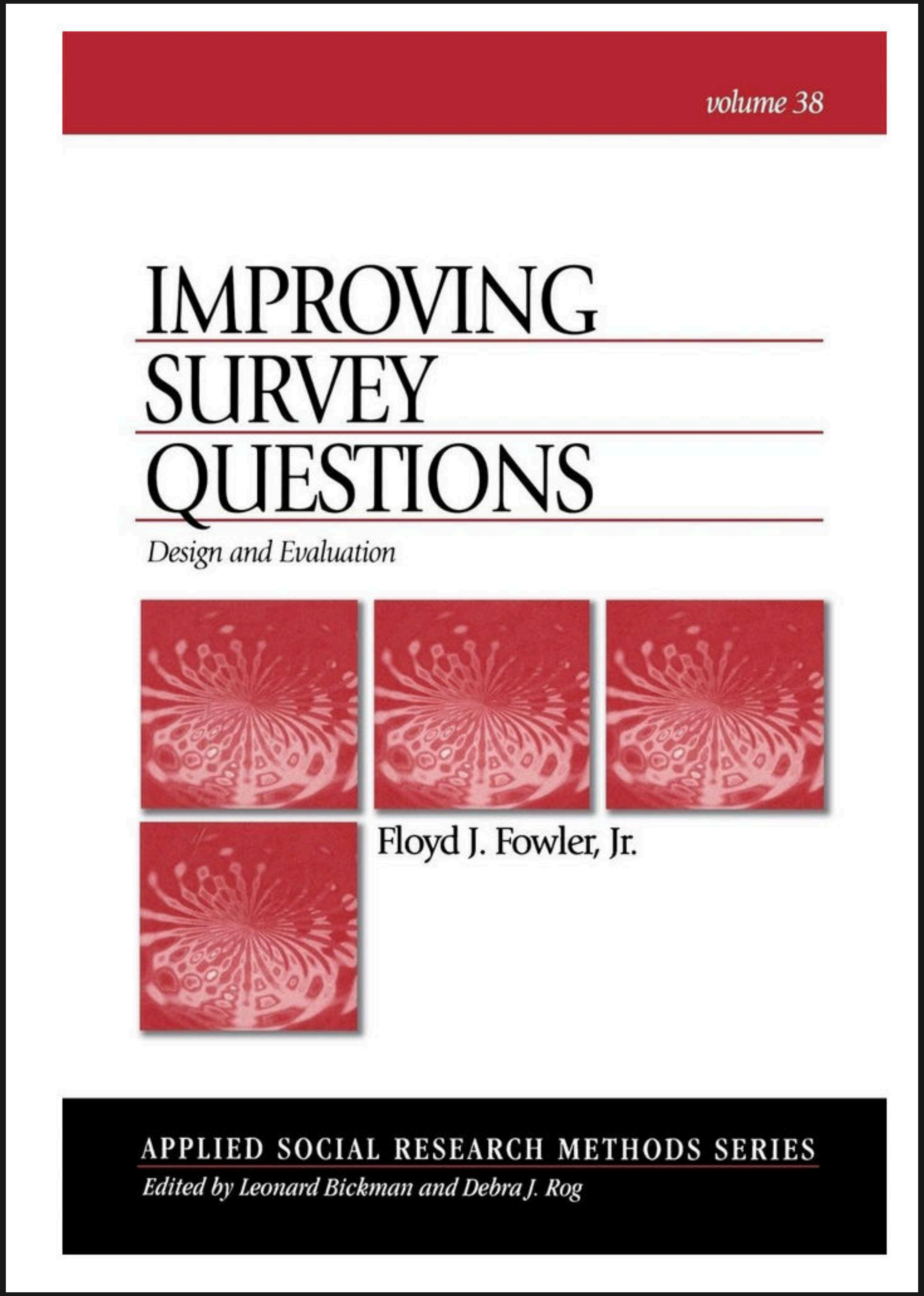
○ One week to one month

○ One day to one week

○ Less than one day

○ Less than one hour

# Myth: Survey data is unreliable



volume 38

# IMPROVING SURVEY QUESTIONS

*Design and Evaluation*

Floyd J. Fowler, Jr.

APPLIED SOCIAL RESEARCH METHODS SERIES

*Edited by Leonard Bickman and Debra J. Rog*

| | Pros | Cons |
|---|---|---|
| **Quant metrics** | • Easy to measure<br>• Objective | • Incomplete<br>• Lacks context |
| **Qual metrics** | • Holistic<br>• Tells you “why” | • Difficult (design, participation, etc.) |