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}
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.²,²⁰
### FIGURE 1: EXAMPLE METRICS

<table>
<thead>
<tr>
<th>LEVEL</th>
<th>SATISFACTION &amp; WELL-BEING</th>
<th>PERFORMANCE</th>
<th>ACTIVITY</th>
<th>COMMUNICATION &amp; COLLABORATION</th>
<th>EFFICIENCY &amp; FLOW</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDIVIDUAL One person</td>
<td>• Developer satisfaction</td>
<td>• Code review satisfaction</td>
<td>• Number of code reviews completed</td>
<td>• Code review score (quality or thoughtfulness)</td>
<td>• Code review timing</td>
</tr>
<tr>
<td></td>
<td>• Retention</td>
<td>• Code review velocity</td>
<td>• Coding time</td>
<td>• PR merge times</td>
<td>• Productivity perception</td>
</tr>
<tr>
<td></td>
<td>• Satisfaction with code reviews assigned</td>
<td>• # Commits</td>
<td>• Lines of code</td>
<td>• Quality of meetings</td>
<td>• Lack of interruptions</td>
</tr>
<tr>
<td></td>
<td>• Perception of code reviews</td>
<td></td>
<td></td>
<td>• Knowledge sharing, discoverability (quality of documentation)</td>
<td></td>
</tr>
<tr>
<td>TEAM OR GROUP People that work together</td>
<td>• Developer satisfaction</td>
<td>• Code review satisfaction</td>
<td>• # Story points completed</td>
<td>• PR merge times</td>
<td>• Code review timing</td>
</tr>
<tr>
<td></td>
<td>• Retention</td>
<td>• Code review velocity</td>
<td>• Story points shipped</td>
<td>• Quality of meetings</td>
<td>• Handoffs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• # Commits</td>
<td></td>
<td>• Knowledge sharing or discoverability (quality of documentation)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Code review timing</td>
<td></td>
</tr>
<tr>
<td>SYSTEM End-to-end work through a system (like a CD pipeline)</td>
<td>• Satisfaction with engineering system (e.g., CI/CD pipeline)</td>
<td>• Code review satisfaction</td>
<td>• Frequency of deployments</td>
<td>• Knowledge sharing, discoverability (quality of documentation)</td>
<td>• Code review timing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Code review velocity</td>
<td>• Code review (acceptance rate)</td>
<td></td>
<td>• Velocity flow through the system</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Customer satisfaction</td>
<td>• Customer satisfaction rate</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Reliability (uptime)</td>
<td>• Reliability (uptime)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1 Use these metrics with (even more) caution — they can proxy more things.
<table>
<thead>
<tr>
<th></th>
<th>ICs define own productivity</th>
<th>Managers define team’s productivity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>8%</td>
<td>9%</td>
</tr>
<tr>
<td>P</td>
<td>35%</td>
<td>67% (†)</td>
</tr>
<tr>
<td>A</td>
<td>50%</td>
<td>21% (†)</td>
</tr>
<tr>
<td>C</td>
<td>24%</td>
<td>33%</td>
</tr>
<tr>
<td>E</td>
<td>38%</td>
<td>45%</td>
</tr>
<tr>
<td></td>
<td>ICs think managers define productivity</td>
<td>Managers define team’s productivity</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>S</td>
<td>5%</td>
<td>9%</td>
</tr>
<tr>
<td>P</td>
<td>37%</td>
<td>67% (*</td>
</tr>
<tr>
<td>A</td>
<td>53%</td>
<td>21% (*</td>
</tr>
<tr>
<td>C</td>
<td>19%</td>
<td>33%</td>
</tr>
<tr>
<td>E</td>
<td>12%</td>
<td>45% (*</td>
</tr>
</tbody>
</table>
“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
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
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
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
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.
<table>
<thead>
<tr>
<th><strong>Quantitative metric</strong></th>
<th><strong>Goal</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>PR cycle time</td>
<td>Knowing if developers work on small changes.</td>
</tr>
<tr>
<td>Commit frequency</td>
<td>Knowing if developers stay in the zone while coding.</td>
</tr>
<tr>
<td>Time to first review</td>
<td>Knowing how quickly code reviews get completed.</td>
</tr>
<tr>
<td><strong>Number of comments per review</strong></td>
<td>Knowing the quality of code reviews being performed.</td>
</tr>
</tbody>
</table>
What if we just asked developers...
What if we just asked developers…

a.k.a. qualitative metrics
<table>
<thead>
<tr>
<th>Quantitative metric</th>
<th>Qualitative metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>PR cycle time</td>
<td>I work on small, iterative changes.</td>
</tr>
<tr>
<td></td>
<td><em>Never</em></td>
</tr>
<tr>
<td></td>
<td><em>Rarely</em></td>
</tr>
<tr>
<td></td>
<td><em>Sometimes</em></td>
</tr>
<tr>
<td></td>
<td><em>Very often</em></td>
</tr>
<tr>
<td></td>
<td><em>Always</em></td>
</tr>
<tr>
<td>Commit frequency</td>
<td>I have uninterrupted time for deep work.</td>
</tr>
<tr>
<td></td>
<td><em>Never</em></td>
</tr>
<tr>
<td></td>
<td><em>Rarely</em></td>
</tr>
<tr>
<td></td>
<td><em>Sometimes</em></td>
</tr>
<tr>
<td></td>
<td><em>Very often</em></td>
</tr>
<tr>
<td></td>
<td><em>Always</em></td>
</tr>
<tr>
<td>Time to first review</td>
<td>I receive code reviews in a timely manner.</td>
</tr>
<tr>
<td></td>
<td><em>Never</em></td>
</tr>
<tr>
<td></td>
<td><em>Rarely</em></td>
</tr>
<tr>
<td></td>
<td><em>Sometimes</em></td>
</tr>
<tr>
<td></td>
<td><em>Very often</em></td>
</tr>
<tr>
<td></td>
<td><em>Always</em></td>
</tr>
</tbody>
</table>
When using GitHub Copilot...

**Perceived Productivity**
- I am more productive: 88%

**Efficiency and Flow**
- Faster completion: 88%
- Faster with repetitive tasks: 96%
- More in the flow: 73%
- Less time searching: 77%
- Less mental effort on repetitive tasks: 87%
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.
“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

<table>
<thead>
<tr>
<th>Component</th>
<th>Specific Processes</th>
</tr>
</thead>
</table>
| Comprehension |Attend to questions and instructions  
Represent logical form of question  
Identify question focus (information sought)  
Link key terms to relevant concepts |
| Retrieval |Generate retrieval strategy and cues  
Retrieve specific, generic memories  
Fill in missing details |
| Judgment |Asses completeness and relevance of memories  
Draw inferences based on accessibility  
Integrate material retrieved  
Make estimate based on partial retrieval |
| Response |Map judgement onto response category  
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
<table>
<thead>
<tr>
<th></th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Quant metrics</strong></td>
<td>- Easy to measure</td>
<td>- Incomplete</td>
</tr>
<tr>
<td></td>
<td>- Objective</td>
<td>- Lacks context</td>
</tr>
<tr>
<td><strong>Qual metrics</strong></td>
<td>- Holistic</td>
<td>- Difficult (design,</td>
</tr>
<tr>
<td></td>
<td>- Tells you “why”</td>
<td>participation, etc.)</td>
</tr>
</tbody>
</table>