Developer Thriving:

The four factors that drive Software Developer Productivity across Industries

Authors: Cat Hicks, PhD., Carol S. Lee, PhD., Morgan Ramsey
The Developer Success Lab | devsuccesslab.com

2023

Suggested citation: Hicks, C., Lee, C. S., Ramsey, M. Developer Thriving: The four factors that drive Software Productivity across industries [research report]. The Developer Success Lab at Flow (2023). [link]

Table of Contents

EXECUTIVE SUMMARY

What helps developers thrive?

- Understanding how engineering investment leads to business impact is a critical challenge; organizations that succeed at this can maximize the value of engineering work. Maintaining software development velocity, and sustaining meaningful feedback loops between developer effort and impact, is central to success for organizations that rely on software work. Because of this, both understanding and then maintaining long-term developer productivity is a top priority for leaders, engineering managers, and developers themselves.

- In this report, the Developer Success Lab at Flow shares what we've learned about the key sociocognitive factors that lead to highly productive software teams. With three research projects and 1200+ real-world software developers, we used empirically validated factors from social science and software research to create a framework for *Developer Thriving* and *Visibility and Value,* both of which significantly predicted productivity. These factors are corroborated by situated examples from qualitative interviews with developers and managers. We also share what we've learned about how developers *measure* their productivity, reporting on the use of software metrics and its impact.

- We include recommendations based on both large-scale quantitative research and in-depth qualitative research, and key examples for how the factors in Developer Thriving may show up inside of organizations.

To create new technologies, developers must collaborate well on complex, iterative, and distributed code. Developers and their teams need to balance personal productivity, project constraints, organizational context, and business impact alongside pushing the boundaries on what code can do in the world. Against this complexity, some estimates of the overall success rates of software projects claim that the *majority* of software projects deliver late, deliver out of scope with planned budget, and fail to drive business impact (Reel, 1999; Verner et al., 2008).

Nevertheless, developers work together in code and make progress every day. Extensive research on how software engineers problem-solve in complex code paints a much more optimistic picture of software work. Developers are highly motivated to do cognitive knowledge work (e.g., Beecham et al., 2008), experts at triaging and learning through failure (e.g., Petre, 2009), and successfully create technologies that are interacted with by nearly every human on earth. Understanding this problem-solving is key to maintaining this innovation.

The Developer Success Lab at Flow set out to study what factors underlie how developers and their teams at work maintain productivity, work deeply and collaboratively, and have real-world impact. This report summarizes three in-depth research studies with software teams on *how real-world developers and their teams achieve success*. Across quantitative data from **1282 developers**, and rich qualitative data from **15+ hours** of conversation in interviews and focus groups, we share our initial findings about how developers experience their working environments, how experiences of learning culture, agency, motivation, and belonging impact developer productivity, and the strategies that developers use to navigate these complex environments.

We introduce a key concept that we believe is foundational to software team success: Developer Thriving.

One of our key findings is that high-quality, sustainable software work is unlocked by the structural elements of iterative, collaborative problem-solving that allow developers to *thrive.* Drawing from important empirical research in human wellbeing, learning, and achievement, we created an original measure of *Developer Thriving:* a growth-oriented measure of developers' environments that captures whether developers have four key sociocognitive dynamics inside of their teams that enable knowledge work. In our study, Developer Thriving significantly predicted developers' productivity.

We introduce an important dimension that determines how software teams see their work's connection to business impact: Visibility & Value.

We also developed a new and original measure for understanding the larger organizational context around developers' individual work: how much developers feel that their work is visible to the right people (both teammates and leaders), and how much it feels valued. We see this perception as a missing element that bridges the gap between individual developer experience and how engineering organizations' systemic choices about their teams change developer behavior. We call this concept *Visibility and Value,* and it too was a significant predictor for *both* developers' productivity and Developer Thriving.

We describe how software teams interact with software metrics, and share the benefits of thoughtfully tracking parts of the coding process for both teams and individuals.

We then turned to asking developers about *measurement:* how do software teams think about tracking their progress, and do they see a benefit from it. While the use of software metrics is reported relatively unevenly across software teams, we find that bringing team-level metrics into rituals like sprint planning and project retrospectives boosts *Visibility* and *Developer Thriving*. Crucially, we find that this benefit is unlocked when measurement happens on the team level, ties directly to process and effort, and is aligned with developers' perceptions of how the organization values their work.

Finally, we explore these themes in rich, qualitative interviews and focus groups.

Stories and insights directly from developers reinforced the *Developer Thriving cycle*, and helped us understand how *Visibility* and *healthy measurement* was experienced in real developers' work lives. Developers shared key themes around what was working well and what wasn't in their environments, and we surfaced examples of all key constructs in Developer Thriving, along with barriers and friction points that disrupt these good problem-solving elements. In particular, developers noted that organizational transparency and a reliable expectation of authentic recognition helped motivate their problem-solving and ensure they were solving the right things. Managers shared stories of advocating for developers, and also the frictions in frequently feeling overwhelmed or solely responsible for making their teams' work visible.

Selected Recommendations	
Finding	Recommendation
Developer Thriving unlocks productivity across roles and industries.	**Developers** should be mindful about how they support their teammates' learning, belonging, and other key factors in crucial collaboration moments such as code reviews and team retrospectives **Managers** seeking to increase productivity should diagnose lacks of and advocate for investments in Developer Thriving factors
Developers benefit when their work is seen and recognized by their teammates, managers, and organization, but struggle to find and maintain this visibility.	**Managers** should commit time to learning reports as individual people; well enough to understand how to effectively advocate on their behalf **Managers** should seek out opportunities to credit developers directly in business impact, such as representation in demos, and recognition in launches **Leaders** should diagnose whether there are "visibility gaps" inside of their engineering organizations, paying special attention to teams or types of engineering work that do not get shared broadly **Organizations** should invest in systems that explicitly recognize and reward teams for the technical progress they make, particularly work that was unexpectedly challenging, required new skills, or fixed long-standing problems
Using software metrics in a thoughtful and healthy way can increase visibility into developers' technical work.	**Developers** should integrate tracking their own code processes in a way that helps them contextualize work over time (e.g., the balance of code reviews they do, the types of tickets consuming time, the cadence of work) into a reflective practice. Avoid using a single measure, and start with documenting success and effort to reveal "easy wins" for increasing recognition of technical progress, and justifying learning investments **Managers** should consider bringing reflection on over-time metrics into 1x1 conversations with developers and using overall team metrics during planning processes. Managers should highlight and explain when success metrics need to change for a different context **Organizations** should assess whether engineering effort is evaluated with metrics that drive change and decisions, and ensure that long-term impact is tracked over time and across engineering projects

FULL RESEARCH REPORT

Introducing a Framework for Developer Thriving

To study productivity and success with real-world developers, we took inspiration from the science of developer productivity. Recent waves of software research argue for a new grounding in human-centered, evidence-based science by 1) learning from foundational evidence about what truly improves problem-solving during coding and software work, 2) doing research directly on the real-world experiences of modern software teams, and 3) avoiding major misconceptions in measuring productivity, such as defining developer productivity only as crude output measures such as lines of code, or setting a single metric goal and using it as a threshold evaluate all software work regardless of differing contexts and needs (e.g., Bouwers et al., 2012; 2013; Forsgren et al., 2021; Grieler et al., 2022; Sadowski, & Zimmermann, 2019; Storey et al., 2021; Storey, Houck, & Zimmermann, 2022).

We believe that this grounding provides a way out of the two *measurement traps* that leaders experience when attempting to increase organizational developer productivity: **1)** fixating on surface definitions of productivity and measuring and incentivizing the wrong things, or **2)** becoming paralyzed by complexity and context and measuring and incentivizing nothing.

The SPACE framework, which characterizes *developer productivity* in terms of satisfaction and wellbeing, performance, activity, communication, and efficiency is one impactful recent summary (Forsgren et al., 2021; Storey et al., 2021). The SPACE framework provides an example of systematically broadening "productivity" definitions with dimensions such as job satisfaction. Nevertheless, while the SPACE framework includes satisfaction as a key piece of productivity, understanding *what* real developer satisfaction is remains an important question. Developers, managers, and engineering may still struggle to understand *how* to begin breaking down and diagnosing how developers' sociocognitive experiences connect to more robust cycles of productivity for teams.

Developer Thriving builds on the known connection between developer satisfaction and productivity, but helps to answer *what* drives satisfaction in the first place. In creating this construct, we took a survey of evidence across rich research areas in human achievement and wellbeing. From this, we developed four original measures to capture key elements of developer experience. Each element draws on robust theories which have proven directly impactful to human problem-solving and achievement, and adapts them to software teams (Table 1). These elements are *agency, motivation & self-efficacy, learning culture,* and *support & belonging*. A "thriving" software team has each element in their environment.

We believe the sociocognitive elements that create *Developer Thriving* are impactful because they create "virtuous cycles": positive beliefs, perceptions, and expectations about code work and problem solving.

These cycles work to reinforce developers' sense of progress and problem-solving *even and especially* when developers encounter difficulty, friction, and failure. Across intervention science in human behavior, these positive metacognitive beliefs, perceptions, and environmental factors have been found to drive human achievement change (e.g., Yeagar et al., 2013). Organizations can either enhance or subvert these important cycles: when teams and organizations put effort into creating a positive problem-solving culture, it sustains long-term achievement, iterative improvement, and reflective, collaborative problem-solving.

The "virtuous cycles" unlocked by a culture of Developer Thriving provide the most effective way out of the two measurement traps in "developer productivity." Good problem-solving elements *do* scale across contexts, and meaningfully lead to productivity. Learning to see these cycles gives engineering leaders and software teams a place to begin diagnosing the barriers to thriving, and make visible the most valuable practices which protect it. In the following three studies, we'll test the impact of Developer Thriving and Visibility, explore how managers' and developers' experiences inside of organizations shape their success and productivity, and share what we've learned about how healthy measurement can benefit software teams.

The Behavioral Science behind Developer Thriving		
Agency	A developer is: 1) able to voice disagreement with team definitions of success 2) has a voice in how their contributions are measured	Billett, 2011 Gobeli et al., 1998 Hicks, 2022 Meyer et al., 2019
Motivation & Self-Efficacy	A developer is: 1) motivated when working on code at work 2) can see tangible progress most of the time 3) is working on the type of code work they want to work on 4) is confident that even when working in code is unexpectedly difficult, they will solve their problems	Bandura & Adams, 1977 Kim et al., 2023 Robinson et al., 2019 Sherer, 1982
Learning Culture	A developer is: 1) learning new skills as a developer 2) able to share the things they learn at work	Hicks, 2022 Scott & Ghinea, 2013 Luxton-Reilly et al., 2018
Support & Belonging	A developer is: 1) supported to grow, learn, and make mistakes by their team 2) agrees they are accepted for who they are by their team	Anderson-Butcher & Conroy, 2002 Pardede, Gausel, & Høie, 2021 Rattan et al., 2018 Wilson et al., 2010

Table 1.

Research Implementation & Methods

The Developer Success Lab takes a *mixed-methods* approach to our research. Therefore, our approach was both quantitative (a large-scale survey of 1200+ developers across 12 industries) and qualitative (in-depth interviews and focus groups with 19 developers, ranging from junior ICs to managers, and representing diverse characteristics, backgrounds, and engineering areas). Combining these sources of insight allows us to test meaningful evidence about developer experience at scale, while also surfacing recommendations and lived experience directly from developers on their own thriving, barriers, and possibilities.

For our **quantitative** research, we opened an online survey to individual contributor (ICs) developers and software engineers responsible for technical code work in their role. Our survey was advertised publicly on various social media (e.g. twitter, facebook, mastodon, linkedin, and reddit), from researchers' personal social media accounts, and via direct emails to professional listservs of interest to developers. Our survey was also advertised inside of the Pluralsight Skills platform, embedded in developer-relevant content pages such as internal learning programs, and inside of the Pluralsight Flow platform, shown as a banner advertisement to professional developer users. In all cases, this survey advertisement was optional and not connected to user data on either of these platforms. See Appendix B for a copy of our consent form, which was provided on the initial survey eligibility page.

For our **qualitative** research, we recruited participants from two sources: internally recruiting full-time software developers at Pluralsight (interviews), and external recruiting from participants who had opted in to a follow-up after our survey (focus groups). For internal participants, we conducted 1-hour long semi-structured interviews focused around understanding how developers defined *success, productivity,* and the *barriers* they experienced to both. From these initial internal interviews, we identified major key themes, which we used to create semi-structured interview scripts for a series of 1-hour long focus groups with individual contributors and software team managers.

Key Research Questions

What **organizational factors** and experiences help to **motivate** developers when work is difficult? How do developers make a connection between their engineering effort and business impact?

How does visibility impact **productivity**? How does visibility impact **team-wide success**? How do developers measure their work, and what benefits do they see from **thoughtful measurement?**

How do developers **define success**? What elements do they think matter *most* to successful software work, and what do developers think their managers and teams see as successful?

Participant Consent & Privacy

In our research, we strive to follow best practices for social science research and human behavior data collection. Two key values for the Developer Success Lab are to provide informed consent to all participants prior to their participation, and to take precautionary measures to protect participants' privacy.

For example, we 1) restrict access to all raw participant data to Developer Success Lab researchers 2) anonymize across findings so that specific names, teams, and contexts aren't identifiable 3) only share quantitative data insights in aggregate 4) emphasize in multiple points during data collection that participants should only share what they feel comfortable sharing, and 5) maintain a "continual consent" practice with participants, meaning that participants can opt-out of research at any time during their participation, and 6) do not treat opt-out of identity disclosures as an exclusion criteria, meaning that we analyze data in a way that ensures participants who opt-out of sharing personal information such as demographics are still able to participate in other research questions, where possible.

This information was communicated to our participants through a consent form shared at the beginning of the quantitative survey, via email several days prior to the qualitative interviews and focus groups, and in verbal communication from researchers during our qualitative research.

Representation, Data Quality, and Designing for Quality of Response

Our quantitative research recruited 1409 individual contributor developers. Of the 1409 participants, 121 did not move past the first 2 questions of the survey and were thus dropped from our sample. Of the remaining 1288 participants, six were removed for writing identity-based discriminatory responses in our open text demographic fields. Our final sample consisted of 1282 participants (Figure 1). As a token of appreciation for participation, our research team made a donation to an open source software nonprofit, chosen based on participant voting.

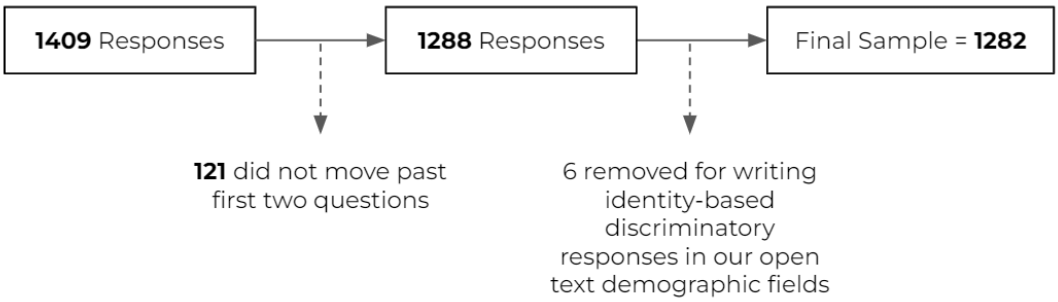

Fig 1. Flowchart of participant drop out and final sample size.

Throughout the survey, participants were allowed to leave any question blank. In order to center participant consent and opt-in, identity-based questions were also marked with [OPTIONAL] in the item description, and *"prefer not to respond"* was given as an explicit option. This resulted in a significant drop-off of participant response to many of the identity and firmographic questions, which is expected in this survey design. To reduce the risk of bots and multiple responses, we enabled bot detection and security scan monitoring on our survey.

In order to reduce response biases within-survey, we used a semi-randomized survey design. All participants answered key construct measures *before* being asked to answer measures that may influence their responses. For example, to avoid stereotype threat, participants rated their productivity *before* being asked to answer any questions about demographic characteristics, or about their sense of belonging (Spencer, Steele, & Quinn, 1999; Steele & Aronson, 1995). Within the key construct measures, the order of presentation was randomized to control for order effects.

During our qualitative research, participants were allowed to not answer questions, or to leave the conversation at any time. Researchers prompted any confusion, hesitation, or non-response with clarifying support such as, "there are no right or wrong answers." All qualitative participants participated in the full time of the research session, and none withdrew any content from their transcripts afterward.

Quantitative Measures		
Construct	**What it measured**	**Response format**
Perceived Productivity Rating (PPR)	An individual developer's productivity over the last month	Rating: 1-5 Likert scale ("Not at all Productive" - "Extremely Productive")
Team Metrics Use (TMU).	Whether an individual developer reported that their team used software metrics	Categorical: Yes, No, Sometimes
Healthy Metrics Use (HMU)	An original measure of team metrics use and the extent to which developers agree that their teams are measuring the "right" things.	Rating: 1-3 Likert scale ("Yes", "Sometimes", "No") and a binary scale ("Yes", "No"). Scores averaged.
Developer Thriving Scale (DTS)	An original measure of four primary factors that we propose as important to developer thriving, drawing from key research in wellbeing, motivation, and learning sciences: motivation, belonging, learning culture, and agency	Rating: 1-5 Likert scale. Scores averaged.
Visibility and Value Questionnaire (VVQ)	An original measure of perceived visibility and value of an individual developer's technical work, visibility by teammates and by manager, and sense of value overall.	Rating: 1-5 Likert scale. Scores averaged.
Impact of Measuring Code Work (IMCW)	An original measure of how developers agree with the usefulness of software metrics in 1) helping a team to work better, such as helping with trade-offs in decision making and 2) helping an individual to work better, such as understanding one's own productivity 4) agreement that tracking & quantifying coding work increases teammates' & manager's visibility into a developers' technical work, and 5) the progress of technical work overall Items adapted from Wallace & Sheetz (2014)	Rating: 1-5 Likert scale. Scores averaged.

Table 2. How we measured key questions on the quantitative survey

Creating our Quantitative Measures

Throughout the quantitative measures on this survey, participants answered using a Likert scale. Unless otherwise indicated (see Table 2), the scale included a neutral midpoint; scores greater than 3 indicate agreement, while scores less than 3 indicate disagreement. Where multiple items are used in a subscale, scores were averaged.

Perceived Productivity Rating (PPR). There is no standard measure for developer productivity (Sadowski & Zimmerman, 2009) and developers define productivity in multiple ways; software research has therefore frequently used self-report ratings of productivity (Meyer et al., 2017). To operationalize this complex concept, we also chose to ask developers to rate their own productivity over a recent period of time. This approach allows us to let developers summarize across their complex contexts, different industry paces of work, and working environments. In our study, the *PPR* is a self-report, single-item measure adapted from a similar rating utilized

by Cheng and colleagues (2022). In keeping with our aim of reducing within-survey response effects, this question was shown first to reduce biases that might arise from respondents' reflecting on questions about belonging, measurement and software metrics.

Healthy Metrics Use (HMU). Healthy metrics use was operationalized as a two-item composite rating created for the purpose of this study. The first item asked participants to report their team's use of metrics. The second item asked participants to report if they believed their team used the "right" metrics for their team and agreed that "they measure the right things." The two scores were averaged to create a single composite score.

Developer Thriving Scale (DTS). The DTS is a ten-item measure created for the purpose of this study, abbreviated in order to be accessible to participants at scale in an applied research setting (see Appendix D for more details). The measure draws from models of health and psychology to identify four primary factors of satisfaction: motivation, belonging, learning culture, and agency. The items for each factor are adapted from empirically validated psychological measures of these constructs, which we present in the section below (Table 5). The measure had good internal consistency in our sample (α = .86).

Visibility and Value Questionnaire (VVQ). The VVQ is a three-item measure created for the purpose of this study. The measure draws from previous research indicating that recognizing and valuing employees' work predicts employee satisfaction and asks respondents to rate the extent to which they believe their technical work is visible and valued by teammates and managers. The measure had good internal consistency in our sample (α = .83).

Impact of Measuring Code Work (IMCW). Across five items, participants rated the perceived usefulness and impact of software metrics, defined as 1) helping them understand teammates' work and make trade-off decisions, and 2) helping them understand their own productivity 3) increasing visibility and 4) helping them make progress in technical work. This measure was only shown to the subset of participants who had indicated they sometimes or always used *either* team or individual software metrics.

Demographics & Firmographics

Participants could choose to provide information on their team size, team type, industry, organization size, engineering area, and percent of time spent writing code. Participants also could choose to provide information on their years of experience, where they learned to code, gender identity, sexual orientation, race, education, and country of residence. The purpose of these items was to accurately represent our sample, as well as to control for any significant effects of specific demographic characteristics on our primary variables of interest.

Creating our Qualitative Measures

Qualitative Script Development

Qualitative interview scripts were designed with feedback from internal software developer stakeholders and designed in conjunction with our survey items.
The sequence of questions in our script was also intentionally designed. We began our discussion by asking broad questions around definitions to better understand developers' initial descriptions of "success." Next, we probed deeper into how developers understood the path to success operating inside of their organizations. We asked participants to reflect on moments of collaboration, friction, or doubt, and how peers and/or managers served as barriers or bridges to achieving that success. We further unpacked these instances by asking developers for examples of feeling valued, appreciated, or understood during those collaborations. We wanted to see if these moments had an impact on the participants' workflow, motivation, or their definition of "success."

For the focus groups, we aimed to gather participants in groups matched in individual contributor or manager roles and years in role (see Table 3 for a summary of all qualitative participants). Previous research shows that when focus group participants have shared characteristics, they are more likely to have shared experiences, which can increase experiences of comfort, validation, and belonging. This subsequently increases participant openness and experiences of safety (Roller & Lavrakas, 2015). We also aimed for group sizes of 4 or fewer participants. This was to combat biases that are more likely to occur in larger groups of people, such as group-think or one person dominating the conversation. Two researchers were present in each group; one to primarily take notes and the other to lead the discussion. We found that the 2:4 ratio was large enough to allow the participants space to freely converse amongst themselves, but small enough for the researchers to guide the conversation or engage individual participants who may have been less vocal. However, in Focus Group 3, only one participant (P6) was available on the morning of the session; therefore this session was treated as a semi-structured individual interview, using the same script as the other focus groups.

Qualitative Participants		
	Demographics	**Firmographics**
Focus Group 1		
P1	Male, South/Southeast Asian	Individual Contributor, Full-stack developer, 3-5 years in role, Media/Entertainment
P2	Male, East Asian	Individual Contributor, Backend developer, 3-5 years in role, Technology
P3	Male, South/Southeast Asian	Manager, Backend/Database admin, 3-5 years in role, Financial Services
Focus Group 2		
P4	Male, chose not to disclose further demographics	Manager, Full-stack developer, 3-5 years in role, Retail/e-Commerce
P5	Male, White	Manager/Leader, 10+ years in role, Technology
Focus Group 3		
P6	Male, White	Individual Contributor, Backend developer, 0-1 years in role, Media/Entertainment
Interviews (9 participants). For internal participant confidentiality, fewer demographics were surveyed in our interviews.		
5 Men 4 Women	8 US 1 India	Senior Manager (1) Principal Software Engineer (2) Senior Software Engineer (1) Software Engineer (4) Contractor (1)

Table 3. Qualitative Research Participants

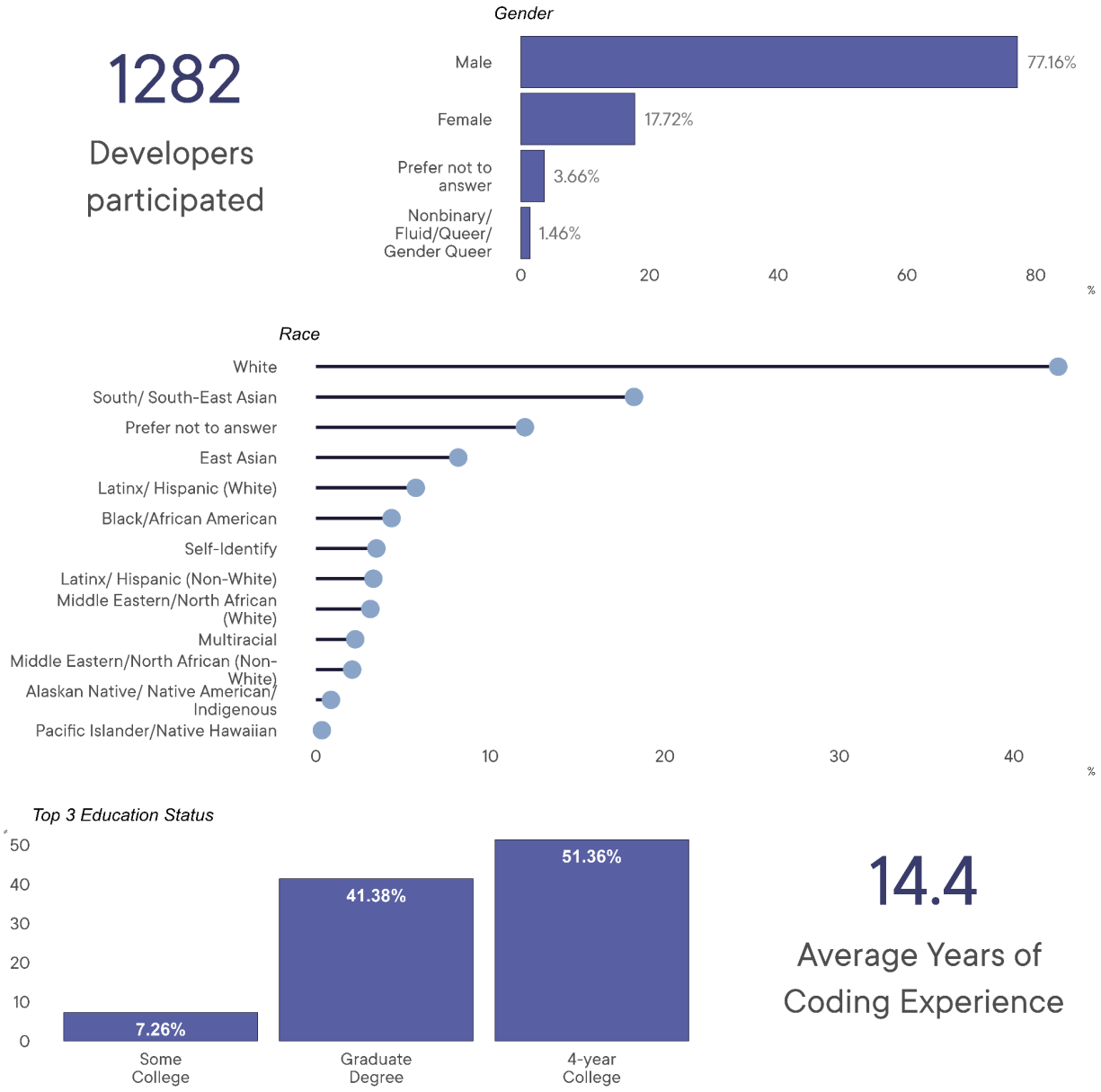

Fig. 2: A Summary of Quantitative Participant Demographics

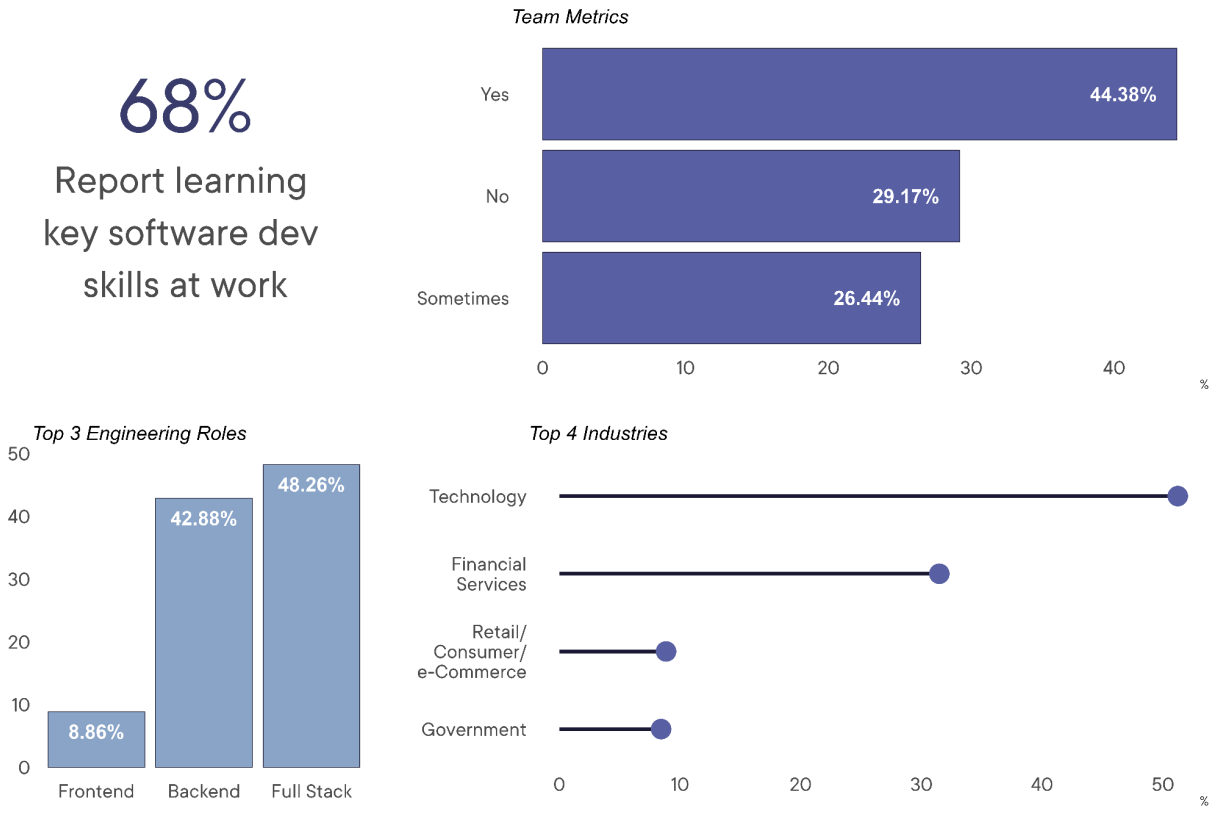

Fig. 3: A Summary of Quantitative Participant Firmographics

The Developer Thriving Scale and Productivity

Average scores shown by role and industry. Subgroups smaller than 3 were excluded.

	Average Developer Thriving Scale Score	Average Perceived Productivity Rating
Technology (*n* = 249)		
Backend (*n* = 84)	4.15	3.55
Frontend (*n* = 24)	4.36	2.67
Full Stack (*n* = 91)	4.29	3.55
Financial Services (*n* = 153)		
Backend (*n* = 63)	4.28	3.61
Full Stack (*n* = 56)	4.26	3.64
Retail/ Consumer/ e-Commerce (*n* = 43)		
Backend (*n* = 16)	3.90	4.00
Full Stack (*n* = 18)	4.41	3.50
Government (*n* = 41)		
Backend (*n* = 8)	4.06	3.33
Frontend (*n* = 3)	3.92	3.00
Full Stack (*n* = 23)	4.05	3.27
Unreported (*n* = 650)	4.32	3.42

Table 4. Average scores for Developer Thriving and Productivity, shown by industry and engineering role

Study 1: Developer Thriving and Visibility create a holistic ecosystem that unlocks Productivity

Study 1 High Level Summary

Developer Thriving (developers' overall ratings of their agency, motivation, learning culture, and sense of belonging) was the strongest predictor of Productivity.

In a serial mediation model, we found that Visibility & Value indirectly increased Productivity by directly increasing Developer Thriving.

Additionally, Healthy Team Metrics Use indirectly increased both Developer Thriving and Productivity, by directly increasing Visibility and Value.

Are developers thriving?

When looking at our entire analytic sample (*N* = 1282), we found that developers reported positive levels of agency, belonging, motivation, and learning culture (Table 4). Overall, developers scored the highest on learning culture and the lowest on agency. Taken together, the findings show that developers are thriving overall, but that there may be a slight imbalance between the factors of developer thriving, though this difference was not statistically significant.

One concern with examining these constructs might be if our sample reported systematically different scores on Developer Thriving constructs because of differences in engineering work or context–for example, perhaps developers score very highly in agency in one industry, but not another. However, we found no large patterns of difference in developers' thriving scores by years of coding, type of engineering role, and industry, indicating that the measures in developer thriving can be used across diverse types of engineering work and different organizations. Then, we conducted a correlation analysis and found that thriving scores, developers' productivity scores, their visibility, *and* healthy metrics use were all positively associated (Fig 5). In particular, Developer Thriving and whether developers believed their work was visible to and valued by managers and teammates was strongly correlated, at **.73**. . This provided initial evidence that developer thriving, healthy metrics use, and visibility impact how real developers work, and that these factors positively reinforce each other inside of developers' workplaces. Productivity is highly multifaceted, and driven by many factors beyond the individual–it is therefore not surprising that the correlations between Productivity

and the other factors are not as large, but these relationships remained statistically significantly positive and noteworthy.

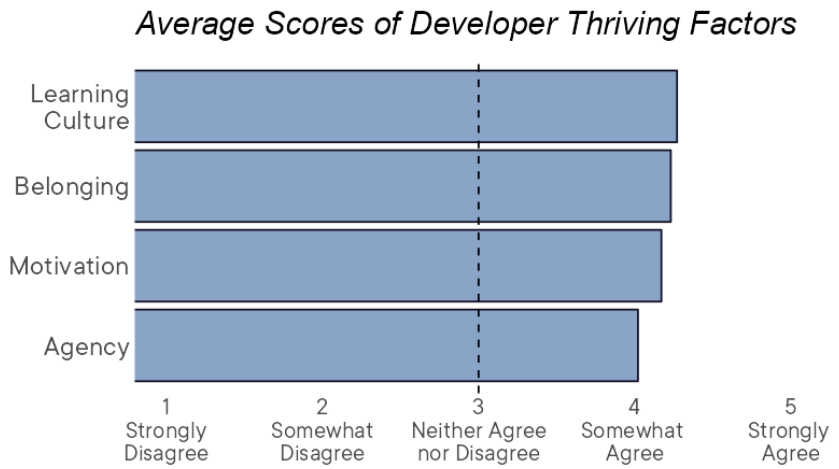

Fig 4. Average scores of the Developer Thriving Scale's factors

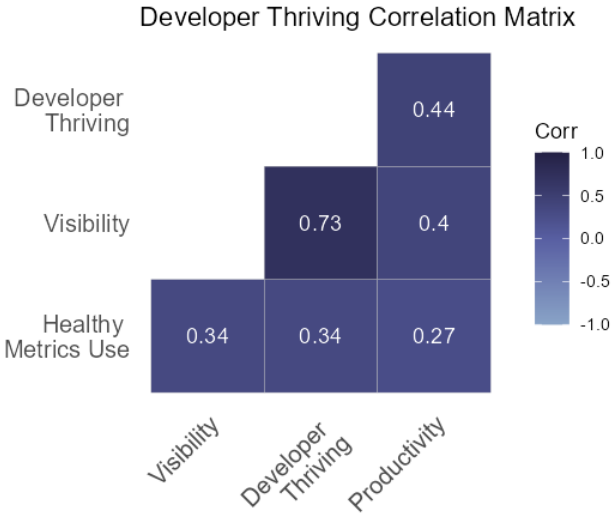

Fig 5. Correlation matrix of the Developer Thriving Scale, Visibility and Value Questionnaire, Healthy Metrics Use, and Perceived Productivity Rating.
All correlations shown were statistically significant.

Testing how Developer Thriving, Metrics use, and Visibility lead to Productivity

From previous research (e.g. Fagerholm & Münch, 2012; Forsgren et al., 2021; Mikkonen, 2016; Morales et al., 2019; Storey et al., 2021), we know that a positive developer experience and increasing developer satisfaction are among the best ways to increase developer productivity. With our study, we wanted to expand the concept of satisfaction and look at whether the factors in developer *thriving* impact productivity. We also wanted to see if implementing team-level tools and processes such as healthy metrics and increased visibility could improve

developer thriving and productivity, even after controlling for factors like years of experience and time spent coding.

To ask these questions, we conducted a serial mediation conditional process analysis. This analysis allows us to look at multiple relationships between variables at once and test a proposed *path* towards an outcome (in this case, developer productivity). This model allows us to explore evidence for both direct and indirect effects. In mediation analyses, direct effects refer to the effect that a predictor variable has on an outcome variable. Indirect effects refer to the effect that a predictor variable has on an outcome variable by working through a mediating variable (mediator; Hayes, 2022).

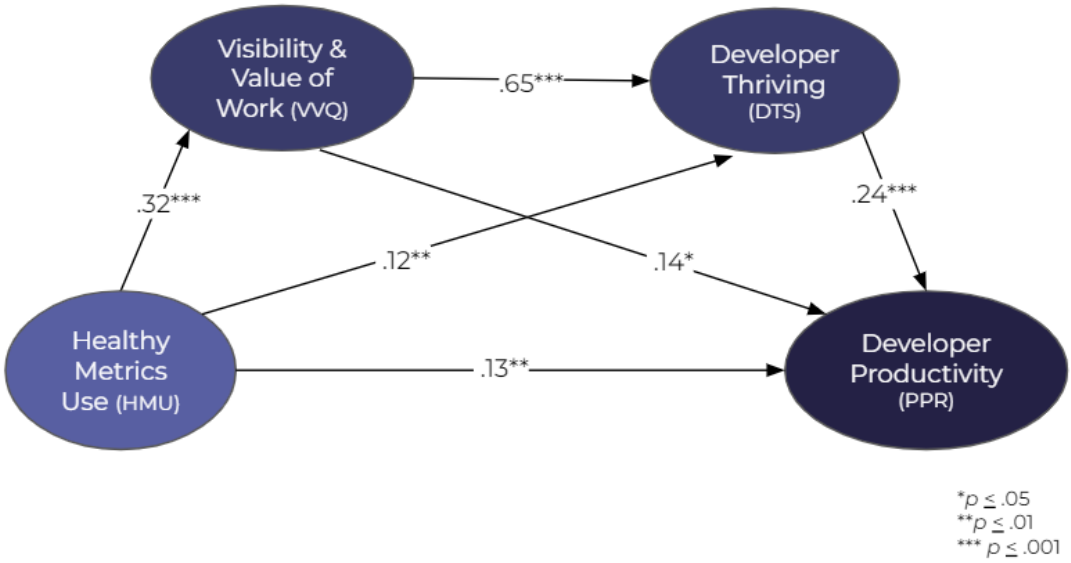

Fig 6: Serial Mediation Model with standardized regression coefficients.

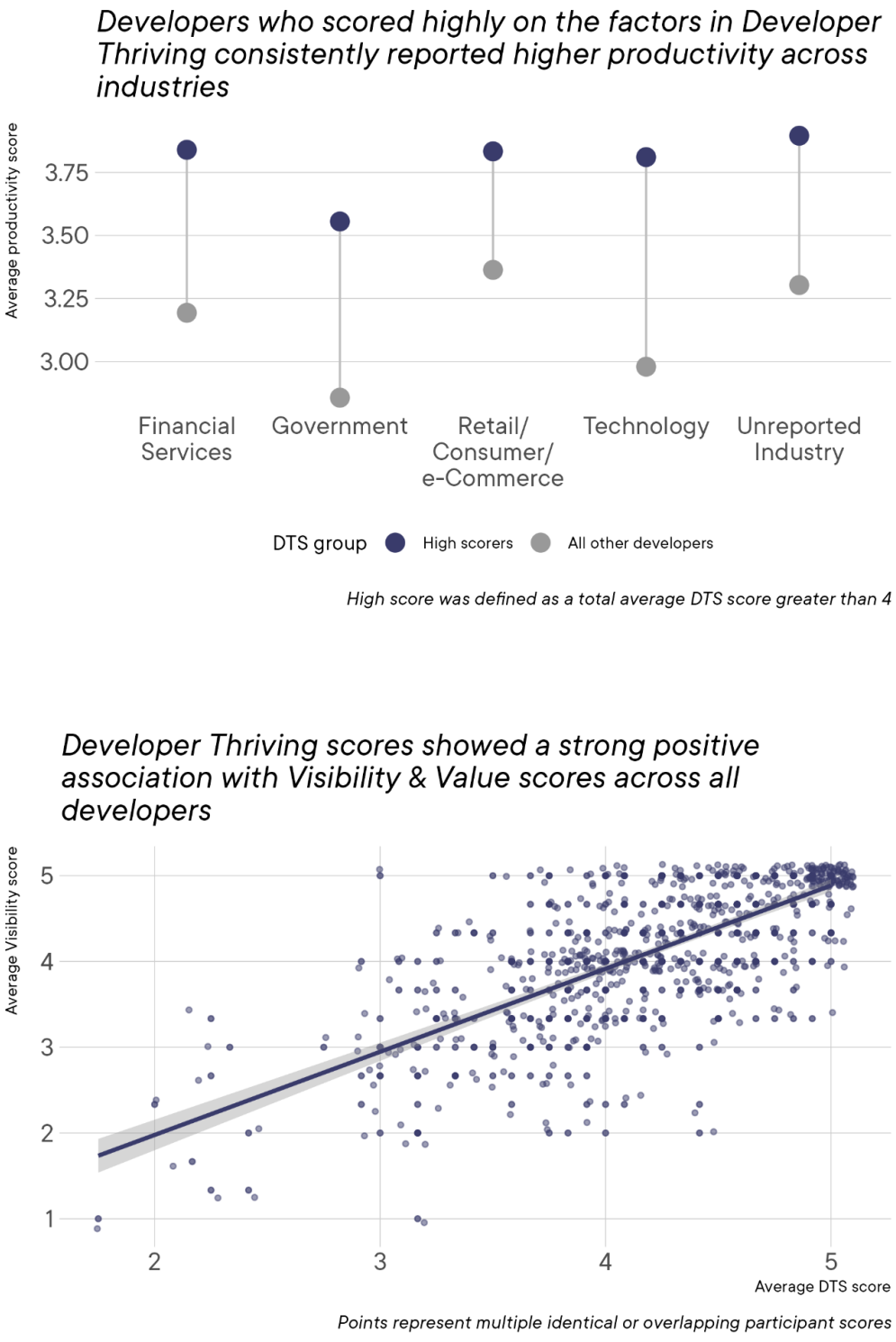

Fig 7: Developer Thriving scores shown in relation to Productivity, and Visibility

Our model finds evidence that across these factors, ***improving Developer Thriving was the most effective path in this model to improving productivity*** (Fig 6-7). This finding supports previous software and employee wellbeing research that highlights satisfaction as the

strongest predictor of productivity (e.g. Hackman & Oldman, 1975; Storey et al., 2021). Importantly, our model also expands on this connection to highlight visibility as the key to not only directly increasing developer thriving, but also boosting the effect of thriving on developer productivity.

That is, developers need thriving and all its elements inside of their immediate problem-solving environment, but they also need to believe that their *individual* productivity will go beyond their teams. As one senior, tech lead developer expressed in our qualitative interviews, *"peak success is [code is] written in a way that is composable [and] easy to digest for other developers, you know, potentially has documentation around it."* For this technical expert, true "success" in software work is not accomplished until the loop of visibility is completed. Our **Visibility & Value** construct is a step towards naming and measuring the missing piece that helps explain an important connection between individual developer productivity, and how the organization's valuing and recognition of it flows back down to software teams.

The unique benefit of both *expecting and planning for visibility,* and *getting feedback from a visibility cycle,* echoes scientific evidence around human wellbeing, health sciences, and organizational psychology. For example, research on behavioral change in healthcare settings highlights the value created from recognition and visibility as one of the strongest predictors of behavioral engagement, performance, and productivity of both individuals and team members (Dawson, Mullan, & Sainsbury, 2015; Johnston & White, 2003; Stecker, McGovern, & Herr, 2012; O'Flaherty et al., 2022). This impact on developer *motivation* was a key theme underlined by both individual contributor developers in our qualitative research, who describe *expecting* and *anticipating* moments of recognition as key motivators, and by managers, who describe a pivotal responsibility of *making their team's work visible*. In **Study 2**, we dive more deeply into this experience.

Increased measurement leading to positive outcomes echoes a significant body of research in the clinical and behavioral sciences, which indicates that we tend to forget or lack awareness of the amount of work we have done, leading us to devalue and minimize our progress. Tracking behavioral and psychological processes has been shown to mitigate this effect by providing us concrete evidence of our progress and accomplishments. Having this evidence not only increases mindful attention and awareness, but also increases our sense of value and mastery over our work, increases empathy and self-compassion, boosts coping abilities and distress tolerance, empowers us to recognize and set boundaries, and drives behavioral engagement for both groups and individuals (Bornstein, Hamilton, & Bornstein, 1986; Cohen et al., 2013; Ehlers et al., 2003; Foster et al., 1999; Kavanagh et al., 1999; Korotitsch & Nelson-Gray, 1999; Jason, 1975; Lambert et al., 2001; Latner & Wilsom, 2002). And with developers specifically, research has found that self-reflection in a repeated cadence increased developers' awareness of their habits and led to positive behavior change for both output and wellbeing (Meyer et al., 2019). In **Study 3**, we dive more deeply into how developers report their teams' measurement practices.

Developer Thriving in Qualitative Research			
Agency	A developer is: 1) able to voice disagreement with team definitions of success 2) has a voice in how their contributions are measured	Billett, 2011 Gobeli et al., 1998 Hicks, 2022 Meyer et al., 2019	“An environment where you know [speaking up is] encouraged makes you want to chime in if you're anticipating something, or think that maybe another direction [should be explored] definitely helps” - IC Participant, Focus groups
Motivation & Self-Efficacy	A developer is: 1) motivated when working on code at work 2) can see tangible progress most of the time 3) is working on the type of code work they want to work on 4) is confident that even when working in code is unexpectedly difficult, they will solve their problems	Bandura & Adams, 1977 Kim et al., 2023 Robinson et al., 2019 Sherer, 1982	“I know for me and some of the engineers that I have managed, it comes down to ‘are you able to have an impact on the product...the company...have you been able to drive that forward?” - Manager Participant, Focus groups "[I want] recognition that I'm a good problem-solver...that I'm given problems to solve not given solutions and told to implement them. That's where you move toward [technical leadership]” - IC Participant, Interviews
Learning Culture	A developer is: 1) learning new skills as a developer 2) able to share the things they learn at work	Hicks, 2022 Scott & Ghinea, 2013 Luxton-Reilly et al., 2018	“Being an engineer it's always about growth. Your job is to do the job and to learn and grow because you're going to have to be able to take on more challenges, and in order to grow as a leader. It’s a never ending cycle unless you're planning on being stagnant” - IC Participant, Interviews "Having that time to learn gave me greater confidence as I moved into the task of actually writing the code...and I think ultimately allowed me to be more successful” - IC Participant, Interviews
Support & Belonging	A developer is: 1) supported to grow, learn, and make mistakes by their team 2) agrees they are accepted for who they are by their team	Anderson-Butcher & Conroy, 2002 Pardede, Gausel, & Høie, 2021 Rattan et al., 2018 Wilson et al., 2010	"It's very valuable to be seen by a manager [as] an existing human being. I'm not a robot...I'm a finite human being and I need a break ... because it's wearing me out...my manager's like yeah...thanks for saying [it's not working]" - IC Participant, Interviews “I feel safe when ideas and contributions are valued equally.. whether it's an architect, person that's been at the company for like 15 years, and built the whole thing, or some intern that just started. I feel like good ideas can really come from anywhere” - IC Participant, Focus Group

Table 5. The Developer Thriving Framework

Study 1 Summary

Overall, we find evidence that supports describing developer productivity as an emergent result of a holistic ecosystem. This ecosystem includes 1) the structural elements of Developer Thriving that support good problem-solving on software teams and for individuals 2) a culture of recognition and connected visibility for software work between teams, and throughout the engineering organization 3) the right tools and processes to facilitate Thriving and Visibility, such as *healthy metrics.* In Table 5, we again summarize the framework of Developer Thriving, along with example quotes from our qualitative research for each construct. The following sections dive more deeply into 1) this qualitative research, specifically looking at *how developers describe visibility,* and 2) what we've learned about *healthy metrics*.

Recommendations from Study 1

Finding	Recommendation	Potential Impact
Developer Thriving is the strongest predictor of productivity	**Developers** should be aware of whether they are supporting their teammates' learning, belonging, and other key factors in crucial collaboration moments such as code reviews and team retrospectives **Managers** seeking to increase productivity should diagnose possible gaps in Developer Thriving elements on their teams, documenting baselines and advocating for investments in Developer Thriving factors **Organizations** should commit to monitoring how engineering functions are achieving Developer Thriving between and across teams, considering factors such as enough learning time, strong supportive cultures, the opportunity to give feedback, and recognition for effort work and difficult problem-solving.	⬆ Developer Experience ⬆ Code velocity ⬇ Problem-solving roadblocks
Developers benefit when their work is seen and recognized by teammates, managers, and the organization.	**Leaders** should diagnose whether there are "visibility gaps" inside of their engineering organizations, paying special attention to teams or types of engineering work that do not get shared broadly **Leaders and Managers** should assess whether engineering effort is evaluated with metrics that drive change and decisions, and ensure that long-term impact is tracked over time and across engineering projects **Organizations** should invest in systems that explicitly recognize and reward teams for the technical progress they make, particularly work that was unexpectedly challenging, required new skills, or fixed long-standing problems	⬆ Employee Satisfaction ⬆ Employee Retention ⬇ Planning Time

Table 6. Study 1 Recommendations

Study 2: How developers describe the impact of visibility on their motivation and success

Study 2 High Level Summary

Managers and ICs both shared multiple examples of the importance of tying the effort work of engineering to the impact on the real world.

One key theme was a gap of understanding between the work engineers know they need to do, and the real business impact. Many managers and senior engineers spoke of unrelenting pressure to do translation work between engineering investment and business impact.

An important counterpoint to "visibility" arose as a subtheme: when developers felt that their work was less understood by their organization, they spoke to the importance of protecting focus time and "real work."

In Study 2, we present a qualitative investigation across interviews and focus groups where developers shared their definitions of success, experiences with complex software work, and reflections on barriers. For the qualitative study, we first conducted 1) in-depth individual interviews with **9 developers** from within Pluralsight, and then built on these initial conversations to create a deeper script for 2) three hour-long focus groups with **6 developers** from outside organizations. This two-part process allowed us to iterate our investigation as we began to identify the importance of "visibility" to better understanding developer productivity.

Qualitative research is necessarily open-ended; we created a semi-structured interview script for all sessions, but participants are invited to share their experiences naturally. This exploratory approach allows researchers to follow the themes that are the most important to participants, and to adapt the question script in response to what participants find most important to share. A sample of our qualitative scripts can be found in Appendix C.

Qualitative Analysis

To analyze across our interview and focus group conversations, we used thematic analysis (Braun & Clarke, 2006). Thematic analysis focuses on examining every conversational statement made by individuals across our interviews and focus groups, and categorizing them into an overarching "theme" which the specific example illustrates. Themes are meant to represent large patterns in participants' experiences, which capture an important lens on

the research topic. These themes are often overlapping, and participants may speak to more than one theme at a time.

Themes and subthemes across all sessions were identified in collaborative qualitative coding sessions with three researchers, and inter-rater agreement was achieved for each major theme. While not every theme was mentioned by every participant, in order to qualify as a major theme, a topic had to be mentioned by at least four out of six focus group participants, or seven out of nine interview participants.

Qualitative Findings

We identified **3 major themes** relating specifically to the work of making engineering effort visible (Table 7).

Making Engineering Visible: Top Themes		
Theme 1	Visibility impacts individual motivation	*"...I would be more willing to pick up something again that was buggy or stuck because I know that I am being appreciated for it."* - IC Participant, Focus Groups
Theme 2	Visibility impacts business decisions and goals	*"As leaders, you're not necessarily in the weeds...but the very minute details actually do matter...giving [leaders] more information is valuable so they know how to kind of navigate those nuances."* - Manager Participant, Focus Group
Theme 3	Visibility is generated through advocacy, but good advocacy relies on careful understanding of individuals	*"... there is essentially a huge set of internal tools and stuff which never really see the light of day....that's the role of the tech lead [or manager] ...to create the visibility that 'I have like 2 or 3 folks on my team. They're super talented... just because they're not doing [feature] work doesn't mean that they're not contributing to the business...[making that visible is] a huge responsibility."* - Manager Participant, Focus Groups

Table 7. Qualitative Research Themes

Theme 1: Visibility impacts individual motivation. Providing transparency into an IC's accomplishments to the broader organization strengthened a developer's self-confidence and thus motivation. In our conversations with Developers, many developers felt that the more visible they were to the organization, the more motivated they felt to do the work. Managers were essential to the creation of two forms of visibility, first "bottom-up," communicating engineering effort to organizations, and the second, "top-down," redirecting and steering engineering work and energy in response to business priority. Examples of the first visibility were especially impactful when developers knew their progress was directly

shared with leaders, and when they felt recognized for conquering truly difficult work and the seemingly "invisible" effort behind impact.

> **"I would be more motivated if someone said, "oh, wow! Nobody had gotten it done, that's so amazing". I would be more willing to pick up something again that was buggy or stuck because I know that I am being appreciated for it, and someone is looking at it."**
> **-** IC Participant, Focus Groups

> **"[in a feature demo] the credit was always given [to me as the engineer] and I think that gave me a huge sense of not only like ownership, but it's also quite empowering [...] because you know the end user of this product."**
> -IC Participant, Focus Groups

> **"If you're going to ask a developer to do something, you have to tell them why they're doing such a thing [and] what the bigger picture is. It's easy to say exactly what [impact the whole group will have], but it's really difficult to get people motivated to do [their individual task]...that's why giving context is really important."**
> - Manager Participant, Focus Group

> "...**giving credit where it's due at the right time, like a showcase, is huge."**
> - Manager Participant Focus Groups

In our conversations, managers also recognized visibility as a key component to motivation, and emphasized that these cycles of recognition often needed to come *before* explicit accomplishments.

> **"Giving visibility early on as developers are learning and making sure that others are actually seeing and celebrating that growth is a very, very key piece of that puzzle."**
> - Manager Participant, Focus Groups

> **"As leaders, you're not necessarily in like in the weeds...but the very minute specific details actually do matter and play a big role when you're finally putting something out [even though it may not have been] visible to you as a leader. Giving [leaders] more information is valuable so they know how to kind of navigate those nuances that come with any given project."** - Manager Participant, Focus Group

Research further corroborates these sentiments by exploring how visibility not only directly increases motivation (Ajzen, 1991; Eccles et al., 1983; Roemer & Orsillo, 2009), but that it also increases confidence in one's abilities, which similarly increases levels of motivation (Ajzen, 1991; Bandura & Adams, 1977). This increased level of recognition can be a strong determinant

of developer's productivity and overall project success (Baddoo et al., 2006). Without a strong cycle of reflection and visibility for engineering effort, one manager shared an example of new developers hitting and replicating old barriers:

> **"There were specific, kind of technical, bad decisions that we made that we would actually never make today. But [new developers were] learning off the technical bad decisions."**
> -Manager Participant, Focus Groups

Theme 2: Visibility impacts business decisions and goals. The second level of visibility can be "top-down." This is when organization-wide impact, progress and/or decisions are shared with developers. From the leader's perspective, all managers in this study spoke about how transparency and real-time insight into what work is in progress, completed, or blocked, helps create clearer business expectations. Such visibility can help leaders re-prioritize projects or shift company-wide OKRs (Craig, 2018). Managers spoke about how giving individual contributors' visibility into the "bigger picture" can also be necessary and beneficial to the business's bottom line. Individual contributors also shared that developers often need to play a key role as a "signal" to leadership about the realities of abstract business goals.

> **"Providing information to leaders about how things are going, and then letting reports have some transparent insight into what's the conversations that are happening at the leadership level [is critical]."**
> - Manager Participant, Focus Group

> **"Engineers and individual contributors can impact their own developer experience by speaking up if they think a deadline is unreasonable. Developers have agency to change their experience through things like advocating for more learning time."**
> - IC Participant, Interview

> **"A big part of leadership is to help provide that visibility and also help navigate so you can put any person in the best position to both see what's happening, and also contribute in a way that they know the business will appreciate and essentially help them grow."**
> - Manager Participant, Focus Group

Theme 3: Visibility is generated through advocacy, but good advocacy relies on careful understanding of individuals. Given a host of complex societal factors, some developers may come from backgrounds where feeling "seen" or valued is not a salient concern as long as they produce strong technical work. Others come from underrepresented backgrounds, which can increase adverse experiences that make it harder to feel valued or fairly assessed, even in technical work (e.g., Roberson & Kulik, 2007). In Study 2, several developers shared that discretion and care in the ways that managers made their work visible was a critical

component to feeling valued. Throughout this theme, participants noted the difficulty of maintaining this advocacy and the long-term relationships and information exchange required between managers and developers when an organization relies heavily on individual managers to be the only source for translating engineering effort to the rest of the business.

> **“People are made differently. There are folks who would love to just work on code and they don't care if people see it right away. They don't care about recognition. Then there are folks who derive value from the fact that there are some eyes on it [earlier]. You have to really deploy [visibility] it on a case to case basis.”**
> - Manager Participant, Focus Group

> **“Conversations are really really really important for building trust [with direct reports]. Every developer is different. I think that's a really crucial point. Every conversation is different. Understanding how that person operates ...what motivates this person is super important. Finding as many early and often opportunities to provide feedback, both positive and constructive, and giving [visibility] to those things so you can establish a cadence of feedback, helps build the trust.”**
> - Manager Participant, Focus Group

> **“... there is essentially a huge set of internal tools and stuff which never really see the light of day....that’s the role of the tech lead [or manager] ...to create the visibility that ‘I have like 2 or 3 folks on my team. They're super talented... just because they're not doing [feature] work doesn't mean that they're not contributing to the business...[making that visible is] a huge responsibility.”**
> -Manager Participant, Focus Group

Despite this individual complexity, overall developers agreed that when managers *publicly* advocated for a developer’s skillset and/or made their contributions widely known, developers felt validated, reassured that their accomplishments would endure, and felt more confident and valuable bringing work to their team. Individual advocacy can be a “powerful tool” to increase employee satisfaction, productivity and even retention (Johnson, 2022). But advocacy was not only a tool for managers: developers mentioned that seeing this advocacy showed them a skill they could cultivate for themselves, and that senior colleagues or tech leads could also contribute to a software team.

> **“...it doesn't necessarily have to be the manager either that understands [you and your work]. If <u>someone</u> understands, a peer..a junior developer....they can [advocate for you and make your work more visible]“**
> - IC Participant, Focus Group

> **“If a good manager is paying attention, you're good at advocating for yourself or other people are willing to advocate for you it will [help show] that you are a good team player….”**
> - IC Participant, Interviews

However, it should be noted that successfully advocating required significant skill, time and effort from technical managers. Managers spoke of translating engineering effort at the right level, time, and place, and also the need to occasionally “hide” engineering effort when they felt the business was liable to misunderstand it. This tension weighed on the managers in our sample. Additionally, throughout this theme, we heard that if used to elevate developers who already felt *over*exposed in an organization, advocacy may have adverse effects (i.e. overshadowing or silencing other team contributions, or making underrepresented team members feel less important).

All managers in this study shared that they often did not feel they had the proper resources to prepare to advocate and instead pulled from their own experiences as a source of quick judgment or assumptions about what a developer needed and why they worked in a certain way. This can be dangerous as *“no two developers are the same,”* and managers and developers may frequently be concerned with and prioritizing different forms of “productivity” (Storey et al., 2022). Likewise, individual contributors shared that relying on a manager to surface their work at the right time and place could feel highly variable. This uncertainty could introduce significant stress and tension for developers.

> **“If you have a <u>good</u> manager, you won't feel the pressure that you just have to make sure your data is perfect… A good manager is going to be hands-on but also have context. [For example] your metrics might look really bad but your Manager's going to know that’s because you've been researching something for a week and a half that doesn't involve committing code or reviewing code**.”
> - IC Participant, Interviews

> **“I think a huge part of the role of the leader is to set up the structures for [priorities and metrics] to be aligned. To make sure the reports’ clear, make sure you manage up, and that leaders understand what that report's strengths and weaknesses are. Then build the structure, whatever that might be, to allow [expectations] to be aligned.”**
> - Manager Participant, Focus Group

> **“It doesn't matter how well you manage your team if you're not aligned or bring visibility into the place that your team or department has within the organization. Those pieces are critical.”**
> - Manager Participant, Focus Group

Advocacy, when applied cautiously and appropriately, can create greater outcomes for certain developers. But under this theme, both managers and IC developers highlighted the *precarity* of relying entirely on individual advocacy. Developers highlighted a need to move from relying solely on highly skilled individuals to do this translation work, and toward structural and organizational support that could replace ad hoc, individual advocacy with more systematic insight into the contributions of engineering.

> **"I think I've realized the importance of taking myself out of the way...A lot of my approach was based around [...] direct personal interactions which is important. But I think I leaned so heavily on those that, after having a few circumstances where I left the company or [stopped managing that person] [it didn't] necessarily set the reports up for the most success without me in the picture."**
> - Manager Participant, Focus Group

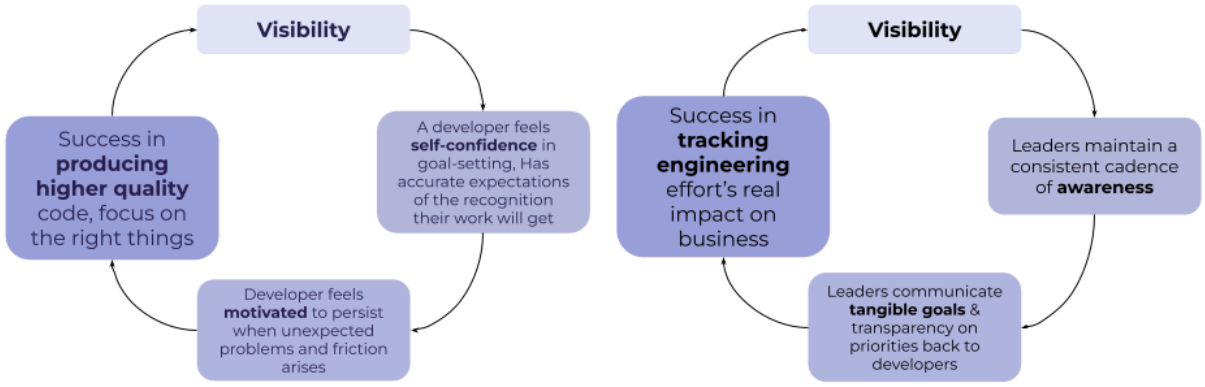

Fig 8. The "visibility cycle" inside of an engineering organization as described by IC developers and managers in Study 2

Study 2 Summary

The importance of managing trade-offs. In our interviews, developers consistently highlighted one important aspect of high-quality software: making and sharing the right decisions about trade-offs, priorities, and investments. Developers noticed that tasks which were categorized as "technical" frequently had overlapping consequences to "non-technical" work, and vice versa. This led many developers to discuss the fact that the most impactful forms of decision making were *both.* For example, senior developers discussed how the social and mentoring work of guiding junior teammates to consider alternative approaches would yield more flexibility in later code decisions made by those teammates. This type of successful problem-solving, and this cycle of influencing others, led many developers to question the divisions between "technical" code work and "non-technical" contextual work of mentorship, team planning, and communication. In particular, *team*-level definitions of success felt

important to this area of success, and *individual measures of productivity* felt inadequate and even threatening to this arena of work.

The importance of visibility. Visibility was frequently defined as the skill of translating and creating transparency around how engineering effort leads to impact, for both employees and leadership. Developers shared how such transparency can not only improve employee satisfaction and trust, but can also positively impact company metrics and ultimately profitability. When done carefully, promoting a company culture of visibility was seen as a recipe for success in software development organizations and a key dependency for developer "success". Developers and their managers highlighted examples of when visibility was critical to individual developers' motivation to solve hard problems, and manager's confidence that their teams were going to achieve success. Overall, being able to count on visibility that went "beyond your software team" raised developer confidence and motivation, and also brought information back to engineering teams about business impact, driving improvements to overall software quality.

The importance and difficulty of tying engineering investment to business impact. Across the board, participants agreed that more visibility into team progress can also be beneficial for leaders, as it created opportunities for realistic goal-setting and appropriate company-wide changes. Developers shared the need for leaders' business strategy, goals, and planning to be informed by "the possible" in engineering. In particular, managers and more senior developers spoke strongly of wanting to be a voice for engineering decision-making, yet navigated constant complexity in guessing where, and when, to have this voice. Our participants contrasted moments of productive, shared visibility with moments of dealing with uncertain, "black box" moments inside of their organizations. Echoing our quantitative findings, shared visibility helped to mitigate these "black box" moments. Overall, nearly all participants vocalized that visibility can be a change-agent in fostering developer trust, increasing employee satisfaction and even strengthening org-wide strategy, but also saw this benefit unevenly applied in their organizations.

Recommendations from Study 2		
Finding	Recommendation	Potential Impact
Developer motivation is influenced by **self-confidence**; which is impacted by moments of visibility and recognition.	**Managers** should make their reports more visible by publicly recognizing and advocating for their work. Where possible, seek out opportunities to credit developers directly in business impact, such as representation in demos, and recognition in launches.	⬆ Employee Satisfaction ⬆ Employee Retention
Org metrics can be impacted by a **lack of transparency** around team progress. Tying engineering investments to business impact can lag over time, and relies heavily on individual manager advocates.	**Leaders** should provide opportunities to act on transparency around team progress Assess whether engineering effort is evaluated with metrics that drive change and decisions, and ensure that long-term impact is tracked over time and across engineering projects	⬆ Technical Roadmapping ⬆ Tangible Org Metrics ⬇ Missed Org Targets ⬇ Wasteful Project Spending
Some managers and leaders struggle with **how to advocate** for their reports. When orgs depend on individual managers for reporting and visibility, and **those managers leave**, individual contributors can struggle and experience loss of recognition/opportunity	**Organizations** should provide educational resources on how to advocate effectively across lines of difference **Managers** must get to know their reports well enough to know how to effectively advocate. **Leaders** should ensure there are documentation and recognition structures that maintain a “life-cycle” view of developers’ work within an organization, helping make IC contributions visible beyond individual advocates	⬆ Employee Satisfaction ⬆ Employee Retention ⬆ Equity & Inclusion ⬇ Knowledge loss

Table 8. Study 2 Recommendations

Study 3. “Healthy metrics”: what we learned about how software teams measured their work

Study 3 High Level Summary

Only 20-30% of developers report being on a team that consistently uses team-level software metrics.

For developers who do report being on a team that uses software metrics, many report common pitfalls such as not measuring enough things, measuring things without context, or being concerned primarily with the *appearance* rather than *meaning* of the metric.

Despite this, developers are positive on the overall potential *benefits* of using metrics, with high levels of agreement that tracking parts of their coding process increases visibility, helps their teams make trade-off decisions, and helps them understand their own productivity.

Racially minoritized developers were *more likely* to agree that tracking parts of their coding process, and that *team* tracking of coding processes, was useful and impactful to their work. This may reflect a greater concern from some developers with obtaining visibility and recognition for technical work, which previous research has documented for people who hold minoritized identities in STEM fields.

In Study 3, we investigated what developers told us about *measurement* and the use of software metrics on their teams, as well as their usage of software metrics as individuals. Measuring software teams’ activity and output, and comparing this between teams, is difficult. Between teams, software teams may use vastly different measurements as their “metrics,” and the expected values and impactful change on these metrics depends heavily on the type of engineering work being done, and may provoke misconceptions from people not directly involved in the context of the software metric (for more commentary on this, see Sadowski & Zimmerman, 2019). Even within a single software team many different metrics may be used, emphasized, or discarded over time to correspond to changing definitions of team success.

To grapple with this complexity, we used initial pilot testing, qualitative interviews with developers, and a survey of previous research to develop a set of questions describing *how*

developers perceive the usefulness and impact of metrics in their workplace, rather than a simple canvas of *which* metrics they are using (refer to Table 2 for a summary of quantitative measures across Studies 1 and 3).

Looking at metrics usage. For Study 3, we focused on asking developers about the activity of measurement and its impact in several different ways. First, developers reported whether they used software metrics as individuals and/or their team, in what rituals these were used, and whether they believed their teams were falling into one of four *common metrics implementation pitfalls*. These "pitfalls" are based on recommendations from the Software Improvement Group and their experience with evaluating the implementation of software metrics (Bouwers et al., 2012; 2013). We adapted these patterns into a "metrics diagnostic" item for this study (Table 9). While our adapted survey item was not a comprehensive summary of all the ways that metrics implementations can go wrong, it provided a helpful starting point for characterizing developers' perceptions of *how* their teams are implementing metrics.

Common Metrics Pitfalls
1. *We tend to measure things without enough context*.
2. *We're concerned about the appearance, but not really the meaning of what we've measured*.
3. *We have not measured enough things*.
4. *We measure many things, but they do not feel related to each other*.

Table 9.

Looking at the Impact of Measuring Coding Work (IMCW). Finally, developers who indicated that they or their teams used metrics answered a series of questions aimed at identifying whether developers found measurement of software work 1) *useful and impactful to the progress of immediate technical work* 2) *increasing visibility of technical work* and 3) helpful to individual developers in *understanding their own productivity*. Previous research has found that developers' perceptions of the *usefulness of metrics* can be a large barrier to their adoption (Wallace & Sheetz, 2014).

Across industries and engineering roles, only 21-32% of developers report being on a team that consistently uses software metrics. Uncertainty around whether teams even use metrics emerged as a significant theme.

Overall, developers reported mixed software metrics usage (see Figures 9-11 for some selected views of this data by groups). Across the entire analytic sample of developers (*N* = 1282), **29%** reported being on a team that used software metrics, and **23%** reported using *individual*

metrics about their software work. These groups overlapped: developers who reported using team metrics were also significantly more likely to report using *individual* metrics [$\chi^2(4, N = 1282) = 663.62, p < .01$].

A notable percentage of developers answered *Uncertain/Maybe* on their team metrics use (**26%**), which may indicate that they are unaware of whether their managers and leaders track software metrics, or may indicate an uneven and inconsistent usage of different metrics. When we asked developers *when* they saw software metrics used, a larger percentage reported usage (e.g., **41%** of developers reported the use of software metrics during sprint planning; Fig 9).

Taken together, these findings suggest that *many developers feel uncertainty* about metrics usage on their team, and may be experiencing uneven and often-changing uses of measurement. This may reflect a lack of clarity on what managers see and track, an uncertainty which was also mentioned in our qualitative analysis. This pattern may also reflect the difficulty of assessing "standard practice" in how developers measure their work, particularly because developers likely see different practices across their teams. Coding work requires developers to frequently collaborate across and between formal hierarchies; out of the developers who answered firmographic items on our survey, more than **63%** agreed that they collaborated closely with developers who had a different manager than theirs.

Looking at *when* software teams use metrics can help contextualize what role those metrics may be playing in developers' work. It's important to note, across this usage, that not all developers experience the same software rituals, as teams frequently use different strategies and processes (e.g., not all teams hold sprint planning meetings). However, the top three most frequent software rituals in which developers reported using metrics were *Sprint Planning*, *Standups,* and *Project Retrospectives* (Fig 9). Notably, a much smaller percentage of developers indicated that their organizations were using metrics during monthly or quarterly business meetings, which suggests that software metrics are either not making it outside of team-focused rituals, or their usage outside of specific software teams is relatively opaque to developers themselves. This finding underlines the tension and uncertainty that our qualitative research participants expressed in Study 2 in whether or not their work would become visible to the organization overall.

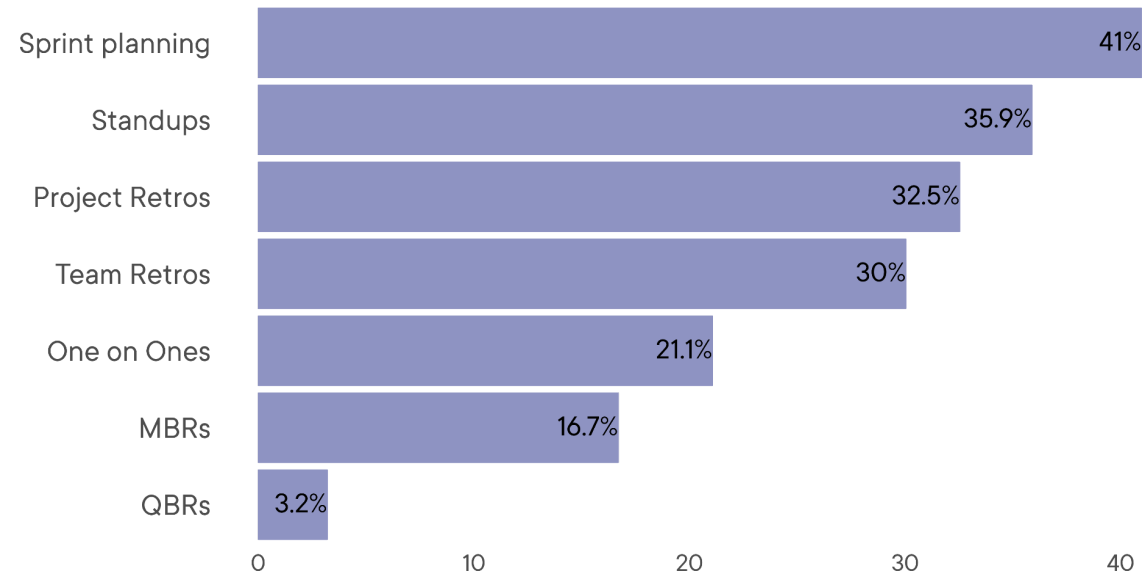

Fig 9. Software rituals in which developers reported their teams using software metrics. Developers could select multiple options, so each percentage is shown out of the entire sample, and responses add up to more than 100%. Based on full sample size of 1282

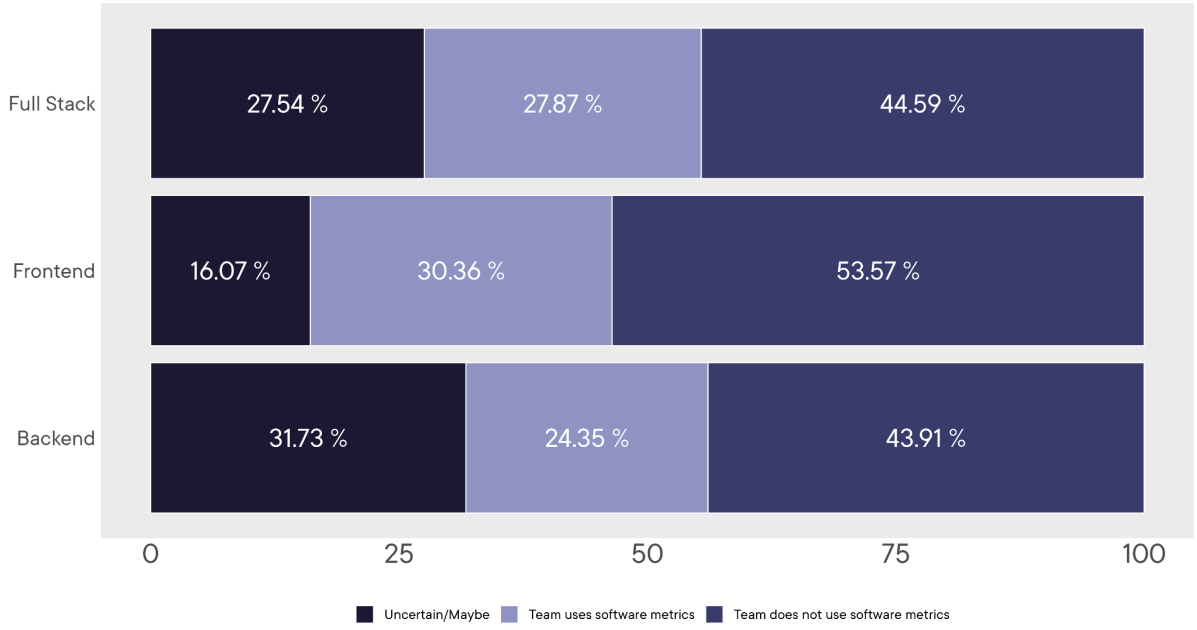

Fig 10: Whose team uses software metrics by top engineering role types

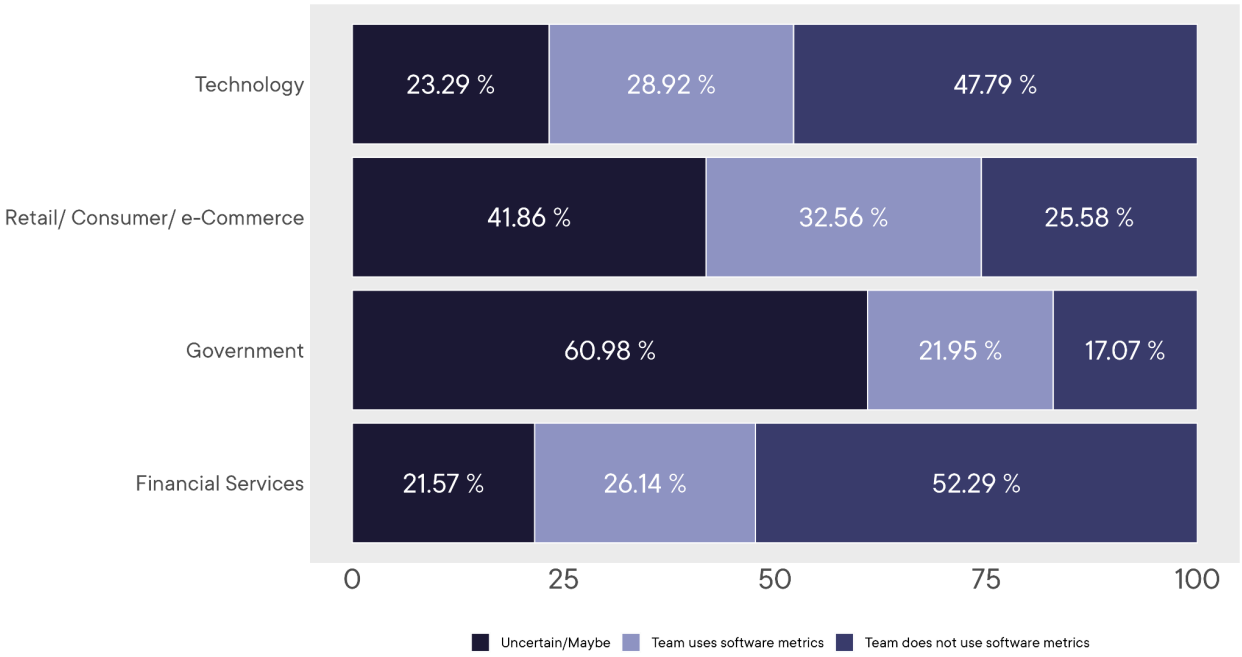

Fig 11: Whose team uses software metrics, by top industries

For developers whose teams use metrics, the most commonly reported pitfalls were not measuring enough things, and measuring things without enough context.

In Study 1, we showed that *Healthy Metrics Use* was associated with increased developer productivity. We explored this with a composite score: *Healthy Metrics Use* was defined as developers who are on teams that both use metrics consistently, *and* who agree, on our "metrics diagnostic" item, that their team consistently uses the metrics that are "right for us." Compared to developers who match either only one, or neither of those criteria, healthy metrics users show a benefit. This group was relatively small: only **14%** of developers in our survey overall were "healthy metrics users."

For the metrics diagnostic question, over the whole subsample of developers who were on teams using metrics, **69%** of developers reported one of the key pitfalls. Developers reported a range of uses, which continues to suggest that developers see variable implementation of metrics, and that developer perceptions around this usage is an important thing for managers and leaders to understand (Fig. 12). We also explored for but found no evidence in this sample that developers were *more* likely to detect, report, or be exposed to the metrics pitfalls depending on characteristics such as their engineering role, whether or not they worked closely with developers who had a different manager, and the demographics collected in this study.

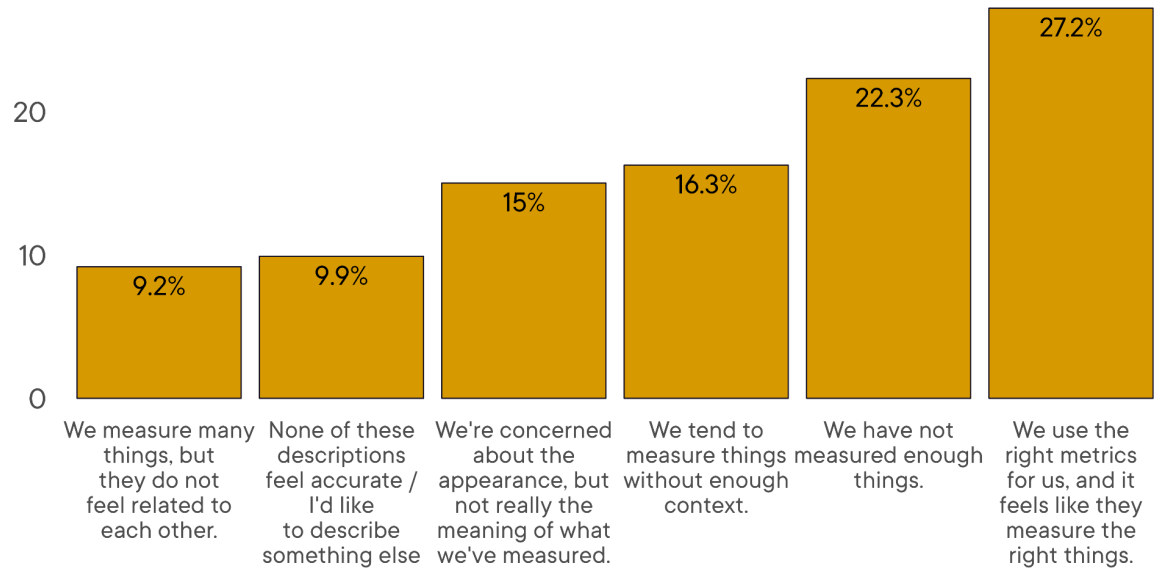

Fig 12. Developers reported a range of use on the Metrics Diagnostic, including four key "metrics pitfalls"

Despite the frequent occurrence of these pitfalls, developers were strongly positive on the impact of tracking and quantifying engineering work. This was true *even for the developers*

who saw pitfalls and mistakes in their team's use of metrics, who were still very positive on the overall impact and utility of metrics. Across the entire IMCW scale, **87%** of developers agreed that tracking parts of their coding process would help their teams make trade-off decisions, increase the visibility of technical work, and help them see their *own* productivity (Fig 13). This last finding once again echoes the research on the benefits of self-monitoring and meaningful information gathering mentioned in Study 1, which suggests that developers will benefit from measuring their work not only to assess project success, but also in order to reflect and improve on their own productivity.

In the qualitative data gathered in Study 2, we also heard examples of why developers see benefits from improving the over-time measurement of their work. One developer shared an example of tracking their coding work as an exercise in diagnosis and reflection: "[metrics let me ask] am I coding every day, am I coding efficiently. Am I writing new code that endures? Is my code getting better...that's a little subjective, but how long does it take a PR from getting submitted to being merged...my manager and I have talked about some of those [metrics] in the past -- seeing my code getting better helps me." In another quote, a participant shared a similar growth mindset around how tracking represented an opportunity for improvement: *"I just view [metrics] as ways to remind me of how I can do better."*

Developers' positivity toward the benefits of measurement suggests developers both want and see the potential impact from better measurement and tracking of coding processes, both for themselves and across their teams – even when there are challenges with its implementation. Their positivity about measurement is mitigated, however, when developers feel uncertainty in business and leadership's genuine understanding of their effort. For example, one engineering manager shared a counterpoint view on visibility, emphasizing the need for managers to calibrate the pressure that engineering effort may come under: *"[sometimes my team needed] a kind of comfortable bubble [to keep focus]...sometimes organizations [have many] ad hoc requests...and when folks are pinging your team right and left [you need to] tell folks, we'll get back to you."* Another manager shared, *"I think it's like a balancing game, really right. You want the visibility to come in, but you don't want to be over exploited. So I think that's the kind of balancing game you're playing."*

Individual contributors were also cognizant of the ways that important elements of engineering work remain difficult to measure. Across our qualitative research, developers spoke in particular about the difficulty of measuring collaboration. However, multiple participants also spoke about strategies for using measurement as a tool for recognizing shared work, for example: *"We try to be intentional if we pair, that we commit code that both of our names are on it."* Individual contributors also highlighted the possibility of gamifying metrics inside of a system where measures were not thoughtfully connected to real impact, or the possibility that metrics are more or less accurate for different software practices: "*Some people will game a lot [...] if you follow the guidelines of development patterns, metrics are definitely more accurate with certain development patterns, like frequent check-ins and a sort of more agile approach."*

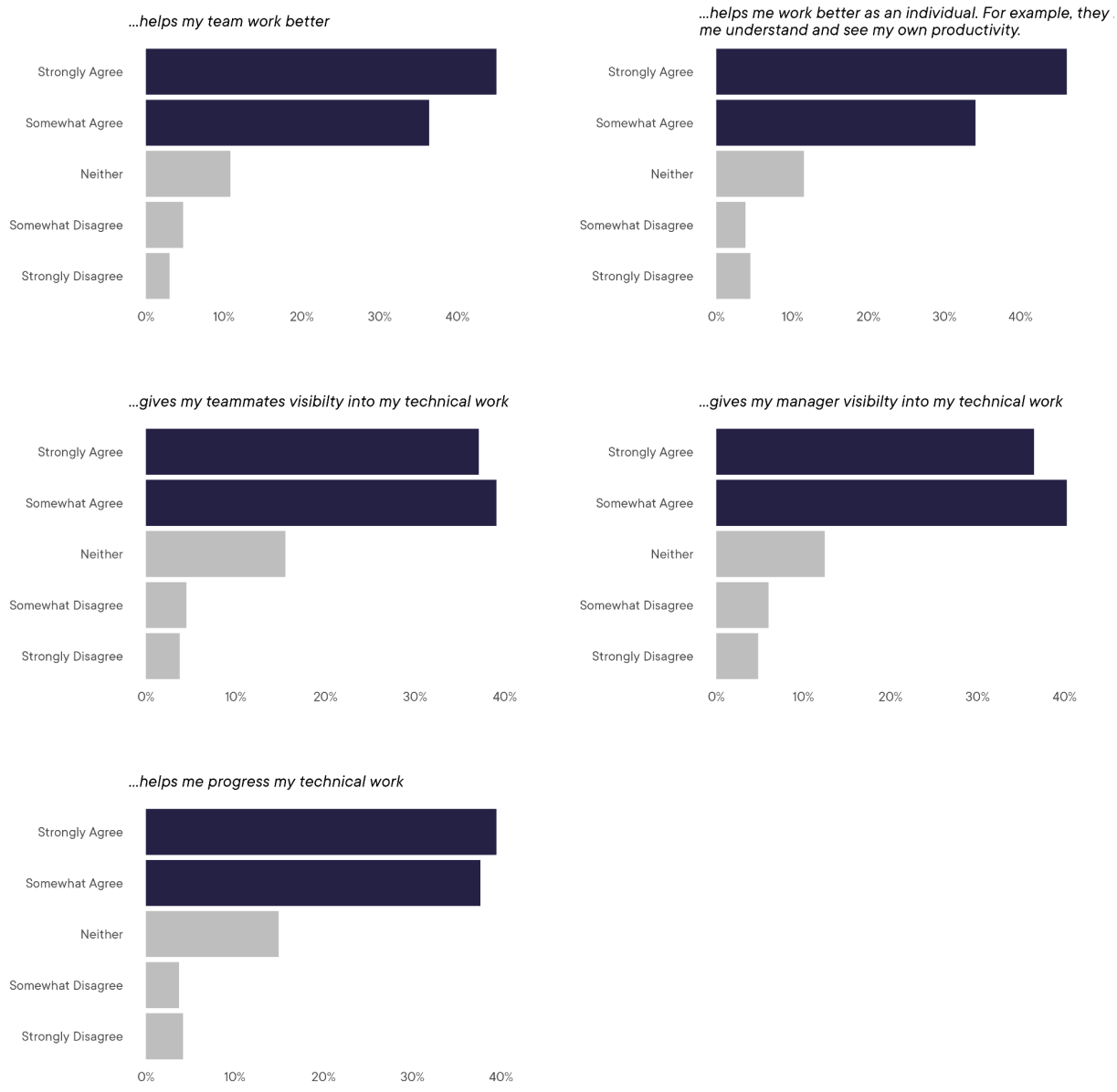

Fig 13. Developers were strongly positive overall on the IMCW items.

Racially minoritized developers are more likely to agree that tracking parts of their coding process was *both* useful *and* that it increased visibility and technical progress.

Across our participant characteristics, we wanted to know whether there were any signals about whether some developers may *perceive metrics as more useful and impactful* compared to others. To answer this question, we used a regression model to ask whether engineering role, industry, team size, reporting to a different manager than collaborators, gender, sexual orientation and/or gender identification minority status, educational status, and racial identification groups showed any differences in how developers scored on the Impact of Measuring Coding Work (IMCW) scale. Accounting across these characteristics, we found one significant difference: **racially minoritized developers were more likely to agree that tracking & quantifying parts of their coding process was beneficial to both team and self** (Figure 14; see Appendix D for full details). We did not find any significant subgroup differences for being more likely to say their team "used the right metrics," nor in being more or less likely to belong to a team that used metrics in the first place.

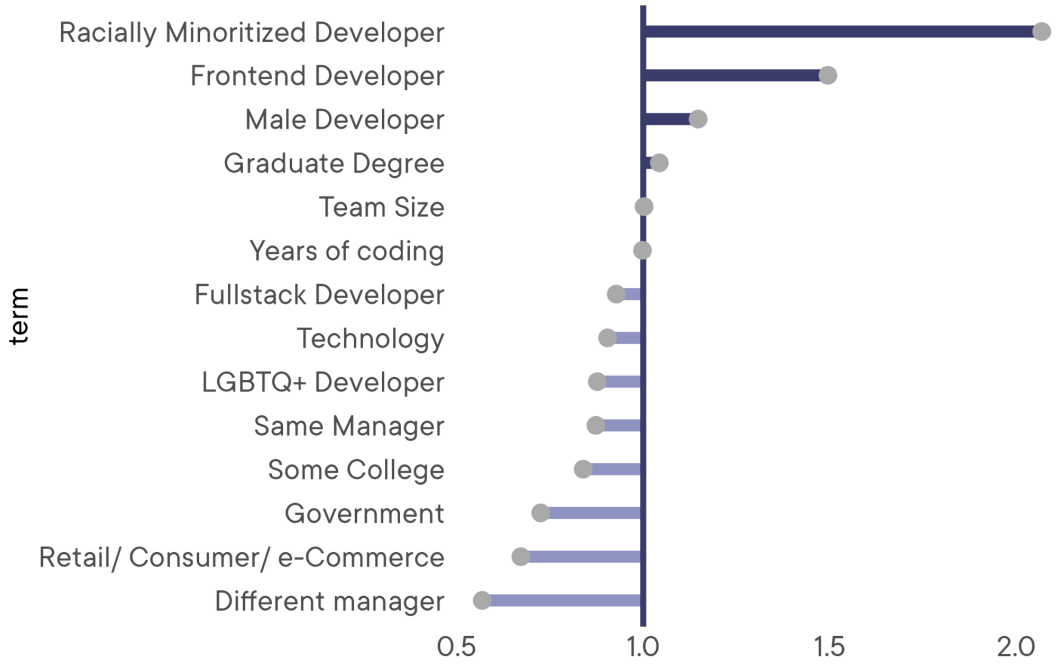

Fig. 14: Subgroup differences as shown in odds ratios: only Racially minoritized developers had significantly higher odds of scoring high on the impact of coding work measure. For full model details, see Appendix D.

While exploratory, as our study was not a representative sample across these identities among professional developers, this finding raises the interesting and important possibility that capturing a history of clear, and shared measurements about developers' *effort* and

process may be *particularly* impactful and appreciated by racially minoritized developers.[1] This interpretation is supported by research on related differences of experience in STEM fields. People at higher risk of adversity and discriminatory experiences often report a deeper concern with gatekeeping, harsh evaluation, and lack of credit for technical work (e.g., Quadlin, 2018). For developers who have experienced and anticipate systematically different, and cumulatively steeper, career impact from discrimination and overly punitive evaluations compared to majoritized developers, it may be that concerns about *mis*measurement are outweighed by concerns about what will happen *without* measurement at all. While our qualitative data in Study 2 did not specifically ask about the experiences of racially minoritized developers, some comments from developers supported this interpretation:

> ***"...speaking of someone to elevate them or giving them opportunities that may not have [originally] been identified for them [is important]. You find that across lines of gender, race, age, just difference, that some folks [don't have] a fair shot. Sometimes it's harder [to get recognition or visibility] if you're from a certain demographic versus others....I think it's very important that somebody speak of you in order to help you in your path."*** - IC Participant, Interviews

> ***"And then I guess when you're talking about [...] along gender lines and things like that... I've heard [people] talk about how the metrics have been equalizers for them compared to past development environments that they've been in."*** - IC Participant, Interviews

We designed our demographic questions to allow for robust self-identification, including allowing multiple category choice, and identity categories drawn from existing best practice recommendations for equity and accuracy in measurement of identity (see Appendix B for further details). Nevertheless, we are conscious that survey questions about identities can never perfectly represent individual identity, and many participants chose not to disclose this information in this research. Further, participants could either choose to not answer any demographic questions or self-identify as a text response. All of these choices led to a reduced sample size across identity questions. To avoid potentially misleading signals, we chose a conservative approach by comparing developers who had answered any of the categories that included "white" with developers who had answered any category that did not include "white" (see Appendix D for full model details). This is a very rough categorical comparison, which does not represent the complex experiences included within and across groups. This analysis is not a final statement, but a suggestive signal that future in-depth work is needed to understand how different developer groups experience the impact of measurement practices. We recognize the limitation in this description and plan to continue investing in specific recruitment strategies to better represent complex identities (e.g., distributing our

[1]In keeping with best practice suggestions from sociology and social sciences, we chose the term "*racially minoritized*" to bring focus to the ways in which this group of developers experience being on teams with systematically fewer people who share their backgrounds and identities, and the fact that this experience is the result of environmental and industry-wide, rather than individual factors. See Appendix B for more about how we chose to measure identities for developers.

research recruitments to affinity groups in tech, recruiting intentionally for qualitative research).

Overall, we highlight the need for leaders and managers of engineering organizations to understand the diverse experiences of developers, and to learn from *all* developers in engineering organizations, particularly those less represented in the majority of software research, such as non-US based developers, global majority developers, developers navigating adverse experiences such as discrimination, and other key experiential differences that impact how developers relate to success, performance evaluation, and safety in their environments. Along with many other researchers in software engineering and social science (e.g., Cole, 2009; Gren, 2018; Rodríguez-Pérez et al., 2021), our finding echoes the need for continued intentional recruiting of underrepresented participants in software research and intersectionality in analyses, both in order to understand these unique experiences, and for continued insight into how these dynamics create or dampen developer productivity.

Study 3 Summary: Moving towards "Healthy Metrics"

Much research on developer experience has noted the dangers of measuring productivity badly. Such bad measures can dampening developer productivity altogether; metrics that make developers feel that important process and effort work is punished can cultivate a performance culture in which developers feel obliged to do busywork and perform "to the metric", focusing on the appearance of their work instead of authentic quality (e.g., Hicks, 2022; Harackiewicz et al., 2000; Zingaro, 2015). Recent research also points out dangers in measuring developer productivity such as widespread misalignments in how managers and individual contributors define productivity, and misunderstand each other's definitions (e.g., Sadowski & Zimmermann, 2019; Storey, Houck, & Zimmermann, 2022).

In Study 3, we explore the idea of **healthy metrics.** Focusing on better definitions for healthy metrics is an important path forward for software teams. In healthy metrics use, measurement and tracking can be seen as the start of a conversation rather than the entire conversation. Nearly every participant in our qualitative research made the point that metrics never capture all their important work. One senior developer in our qualitative research described it this way: *"there are things which couldn't be captured in [software] metrics which I believe personally matter tremendously... like the research work that you are doing for one of the projects. I cannot capture day-to-day research work [in standardized metrics]."*

Yet incomplete measures can still be highly informative, particularly when they provide unique insight over time and drive reflection for teams and individuals. While software metrics are not used by most teams, and many developers report concern with how their teams use metrics, most developers who are on software teams that use metrics agree that this use is healthy. In Study 1, we found that healthy metrics usage had a significant relationship with developers' self-reported thriving and visibility. In Study 2, we found that managers and developers both struggled to tie engineering impact to outcomes, but benefitted from doing so.

Across Study 3, we found uneven and uncertain usage of metrics, but also that most developers were supportive of quantifying and tracking their coding process. We did not find any attributes which significantly predicted which developers were more likely to agree that their teams "use the right metrics for use." Overall, metrics usage is most likely driven by managerial choices and group processes such as conventions of sprint planning, not individual developer preferences. This means that the structural manager and leadership decisions around how software teams are measured are even more important to get right, in order to maximize beneficial impact on developer productivity.

As described in Study 2, the responsibility to track and communicate coding effort and impact weighed on managers and leaders. Speaking of what they would define as their own success as an engineering manager, one participant in our qualitative research said: *"it's both helping [developers] grow and be successful from their own personal ends, and then helping them contribute to the business in a way that the business can see as well."* Yet another manager concluded: *"There could even be some folks in the business, marketing or sales, who don't even know that, you know something like [our biggest engineering effort] is happening, you know. They just [don't] see that part of it."*

Recommendations from Study 3

Finding	Recommendation	Potential Impact
Many teams report inconsistent use of metrics	**Managers** should audit whether measurement practices are being applied consistently across teams and planning cycles, and assess whether developers experience friction in how their work is measured **Developers** should share which metrics they find most useful, and report & discuss inaccuracies of measurement as a team **Organizations** should take steps to ensure that engineering effort is captured across time and shared to decision makers	⬆ Developer Recognition ⬆ Collaboration ⬆ Technical Roadmapping ⬇ Missed Org Targets ⬇ Inaccurate benchmarking between engineering teams
Many developers feel positively about the benefits of tracking their coding work, but this benefit is mitigated by trust and transparency in	**Developers** should consider how thoughtful measures can be used amplify and document collaboration, celebrate upskilling, and encourage junior teammates' progress	⬆ Trust ⬆ Developer Experience ⬇ Implementation failures for new initiatives

the organization	**Managers** should invite developers into decisions about measuring progress between teams, and clarify expectations for measurement between different types of engineering work	
Racially minoritized developers may report stronger desire for tracking their coding work	**Organizations** should include an examination of how performance and technical work is measured and made visible among the concerns of focus in diversity, equity, inclusion & belonging initiatives, reflecting the potential differences of experience for minoritized developers	↑ Employee Satisfaction ↑ Employee Retention ↑ Equity & Inclusion

Table 10. Study 3 Recommendations

TAKE-AWAYS & CONCLUSION

In this report, we've shared key factors that we believe work together to create an ecosystem for sustainable, high-quality productivity inside of engineering organizations. Understanding what truly drives developer productivity is challenging, but critical. Whether engineering organizations are working on the right things and doing it well is a key priority for businesses, engineering leaders, and developers themselves. Our studies across 1200+ developers reveal important stories about developer productivity:

- In Study 1, we presented both Developer Thriving & Visibility as a framework to understand the core facets of 1) good problem-solving environments for software teams that enable innovation and 2) the organizational factors that increase developer motivation via the recognition of engineering effort and impact. The factors of **Developer Thriving**, **Visibility**, and **Healthy Metrics** were all shown to significantly predict greater productivity. In Table 11, we summarize examples of developer experiences that either lifted or lowered the four constructs in Developer Thriving.

- In Study 2, **truly understanding engineering effort** and **advocacy** were significant aspects to how software teams thought about meaningful visibility cycles and software success. In interviews and focus groups, developers broadened the idea of individual developer satisfaction by sharing the impact of whether their technical work was known and valued by others. Visibility brings attention to how what is happening outside of software engineers' individual work and their immediate teams changes developers' perceptions, motivation and planning. Our findings emphasized the difficulty and uncertainty that many developers face in finding this visibility, and the fragile nature of relying only on individual managers to sustain it for their teams.

- In Study 3, we surveyed **how developers are measuring their work**. We found more evidence that many developers find strong positive benefits in tracking coding work. But developers are also highly cognizant of the fact that metrics are used inside of organizations, and that trust, collaboration, and authenticity distinguishes healthy metrics usage from inauthentic and piece-meal approaches.

Developers feel the strain of making their work "visible." Our qualitative research surfaced a critical disconnect that many developers struggle to navigate, and our investigation of measurement shows that many developers are uncertain about how their work is measured. For early career developers, managers, and tech leads across industries, it remains difficult to ground engineering work in real business impact, to gather meaningful data about their work over time, and to understand when to calibrate engineering investments and readjust course in response to changing priorities.

Increasing visibility may be a lever for big impact. Reminiscing about early career, one manager in our qualitative study described the impact of "a really good boss" who involved them in a tangible demo with real customers, remembering it as *"quite empowering, because you know the end user of this product."*

Even though thoughtful and fit-for-purpose measurement may help to increase developers' sense of the value of their work and give software teams a tool for increasing the visibility of engineering's impact in a business, fewer than one in four developers in our quantitative sample of 1200+ reported being on a team that consistently used software metrics. Focusing on models for "healthy measurement" may have a deep impact on developer's individual cadence of work and software teams' collaborative problem solving. Metrics can work to increase multiple forms of awareness for engineering work: within-team awareness of progress over time, between-team recognition and comparison, and overall accuracy in organization awareness of engineering efforts. Thoughtful measurement can provide concrete data to developers and managers to champion and advocate their performance and progress, thus improving satisfaction and thriving.

One developer shared that joint clarity and collaboration around measurement was a reflective process, as opposed to using metrics as a "weapon" to punish: *"[good managers] walk the talk. [My manager] is looking at metrics to figure out where I'm blocked and where I can improve...he's not weaponizing them either. I think he made that clear that's not his goal."* Connecting true engineering effort to measures that feel valid, effort-oriented, and useful to developers' active problem-solving is a core differentiator for "healthy measurement." And for managers and leaders, tying engineering impact to business impact is the highest priority. As one manager put it, the ultimate form of developer productivity and success is real-world impact: *"It comes down to, are you able to have an impact on the product? That company? if you believe in the mission of that company. Have you been able to drive that forward?"*

Lifting or Lowering Developer Thriving: Examples across our research	
Construct in Developer Thriving	Examples participants gave
⬆ Lift Agency	Team and org-level recognition of developer-driven initiatives, such as code “clean up” Teams and managers hold regular conversations about goals and success definitions, acknowledging that software teams work towards multiple goals Weave measurement into teams’ existing software rituals and give developers a platform for adding context to changes on metrics
⬇ Lower Agency	Metrics used to evaluate all engineering work that are only appropriate for some, e.g., failing to recognize constraints of legacy systems for some teams Abrupt and unpredictable disruptions to developers’ work and process, such as frequent initiative redirections Lack of developer-focused documentation to aid onboarding into new codebases and unfamiliar parts of the codebase Rigidly defining success across developers without considering environmental factors such as experience, team needs, and organizational friction Business decision-making that fails to recognize there may be multiple technical approaches in a given problem space, and punishes developers for not choosing “the right” approach before evaluation and testing
⬆ Lift Motivation & Self-efficacy	Developers track parts of their coding process over time and reflect on whether their cadence of work is blocked in surprising ways Junior developers are given frequent and attainable “wins” when onboarding to a new team and codebase Out-of-the-box or unexpected problem-solving is noticed and celebrated by tech leads and other influential senior colleagues
⬇ Lower Motivation & Self-efficacy	When team-level planning does not adapt to account for and reward moments of solving unexpectedly difficult code effort, such as triaging and debugging Assigning large, complex, or vague tasks without providing adequate support, time, or resources Persistent scope creep and rapid change of metrics used to evaluate “good work” during planning moments vs evaluation moments Infrequent manager check-ins, requiring many “restarts” and inefficiencies in developers communicating their progress

⬆ Lift Learning Culture	Team encourages paired programming and mobbing and shares credit for "support" work between developers A reliable cadence of code review practices that encourage meaningful feedback between senior and junior developers A manager or tech lead giving a developer space pursue a new technological approach even if it does not match previous conventions Team retrospectives that track, validate, and share examples of learning new skills Post-mortem rituals that include tracking improvements made in the future in response to identified problems
⬇ Lower Learning Culture	Discouragement of documentation and other forms of long-term knowledge sharing Emphasizing productivity metrics out of context that only reflect quantity, discouraging the recognition of quality or effort Failure to acknowledge that "time cost" determining a potential solution dead-end is sometimes a necessary contribution in technical work Senior developers/tech leads uncertain whether their time spent on mentorship is visible "counts" to managers
⬆ Lift Support & Belonging	Teams celebrating unexpected contributions and new approaches from developers Developers have the opportunity to spend informal, social time together Teams and organizations ensure that diversity in backgrounds, career levels, and experience are represented throughout leadership opportunities, such as speaking roles in high impact presentations Leaders model a culture of "thoughtful" measurement for team success which prioritizes accuracy and transparency
⬇ Lower Support & Belonging	Secret "rules," unclear or inconsistent expectations and norms in code work, such as unnecessarily punitive code reviews Promotions and recognitions that developers feel skew towards rewarding certain types of engineering roles over others Developers believing that only certain backgrounds or tenure in the organization are "allowed" org-level visibility

Table 11. Selected examples of ways engineering organizations can lift or lower the "virtuous cycles" of Developer Thriving.

REFERENCES

Ajzen, I. (1991). The theory of planned behavior. *Organizational Behavior and Human Decision Processes, 50,* 179-211. https://doi.org/10.1016/0749-5978(91)90020-T

Anderson-Butcher, D., & Conroy, D. E. (2002). Factorial and criterion validity of scores of a measure of belonging in youth development programs. *Educational and Psychological Measurement, 62*(5), 857–876. https://doi.org/10.1177/001316402236882

Arel-Bundock, V. (2022). "modelsummary: Data and Model Summaries in R." *Journal of Statistical Software, 103*(1), 1–23. https://doi.org/10.18637/jss.v103.i01

Baddoo, N., Hall, T., & Jagielska, D. (2006). Software developer motivation in a high maturity company: A case study. *Software Process: Improvement and Practice*, *11*(3), 219–228. https://doi.org/10.1002/spip.265

Bandura, A., & Adams, N. E. (1977). Analysis of self-efficacy theory of behavioral change. (1977). *Cognitive Therapy and Research, 1*(4), 287-310. https://psycnet.apa.org/doi/10.1007/BF01663995

Beecham, S., Baddoo, N., Hall, T., Robinson, H., & Sharp, H. (2008). Motivation in Software Engineering: A systematic literature review. *Information and software technology*, *50*(9-10), 860-878. https://doi.org/10.1016/j.infsof.2007.09.004

Billett, S. (2011). Subjectivity, self and personal agency in learning through and for work. In M. Malloch, L. Cairns, K. Evans & B. O'Connor (Eds.), *The SAGE handbook of workplace learning* (pp. 60-72). Los Angeles, CA: SAGE

Bornstein, P.H., Hamilton, S.B. & Bornstein, M.T. (1986) Self-monitoring procedures. In A.R. Ciminero, K.S. Calhoun, & H.E. Adams (Eds) Handbook of behavioral assessment (2nd ed). New York: Wiley.

Bouwers, E., Visser, J., & Van Deursen, A. (2012). Getting what you measure: Four common pitfalls in using software metrics for project management. *ACM Queue*, *10*(50), 50-56. https://doi.org/10.1145/2208917.2229115

Bouwers, E., van Deursen, A., & Visser, J. (2013). Software metrics: pitfalls and best practices. *35th International Conference on Software Engineering (ICSE)*, 1491-1492.

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology, 3*(2), 77–101. https://doi.org/10.1191/1478088706qp063oa

Cheng, I., Murphy-Hill, E., Canning, M., Jaspan, C., Green, C., Knight, A., Zhang, N., & Kammer, E. (2022). What improves developer productivity at Google? Code quality. *30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '22)*. https://doi.org/10.1145/3540250.3558940

Cohen, J.S., Edmunds, J.M., Brodman, D.M., Benjamin, C.L., Kendall, P.C. (2013), Using self-monitoring: implementation of collaborative empiricism in cognitive-behavioral therapy. *Cognitive and Behavioral Practice, 20*(4), 419-428. https://psycnet.apa.org/doi/10.1016/j.cbpra.2012.06.002

Cole, E. R. (2009). Intersectionality and research in psychology. *American Psychologist, 64*(3), 170.

Craig, W. (2018, October 16). *10 things transparency can do for your company*. Forbes. Retrieved February 15, 2023, from https://www.forbes.com/sites/williamcraig/2018/10/16/10-things-transparency-can-do-for-your-company/?sh=7bab55e625d0

Dawson, L., Mullan, B., & Sainsbury, K. (2015). Using the theory of planned behaviour to measure motivation for recovery in anorexia nervosa. *Appetite, 84*, 309-315. https://doi.org/10.1016/j.appet.2014.10.028

Eccles, J., Adler, T. F., Futterman, R., Goff, S. B., Kaczala, C. M., et al. 1983. Expectancies, values, and academic behaviors. In J. T. Spence (Ed.), *Achievement and Achievement Motivation* (pp. 75–146). San Francisco: Freeman

Ehlers, A., Clark, D.M., Hackmann, A., McManus, F., Fennell, M., Herbert, C., & Mayou, R. (2003). A Randomized Controlled Trial of Cognitive Therapy, a Self-help Booklet, and Repeated Assessments as Early Interventions for Posttraumatic Stress Disorder. *Archives of General Psychiatry, 60*(10), 104-1032. https://doi.org/0.1001/archpsyc.60.10.1024

Fagerholm, F., & Münch, J. (2012, June). Developer experience: Concept and definition. In *2012 international conference on software and system process (ICSSP)* (pp. 73-77). IEEE.

Forsgren, N., Storey, M. A., Maddila, C., Zimmerman, T., Houck, B., & Butler, J. (2021). The SPACE of Developer Productivity: There's more to it than you think. *Queue, 19*(1), 20-48. https://doi.org/10.1145/3454122.3454124

Foster, S. L., Laverty-Finch, C., Gizzo, D. P., & Osantowski, J. (1999). *Practical issues in self-observation. Psychological Assessment, 11(4), 426–438.* https://doi.org/10.1037/1040-3590.11.4.426

Gobeli, D. H., Koenig, H. F., & Bechinger, I. (1998). Managing conflict in software development teams: A multilevel analysis. *Journal of Product Innovation Management: An International Publication of the Product Development & Management Association, 15*(5), 423-435. https://doi.org/10.1111/1540-5885.1550423

Greiler, M., Storey, M. A., & Noda, A. (2022). An actionable framework for understanding and improving developer experience. *IEEE Transactions on Software Engineering*. https://doi.org/10.48550/arXiv.2205.06352

Gren, L. (2018). On gender, ethnicity, and culture in empirical software engineering research. *11th International Workshop on Cooperative and Human Aspects of Software Engineering*, 77-78. https://doi.org/10.1145/3195836.3195837

Hackman, J. R., Oldham, G. R. (1976). Motivation through the design of work: Test of a theory. Organizational Behavior and Human Performance, 16, 250-279. https://doi.org/10.1016/0030-5073(76)90016-7

Harackiewicz, J. M., Barron, K. E., Tauer, J. M., Carter, S. M., & Elliot, A. J. (2000). Short-term and long-term consequences of achievement goals: Predicting interest and performance over time. *Journal of Educational Psychology*, *92*(2), 316. https://psycnet.apa.org/doi/10.1037/0022-0663.92.2.316

Hayes, A. F. (2022). Introduction to mediation, moderation, and conditional process analysis. Guilford Press.

Hicks, C. It's Like Coding in the Dark: The need for learning cultures within coding teams [White Paper], Catharsis Consulting. [https://www.catharsisinsight.com/reports]

Jason, L. (1975). Rapid improvement in insomnia following self-monitoring. *Journal of Behavior Therapy and Experimental Psychiatry, 6*(4), 349–350. https://doi.org/10.1016/0005-7916(75)90079-8

Johnson, D. (2022, September 12). *The importance of managers as advocates*. MIT Sloan Management Review. Retrieved February 15, 2023, from https://sloanreview.mit.edu/article/the-importance-of-managers-as-advocates/

Johnston, K. L., & White, K. M. (2003). Binge-drinking: A test of the role of group norms in the theory of planned behaviour. *Psychology & Health, 18*(1), 63-77. https://doi.org/10.1080/0887044021000037835

Kassambara, A., & Patil, I. R package Visualization of a Correlation Matrix using 'ggplot2'. http://www.sthda.com/english/wiki/ggcorrplot-visualization-of-a-correlation-matrix-using-ggplot2

Kavanagh, D. J., Sitharthan, T., Spilsbury, G. Vignaendra, S. (1999). An evaluation of brief correspondence programs for problem drinkers. *Behavior Therapy, 30*(4), 641–656. https://doi.org/10.1016/S0005-7894(99)80030-6

Kim, Y. E., Yu, S. L., Wolters, C. A., & Anderman, E. M. (2023). Self-regulatory processes within and between diverse goals: The multiple goals regulation framework. *Educational Psychologist*, 1-22. https://doi.org/10.1080/00461520.2022.2158828

Korotitsch, W. J., & Nelson-Gray, R. O. (1999). An overview of self-monitoring research in assessment and treatment. *Psychological Assessment, 11*(4), 415. https://psycnet.apa.org/doi/10.1037/1040-3590.11.4.415

Kronk, C. A., Everhart, A. R., Ashley, F., Thompson, H. M., Schall, T. E., Goetz, T. G., Hiatt, L., Derrick, Z., Queen, R., Ram, A., Guthman, E. M., Danforth, O. M., Lett, E., Potter, E., Sun, S. D., Marshall, Z., Karnoski, R. (2022). Transgender data collection in the electronic health record: Current concepts and issues. *Journal of the American Medical Informatics Association, 29*(2), 271–284, https://doi.org/10.1093/jamia/ocab136

Lagos D, & Compton D. (2021). Evaluating the use of a two-step gender identity measure in the 2018 General Social Survey. *Demography, 58*(2), 763-772. https://doi.org/10.1215/00703370-8976151

Lambert, M. J., Hansen, N. B., & Finch, A. E. (2001). Patient-focused research: Using patient outcome data to enhance treatment effects. *Journal of Consulting and Clinical Psychology, 69*(2), 159–172. https://doi.org/10.1037/0022-006X.69.2.159

Latner, J. D., & Wilson, G. T. (2002). Self-monitoring and the assessment of binge eating. *Behavior Therapy, 33*(3), 465–477. https://doi.org/10.1016/S0005-7894(02)80039-9

Leary, M. R., Patton, K. M., Orlando, E., and Funk, W. W. (2000). The impostor phenomenon: self-perceptions, reflected appraisals, and interpersonal strategies. *Journal of Personality, 68*, 725–756. https://doi.org/10.1111/1467-6494.00114

Lüdecke, D. (2022). *sjPlot: Data Visualization for Statistics in Social Science*. R package version 2.8.12, https://CRAN.R-project.org/package=sjPlot

Luxton-Reilly, A., Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., ... & Szabo, C. (2018). Introductory programming: a systematic literature review. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education* (pp. 55-106). https://doi.org/10.1145/3293881.3295779

Lynn, P. (2018). Tackling panel attrition. The Palgrave handbook of survey research, 143-153.

Mak, K. K. L., Kleitman, S., & Abbott, M. J. (2019). Impostor phenomenon measurement scales: A systematic review. *Frontiers in Psychology, 10*. https://doi.org/10.3389/fpsyg.2019.00671

Meyer, A. N., Zimmermann, T., & Fritz, T. (2017). Characterizing software developers by perceptions of productivity. *IESEM '17: Proceedings of the 11th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement,* 105-110. https://doi.org/10.1109/ESEM.2017.17

Meyer, A. N., Barr, E. T., Bird, C., & Zimmermann, T. (2019). Today was a good day: The daily life of software developers. IEEE Transactions on Software Engineering, 47(5), 863-880. https://doi.org/10.1109/TSE.2019.2904957

Meyer, A. N., Murphy, G. C., Zimmermann, T., & Fritz, T. (2019). Enabling good work habits in software developers through reflective goal-setting. *IEEE Transactions on Software Engineering*, *47*(9), 1872-1885. https://doi.org/10.1109/TSE.2019.2938525.

Mikkonen, T. (2016). Flow, intrinsic motivation, and developer experience in software engineering. *Agile Processes in Software Engineering and Extreme Programming*, 104.

Morales, J., Rusu, C., Botella, F., & Quiñones, D. (2019). Programmer eXperience: A systematic literature review. *IEEE Access*, *7*, 71079-71094.

O'Flaherty, S., Mitchel, C., Sanders, M., Walker, E., Unger, D., Wilans, A., & Daniels, K. (2022). Happier, Healthier Professionals: Phase Two: RCTs and Pilots Conducted with Public Sector Workforces. *What Works Centre for Children's Social Care*. https://ueaeprints.uea.ac.uk/id/eprint/83176/1/Published_Version.pdf

Pardede, S., Gausel, N., Høie, M. M. (2021). Revisiting the “The Breakfast Club”: Testing different theoretical models of belongingness and acceptance (and social self-representation). *Frontiers in Psychology, 11*. https://doi.org/10.3389/fpsyg.2020.604090

Petre, M. (2009). Insights from expert software design practice. *Proceedings of the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, 233-242, https://doi.org/10.1145/1595696.1595731

Quadlin, N. (2018). The mark of a woman's record: Gender and academic performance in hiring. *American Sociological Review*, *83*(2), 331-360, https://doi.org/10.1177/0003122418762

Rattan, A., Savani, K., Komarraju, M., Morrison, M. M., Boggs, C., & Ambady, N. (2018). Meta-lay theories of scientific potential drive underrepresented students' sense of belonging to science, technology, engineering, and mathematics (STEM). *Journal of Personality and Social Psychology*, *115*(1), 54. https://psycnet.apa.org/doi/10.1037/pspi0000130

Reel, J. S. (1999). Critical success factors in software projects. *IEEE software*, *16*(3), 18-23. https://doi.org/https://doi.org/10.1109/52.765782

Roberson, L., & Kulik, C. T. (2007). Stereotype threat at work. *Academy of Management Perspectives*, *21*(2), 24-40.

Robinson, K. A., Lee, Y.-k., Bovee, E. A., Perez, T., Walton, S. P., Briedis, D., & Linnenbrink-Garcia, L. (2019). Motivation in transition: Development and roles of expectancy, task values, and costs in early college engineering. *Journal of Educational Psychology, 111*(6), 1081–1102. https://doi.org/10.1037/edu0000331

Rodríguez-Pérez, G., Nadri, R., & Nagappan, M. (2021). Perceived diversity in software engineering: a systematic literature review. *Empirical Software Engineering, 26,* 1-38. https://doi.org/10.1007/s10664-021-09992-2

Roemer, L. & Orsillo, S. M. (2009). *Mindfulness- and acceptance-based behavioral therapies in practice*. New York, NY: Guilford Press.

Roller, M. R. & Lavrakas, P. J. (2015). *Applied qualitative research design: A total quality framework approach.* Guilford Press.

Rosseel, Y. (2012). lavaan: An R package for structural equation modeling. *Journal of Statistical Software, 48*(2), 1–36. https://doi.org/10.18637/jss.v048.i02

Sadowski, C., & Zimmermann, T. (2019). *Rethinking productivity in software engineering*. Springer Nature.

Scott, M. J., & Ghinea, G. (2013). On the domain-specificity of mindsets: The relationship between aptitude beliefs and programming practice. *IEEE Transactions on Education*, *57*(3), 169-174. http://dx.doi.org/10.1109/TE.2013.2288700

Sherer, M., Maddux, J. E., Mercandante, B., Prentice-Dunn, S., Jacobs, B., & Rogers, R. W. (1982). The self-efficacy scale: Construction and validation. *Psychological reports*, *51*(2), 663-671. https://psycnet.apa.org/doi/10.2466/pr0.1982.51.2.663

Simard, C., Henderson, A., Gilmartin, S., Schiebinger, L., Whitney, T., (2008). *Climbing the technical ladder: Obstacles and solutions for mid-career women in tech.* https://gender.stanford.edu/publications/climbing-technical-ladder-obstacles-and-solutions-mid-level-women-technology

Spencer, S. J., Steele, C. M., & Quinn, D. M. (1999). Stereotype threat and women's math performance. *Journal of Experimental Social Psychology, 35*(1), 4–28. https://doi.org/10.1006/jesp.1998.1373

Stecker, T., McGovern, M. P., & Herr, B. (2012). An intervention to increase alcohol treatment engagement: A pilot trial. J*ournal of Substance Abuse Treatment, 43*(2), 161-167. https://doi.org/10.1016/j.jsat.2011.10.028

Steele, C. M., & Aronson, J. (1995). Stereotype threat and the intellectual test performance of African Americans. *Journal of Personality and Social Psychology, 69*(5), 797–811. https://doi.org/10.1037/0022-3514.69.5.797

Storey, M. A., Zimmermann, T., Bird, C., Czerwonka, J., Murphy, B. & Kalliamvakou, E. (2021). Towards a theory of software developer job satisfaction and perceived productivity. IEEE Transactions on Software Engineering, *47*(10), 2125–2142. https://doi/org/10.1109/TSE.2019.2944354

Storey, M. A., Houck, B., & Zimmermann, T. (2022). How developers and managers define and trade productivity for quality. *CHASE '22: Proceedings of the 15th International Conference on Cooperative and Human Aspects of Software Engineering*, 26-35. https://doi.org/10.1145/3528579.3529177

Verner, J., Sampson, J., & Cerpa, N. (2008). What factors lead to software project failure?. *2008 second international conference on research challenges in information science*, 71-80. https://doi.org/10.1109/RCIS.2008.4632095

Wadsworth, L. P., Morgan, L. P., Hayes-Skelton, S. A., Roemer, L., & Suyemoto, K. L. (2016). Ways to Boost Your Research Rigor Through Increasing Your Cultural Competence. *The Behavior Therapist, 39,* 76-92.

Wallace, L. G., & Sheetz, S. D. (2014). The adoption of software measures: A technology acceptance model (TAM) perspective. *Information & Management*, *51*(2), 249-259. https://doi.org/10.1016/j.im.2013.12.003

Wang, H. L. (2023). New 'Latino' and 'Middle Eastern or North African' checkboxes proposed for U.S. forms. NPR: All Things Considered. Retrieved February 15, 2023 from: https://www.npr.org/2023/01/26/1151608403/mena-race-categories-us-census-middle-eastern-latino-hispanic

Wickham et al., (2019). Welcome to the tidyverse. *Journal of Open Source Software, 4*(43), 1686, https://doi.org/10.21105/joss.01686

Wilke, C. O., & Wiernik, B. M. (2022). *R Package* ggtext: Improved Text Rendering Support for 'ggplot2'. https://wilkelab.org/ggtext/

Wilson, D. M., Bell, P., Jones, D., & Hansen, L. (2010). A cross-sectional study of belonging in engineering communities. *The International Journal of Engineering Education, 26*(3), 687-698.

Yeager, D., Walton, G., & Cohen, G. L. (2013). Addressing achievement gaps with psychological interventions. *Phi Delta Kappan*, *94*(5), 62-65. http://dx.doi.org/10.1177/003172171309400514

Zingaro, D. (2015). Examining interest and grades in Computer Science 1: a study of pedagogy and achievement goals. *ACM Transactions on Computing Education (TOCE), 15*(3), 1-18. https://doi.org/10.1145/2802752

Supplemental Materials

APPENDIX A. Sample Participant Consent Form

Developer Success Lab
This survey is from the Developer Success Lab at Flow. We are interested in learning more about developers at work. This survey is a part of our ongoing research. By participating in this survey, you are giving consent for us to use your survey responses in our ongoing research, including public reports, white papers, and conference presentations. Please read this consent form carefully to ensure you understand how your data will be used.

It is important to know that there are no wrong or right answers on this survey. Our research team is interested in hearing your most authentic answers to any question.

Participant data & analysis
The Developer Success Lab cares deeply about participant privacy. Your responses on this survey will be analyzed only by our researchers. In order to learn from this data, researchers will have access to everything that you share on this survey. Please do not share anything you do not wish to share with our research team.

Findings
Our research is communicated to an external audience as part of Flow's contributions to the tech community, where we believe research findings can benefit developers and their teams (for example, conference presentations, public white papers). When we share research, all findings will be anonymized, removing any names or specific team contexts. Quantitative insights will be summarized in aggregate as part of a large research report. For example, we may share statements like: "10% of respondents agreed that..."

Any quotes and text responses that you give as a part of this research may also be shared in our external reports and may be quoted directly. We will anonymize all quotes, removing any specific mention of teams, names, or contexts that might be personally identifiable. Where necessary, we may provide anonymized contextual details (for example, "a senior engineer working primarily on backend team noted that....").

Questions, Concerns, and Opt-out.
While we hope that you will enjoy completing this survey and sharing your insights with us, you may choose to leave this survey at any time. You may also leave any questions blank that you do not wish to answer.

If you have any questions or concerns you can reach out to the Developer Success Lab directly [EMAIL]

APPENDIX B. Asking about Identity

We chose to ask about demographic information such as race, gender, and sexual orientation to better describe, represent, and contextualize our participants. Although these categories do not fully capture the complexities of each individual's experience, they were an attempt to reflect the diversity of people's identities. Participants were also reminded that they could skip items they did not feel comfortable answering.

Racial Identity

When asking about racial identity, we chose to utilize a "check all that apply" approach that included a free-text response option. This approach creates some structure for coding purposes, while providing participants greater freedom in how they identify. While a case could be made that simply providing the options of "multiracial/biracial" is sufficient, we wanted to reflect that the biracial and multiracial experiences are distinct and may not encompass how participants are racialized (Wadsworth et al., 2016). That is, people may identify as holding multiple racial identities, but not necessarily identify as "multiracial."

We also asked participants about their "racial/ethnic" identity. While racial identity is distinct from ethnicity, we chose to include ethnicity in order to capture ethnicities that have been racialized (e.g. Native Hawaiian).

Additionally, we split our racial categories of "Latinx/Hispanic" and "Middle Eastern/ North African" into subcategories of "white" and "non-white." This was to allow space for individuals who may be racialized as white by others (and are typically forced to identify as white in national census data; Wang, 2023), but are systemically minoritized based on factors such as cultural practices, appearance of family members, and name.

Racial Identity Question
[OPTIONAL] Which group(s) below most accurately describes your racial/ethnic background? (check all that apply) ☐ Alaskan Native/Native American/Indigenous ☐ Black/African American ☐ East Asian ☐ Middle Eastern/North African (Non-White) ☐ Middle Eastern/North African (White) ☐ Latinx/Hispanic (Non-White) ☐ Latinx/Hispanic (White) ☐ Pacific Islander/Native Hawaiian ☐ South/South-East Asian ☐ White ☐ Multiracial ☐ I would like to self-identify: _____

☐ Prefer not to answer

Finally, throughout the report, we used the term “racially minoritized.” The use of this term is consistent with best practices in social science research and best reflects the systemic ways in which people are treated as inferior or deficit based on the way they are racialized by others, despite being the global majority. We chose not to use the term “marginalized,” as it can imply a deficit narrative and can be stigmatizing. We also chose not to use the term “under-represented,” because it ignores the experiences of those who may be well-represented in tech, yet be systemically and socially minoritized by others. Finally, we opted not to use the term “people of color,” as it has been historically viewed as inaccessible to Native/Indigenous, Asian, and Latinx-identifying individuals.

Gender Identity and Sexual Orientation

We chose to ask about gender and transgender identity separately. This “two-step” approach was intentional and is the current recommended approach for asking about gender identity (Kronk et al., 2022). This approach also avoids asking individuals to “qualify” gender identity (e.g. forcing a choice between “transgender woman” and “woman”), which is not only inaccurate, but stigmatizing. This approach also further avoids conflating gender and sex assigned at birth (Lagos & Compton, 2021). We asked about transgender identity to reflect and acknowledge the additional barriers gender minorities face in the workplace.

We asked about sexual orientation using a “check all that apply” approach that included a free-text response option. This approach was to better reflect the fluid nature of sexual orientation.

Gender Identity Questions
[OPTIONAL] Gender: ☐ Male ☐ Female ☐ Nonbinary/Fluid/Queer/Gender Queer; ☐ I would like to self-identify: _____ ☐ Prefer not to answer
[OPTIONAL] Do you identify as transgender? ☐ Yes ☐ No ☐ Prefer not to answer

Sexual Orientation Question

[OPTIONAL] Which group(s) below most accurately describes your racial/ethnic background? (check all that apply)

- ☐ Asexual/ Aromantic
- ☐ Bisexual
- ☐ Fluid
- ☐ Gay
- ☐ Lesbian
- ☐ Pansexual
- ☐ Queer
- ☐ Questioning or unsure
- ☐ Straight/ Heterosexual
- ☐ I would like to self-identify: _____
- ☐ Prefer not to answer

APPENDIX C. Example Qualitative Script

Interviews and focus groups were both semi-structured conversations: this means that we use a set of scripted questions, but allow for dialogue with participants, who can share tangents, observations, and unexpected topics as they arise in the course of participants' reflection. That means that researchers used the following example questions as *initiating* items, but researchers asked natural follow-up questions and probed (e.g., "can you help me understand [*x*]" or "when you experienced [*x*], can you describe whether that changed how you worked…").

Initiating Questions	
Topic	Questions
Defining Success & Performance	How do you define "success"? If your manager says you were "successful" what does that mean/ look like? How does your team measure success? Are there parts of software projects that are harder to measure? How do you see what your colleagues are working on When I say software metrics, what does that mean to you Has there ever been a time when you've had to align with your manager on performance metrics (i.e. misunderstanding, needing clarification, changing scope)? What prompted that alignment conversation? What was the result of that conversion? (Short term/long term) Did you feel that the conversation affected your motivation? If so, how? Did you feel that conversation affected your perception of your performance? If so, how?
Collaboration	What is the role of "collaboration" in your world? Can you walk me through a time when you had to "collaborate" while on the job? One of the things we've heard people mention is that feeling like your managers really SEES and UNDERSTANDS your technical work in code can be super important to feeling successful as a developer." Does this resonate with you? Are there ways you don't agree? Can you tell me about a time your workflow changed as a result of someone really seeing and understanding your work?

Safety	Something we've heard from developers is the importance of being able to experiment and iterate when they're writing code. Sometimes we've also heard that if developers don't feel very "safe" on a team, they find it hard to do this. Does this resonate with you? Are there ways you don't agree? Can you tell me about a time your workflow changed as a result of someone making you feel safe?

APPENDIX D. Additional statistics

Productivity between industries. In our main analysis in Study 1, we explored Productivity across all developers in our study. Looking specifically *within* the top four industries in our study, the relationship between the Developer Thriving Measures and developers' Productivity remained significant and positive. However, between industries, developers reported differing levels of productivity. This is not surprising, due to the different circumstances, technology resources, and likely distributions of types of engineering work in these industries. Future research should continue to explore developer productivity *within* industries, particularly to better understand the experience of software teams outside of large technology companies.

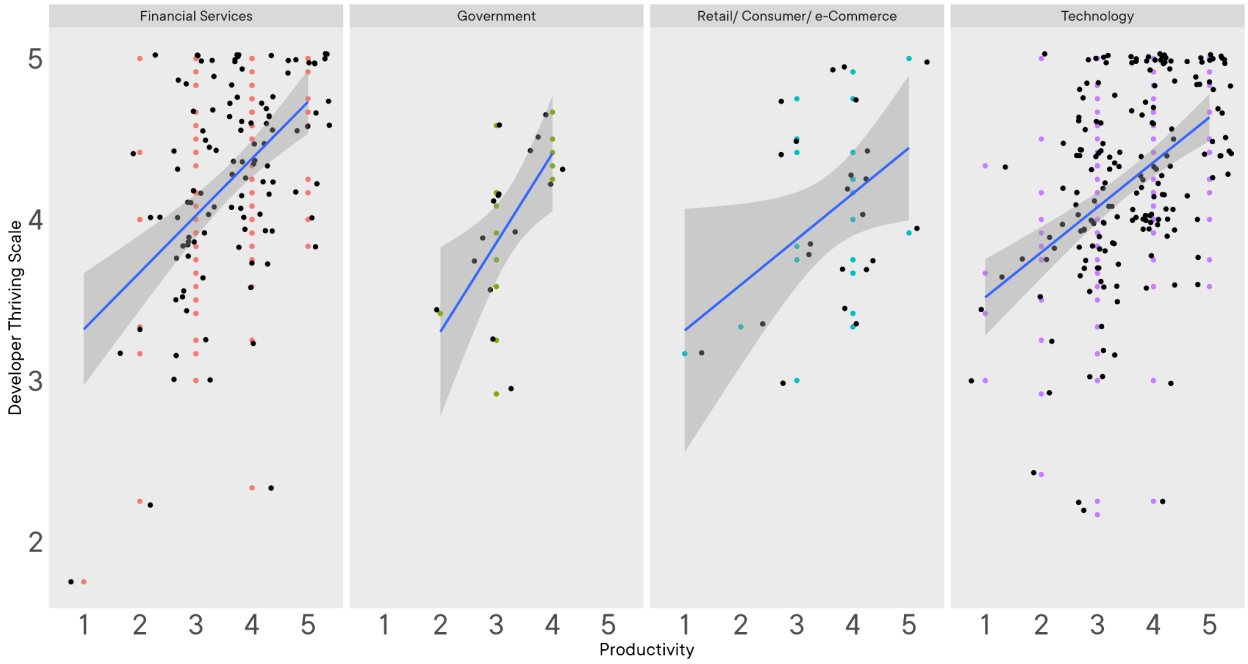

Fig 15. Relations between Developer Thriving Productivity across industries

Overall descriptives of key measures & item analysis. In the tables below, we show descriptive statistics from our quantitative survey, as well as the full details for the Serial Mediation Model reported in Study 1. Because of the limitations of running a public-facing survey at scale, and in order to prioritize participants' ability and willingness to complete a large survey, the Developer Thriving Scale presented an *abbreviated* set of subscales for each

construct. This challenge is common in applied research (e.g., Lynn, 2018). Future research can build on these subscales with techniques to increase the *depth* of measurement (e.g., surveying with a larger subscale for each construct to create more statistically reliable subscales) as well as *breadth* (e.g., surveying over time using within-group repeated measures, to better capture the stability or variability in these aspects of developer experience and behavior).

Overall Descriptives of Key Measures

Variable	Mean (SD)	Skewness	Kurtosis	HMU	VVQ	DSS	PPR
HMU *n* = 958	1.23 (0.52)	0.03	-1.15	1.00	–	–	–
VVQ *n* = 821	3.99 (0.91)	-0.86	.27	.33*	1.00	–	–
DTS *n* = 562	4.26 (0.61)	-0.83	0.81	.34*	.73*	1.00	–
PPR *n* = 1280	3.45 (0.92)	-0.28	0.04	.26*	.41*	.43*	1.00

Note. HMU = Healthy Metrics Use; VVQ = Visibility and Value Questionnaire; DTS = Developer Thriving Scale; PPR = Perceived Productivity Rating.
*$p < .001$

Serial Mediation Model

	Consequent					
	VVQ (m1)		DTS (m2)		PPR (y)	
Antecedent	*β*	*p*	*β*	*p*	*β*	*p*
Direct Effects						
% Time Code (cov1)	0.12	< .01	0.13	< .001	0.05	0.21
Years Code (cov2)	0.17	< .001	0.05	0.15	0.16	< .001
HMU (x)	0.32	< .001	0.12	< .01	0.13	< .01
VVQ (m1)	—	—	0.65	< .001	0.14	< .05
DTS (m2)	—	—	—	—	0.24	< .001
Indirect Effects						
HMU (x) via VVQ (m1)	—	—	0.21	< .001	0.04	< .05
TMU (x) via DTS (m2)	—	—	—	—	0.03	.001
VVQ (m1) via DTS (m2)	—	—	—	—	0.16	< .001
TMU (x) via VVQ (m1) and DTS (m2)	—	—	—	—	0.05	0.001

Note. VVQ = Visibility and Value Questionnaire, DTS = Developer Thriving Scale, PPR = Perceived Productivity Rating, % Time Code = Percent of time spent coding, Years Code = Years of coding experience, HMU = Healthy Metrics Use, cov1 = covariate 1, cov2 = covariate 2, x = predictor variable, m1 = mediator 1, m2 = mediator 2, y = outcome variable

Subgroup differences on the IMCW. In Study 3, we explored whether there was evidence for differences in any of the subgroups we observed across our demographic and firmographic characteristics. As reported in the summary of Study 3, *Racially Minoritized Developers* showed a statistically significant difference on this outcome measure, reporting overall higher agreement with the Impact of Measuring Coding Work. We caution, as in Study 3, that this is an exploratory finding, and that we were unable to explore this question with our full survey sample. We recommend further research that explores this potential difference.

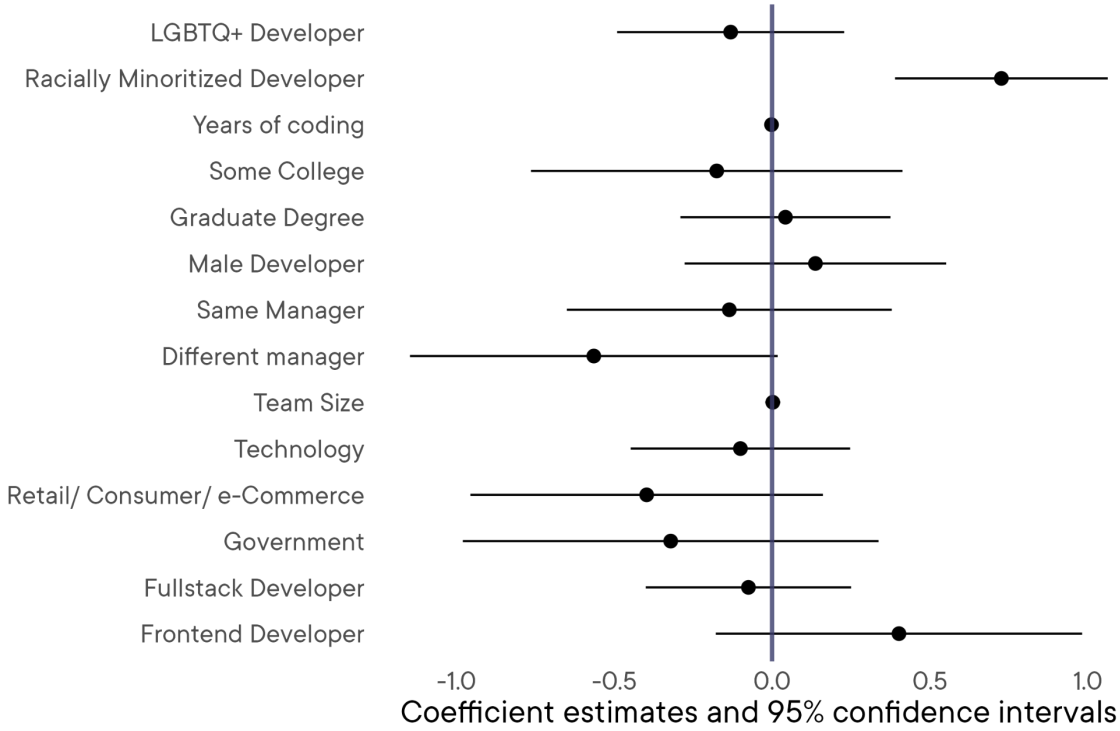

Figure 16. Coefficient estimates for subgroup differences on the impact of coding work measure.

IMCW Model

Characteristic	β	95% CI1	*p*-value
Engineering Role			
Backend	—	—	
Frontend	0.40	-0.18, 0.98	0.2
Full Stack	-0.07	-0.40, 0.25	0.6
Top Industries			
Financial Services	—	—	
Government	-0.32	-0.98, 0.34	0.3
Retail/ Consumer/ e-Commerce	-0.40	-0.96, 0.16	0.2
Technology	-0.10	-0.45, 0.25	0.6
Team Size	0.00	-0.02, 0.03	0.9
Team Type			
Different Manager	-0.56	-1.1, 0.02	0.057
Same Manager	-0.14	-0.65, 0.38	0.6
Gender			
Female	—	—	
Male	0.14	-0.28, 0.55	0.5
Prefer not to answer	0.59	-0.33, 1.5	0.2
Education Completed			
4-year College	—	—	
Graduate Degree	0.04	-0.29, 0.38	0.8
Some College	-0.18	-0.76, 0.41	0.6
Years Coding	0.00	-0.02, 0.01	0.8
Racially Minoritized Status			
Non-racially Minoritized Developer	—	—	
Racially Minoritized Developer	**0.73**	**0.39, 1.1**	**<0.001**
LGBTQ Status			
Non-LGBTQ+ Developer	—	—	
LGBTQ+ Developer	-0.13	-0.49, 0.23	0.5

Note. 1CI = Confidence Interval