

How Developers and Managers Define and Trade Productivity for Quality

Margaret-Anne Storey
University of Victoria
Victoria, BC, Canada
mstorey@uvic.ca

Brian Houck
Microsoft Corp.
Redmond, WA, USA
brian.houck@microsoft.com

Thomas Zimmermann
Microsoft Research
Redmond, WA, USA
tzimmer@microsoft.com

ABSTRACT

Background: Developer productivity and software quality are different but related multi-dimensional lenses into the software engineering process. The terms are used liberally in industry settings, but there is a lack of consensus and awareness of what these terms mean in specific contexts and which trade-offs should be considered.

Objective & Method: Through an exploratory survey study with developers and managers at Microsoft, we investigated how these cohorts define productivity and quality, how aligned they are in their views, how aware they are of other views, and if and how they trade quality for productivity.

Results: We find developers and managers, as cohorts, are not well-aligned in their views of productivity—developers think more about work activities, while more managers consider performance or quality outcomes. We find developers and managers have more aligned views of what quality means, with the majority defining quality in terms of robustness, while the timely delivery of evolvable features that delight users are also key quality aspects. Over half of the developers and managers we surveyed make productivity and quality trade-offs but with good reasons for doing so.

Conclusion: Alignment on how developers and managers define productivity and quality is essential if they are to design effective improvement interventions and meaningful metrics to measure productivity and quality improvements. Our research provides a frame for developers and managers to align their views and to make informed decisions on productivity and quality trade-offs.

CCS CONCEPTS

• Software and its engineering → Software creation and management.

KEYWORDS

productivity, quality, software development, trade-offs

ACM Reference Format:

Margaret-Anne Storey, Brian Houck, and Thomas Zimmermann. 2022. How Developers and Managers Define and Trade Productivity for Quality. In *15th International Conference on Cooperative and Human Aspects of Software*

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](https://permissions.acm.org).

CHASE'22, May 21–29, 2022, Pittsburgh, PA, USA

© 2022 Association for Computing Machinery.

ACM ISBN 978-1-4503-9342-3/22/05...\$15.00

<https://doi.org/10.1145/3528579.3529177>

Engineering (CHASE'22), May 21–29, 2022, Pittsburgh, PA, USA. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3528579.3529177>

1 INTRODUCTION

Software organizations strive to develop software quickly and efficiently, while achieving and maintaining high levels of quality. At several large companies we have worked with, we heard many discussions about how to improve “productivity” and “quality,” but these discussions, many of which aimed to propose actionable productivity and quality metrics, often lacked a clear definition of what these nuanced terms mean.

Sparked in part by the needs of industry, developer productivity has been studied extensively in software engineering research in recent years. Researchers have reported on how developers perceive productivity, sharing that developers consider productivity not just about writing code or delivering features quickly, but is also related to their ability to deliver high-quality work [18], their ability to work without interruptions [12], and is impacted by their satisfaction and well-being [22], their happiness at work [7], and many other social and technical factors [15, 25]. Despite these many research findings, how to define and measure productivity in a way that provides meaningful insights and pragmatically supports change remains challenging.

There are established formal models [16] and definitions of software quality [9], and many proposed metrics [17] for measuring system and process quality, including defects in delivered code, unplanned rework, code churn, reusability, code readability, test coverage, how code is reviewed, and customer satisfaction. It is also an ongoing debate in practitioner blogs which metrics to use¹. Despite so many different views, Kitchenham and Pfleeger note that although it is not easy to define quality, we can recognize it when we see it [9]. In our experience working in and with industry, how software developer practitioners define and think about quality in their day-to-day work varies greatly. To some, quality is about satisfying the visible functional requirements for a product, while to others it is more important to consider quality in terms of non-functional requirements, such as performance and reliability of the code or process. These different priorities for quality can lead to misaligned values and goals during the design and evolution of software, and can create tension among developers and managers when they try to balance developer productivity in terms of shipping high-quality features or completing tasks.

In this paper, we present the findings from a study at Microsoft that investigated how developers and managers define and trade

¹<https://www.bmc.com/blogs/software-quality-metrics/>,
<https://xboosoft.com/software-qa-consulting-services/definition-software-quality/>,
<https://www.testbytes.net/blog/what-is-software-quality/>

off productivity and quality. Through a survey answered by 165 participants, we consider whether developer views of productivity and quality are aligned with each other and with the views of managers. We further explore how developers and managers think about quality, whether they report making trade-offs between them, and why those trade-offs are made.

Our research has revealed that developers and managers, as cohorts, are not well-aligned in their views of what it means to be **productive**. Notably, more developers think of productivity in terms of **activities** and task completion, while more managers think of productivity in terms of **performance and quality of their outcomes**. We also found that developers are not accurate at predicting their managers' views of what it means for their team to be productive.

In terms of **quality**, we found that although individual developers and managers have quite varied views of what quality means, the two cohorts are closely aligned in these different views, with the majority in both groups defining quality in terms of software **robustness**. We also found that over half of the developers and managers that answered our survey consider quality as something that can be **traded** for higher productivity. Conversely, approx. one-third of the developers and managers that answered our survey consider quality as a necessary part of productivity. We conclude with the recommendation that productivity and quality should be considered as related but multi-dimensional lenses into the software engineering process and that there is a need for more discussion by developers and managers to align on what these terms mean in their contexts and to use this understanding to drive how they approach measuring and changing how they approach improving productivity and quality.

2 METHODOLOGY

To understand how developers and managers define and trade productivity for quality, we conducted a survey study with employees at Microsoft. Our research questions were as follows:

- RQ1** How do developers and managers define **productivity**? How **aligned** are developers and managers in their definitions of productivity?
- RQ2** Do developers and managers **understand how the other cohort defines** productivity?
- RQ3** How do developers and managers define **quality**? How **aligned** are developers and managers in their definitions?
- RQ4** Do developers and managers **trade** quality for productivity? What **reasons** do they give for trading or not trading quality for productivity?

2.1 Survey Design

To help us prepare our survey questions, we conducted an exploratory focus group with six developers and two interviews with managers. We developed the survey in an iterative manner, with a pilot helping refine the survey questions. The survey was distributed in Jan 2020. We invited 1,500 developers and 720 managers at Microsoft to participate in the survey via personalized email. The survey was advertised as a general survey on software productivity. After completion of the survey, participants could enter

a sweepstakes to win one of four \$100 Amazon.com Gift Certificates. The survey was open for two weeks, and we received 131 responses from developers and 34 responses from managers (response rates of 9% and 5%, respectively). The ethics for this study were reviewed and approved by the Microsoft Research Institutional Review Board (MSRIRB), which is an IRB federally registered with the United States Department of Health & Human Services (Reference: MSRIRB #586). The survey instruments are available as supplemental material [21]. In our initial interviews, we found that developers and managers believed they understood how each defined productivity, but they did not have a good sense of how the other cohort defined quality—so we omitted questions about how the other cohorts defined quality in our surveys.

The questions in the **developer** survey that we analyzed for this research are as follows:

- When thinking about your work, how do you define productivity?
- How do you think your manager defines productivity?
- When thinking about your work, how do you define quality?
- Have you ever had to make trade-offs between productivity and quality? If so, please explain.

The questions we asked in the **developer manager** survey are:

- When thinking about your team, how do you define productivity?
- How do you think the developers you manage define productivity?
- When thinking about your team, how do you define quality?
- Has your team had to make explicit trade-offs between productivity and quality? If so, please describe those tradeoffs.

2.2 Data Analysis

We coded the open-ended responses to the survey in an iterative and inductive manner, where two paper authors coded all responses, then compared all codes in agreement sessions with all three authors present to refine the codes and check the coding of all responses. After three cycles of coding, we arrived at five codes for the productivity questions and quality questions. For the trade-off question, we noted several themes to describe why some developers and managers said they did or did not make trade-offs.

The five main codes that emerged from coding the productivity definition question are as follows: developer **satisfaction and well-being**, **performance or quality outcomes of developer's work**, the **activities** developers achieve in a certain time frame, the work they do **collaborating** with others, and the ability of developers to work **efficiently or with fewer interruptions**. We noted that these five emergent codes from our inductive coding aligned with the five dimensions proposed by the SPACE developer productivity framework [4] (see Table 1).

Table 1: Developer productivity codes and definitions [4].

S:	Satisfaction with work and personal well-being
P:	Performance and quality of development outcomes
A:	Activity as the count of actions or outputs
C:	Collaboration and communication among people
E:	Efficiency and flow of work with minimal interruptions

Table 2: Quality codes and definitions.

T:	Timely and predictable delivery of features
R:	Robustness of code (reliable, tested, secure, scales, etc.)
U:	Meets user, customer, stakeholder requirements/needs
C:	Readable, documented code to facilitate collaboration
E:	Evolvable design with tests to support future changes

Five key codes also emerged from inductive coding of the quality questions (see Table 2 for a description of these codes). When the productivity and quality questions were coded, we counted the responses so that we could compare how aligned developers and managers were across the two cohorts and how closely they were able to predict how the other defined productivity and quality. Some responses were coded with multiple codes. We show the frequency of the coded responses in the results below. To check alignment, we used Fisher’s exact test to check for statistically significant differences for research questions 1, 2 and 3. We used Fisher’s exact test rather than CHI square test due to our small sample size.

2.3 Limitations

Our exploratory study is a first step in understanding how productivity and quality are defined and traded by both managers and developers. Future studies should address the following limitations.

External validity: Since our survey was conducted at a single company, our results may not generalize to other companies. Furthermore, our sample was small and our results may not generalize to other developers and managers. The response rate for the manager cohort was low.

Construct validity: The survey’s open-ended questions may have been ambiguous and there may be response bias. The order of questions may have influenced how our respondents answered subsequent questions in the survey (e.g., the trade-off question). Furthermore, we asked the developers and managers to think about their own work and their team’s work, respectively, which may have influenced how they answered questions about productivity and quality. Our wording was intentionally vague as the productivity and quality terms are often not clarified when used in industry.

Internal validity: How we coded the questions may not match how other researchers would have coded our data, and we do not claim that our findings are reproducible. We tried to offset potential biases in how we coded the data by having two of us code all of the data independently. We then discussed our codes in agreement sessions with all three researchers present, with the third researcher breaking any ties in the case of disagreements (this was rare), but some responses were ambiguous and in those cases we did not assign codes to those responses. As this was conducted in a company, we do not have permission to share all responses but we include more examples of our coding in Tables 3 and 7.

3 COMPARING HOW DEVELOPERS AND MANAGERS DEFINE PRODUCTIVITY

Through our surveys, we asked developers and managers to share their definitions of productivity as well as how they thought their managers or team members, respectively, might define the term. As mentioned, we inductively coded these responses, finding that the

top-level codes that emerged aligned with the five themes captured by the SPACE framework for developer productivity (see Table 1). Examples of coded responses for productivity are shown in Table 3.

3.1 How Developers and Managers Define Productivity

To answer RQ1, we asked Individual Contributors [ICs]² and managers for their definitions of productivity (the exact wording of the questions in the survey is listed in Section 2.1). We show how their responses compare in Table 4, and discuss their responses below. Note we refer to quotes from specific developers and managers using D# and M#.

3.1.1 How developers define productivity. We note from Table 4 that developers are more likely than managers to consider productivity in terms of **activity** (writing code, fixing bugs, issuing pull requests, and other tasks completed in a given period of time). For example, D21 described their productivity this way: *“How many artifacts I produce, example: Pull Requests, check-ins, dev docs, emails, etc.”* But many consider productivity in terms of **performance**, such as quality of the work done or features delivered to the customers, as D98 shared: *“How efficiently I’m completing my work items for the given sprint while maintaining a high level of quality in my code.”* Many ICs, however, define their productivity in terms of **efficiency** and their ability to achieve a state of flow [13] while working. As D27 mentioned: *“Percentage of my time spent doing actual work versus time spent in meetings, time spent doing manual tasks I shouldn’t be doing, or wandering around looking for someone with subject-matter expertise (basically work versus meta-work).”*

Some ICs also mentioned the importance of **collaboration**, for example, D119 defined their productivity as follows: *“Amount of useful ‘work’ (feature implemented, customer issues resolved, colleagues helped) done compared to ‘distractions’ (interruptions, planning meetings, context switching).”* This response was also coded with P and E. Only a few mentioned developer **satisfaction** and well-being as aspects of productivity, for example, D134 mentioned how being engaged is important to them: *“I define productivity as how well I felt engaged in the work I am doing, as well as the lack of feeling stopped or held back. I attest to this as I notice if I am truly engrossed within a task I am just naturally more productive (through emotion) and have a strong desire to solve [the] problem given to me.”*

3.1.2 How managers define their team’s productivity. When we asked managers to define productivity when thinking about their team, their responses were quite different to how ICs define productivity (see Table 4). Managers are more likely to define productivity for their team in terms of **performance** and/or **efficiency** rather than **activity**. M20 shared that their view of their team’s productivity is to: *“Tackle the right problem and get the job done efficiently & high quality”*. Note we coded this response with P and E.

About one-third of the respondents to the manager survey focused on **collaboration** as an important aspect of productivity. M18 mentioned: *“Being able to get out of meetings with action items, and proper end result. Having the right folks in the room so that we can close on things and move on.”* Some of the manager responses for

²We use the terms ICs and developers interchangeably throughout the paper.

Table 3: Examples of how the responses to the questions about productivity were inductively coded.

Code	Example Quotes of How Developers Define Productivity	Participant ID
Satisfaction and well-being	"My productivity is working on the real tool development and learning new skills."	D63
Performance	"Delivery of projects in the long term."	D92
Activity	"Number of tasks completed / iteration."	D11
Collaboration	"Ability to brainstorm high-quality ideas and provide feedback for teammates ideas, produce high-quality code/code design quickly and provide valuable feedback to code/code design of teammates."	D114
Efficiency and flow	"Having the information and tools needed to do my work efficiently. Having a quiet environment with minimal distractions to focus and remain in the flow state. Having as few meetings as possible."	D84

Table 4: How ICs and managers define productivity. The differences with how they defined productivity in terms of performance (P) and activity (A) are statistically significant (*).

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

their definition of their team productivity crosscut several dimensions, as M6 noted: "People are able to predictably deliver features and fixes that keep our customers happy while learning and growing, constantly improving our culture, and staying happy themselves" (this response was coded with S, P, C, and E).

➔ **Takeaway:** Developers and managers defined productivity in terms of all five dimensions of SPACE, but only a few mentioned developer satisfaction and well-being. Notably, developers and managers, as cohorts, were not that closely aligned in their views of productivity as *developers are more likely to define productivity in terms of activity, while managers are more likely to define productivity in terms of performance*.

3.2 How Developers Think Managers Define Productivity

To answer RQ2, we used the same five codes as RQ1 to code responses to the question we posed to developers about how they think their managers define productivity. Table 5 shows that the IC perceptions of how managers define productivity does not align with how managers define productivity, especially with respect to the distributions of answers coded with Performance, Activity and Efficiency. Managers are more likely to think about productivity in terms of **performance** (as discussed above), but ICs anticipate managers define productivity more in terms of **activity**. For example, D82 is one of many developers that considers that their manager defines productivity in terms of their **activity** only: "codes have been checked in, bugs have been fixed." Some do consider that their managers define productivity in terms of **performance**, and

Table 5: How ICs think managers define productivity compared with how managers actually define productivity. The differences in terms of performance (P), activity (A), and efficiency (E) are statistically significant (*).

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

a few anticipate that managers consider **collaboration** as an aspect of productivity. For example, D81 responded: "How much impact is created. How much [value] added to others work and how much of others work is leveraged" (coded with both P and C). Furthermore, some developers underestimated the number of managers that consider **efficiency** as an important aspect of productivity. Only a few mentioned that their managers consider **satisfaction** and well-being, with one of the few notable exceptions by D44: "making sure all are happy." Of note is that five ICs were not sure how their managers define productivity, suggesting that a conversation about productivity had not taken place.

➔ **Takeaway:** Developer anticipation of how their managers define productivity is not aligned with how managers actually define productivity, as *managers are more likely to define productivity in terms of performance rather than activity. Developers also underestimated how often managers mentioned efficiency as an important aspect of productivity*.

3.3 How Managers Think Developers Define Productivity

From Table 6 we can observe that managers, as a cohort, are quite accurate at estimating how ICs define productivity (RQ2). This contrasts with how many ICs do not have such an accurate perception of how managers define productivity. Many managers anticipate that many ICs will define productivity in terms of activity. For example, M37 suggested that ICs on their team define productivity in terms of **activity** in contrast to how they (as managers) or their

Table 6: How ICs define productivity compared with how managers think ICs define productivity. No differences are statistically significant.

	ICs define own productivity	Managers think ICs define productivity
S	8%	15%
P	35%	24%
A	50%	52%
C	24%	12%
E	38%	42%

peers define productivity: “Achieving current sprint deliverables. We use two weeks sprint cycles with monthly integration cycles. Direct reports are more focused along these more immediate time lines that myself or my peers.” Many managers, however, recognize that their ICs think about efficiency and flow when defining productivity, as M11 mentioned their ICs consider: “How many hours a day can they spend in uninterrupted coding activity.” Of note, and as discussed above, is that few ICs thought their managers considered **efficiency** and flow in their definitions of productivity (see Table 5).

➔ **Takeaway:** Many managers have accurate insights into how ICs define productivity (notably in terms of activity, performance and efficiency), even though those views are not well-aligned with their own definitions of productivity.

4 COMPARING HOW DEVELOPERS AND MANAGERS DEFINE QUALITY

We asked developers and managers how they define quality (to answer RQ3). We did not ask them how they anticipated the other cohort may define quality as we found in the exploratory interviews that asking about their own views of quality was already challenging to answer.

4.1 How Developers and Managers Define Quality

Five key codes emerged from our inductive coding of the responses to the questions “when thinking about your work, how do you define quality” posed in the developer survey, and “when thinking about your team’s work, how do you define quality” posed in the manager survey. The five core codes are shown in Table 2 above, and samples of coded responses are shown in Table 7.

The most common theme (see Table 8 for the distribution of answers coded with the five emergent quality codes) for both developers and managers is defining quality in terms of the **robustness** of the software, which can refer to reliability, performance, how well tested the code is, and adherence to security concerns.

The next most frequent response for both developers and managers referred to how **evolvable** the software is (i.e., how its design and test suites support future changes). For example, D42 felt that quality meant “maintainable, and defect free”, while M23 mentioned that “quality is work that is designed for the future, unit and manually tested” (both responses were coded with Robust (R) and Evolvable

(E)). A similar relative number of developers and managers defined quality in terms of the delivery of features that **meet user needs** (coded with U). As M19 put it: “Quality is doing what the customer wants.”

Some developers and managers noted that quality is about writing software that others, not the just authors, can build on explicitly and thus support **collaboration**. As D112 put it: “well documented code that is easy for newcomers to the team to get up to speed with.” And lastly, a few developers mentioned the **timeliness** of the features that are delivered as a core aspect of quality, as M26 said: “less bugs after release, followed by time to market” (coded with R and T).

Some of the open-ended responses we received to this question addressed three or more of the five dimensions of quality that emerged from our coding. For example, M11’s response to this question was coded with R, U and E: “Quality isn’t the absence or pretense of some metric, rather it is an emergent property of a complex system. The property incorporates some sense of ‘bug free’ and ‘suitability for a specific purpose’ and also includes a bunch of other things like maintainability, diagnosability, security, robustness, availability, etc.” And D84’s definition captures all five codes (T, R, U, C, E): “Available to customers at the originally planned date. Design that is clear to peers who will review the final work (code). Bug free and with full test coverage. Well documented for other peers to understand. Meets all planned and emergent requirements.”

4.2 How Aligned Are Developers and Managers in How They Define Quality

We compared how the two cohorts were aligned in their views of quality. Interestingly, and surprising to us, developers and managers as cohorts considered quality in similar ways in terms of distribution of the different responses, but within each cohort there were varied views of what quality refers to (as presented above, some definitions of quality were quite narrow, while others covered multiple dimensions). We expected and did see more quality definitions concerning Robustness (R), Evolvability (E), and meets User needs (U). Fewer managers and developers mentioned Timeliness (T) and Collaboration (C), but those that did clearly articulated how these aspects were very relevant to quality according to their view. Although more managers mention robustness than developers (proportionally), managers were also more likely to provide more elaborate definitions of quality.

➔ **Takeaway:** Developers and managers defined quality, as cohorts, with the same inductively derived five themes (timeliness, robustness, delights users, meets collaboration needs, supports evolution). Over 70% of both cohorts mentioned robustness in their definitions of quality, while around a third mentioned “delights users” and is “evolvable”.

5 DO DEVELOPERS AND MANAGERS TRADE QUALITY FOR PRODUCTIVITY?

We asked developers and managers, through an open-ended question, if they or their teams traded quality for productivity (per RQ4). We classified their answers into “Yes, they report trading quality in favour of productivity”, or “No” — they either report not trading

Table 7: Examples of how responses to the question about quality were inductively coded.

Code	Example Quotes of How Developers Define Quality	Participant ID
Timeliness of features delivered	<i>"Shipping code that works, and on time."</i>	D13
Robustness of features	<i>"Robustness of code."</i>	D9
User delight	<i>"Degree to which the product meets customer needs."</i>	D46
Collaboration needs met	<i>"Quality is code or solutions that solve a problem and don't need undue maintenance or lengthy handoff. If I get hit by a bus and the company can still easily use the code I've written, I've made a quality solution."</i>	D129
Evolution needs met	<i>"Easy to test, easy to change."</i>	D15

Table 8: How individual contributors (ICs) and managers define quality. The differences with how they define quality in terms of robustness (R) are statistically significant (*).

	ICs define quality	Managers define quality
T	■ 7%	■ 15%
R	■ 71%	■ 88% (*)
U	■ 38%	■ 39%
C	■ 17%	■ 18%
E	■ 44%	■ 33%

quality for productivity, or they are more likely to trade productivity (in terms of feature delivery) for higher quality. For the two cohorts, we saw a similar distribution of responses to this question (see Table 9). We further analyzed their open-ended responses to identify themes to explain their answers. We share insights from this analysis below.

5.1 Why and How Developers and Managers Trade Quality for Productivity

Over half of the developers and managers indicated they trade quality for productivity.

Many reported that **meeting deadlines led to lower quality**. D36 mentioned there is **insufficient time prioritized for quality** goals: *"All the time. High-quality work takes much longer during all phases of development (design, implementation, testing, documentation) and there is seldom sufficient time available to meet productivity goals if quality were a priority."* D13 also mentioned they had insufficient focus time to meet their deadlines: *"Yes. I often have to take shortcuts to meet tight deadlines. The constant randomization my team faces is the main driver of this."* Note that D13 recognizes that interruptions may lead to quality trade-offs. In contrast, D7 clarified that **meeting deadlines is itself a form of quality**: *"SW engineers including me have a tendency to over complicate solutions and these short-solutions keep us grounded on the reality that code that runs _right now_ has a quality element of its own!"*

Many participants also explained that working with **legacy code is a barrier to improving quality**, as D11 shared: *"Yes all the time. The code base is so old and there're lots of inefficient / misleading implementation. It's almost impossible to fix them at once and no one dares to do so. So we're always sacrificing quality for productivity."* Furthermore, some mentioned a need to incur **new technical debt**

Table 9: Responses to the question "Do you trade quality for productivity?"

	Yes	No	Unknown
Developers	■ 52%	■ 34%	■ 14%
Managers	■ 59%	■ 38%	■ 3%

to meet deadlines, as D86 shared: *"Yes, occasionally it becomes necessary to take on technical debt in order to ship a feature. This is ok as long as it leads to a better long-term return [...]"* However, D21 noted that this trade-off can backfire: *"That's technical debt. Our product is a monument to what several decades of trading productivity for quality look like. Most of the time, it's impossible to predict how quality will suffer as a result of productivity and so retrospectives take several years to fully develop. But yes, I'll often favor finishing work over doing it 'right' the first time because it needs to go out the door."* Several respondents described that adding new technical debt was the result of **reduced or deferred testing** with potential long-term impacts on the evolvability of the code, as D44 shared: *"Often, for example when building test infrastructure will take more effort than feature itself, we make a trade-off towards simpler test coverage, that hits us in the head later."*

Incurring technical debt and deferring testing were seen as necessary for meeting short-term needs such as **unblocking others**, as M21 shared: *"Bootstrapping a new product on which hundreds of developers are dependent, we will bring up an MVP that largely works, but has non-mainline blocking bugs so they can start moving sooner rather than later"*; or to quickly **meet user needs** and speed up feedback, as D117 said: *"sometimes we are creating feature that are 'scenario enabler', then we can iterate over to make it usable with the best customer experience."* M29 shared there are processes in place to ensure **short-term quality trade-offs are not permanent**: *"We work in an environment where we are encouraged to 'run with safety scissors' - we're encouraged to take risks but do so in a safe way, where rollbacks, failovers, or recovery is possible and quick. So I wouldn't say we make trade-offs on quality, but if we do it is usually short-term with a plan to improve and seek feedback along the way."*

Culture around quality also played a role in priorities of feature delivery over quality, as D54 shared: *"Yes. Again this is a theme on team culture. Quality was secondary. The top priority was getting the feature to work."* In contrast, D102 said that although there are pressures to make compromises, trade-offs are mostly imagined: *"There's always perceived pressure to make compromises on quality"*

in favor of speed, but I think it's mostly imagined." Several participants also noted a tension in terms of the **return on investment** of spending too much time on quality over faster feature delivery to their customers, as D103 shared: "Always. It's the primary rub of my craft. The business needs to reach the customer quickly. There are only so many hours in the day to do everything required for perfect quality. Typically there's a line where 'good enough' is good enough and preferable to quality perfection. At some point, further investments in quality have diminishing returns for customer experience."

Although when we asked about how productivity is traded for quality, we found that the trade-offs shared were more about different dimensions of quality rather than productivity (see Fig. 1).

➔ **Takeaway:** Over 50% of developers and managers reported trading quality for productivity, most to meet short-term needs. Technical debt and insufficient testing were reported as trade-offs to achieve higher productivity with timely feature delivery that meets user needs and unblocks others. Legacy code, lack of time, poor return on investment and a culture that prioritizes faster feature delivery over quality were drivers for these trade-offs.

5.2 Why Some Developers and Managers Do Not Trade Quality for Productivity

Notably, approximately one-third of both developers and managers said that they **do not trade quality for productivity and rather reduce feature delivery to achieve quality**. Bluntly, D52 shared: "The question is invalid. Delivering lower quality work in order to deliver a larger number of features is not productivity" and M17 shares: "No, we do not cut edges. We rather be later than wrong."

Some specifically declared that they consider **quality as a part of productivity**. D126 said: "No. For my job, improving quality IS productivity"; and M6 described: "Sort of. If you just think of productivity as 'how many features did I ship' then yes, it's normal to trade off features vs quality, usually by spending more time on quality and shipping fewer features and unfortunately sometimes the other way around. But I would normally think of the activities that create quality (bug fixing, testing, better design) as just another part of productivity—if someone is getting those things done, I see them as being very productive. So in that sense, no, there's no trade-off, because quality is part of productivity."

M31 mentioned aiming for **quality over feature delivery with a new product**: "Absolutely. We're shipping v1 of our product soon, and we're terminating feature work three months in advance to make sure that we can truly ship a quality product." D15 mentioned that **even with legacy code, quality is important**: "legacy code that is known to be obsolete but must be fixed" (note legacy code was in contrast sometimes noted by others as a reason to trade feature delivery over quality above).

Some respondents mentioned that **trading quality with feature delivery can impact developer satisfaction**. We saw two camps in these responses. For some, they indicated that if they have to trade quality for productivity, they would be less satisfied with their work. As D62 noted: "No, [trading quality for productivity is] the reason for bad work life balance." D33 felt so strongly about this that they shared they are willing to spend more time to ensure they also remain productive: "I prioritize quality even if I may have to take

extra time to ensure that my code is of good quality." Others, however, find prioritizing quality over productivity is less rewarding, as D53 shared: "I have been in this team for 4 months now and I find it very hard to find myself productive because my new team wants things a certain way and they are caring a lot about the code I put in. Their definition of quality is definitely affecting my productivity and hence my motivation." And D79 reported they observed frustration with others: "Yes, in past projects I have had to implement some form of ask mode to ensure that we had enough time to stabilize the code we had already written. This often frustrated engineers who did not want to have their checkins gated on lead/manager approval." That is, not making progress on features in favour of higher quality lowered these developers' satisfaction.

In our analysis, we observed some **tensions with manager priorities** for balancing quality and productivity. D95 shared: "I'm sure my managers at times would've liked me to sacrifice quality to just get a feature done, but I honestly don't really know how to do that. I usually use tests to ensure that the code that I write does what I want it to do, and generally that 'once it's done, it's done' feeling comes from writing good test cases." D129 shared how they also push back: "There's often a push for getting things out quicker than is possible which often means taking shortcuts and not documenting fully. I try to push against this and make solutions that follow best code and documentation practices." In contrast, D113 shared: "My manager consistently demonstrates the willingness to trade productivity for quality; typically by avoiding being date-driven."

Trade-offs between quality and productivity may need to be considered across teams. M5 shared: "If I feel that our quality isn't where it needs to be, we will swarm on whichever part of the system needs it. That means productivity for those other areas will slow down, temporarily. I've worked on projects where the priority has been to help another team's quality and we've effectively stopped our work to help them because the release needed it. Another example might be ramping other resources for better coverage in our codebase: the SME's personal productivity will slow but the benefit is that we'll have multiple people now aware of that area."

➔ **Takeaway:** Approximately one-third of developers and managers indicated they do not trade quality for productivity, as they consider quality an essential aspect of productivity (for new and legacy systems). However, trading feature delivery for more quality can positively or negatively impact developer satisfaction, depending on team and manager priorities.

6 DISCUSSION

We discuss how the main dimensions in the SPACE framework mapped to the definitions developers and managers provided to us in the survey, how the responses to the quality question led to a new framework of thinking about software quality that we call TRUCE, and how SPACE and TRUCE are related but different lenses for thinking about software engineering processes.

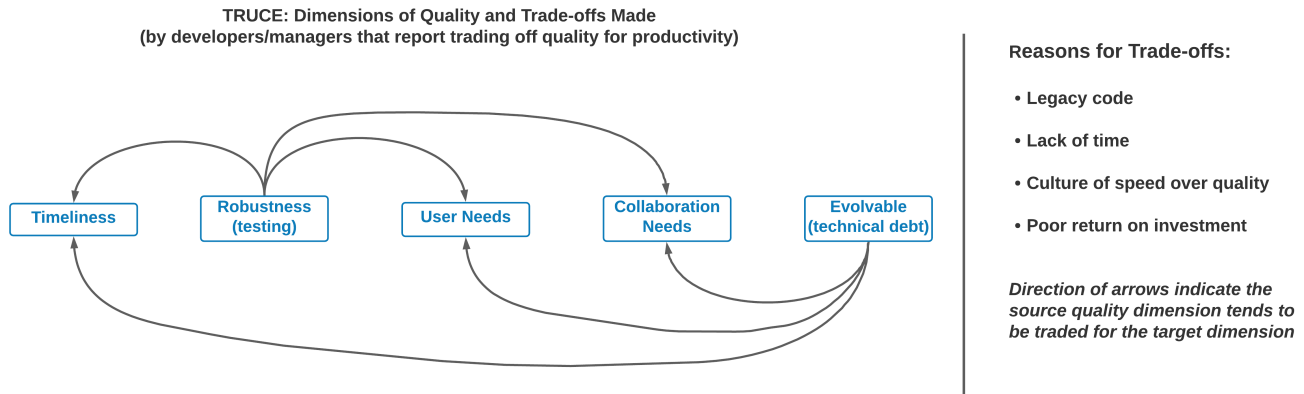


Figure 1: Trade-offs that developers and managers reported making when asked about trading off quality for productivity. We note that many reported quality trade-offs rather than quality for productivity.

6.1 SPACE: Comparing How Developers and Managers Define Developer Productivity

We inductively coded the responses about productivity to see how developers and managers in a company define productivity and if they are aligned in their views. We noticed in our analysis that the five main codes that emerged mapped 1-1 to the five dimensions proposed by the SPACE framework [4], a framework we co-authored. The SPACE framework emerged from the combined insights across several studies rather than from a single empirical study. However, the SPACE framework was developed some months after we conducted this study and after we initially analyzed our data. Later we discovered that the five categories that emerged from our study data aligned with the five dimensions in SPACE. Our study adds support for the dimensions of productivity captured in the SPACE framework and that the SPACE framework adequately captures how developers and managers, collectively, define productivity.

However, as we can see from Table 5, the majority of the developer definitions of productivity concerned **activities**, while the majority of manager definitions mentioned **performance** or quality outcomes. And across both cohorts, fewer developers and managers considered other dimensions such as developer satisfaction, collaboration and efficiency. A few, however, mentioned 4-5 of the five dimensions. This misalignment within and between cohorts shows that more discussion about what productivity means is needed, especially if productivity improvement goals and metrics are being proposed. There is also a need for awareness of how others define productivity (see Table 6) to avoid potential false assumptions and later conflicts about how to improve and measure productivity.

6.2 TRUCE: A Framework for Software Quality

We were interested to learn how developers and managers define quality in their own words. From our inductive coding of the responses we received to the quality question, five main codes emerged (after several iterative rounds of coding and comparison across coders). Although it was not our intent when we designed this study to derive a new definition for quality, we offer TRUCE as a framework for thinking about quality (similar to the SPACE framework for productivity). However, since it emerged from a single

study, it would need to be validated in future studies. We note that the dimensions in this framework come from the “mouths” of the developers and managers we surveyed. TRUCE captures five main quality dimensions: The **Timely** delivery of **Robust** features that meets **User needs**, while enhancing **Collaboration** with others and supporting the product’s future **Evolution**.

As shown in Table 8, fewer responses to the quality definition question mentioned C (collaboration) and T (timeliness), and yet these aspects are critical aspects of quality that should be considered and surfaced more often. Enabling collaboration or use by others is often invisible and hard to measure; this may explain why it was rarely mentioned, and yet it is very important in many development contexts. Similarly, timeliness was infrequently mentioned; we were surprised to see this code surface in our analysis. Yet the responses convinced us that timeliness is indeed a very important aspect of quality. Similar to the breadth of productivity definitions, there is misalignment within cohorts of how developers and managers define quality (Table 8). The majority mention **robustness**, but the other dimensions also come into play when others think about what quality of the product and their process means to them.

6.3 SPACE and TRUCE: Related but Different Lenses Into Software Development

Productivity is a lens that focuses on the developer, while quality is a lens that focuses on the quality of the software produced and process used. Although different lenses, they are related. In fact, when we asked about trade-offs between quality and productivity, we found that the trade-offs mentioned were more about trading quality aspects (see Fig. 1) than trading quality for productivity.

This is not surprising as there is overlap across the dimensions in SPACE and TRUCE. Collaboration is in both frameworks: in SPACE, it captures the idea that one can’t think about productivity without considering how what they do is part of a bigger collaborative effort (e.g., code reviews), while in TRUCE, it captures how the quality of the software facilitates other’s work (e.g., documented code that can be extended by others). Furthermore, the Performance dimension in SPACE captures the quality of the work that is done. Although there is some overlap, the two frameworks can be used

by developers and managers to reflect on and discuss trade-offs between productivity and quality, and potentially to define metrics (see [4] for a discussion of productivity metrics). As M37 put it: *“We are often/always constrained on the challenge of quality vs. time vs. cost/resources. [...] Productivity is a constant and elusive trade-off between immediate and future throughput.”*

7 RELATED WORK

Before concluding our paper, we map our findings to related research on developer productivity and software quality.

7.1 Developer Productivity

There has been extensive research in recent years on developer productivity and how system engineering activity metrics can provide important signals about developer activity and productivity. Wagner and Ruhe’s review of the literature summarize studies that use performance measures such as *lines of code* or *function points* as proxies to productivity [25]. However, many researchers and practitioners emphasize that developer productivity cannot and should not be measured by engineering metrics alone—development work is not mechanized work that can be assessed using a single metric, and doing so may be detrimental to overall and long-term development objectives [10]. For example, developers spend time mentoring newcomers, reviewing each other’s work informally, and learning new skills. Most related to our work, Meyer et al. [11] investigated how developers perceive and think about their own productivity. Meyer et al. also note the importance for developers to work without interruptions [12].

Murphy-Hill et al. [15] asked 622 developers in a survey across three companies (Google, ABB and National Instruments) about productivity factors and self-rated productivity. They found that non-technical factors, such as job enthusiasm, peer support for new ideas, and useful feedback about job performance, correlated most strongly with self-rated productivity. The three factors with lowest variance (across the three companies) were social and environmental rather than technical factors. Developer productivity and satisfaction has been discussed in conjunction with other human aspects of software engineering, such as developer happiness [7] and developer motivation [1, 5, 20], while the impact of the pandemic on team productivity was discussed in [14]. Some recent research led to the SPACE framework of productivity [4]—we saw that its five dimensions of productivity mapped to the different dimensions of productivity mentioned by our survey respondents.

7.2 Software Quality

Research about software quality has been extensive over the past few decades (and especially in the 1990s). But what quality means in any domain is not absolute—it remains somewhat elusive, despite the ISO standards for defining quality³. Some of the earliest research on software quality was done by Basili et al. [23] and led to the powerful Goal-Question-Metric which still stands the test of time in terms of defining quality metrics. Kitchenham and Pleegeer [9], in their foreword to a special issue on software quality, eloquently describe the different views of software quality that include a transcendental view (where quality is an ethereal goal

we never quite reach), the user view (how the product meets users’ needs), the manufacturing view (product quality during production and after delivery), the product view (the product’s inherent internal quality), and the value-based view (value to customers or market groups). These different views also form the basis for an excellent discussion among developers and managers.

Gilliet et al. also share that quality means different things to different people in different contexts and is impacted by numerous multi-dimensional factors [6]. In particular, they note that what quality means in software development is hard to define, and measuring it using only a few quantitative metrics alone can be misleading and may not map to what developers or managers consider to be quality. The five main dimensions we identified in TRUCE correspond closely to the definition of quality provided by Gillies et al. [6], as four of the five dimensions in TRUCE were also in their definition (using slightly different terminology): Only collaboration was not explicitly mentioned in Gillies’ definition. Although collaboration support may be considered as an aspect of evolution, our respondents discussed collaboration in a more immediate sense in that work a developer does has to consider the collaborative needs that are pressing (e.g., making quality trade-offs to unblock others).

Wilson and Hall conducted a study in 1994/95 to compare developer, manager, and software quality practitioner views on software quality from different companies [26]. They found “that management often appear unaware of their developers’ opinions of software quality” and that the “[t]he significant impediments to software quality initiatives are lack of resources and lack of time”. They also found that managers underestimated the amount of effort developers take to ensure quality.

7.3 Where Productivity and Quality Meet

There has been extensive research that considers both quality and productivity (not surprising given how related they are). Notably, the performance dimension of the SPACE framework is about the quality of the outcomes achieved, and Sadowski et al. also suggest that quality is an important aspect of developer productivity [18]. Some researchers have considered the impact of interventions on both developer productivity and software quality. For example, Bissi et al. conducted a review of test-driven development approaches on developer productivity and quality [2], Coupe et al. investigated the impact of computer-aided software engineering tools on productivity and quality [3], and Savor et al. [19] and Vasilescu et al. [24] considered the impact of continuous deployment on productivity, measured by shipped lines of code and number of commits (respectively), and quality, measured by number of failures. Other researchers have explored the impact of both productivity and quality perceptions on adoption of improvement processes [8]. We did not, however, identify other research that has investigated how developers and managers define both quality and productivity, how aligned the two cohorts are in their views, and how they report making trade-offs between them.

In our research, we found that both developers and managers have different views of both productivity (within and across cohorts) and quality (within cohorts, recall we did not ask survey questions about their perceptions of how the other cohort defined quality), and that over half of over half of our participants report trading quality

³See <https://asq.org/quality-resources/learn-about-standards>

for productivity. As Wilson and Hall noted back in 1998, “Many software quality initiatives fail because they do not take account of the range of views that people have of quality” [26]. They also stressed that “new approaches to software quality improvement will not work unless software developers believe in them, no matter how enthusiastic managers may be”. We go one step further, and suggest that developers and managers need to discuss their views of developer productivity, software quality, and when/if they should make trade-offs, and why or why not.


8 CONCLUSIONS

We found that many developers were not aware how their managers perceived productivity, and many were nervous about having their work measured and evaluated using metrics from telemetry data, especially given the invisible nature of some development activities (e.g., helping others and learning or planning for the future).

The new framing of quality we offer, and the insights on misalignment of what productivity means to developers and managers, suggests a need for developer teams and their managers to regularly reflect and discuss what productivity and quality mean to them. Many companies lately have also been striving to understand and measure productivity and quality; thus, our research calls for attention on how differences in developer and manager views of productivity and quality should be discussed before any metrics or interventions to improve quality and productivity are put into practice. Quality criteria in any domain are seldom independent, and making compromises often brings up conflicts [6], but highlighting which aspects of quality should be traded is important to do. For example, one aspect of quality is timely delivery (and some practitioners advocate this is the most important quality criteria) of software that meets user needs [9], but other practitioners recognize that software that is useful will need to evolve, and writing code with evolution in mind is essential [6]. Through our study, we offered some feedback (and reinforcement) that SPACE can be used for thinking about developer productivity, and how TRUCE may also be used to have richer conversations about trade-offs that are made in terms of software quality.

Despite the limitations with using quantitative metrics for measuring both productivity and quality, many companies strive to use metrics to have fast ways to measure changes and drive insights about possible improvements. Recognizing that metrics will be used despite their known limitations, we suggest using both SPACE and TRUCE for defining metrics. Using both frameworks can help distinguish developer productivity metrics from software quality metrics, and help practitioners identify a more diverse set of metrics that will capture a broader picture of productivity and quality in their companies or projects.

ACKNOWLEDGMENTS

We thank the survey participants, and Jacek Czerwinka, Cassandra Petrachenko and Alessandra Paz Milani for feedback on our paper. Inspired by Courtney Miller and Chanel (Miller @ ICSE 2022), a very special thanks to Kodabear  for keeping the authors company on their many Teams and Zoom calls! #dogsoficse #dogsofchase

REFERENCES

- [1] Sarah Beecham, Nathan Baddoo, Tracy Hall, Hugh Robinson, and Helen Sharp. 2008. Motivation in Software Engineering: A systematic literature review. *Information and software technology* 50, 9-10 (2008), 860–878.
- [2] Wilson Bissi, Adolfo Gustavo Serra Seca Neto, and Maria Claudia Figueiredo Pereira Emer. 2016. The effects of test driven development on internal quality, external quality and productivity: A systematic review. *Information and Software Technology* 74 (2016), 45–54.
- [3] RT Coupe and NM Onodu. 1996. An empirical evaluation of the impact of CASE on developer productivity and software quality. *Journal of Information Technology* 11, 2 (1996), 173–181.
- [4] Nicole Forsgren, Margaret-Anne Storey, Chandra Maddila, Thomas Zimmermann, Brian Houck, and Jenna Butler. 2021. The SPACE of Developer Productivity: There's More to It than You Think. *Queue* 19, 1 (Feb. 2021), 20–48. <https://doi.org/10.1145/3454122.3454124>
- [5] César França, Fabio QB Da Silva, and Helen Sharp. 2018. Motivation and satisfaction of software engineers. *IEEE Transactions on Software Engineering* (2018).
- [6] Alan Gillies. 2011. *Software Quality: Theory and Management*. Lulu.com. Google-Books-ID: XTvpAQAAQBAJ.
- [7] Daniel Graziotin, Xiaofeng Wang, and Pekka Abrahamsson. 2013. Are happy developers more productive?. In *International Conference on Product Focused Software Process Improvement*. Springer, 50–64.
- [8] Gina C Green, Alan R Hevner, and Rosann Webb Collins. 2005. The impacts of quality and productivity perceptions on the use of software process improvement innovations. *Information and Software Technology* 47, 8 (2005), 543–553.
- [9] B. Kitchenham and S.L. Pfleeger. 1996. Software quality: the elusive target [special issues section]. *IEEE Software* 13, 1 (Jan. 1996), 12–21. <https://doi.org/10.1109/52.476281> Conference Name: IEEE Software.
- [10] Amy J. Ko. 2019. *Why We Should Not Measure Productivity*. Apress, Berkeley, CA, 21–26. https://doi.org/10.1007/978-1-4842-4221-6_3
- [11] André N. Meyer, Thomas Fritz, Gail C. Murphy, and Thomas Zimmermann. 2014. Software Developers' Perceptions of Productivity. In *Proceedings of the 22th International Symposium on Foundations of Software Engineering*.
- [12] André N. Meyer, Gail C. Murphy, Thomas Fritz, and Thomas Zimmermann. 2019. Developers' Diverging Perceptions of Productivity. In *Rethinking Productivity in Software Engineering*, Caitlin Sadowski and Thomas Zimmermann (Eds.). Apress, Berkeley, CA, 137–146. https://doi.org/10.1007/978-1-4842-4221-6_12
- [13] Tommi Mikkonen. 2016. Flow, intrinsic motivation, and developer experience in software engineering. *Agile Processes in Software Engineering and Extreme Programming* (2016), 104.
- [14] Courtney Miller, Paige Rodeghero, Margaret-Anne Storey, Denae Ford, and Thomas Zimmermann. 2021. "How Was Your Weekend?" Software Development Teams Working From Home During COVID-19. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. 624–636.
- [15] E. Murphy-Hill, C. Jaspan, C. Sadowski, D. Shepherd, M. Phillips, C. Winter, A. Knight, E. Smith, and M. Jorde. 2019. What Predicts Software Developers' Productivity? *IEEE Transactions on Software Engineering* (2019). <https://doi.org/10.1109/TSE.2019.2900308>
- [16] Padmalata Nistala, Kesav Vithal Nori, and Raghu Reddy. 2019. Software Quality Models: A Systematic Mapping Study. In *2019 IEEE/ACM International Conference on Software and System Processes (ICSSP)*. 125–134.
- [17] Satya Pradhan, Venky Nanniyur, Paddu Melanahalli, Munir Palla, and Sunita Chulani. 2019. Quality Metrics for Hybrid Software Development Organizations – A Case Study. In *2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. 505–506. <https://doi.org/10.1109/QRS-C.2019.00097>
- [18] Caitlin Sadowski, Margaret-Anne Storey, and Robert Feldt. 2019. A Software Development Productivity Framework. In *Rethinking Productivity in Software Engineering*, Caitlin Sadowski and Thomas Zimmermann (Eds.). Apress, Berkeley, CA, 39–47. https://doi.org/10.1007/978-1-4842-4221-6_5
- [19] Tony Savor, Mitchell Douglas, Michael Gentili, Laurie Williams, Kent Beck, and Michael Stumm. 2016. Continuous Deployment at Facebook and OANDA. In *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*. 21–30.
- [20] Helen Sharp, Nathan Baddoo, Sarah Beecham, Tracy Hall, and Hugh Robinson. 2009. Models of motivation in software engineering. *Information and software technology* 51, 1 (2009), 219–233.
- [21] Margaret-Anne Storey, Brian Houck, and Thomas Zimmermann. 2021. *Appendix to How Developers and Managers Define and Trade Off Productivity and Quality*. Technical Report MSR-TR-2021-26. Microsoft Research. <https://www.microsoft.com/en-us/research/publication/appendix-to-productivity-quality-alignment>
- [22] Margaret-Anne Storey, Thomas Zimmermann, Christian Bird, Jacek Czerwinka, Brendan Murphy, and Eirini Kalliamvakou. 2021. Towards a Theory of Software Developer Job Satisfaction and Perceived Productivity. *IEEE Transactions on Software Engineering* 47, 10 (2021), 2125–2142.
- [23] Rini Van Solingen, Vic Basili, Gianluigi Caldiera, and H Dieter Rombach. 2002. Goal question metric (gqm) approach. *Encyclopedia of software engineering* (2002).

- [24] Bogdan Vasilescu, Yue Yu, Huaimin Wang, Premkumar Devanbu, and Vladimir Filkov. 2015. Quality and productivity outcomes relating to continuous integration in GitHub. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. 805–816.
- [25] Stefan Wagner and Melanie Ruhe. 2018. A systematic review of productivity factors in software development. *arXiv preprint arXiv:1801.06475* (2018).
- [26] David N Wilson and Tracy Hall. 1998. Perceptions of software quality: a pilot study. *Software quality journal* 7, 1 (1998), 67–75.