Comparing the Validity of Productivity Metrics in Software Engineering

Perception vs. Reality in a Student Environment

Bachelor of Science Thesis in Software Engineering and Management

SAM HARDINGHAM

KAI ROWLEY

DANIEL VAN DEN HEUVEL

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025

CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Bridging Perception and Reality in Software Productivity
A Comparative Study of Traditional Metrics, the SPACE Framework, and Self-Assessment in Student Software Engineering Teams

Supervisor: CRISTINA MARTINEZ MONTES
Examiner: RICHARD BERNTSSON SVENSSON

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
University of Gothenburg
Chalmers University of Technology
Gothenburg, Sweden 2025

Comparing the Validity of Productivity Metrics in Software Engineering: Perception vs. Reality in a Student Environment

Sam Hardingham
Department of Computer Science
Gothenburg University
Gothenburg, Sweden
gushardisa@student.gu.se

Kai Rowley
Department of Computer Science
Gothenburg University
Gothenburg, Sweden
gusrowkai@student.gu.se

Daniel Van Den Heuvel
Department of Computer Science
Gothenburg University
Gothenburg, Sweden
gusdaniva@student.gu.se

***Abstract*—Given the rise of modern frameworks for measuring productivity like SPACE, it remains unclear how well they capture what software engineers value as productive work. In this study, we explore how objective measures, both modern and traditional, relate to software engineering students' self-perception of productivity by surveying 24 participants throughout the course of a simulated industry project. With this comparison, we also explored the opinions of software engineering students on how they define productivity in an effort to explain the weak observable association between self-perceived productivity and objective metrics. Our results suggest that productivity models should be further refined to more accurately capture social dimensions of work. In addition to this, we propose that future work should seek to explore other perception-based measures in order to understand which of them is most strongly associated with objective metrics.**

I. Introduction

Measuring software engineer productivity has always been difficult. Traditional metrics like code lines and commit counts, more often than not, fail to capture the full scope of software contribution—a point underscored by Jones [1] and Wagner [2]. These simple measurements miss the deeper aspects of engineering work [3], [4], but are popular because they are easy to collect, however; they focus too much on quantity over quality [1], [4]. Companies that depend only on these numbers often get a skewed picture of developer productivity, leading to inaccurate readings, estimates, and potentially missing the complex technical work that takes place behind the scenes [5], [6]. Many important, intricate, or time-consuming tasks do not show up well in simple metrics—tasks like feature design, database work, code cleanup, performance optimization, and infrastructure management. These crucial activities typically get overlooked in productivity assessments [4], [7]. Researchers are however not suggesting we throw out traditional measurements entirely, but rather use them as part of bigger frameworks for a more complete picture of developers' productivity [2], [3], [5]. The SPACE framework, for example, looks at: *Satisfaction*, *Performance*, *Activity*, *Communication* and *Efficiency* to evaluate productivity more holistically [6]. By mixing numbers with self-assessment and team factors, we can better understand real-world productivity [8], [9]. Despite these pushes many teams still default to the surface-level metrics exclusively [1], [4]. This tends to lead to disparate judgments and inconsistent evaluations in productivity assessments, showing that we still have not solved the productivity puzzle in software engineering [2], [5], [10].

Across industries, productivity is commonly defined as the ratio between the value of output and the value of the input required to produce it [11]. While this appears to be a simple definition, there is significant debate as to *how* useful output is quantified, and what impact different kinds of work have on producing it – essentially how inputs should be weighted against each other when compared to the work produced.

However, what constitutes "value" here is generally unclear. In software development, value may not just be an immediate product like code or features completed but also indirect values like facilitation of team communication, technical mentoring, or improvement of project organization. This lack of clarity is even more acute in an academic context where learning and collaboration are also essential objectives. In our research, we seek to understand how students understand what "value" means in the context of productivity.

Many of the limitations in these studies on industry professionals emerge from the absence of a structured study environment [12], and lack of participation from entire teams [13]. In our research, we explore the validity of these metrics for productivity in a university environment and how accurately they correspond to each other. With the current generation of software engineers entering the workforce as some of the first to experience the transition to remote-learning and subsequently remote-work, it is essential to understand how common productivity metrics apply to their learning environments and how they perceive their productivity in this setting. After consideration of the current available literature, we propose two research questions:

RQ1: How well do traditional (LOC, number of commits,

number of code reviews) and modern (SPACE) productivity metrics align with how productive software engineering students *perceive* themselves to be?

RQ2: What kinds of tasks do software engineering students value as "productive" work?

To address these questions, we conducted a sample study by analyzing the commits and lines of code in the GitHub repository for each team member of several student software development teams and ranking the participants against each other. In addition to this, we also used the SPACE framework [6] to rank participants by productivity. We conducted surveys among the teams, with questions focused on their perceived productivity, and what activities they perceive as productive. This allows us not only to compare common productivity metrics with students' self-perceived productivity, but also gives an indication as to what differences in opinion students have of "what is productive?" with the factors these frameworks emphasize.

Ultimately, this research contributes to the ongoing discourse on software productivity measurement by contextualizing traditional and modern metrics in educational settings, and by foregrounding developer perception as an essential component in productivity evaluation.

II. Related Work

A. Evolving Perspectives on Developer Productivity

High-priority or mission-critical projects often rely significantly on the productivity of their developers [14], [15]. Productivity was typically measured in straightforward, numerically oriented methodologies—examining such values as the number of hours developers worked, lines of code they wrote, or documents they produced. This provided a fairly constricted and mechanical definition of what exactly developer performance is. However, more recent research [8], [12], [16] reveals a more complex perspective, with developers and industry professionals now taking into account a broader range of issues, such as social dynamics, emotional wellbeing, and the overall work environment. Studies also demonstrate that productivity can be rather diverse based on the individual, their specific job, and the organizational culture, so a blanket measurement approach is challenging [12], [16].

There has been greater focus in recent years on so-called "soft" factors—such as team morale, safety, developer attitude, and office atmosphere [8], [12]. These so-called soft factors, which were once considered secondary, are now considered to be key to enhancing creativity and performance. Knowledge from cognitive psychology and human-computer interaction has made it obvious how mind states like flow, thinking load, and attention duration affect the quality of the code directly along with the speed of development [17]–[19].

Case studies from the industry reflect this changing mindset, but they also reveal an ongoing tension. On one hand, some still regard hard metrics like output and velocity. On the other hand, methods to understand productivity in a more holistic manner are gaining interest [6], [20]. These companies are beginning to realize that relying on short-term metrics too much can cause serious problems in the long run: developer burnout, technical debt, or deteriorating code quality [20]–[22].

To address these problems, the larger paradigm of modeling considers approaches such as the SPACE framework. These approaches attempt to collect traditional performance metrics with the behavioral and emotional ones [6]. A great deal of research then suggest that productivity must be reconsidered as a complicated, very personal, and context-driven concept [23]. They also highlight the fact that study methods must be built that can link subjective experiences with hard data.

B. Productivity in Educational Contexts: Parallels and Challenges

In a school or university setting, the actual output by software engineering students is usually interpreted differently from the way that industry assesses it [24]. Spending time understanding a codebase, actively debating design trade-offs in study groups, reflecting deeply on feedback to improve problem-solving strategies, and collaboratively troubleshooting concepts before writing any code, albeit being non-artifactual activities, yet do not figure on conventional productivity indicators [6]. These educational challenges are translated into tracking issues in professional software development culture, involving aspects of team interaction, affective experiences, and working habits [8], [12], [25]. According to Vygotsky [26] and Guzdial [27], productivity in education is strongly tied to how students acquire skills, understand concepts, and develop professionally. Sometimes, students get engaged in activities for long periods that do not seem productive by regular standards but are quite necessary for laying strong foundations and sharpening problem-solving skills. This dissonance between what seems productive and what really advances learning foregrounds a larger problem of knowledge-worker assessment. It thus brings into focus teaching productivity alongside technical skills so students are prepared for professional settings, where vastly different expectations from those in the classroom exist.

C. The SPACE Framework: A Multi-Dimensional Model

The SPACE framework is a modern approach for measuring developer productivity introduced by Forsgren et al [6] stating that software engineering productivity must be understood through five different aspects:

- **Satisfaction** - happiness and mental state.
- **Performance** - quality and impact of the contributions.
- **Activity** - measurable actions such as task completion.
- **Collaboration** - interactions within the team.
- **Efficiency** - ability to work effectively.

The premise of the SPACE framework states that productivity should not be reduced to a single measure, as human

factors like well-being and team interactions matter. Forsgren [6] says that multiple dimensions of productivity must be monitored to prevent overlooking important contributions to productivity. The idea of a "productive" developer is one who not only writes quality code but also one who is content and collaborative. Although, not all dimensions are always applied in every study as they may be more applicable in different settings.

D. The Baidu Study

We draw on the study by Bao et al. [23] due to its strong influence and relevance in the field of developer productivity assessment. Conducted over several months, the study provides a comprehensive analysis of the contributions of hundreds of developers and gives a robust look into how productivity shifts in the real world. Several studies emphasize the utilization of a diverse array of metrics and indicators to evaluate developer productivity, as demonstrated in the works of Meyer et al. [28], Wagner and Ruhe [2], and Jones [1]. While some research initiatives or organizations may adopt all, some, or none of the metrics highlighted in the Baidu study within their assessment frameworks—such as commit count, lines of code added or removed, code reviews, build statuses, or release success rates—the industry generally converges on at least one traditional quantifiable measure of productivity, making such metrics still a common reference point in both research and practice [2], [3], [29].

Despite the variation in productivity definitions, studies still include at least one of these measurable indicators. Russo et al. [30] use Bao et al.'s findings to examine developer performance and satisfaction during enforced work-from-home hours using similar coding and bugfixing metrics. In another study, Shah et al. [31] highlight the importance of combining various productivity measures to provide an integrated evaluation framework. Their paper highlights the importance of combining repository data with issue-tracking metadata and code quality scores in order to make a contribution towards high-level understanding of both the quality and rate of software development.

Other research supports a multi-centric approach, highlighted in Oliveira et al.'s study [4], which promotes data triangulation, which combines several measures to get a more thorough picture of developer success. In a similar manner, Lima et al. [32] additionally underscore the importance of a diverse set of indicators such as commit frequency, complexity measures, and bug-related activity—to not only allow leaders to assess performance but also better identify training needs and refine project estimations.

In brief, the aforementioned studies are some of the studies showing the idea of productivity as dynamic interaction of efficiency, effectiveness, and quality, and not one single fixed concept. Organizations can evolve beyond simple notions of productivity to a richer and actionable concept by employing several streams of data, triangulating across various sources for knowledge, and situating metrics relative to real-world development flows—one that balances speed and sustainability, output and influence, and short-term performance and long-term project health.

E. Self Perceived Productivity

Self-perceived productivity allows developers to assess their own effectiveness beyond just measurable outputs. It allows one to measure their own productivity as a whole including teamwork, time spent problem-solving, and learning outcomes—all of which may be missed by existing frameworks [11].

Productivity research reveals a correlation between emotional state and individual sense of productivity that identifies productivity as including not only tangible results but also effort needed to achieving these results [8]. This identifies the difference between what we think productivity is and actual measurable numbers such as lines of code. In student groups, perceived productivity would be affected by many different variables, including but not limited to learning developments and collaboration challenges. These factors are not represented by output measures or SPACE measured productivity [6].

Summary of Related Work

Current research indicates how our conception of productivity in software engineering has changed markedly. The early focus was on counting outputs and measuring efficiency. More recently, researchers are studying the social, emotional, and contextual aspects affecting developer performance. Inevitably, the research has shifted attention to long term sustainability in terms of code quality and developer wellbeing. While short-term productivity measures are still relevant in some capacity, they are increasingly considered inadequate or outright damaging when they do not consider the risks of burnout, growing technical debt, or complicating team dynamics. There is growing agreement in academia and industry about the need for a more holistic perspective to sustain high-quality, effective work in a sustainable way, namely, productivity that is informed by personal experience, long-term impact, and the *messy* reality of software development.

Herein, we explore whether the perceptions of students align with their measured productivity rankings by contrasting self-perceived productivity with established multi-dimensional frameworks of works such as the SPACE model and traditional, output-focused assessments incorporating multiple quantitative metrics as exemplified and used in the Baidu study.

III. Methodology

A. Overview

The methodology we used to compare perceived productivity with traditional methods and productivity scores derived from the SPACE framework involved a combination of survey data and quantitative data gathered from the project

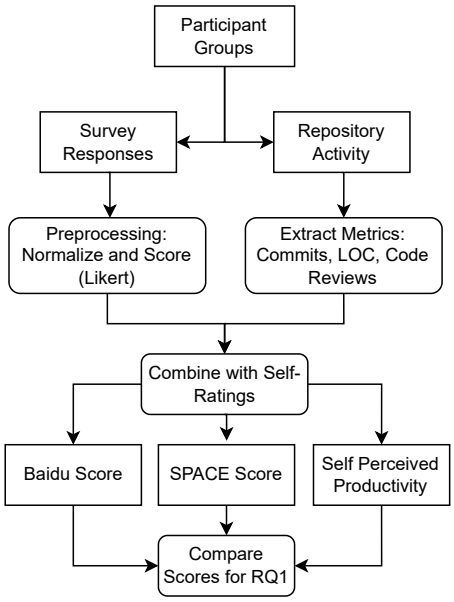

Fig. 1. Overview of the research methodology visualizing participant inputs, collected data, transformation, and how RQ1 is addressed.

repositories of several student software development teams. Teams participating in our study were tasked with creating a mobile application in Android Studio to connect and interact with an Arduino micro-controller via an MQTT broker, as part of a university-assigned project. These teams completed this project over the course of ten weeks in Java and C++, and we applied each of the three methods for a two-week period during the project. In order to control for the time our participants spent learning the required technologies and workflow (Git and Agile), we conducted our study from the fifth week of development onward. Our research methodology overview is detailed in Figure 1.

Our first method for assessing productivity is based on the work of Bao et al. [23], with commit count, LOC added, and code reviews performed included in determining the overall productivity of each member.

In addition to this, we utilized the SPACE framework for measuring productivity. Given the nature of the projects our participants undertook, we chose four of the five dimensions: satisfaction, performance, activity, and collaboration. The final dimension of efficiency was excluded due to the difficulty in obtaining a suitable metric from teams that lack experience in documentation. Since the SPACE framework requires a minimum of three dimensions and we have four, this does not pose a problem. The metrics we used to collect data in these four dimensions is discussed in the next section.

For measuring self-perceived productivity, we used a simple five-point Likert scale for participants to self-assess their productivity. This is consistent with similar research as seen in a previous study on the effects of feelings on perceived productivity [9].

B. Participant Selection

For us to understand the differences between productivity metrics and theorize where they arise from, a structured study environment where the full participation of development teams can be ensured is essential. This was a large part of the rationale for us deciding on students as participants. Unlike surveys sent out over the Internet that rely on a small percentage of responses from teams all over the world, examining a full development team allows us to account for every kind of task in the development process that leads to the eventual product, ensuring that all valuable inputs can be captured. Selecting participants from a structured university environment allowed us to achieve this.

Another reason for this decision is that we have placed special value on the opinions of the next generation of software engineers. While it is the case that industry professionals with more experience would likely have a better understanding and perspective on what work is ”productive”, these opinions have been formed in an environment that reinforces well-established industry norms and practices. By surveying the opinions of the next generation of software engineers before these perceptions have formed, we aim to potentially discover innovative and novel ideas on what defines productivity in the industry.

C. Data Collection

We decided that a range of developer teams was desirable in order to control for social factors and potential anomalous circumstances. Although our sample consists of students rather than working professionals, this choice allowed for better control and accessibility while still providing valuable insights into early-stage perceptions of productivity.

Our study consists of both qualitative and quantitative data, where we followed each of our six participant groups throughout the course of their projects and conducted surveys with each team group for a two-week period, in addition to a review of each team member’s development activities as documented on their GitLab/GitHub repository for the period. For control purposes, participants absent from the data collection session were excluded from the study, as we recognize that responses to the survey are influenced by the environment and time that they are conducted at. An overview of our data collection procedure is illustrated in Figure 2.

The quantitative productivity data collected from the repositories consists of several factors that we determined to be valuable to the development process: number of commits, lines of code committed, and number of code reviews performed. In order to utilize the SPACE framework to measure productivity, we also collected data from the repositories on the number of successfully completed issues per member and merge request comments on the project for the past two weeks.

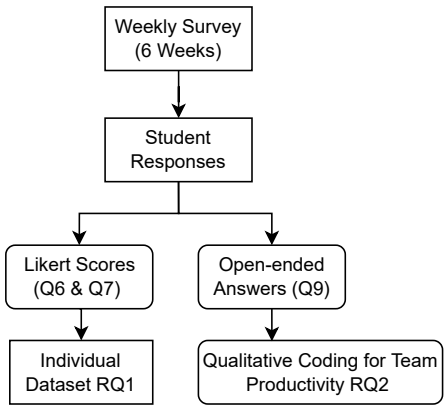

Fig. 2. Steps in the survey data collection process, including the structure, and how RQ1 and RQ2 are supported.

The qualitative data collected are from open-ended questions included in our survey. These questions are intended to help us understand the opinions of software engineering students on what they consider "productive" work. Survey questions are listed in the appendix.

The final question in our survey guide is essential to answer RQ2, which forms the exploratory aspect of our study. In combination with data collected from the git repositories of our participants, the responses from the survey questions enabled us to capture both a traditional and a modern metric for productivity that can be compared with the self-perceived productivity of participants for addressing RQ1.

D. Data Analysis

To examine the collected data, we used two productivity ranking approaches. The first is the Baidu metric-centered method, which focuses on quantitative development activities such as commits and code reviews, as shown in Table I.

TABLE I
BAIDU FRAMEWORK PRODUCTIVITY METRICS

Metric	Data Source	Included
Commits	Git History	✓
Lines of Code (LOC)	Git History	✓
Code Reviews	GitLab Comments	✓

The second approach applies the SPACE framework, which captures a broader set of productivity dimensions alongside quantitative data. Table II summarizes the SPACE dimensions used in this study.

The repository activity data was extracted from each participating group's GitLab project, which we then parsed to retrieve relevant information including issue and merge request authorship, number of commits per participant, lines of code per participant, GitLab actions, as well as the full comment activity across the development cycle. These metrics formed the foundation of both ranking methodologies - with the survey data being included for the SPACE ranking. Our data analysis process is visualized in Figure 3.

TABLE II
SPACE FRAMEWORK DIMENSIONS USED IN THE STUDY

Dimension	Data Source	Included
Satisfaction	Self-assessment (Q7)	✓
Performance	Task completion, self-assessment (Q6)	✓
Activity	Commits, Lines of code	✓
Collaboration	Comments, team communication	✓
Efficiency	Not measured	✗

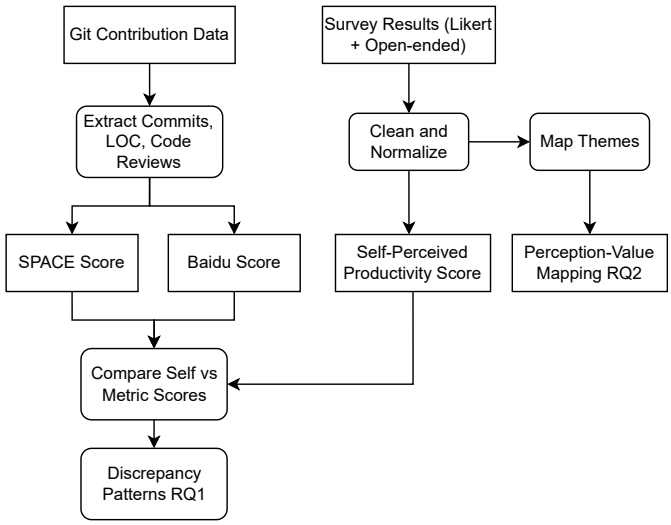

Fig. 3. Our data analysis pipeline. This figure shows how survey and Git analytics feed into productivity scoring and support for RQ1 and RQ2.

1) Baidu-Style Metric-Centered Ranking: The Baidu metric-centered approach relies on three repository metrics: number of commits, lines of code, and code reviews performed. These correspond to the columns *Commits (%)*, *+ lines*, and *comments written* in our dataset. In order to ensure comparability across the different contributors, each metric was normalized using Min-Max scaling, where each participant's score was divided by the maximum observed score within the team, resulting in final scores between 0 and 1.

$$\text{Normalized Score} = \left(\frac{\text{Individual Score}}{\text{Max Score in Team}} \right) \quad (1)$$

The normalized values were then weighted based on their perceived importance in the Baidu study, where commits and lines of code received a weight of 0.4, and code reviews a weight of 0.2. The final Baidu productivity score for each participant was computed as:

$$\text{Final Score} = (\tilde{\text{Commits}} \times 0.4) + (\tilde{\text{LOC}} \times 0.4) + (\tilde{\text{Reviews}} \times 0.2) \quad (2)$$

The results reflect an author's relative contribution intensity within their group based solely on repository metrics.

2) SPACE Framework Ranking: We assess four dimensions of the SPACE framework: Satisfaction (S), Performance (P), Activity (A) and Collaboration (C). Efficiency was not included as we could not reliably extract time-based developer metrics from the GitLab repositories.

To quantify **Satisfaction** (S), we collected the self-reported ratings from each participant submitted through the survey where S represents how satisfied the participant was with their own performance, which was then scaled to a 0–100 range. This is also done for the self-reported productivity rating, which is used in calculating P.

Performance (P) was computed as a weighted average of:

- GitLab actions performed (e.g., assigning issues, creating branches)
- Task completion (measured as the sum of issues and merge requests assigned)
- Self-rated productivity score

Each sub-metric was first normalized using the z-score formula:

$$z = \frac{x - \mu}{\sigma} \tag{3}$$

where x is the raw value, μ is the mean, and σ is the standard deviation of the metric. We set a mean of 50 and a standard deviation of 10, which mirrors the standard T-score - shifting all values into a consistent and positive range. The z-scores were then scaled between 0 and 1 to allow comparability with Baidu using:

$$\tilde{z} = \frac{z - z_{\min}}{z_{\max} - z_{\min}} \tag{4}$$

The final P score was computed as:

$$P = 0.4 \times \tilde{\text{Actions}} + 0.4 \times \tilde{\text{Tasks}} + 0.2 \times selfprod \tag{5}$$

Activity (A) was measured by the number of commits made during the analysis period and **Collaboration** (C) was based solely on the amount of code review activity in the form of written comments. Finally, the SPACE score was calculated using the weighted formula:

$$(S \times 0.2) + (P \times 0.3) + (A \times 0.3) + (C \times 0.2) \tag{6}$$

3) Analysis of Open-Ended Responses: For the responses to the open-ended question Q9, we applied a qualitative content analysis approach, which allowed us to identify common activities in productivity factors. Each researcher independently coded a portion of the responses while inter-coder reliability was assessed using percent agreement to ensure consistency in assigning each of the themes. This resulted in a percent agreement of 93%, which is a high level of reliability, likely due to the clear nature of the responses.

IV. Results

A. Participant Demographics

After contacting first- and second-year Software Engineering and Management BSc students at Gothenburg University, six groups of four to six members agreed to participate in our study.

TABLE III
PARTICIPANT DEMOGRAPHICS FROM THE SURVEY

Attribute	Value
Groups Surveyed	1–6 (Total)
Participants	24
Age (mean)	21.5
Age Range	18–33
Avg. Programming Experience (yrs)	1.42
Experience Range (yrs)	0–5
Prior Degree in Related Field	5

A total of 24 first-year software engineering students completed the survey with a relatively consistent profile. Their ages ranged from 18 to 33 years, with a mean age of approximately 21.5 years. Most students reported 1 to 2 years of programming experience, with an average of 1.42 years and a maximum of 5 years.

While the majority of students had not previously taken a degree in a related field, 5 students indicated that they had background in subjects such as cybersecurity, logistics, or a related technical program.

This demographic consistency allowed a fair analysis of the productivity and satisfaction metrics collected, without major outliers in the level of education or professional background.

B. Quantitative Analysis

After calculating objective metrics for each participant on their personal productivity using the SPACE framework and the method used by Bao et al. [23], we were able to compare them with the participants' own self-perceived productivity, as reported via surveys. Although we were unable to perform statistical tests due to our small sample size (n=24), we have calculated Pearson's correlation coefficient (r) and the coefficient of determination (R^2) to describe the trend in our results. Given that all three measures are intended to measure the same thing, we have assumed a degree of linearity.

Overall, our data shows a weak association between self-perceived productivity and the two metrics we used. Self-reported productivity scores can only weakly predict how students will be evaluated using these common industry standard metrics. See Table IV below.

TABLE IV
BAIDU FRAMEWORK PRODUCTIVITY METRICS

Comparison	r	R^2	Strength
Self-Reported Productivity vs. Baidu	0.31	0.094	Weak
Self-Reported Productivity vs. SPACE	0.31	0.097	Weak
Self-Reported Productivity vs. Satisfaction	0.787	0.619	Strong

The results we plotted comparing each metric with participants' self-perceived productivity are shown in Figure 4 and Figure 5 below.

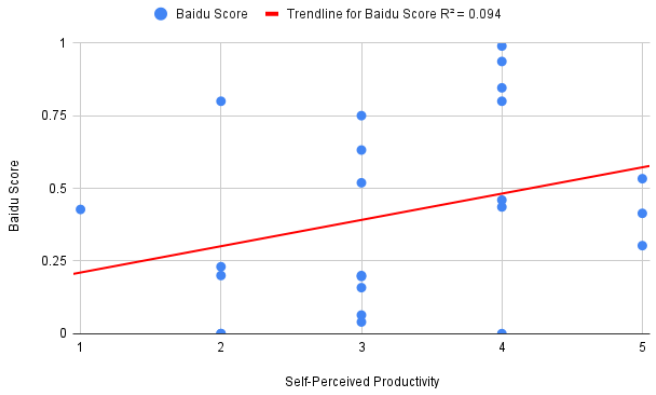

Fig. 4. Self-Perceived Productivity Rating vs. Baidu Score

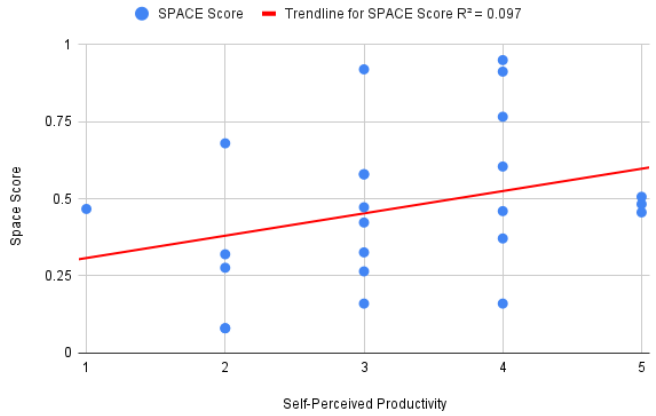

Fig. 5. Self-Perceived Productivity vs. SPACE Score

Although we can observe a weak positive association here, there is a large amount of variance that is not accounted for. The R^2 values between self-perceived vs. Baidu Score and self-perceived vs. SPACE score are 0.094 and 0.097 respectively. The data points in both of these plots are so widely scattered that it makes self-perceived productivity a weak predictor for both metrics.

Since we know that both metrics capture similar data, these values being close together is not unexpected. The R^2 value being slightly higher for the relationship between self-perceived productivity and SPACE score is also not a surprise, since there is a strong association between self-perceived productivity and work satisfaction, as seen in Figure 6 below, which is one of the four components of the SPACE framework that we captured to calculate the scores for each participant.

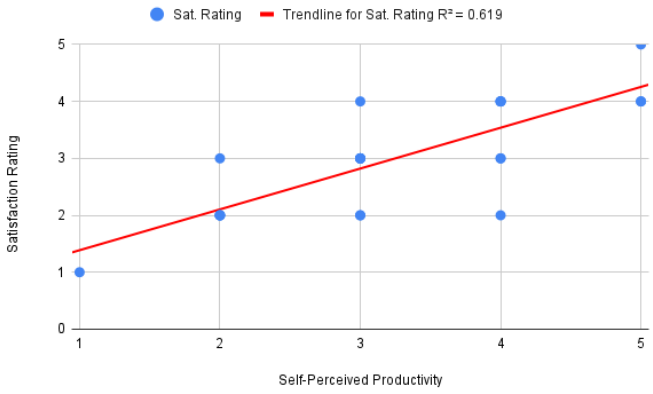

Fig. 6. Self-Perceived Productivity vs. Work Satisfaction

C. Qualitative Analysis

Content Analysis of Qualitative Data

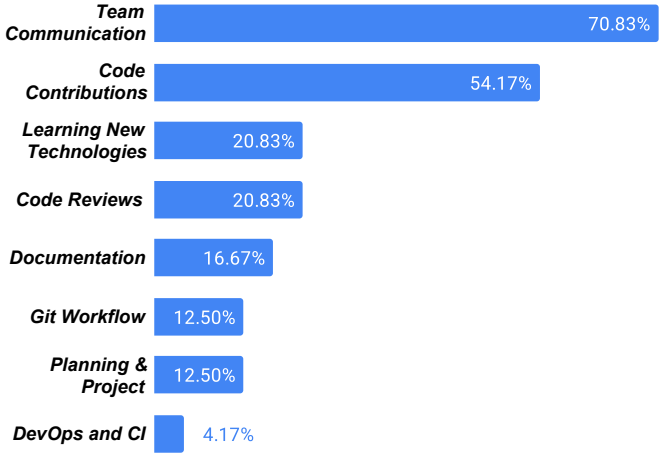

Fig. 7. Categorized themes of productive work as reported by students. Values represent the percentage of participants who mentioned each theme in response to the open-ended question on perceived productivity (Q9).

The chart in Figure 7 presents the results of our content analysis of responses. The most frequently mentioned productive activity was *Team Communication* at 70.83%, even higher than *Code Contributions*, which is often considered the standard measure of productivity. This difference could mean that students view team work as essential to productivity, in that being engaged in communication is not just a secondary activity, but also being a component of productive working. *Code Contributions* was quoted by over half of the students at 54.17%, which still reinforces its relevance as a traditional productivity indicator. However, being reported less frequently than communication implies that code alone is not the only or main output that matters.

Learning New Technologies and *Code Reviews* both rank at 20.83%, seen as productive by a smaller group of students showing value in growth and feedback. Students usually associate learning as a productive investment, valuing team knowledge sharing which, in turn, provides long-term contributions to the project. This also applies to code reviews, where a standard of quality is expected and achieved with valuable feedback.

Tasks like *Documentation*, *Git Workflow*, and *Project Planning* are less frequently mentioned, while still being important to some students. These activities support organization within the team as well as the maintainability of the project, which is usually important to maintain long-term efficiency. However, they are often difficult to detect with traditional metrics unless manually tracked through the GitLab/GitHub interface or the underlying communication channels in use.

In an industry setting, creating pipelines are crucial in order to maintain software stability and streamline development. Hence, the low mention rate of *DevOps and CI* by the students could be an indication that it may be less visible or less understood, especially in a learning environment. It could also simply be that the students might not associate them with their own productivity, even if they benefit the team.

The sharp drop-off after the top two categories, *Team Communication* and *Code Contributions*, indicates that students consider some set of tasks key to productivity, whereas others are seen as secondary. This concentration likely depends on how students distribute their time or the aspects they regard as most influential in team projects. However, the frequency of certain themes may be influenced by how their course projects are designed. For example, if code reviews were not emphasized in any of the course material, students may have not seen the activity as essential to productivity in the first place. This leads to the omission of certain activities like testing, or user feedback, which could suggest students see these as part of coding itself or just don't recognize their separate importance.

Frequency of Themes in Qualitative Data

In addition to mapping the themes, we also examined how many different types of productive activities each student reported (Figure 8). The number of themes mentioned per student ranged from 1 to 4, with most (46%) mentioning only two distinct themes. A smaller portion of the students identified just one (25%) or three (25%) themes, and only one student (4%) mentioned four.

This distribution shows the variation in how broadly students define productivity. While some students focused on a single dominating task, such as coding, others linked productivity with a wider range of activities, including planning and documentation. This suggests that the student's perception of productivity are not uniform and may be influenced by their specific role within the team, prior experience, or priorities in the group work - which highlights the importance of considering different perspectives when evaluating productivity in student teams.

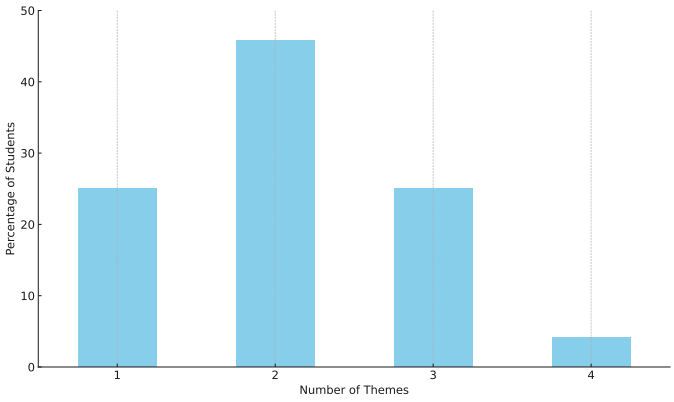

Fig. 8. Percentage of students who mentioned 1 to 4 unique themes of productive work in response to the open-ended question (Q9) on what they considered productive activity

V. Discussion

A. Evaluating Research Question 1: How well do traditional (e.g., LOC, number of commits, number of code reviews) and modern (e.g., SPACE framework) productivity metrics align with how productive software engineering students perceive themselves to be?

Overall, our study was limited in the number of participants, so we could not conduct statistical tests on our results with a sufficiently high level of confidence. The lack of experience of first year students also contributed to the uncertainty of our results, especially given that accurate self-assessments of productivity partly rely on previous experience and expert knowledge of the domain.

The quantitative data we collected in the study shows a very weak association between both traditional and modern methods for measuring productivity and how productive our participants perceived themselves to be. This is not entirely surprising, as previous research has suggested that technical factors are not the strongest predictors of self-perceived productivity [33]. This follows intuitively from the qualitative data we collected about what tasks our participants viewed as productive, where the most common answers were connected to the theme of communication, one that is non-technical and not captured by either objective metric. See Figure 7.

Interestingly, although interpersonal and collaborative work was greatly valued by students, this did not necessarily translate into how students quantified their own productivity. One possibility is that students might underestimate their own non-technical contributions or might view them as less legitimate than more quantifiable measures such as code, even though they value them in others.

Despite this, research around productivity in software engineering still commonly relies on self-perceived measures of productivity to draw conclusions [34]–[36]. Our study raises the crucial question as to whether objective metrics are unsuitable for accurately measuring productivity, or if software engineering students are largely incapable of precisely assessing their own productivity, or both. A key problem here is that students were not given a common definition of what would be considered value in their contributions. Without a shared understanding, students can over- or under-estimate their own productivity on the basis of individual assumptions in estimating value of different kinds of work in other group members. Objective metrics suffer primarily from their inability to capture intangible contributions, such as team members who generate novel ideas or techniques for developing the product, or those who make positive contributions to the team atmosphere. On the other hand, self-perceived productivity ratings are skewed by a range of biases such as differing cultural attitudes, inherent biases in self-perception, and social desirability.

Another productivity study published in 2022 explores the validity of peer evaluations for assessing productivity and found that this method moderately converges with objective metrics [37]. It is possible that by using peer evaluated productivity we can capture the aforementioned intangible contributions without some of the limitations that come with self-perception. This idea is supported by a 2015 meta analysis across other industries which examines multiple methods of evaluating productivity [38], in which peer evaluated productivity converged with objective metrics more so than self-perceived productivity.

The results of our study further reaffirm the idea that self-perception is a weak indicator of productivity, and we propose that future research in this area could deepen our understanding by including supervisor-rated and peer evaluated ratings of productivity in order to compare which of the three converges with objective metrics the most, specifically within the field of software engineering.

B. Evaluating Research Question 2: What kinds of tasks do software engineering students consider "productive" work, and how do these perceptions compare with what is captured by standard productivity measures?

The qualitative findings show that students frame productivity in a broader way that more objective measures are not able to detect. Students most frequently emphasized interpersonal activities as the core of productive work, which implies a more process-oriented understanding of productivity where activity towards coordination, planning, and knowledge sharing counts as valuable even if it creates little measurable trace.

Several students identified some of the activities like *Learning New Technologies* and *Planning & Project*, which foster the success of long-term projects but never spotlighted by conventional metrics. Technical practices like *DevOps and CI* did not receive much emphasis, suggesting that students have a tendency to de-value certain practices or associate them less with their own productivity. This attests to how perceptions are shaped by the visibility of specific tasks in learning contexts.

In general, our results indicate that common measures of productivity do not accurately capture work from the student's own perspective. This indicates a gap between those metrics commonly used to measure developer productivity and the nature of collaborative software development in education. As a result, any assessment framework attempting to understand productivity must treat their work from both a social and developmental context.

VI. Limitations

A number of issues may have imposed constraints and flaws on our research that could have affected how accurately we interpreted our findings as well as the quality of the data we collected. These limitations may result from the characteristics of our sample, the techniques employed to gather data, the manner in which answers are reported, and outside factors.

1) Internal Validity
   - Selection bias: Students may have varied in levels of previous experience (e.g., internships, coursework, or self-taught skills) and familiarity with development tools, which could influence how they report their own productivity. We addressed this issue by collecting demographic and experience data to account for differences in prior knowledge and expertise which ensured that they were considered during the analysis.
2) Construct Validity
   - Hawthorne effect: The students may have altered their behavior by knowing they are being observed. We addressed this by conducting interviews biweekly to minimize the potential for teams to diverge from their would-be workflow as a result of the research presence imposed upon them.
   - Social desirability bias: Self reported productivity may have been inflated towards what students thought was expected. As a measure we emphasized that the responses had no impact on their evaluation, and are anonymous.
3) External Validity
   While our study focused on first- and second-year students to capture the perceptions of emerging software engineers, this choice could be seen as a limitation on the generalization of our findings to more experienced developers. However, this perspective offers a valuable understanding of how productivity concepts develop early in education. We have detailed our methodology to allow future studies to replicate and extend our work with more diverse participant pools

VII. Conclusion

Our study aimed to explore the validity of productivity metrics in software engineering by comparing them with self-

perception scores over a range of student participants. By collecting data to rank our participants on their productivity using several measures, we were able to examine the relationship between self-perceptions of productivity and scores derived from objective contributions. We found a positive association between the objective metrics and self-perception scores, but struggle to justify research based on self-perceived productivity alone given how weak this relationship is. Through open-ended survey questions, we were also able to shed light on student developers' opinions on the meaning of productivity in software engineering. Our findings from these answers support the idea that there are a significant number of intangible contributions not captured by objective metrics, and therefore these methods for assessing productivity are flawed and not representative of how productivity should be defined.

To help align perceptions of value in team projects, future educational settings could benefit from a collaborative exercise/activity where students help create a rubric that defines productive contributions, which would include both technical and non-technical work. This may help bridge the gap between self- and peer-assessed productivity, and reinforce shared accountability within student teams.

Although our study focused on just one course with 24 students across 6 different groups, it opens up larger discussions about the role of metrics in education. Future research could investigate whether opinions change as students transition into the workforce and explore ways to adapt productivity models to better nurture student development and team dynamics. Other methods of assessing productivity within small teams such as peer-evaluated or supervisor-evaluated productivity have been experimented with across other fields to positive results, so we propose that this is a direction for future work.

Acknowledgment

We would like to thank Cristy Martinez Montes, our supervisor, for her guidance, support, and patience throughout the course of this research. We are also deeply grateful to the students who participated in the surveys, their responses were essential to the success of our study. Additionally, we appreciate the encouragement provided by our peers and course coordinators.

- Kai Rowley
  - Section I: Introduction - First and last paragraphs
  - Section II: Related Work - Subsections A, B, D and Summary of Related Work
  - Section VI: Limitations
- Sam Hardingham
  - Section I: Introduction - From Paragraph 2 to 4 / From "Across industries..." to "...the factors these frameworks emphasize."
  - Section III: Methodology - Subsections A, B, and C
  - Section IV: Results - Subsection B
  - Section V: Discussion - Subsection A
  - Section VII: Conclusion
- Daniel Van Den Heuvel
  - Section II: Related Work - Subsections C and E
  - Section III: Methodology - Subsection D
  - Section IV: Results - Subsections A and C
  - Section V: Discussion - Subsection B

Data Availability

The Python scripts used for data preprocessing and analysis are publicly available at: https://github.com/danielvdh24/productivity-frameworks. The repository includes selected figures used in the thesis, anonymized survey responses, and data tables related to theme analysis and productivity rankings. No personal or raw identifying data is shared due to privacy reasons.

References

[1] C. Jones, "Software metrics: good, bad and missing," *Computer*, vol. 27, no. 9, pp. 98–100, 1994.

[2] S. Wagner and M. Ruhe, "A systematic review of productivity factors in software development," *arXiv preprint arXiv:1801.06475*, 2018. 6 pages.

[3] C. H. C. Duarte, "Software productivity in practice: A systematic mapping study," *Software*, vol. 1, no. 2, pp. 164–214, 2022.

[4] E. Oliveira, E. Fernandes, I. Steinmacher, M. Cristo, T. Conte, and A. Garcia, "Code and commit metrics of developer productivity: a study on team leaders perceptions," *Empirical Software Engineering*, vol. 25, pp. 2519–2549, 2020.

[5] M. Sanwal and I. Deva, "A multi-faceted approach to measuring engineering productivity," *International Journal of Trend in Scientific Research and Development*, vol. 8, no. 5, pp. 516–521, 2024.

[6] N. Forsgren, M.-A. Storey, C. Maddila, T. Zimmermann, B. Houck, and J. Butler, "The space of developer productivity: There's more to it than you think.," *Queue*, vol. 19, no. 1, pp. 20–48, 2021.

[7] B. Çaglayan and A. B. Bener, "Effect of developer collaboration activity on software quality in two large scale projects," *The Journal of Systems and Software*, vol. 118, pp. 288–296, 2016.

[8] D. Graziotin, X. Wang, and P. Abrahamsson, "Are happy developers more productive? the correlation of affective states of software developers and their self-assessed productivity," in *Product-Focused Software Process Improvement: 14th International Conference, PROFES 2013, Paphos, Cyprus, June 12-14, 2013. Proceedings 14*, pp. 50–64, Springer, 2013.

[9] D. Graziotin, X. Wang, and P. Abrahamsson, "Do feelings matter? on the correlation of affects and the self-assessed productivity in software engineering," *Journal of Software: Evolution and Process*, vol. 27, no. 7, pp. 467–487, 2015.

[10] S. Biswas, "Predictive metrics: Transforming engineering productivity and software quality," *International Journal of Computer Trends and Technology*, vol. 73, no. 1, pp. 51–56, 2025.

[11] S. Wagner and F. Deissenboeck, "Defining productivity in software engineering," in *Rethinking Productivity in Software Engineering* (C. Sadowski and T. Zimmerman, eds.), p. 29–38, Berkeley, CA: Apress, 2019.

[12] P. Ralph, S. Baltes, G. Adisaputri, R. Torkar, V. Kovalenko, M. Kalinowski, N. Novielli, S. Yoo, X. Devroey, X. Tan, *et al.*, "Pandemic programming: How covid-19 affects software developers and how their organizations can help," *Empirical software engineering*, vol. 25, pp. 4927–4961, 2020.

[13] A. Menolli, T. A. Coleti, and M. Morandini, "Impact of remote work on software teams: A qualitative study," in *Workshop sobre Aspectos Sociais, Humanos e Econômicos de Software (WASHES)*, pp. 51–60, SBC, 2023.

[14] G. S. de Aquino Junior and S. R. d. L. Meira, "Towards effective productivity measurement in software projects," in *Proceedings of the 2009 Fourth International Conference on Software Engineering Advances (ICSEA)*, pp. 241–249, IEEE, 2009.

[15] D. R. Chand and R. G. Gowda, "An exploration of the impact of individual and group factors on programmer productivity," in *Proceedings of the 1993 ACM Conference on Computer Science*, pp. 338–345, ACM, 1993.
[16] C. Jaspan and C. Sadowski, "No single metric captures productivity," in *Rethinking Productivity in Software Engineering* (C. Sadowski and T. Zimmermann, eds.), pp. 13–20, Apress, 2019.
[17] A. Abbad-Andaloussi, "On the relationship between source-code metrics and cognitive load: A systematic tertiary review," *Journal of Systems and Software*, vol. 198, p. 111619, 2023.
[18] D. Graziotin, X. Wang, and P. Abrahamsson, "Happy software developers solve problems better: Psychological measurements in empirical software engineering," *PeerJ*, vol. 2, p. e289, 2014.
[19] N. Forsgren, E. Kalliamvakou, A. Noda, M. Greiler, B. Houck, and M.-A. Storey, "Devex in action: A study of its tangible impacts," *Queue*, vol. 21, no. 6, pp. 47–77, 2023.
[20] C. S. Lee, M. Ramsey, and C. M. Hicks, "Is our organization actually measuring productivity? how contrasting organizational and individual measures of engineering success is an opportunity to drive engineering transformation," *arXiv preprint arXiv:2305.11030*, 2023.
[21] M. Kuutila, M. Mäntylä, U. Farooq, and M. Claes, "Time pressure in software engineering: A systematic review," *Information and Software Technology*, vol. 126, p. 106257, 2020.
[22] R. Ramač, V. Mandić, N. Taušan, N. Rios, S. Freire, B. Pérez, C. Castellanos, D. Correal, A. Pacheco, G. Lopez, C. Izurieta, C. Seaman, and R. Spinola, "Prevalence, common causes and effects of technical debt: Results from a family of surveys with the it industry," *Journal of Systems and Software*, vol. 184, p. 111114, 2022.
[23] L. Bao, T. Li, X. Xia, K. Zhu, H. Li, and X. Yang, "How does working from home affect developer productivity?—a case study of baidu during the covid-19 pandemic," *Science China Information Sciences*, vol. 65, no. 4, p. 142102, 2022.
[24] V. Garousi, G. Giray, E. Tüzün, C. Catal, and M. Felderer, "Closing the gap between software engineering education and industrial needs," *arXiv preprint arXiv:1812.01954*, vol. abs/1812.01954, no. 1, pp. 1–15, 2018.
[25] D. Smite, A. Tkalich, N. B. Moe, E. Papatheocharous, E. Klotins, and M. P. Buvik, "Changes in perceived productivity of software engineers during covid-19 pandemic: The voice of evidence," *Journal of Systems and Software*, vol. 186, p. 111197, 2022.
[26] L. S. Vygotsky, "Mind in society: The development of higher psychological processes," in *Mind in Society: The Development of Higher Psychological Processes* (M. Cole, V. John-Steiner, S. Scribner, and E. Souberman, eds.), pp. 1–174, Harvard University Press, 1978.
[27] M. Guzdial, "Learner-centered computing education for computer science majors," in *Learner-Centered Design of Computing Education*, pp. 83–94, Springer, 2022.
[28] A. N. Meyer, T. Fritz, G. C. Murphy, and T. Zimmermann, "Software developers' perceptions of productivity," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2014)*, pp. 19–29, ACM, 2014.
[29] J. Rashid, T. Mahmood, and M. W. Nisar, "A study on software metrics and its impact on software quality," *Technical Journal, University of Engineering and Technology (UET) Taxila, Pakistan*, vol. 24, no. 1, pp. 69–82, 2019.
[30] D. Russo, P. H. Hanel, S. Altnickel, and N. van Berkel, "Satisfaction and performance of software developers during enforced work from home in the covid-19 pandemic," *Empirical Software Engineering*, vol. 28, no. 2, p. 53, 2023.
[31] H. M. Shah, Q. Z. Syed, B. Shankaranarayanan, I. Palit, A. Singh, K. Raval, K. Savaliya, and T. Sharma, "Mining and fusing productivity metrics with code quality information at scale," in *2023 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 563–567, IEEE, 2023.
[32] J. Lima, C. Treude, F. Figueira Filho, and U. Kulesza, "Assessing developer contribution with repository mining-based metrics," in *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 536–540, IEEE, 2015.
[33] E. Murphy-Hill, C. Jaspan, C. Sadowski, D. Shepherd, M. Phillips, C. Winter, A. Knight, E. Smith, and M. Jorde, "What predicts software developers' productivity?," *IEEE Transactions on Software Engineering*, vol. 47, no. 3, pp. 582–594, 2021.
[34] C. I. M. Bezerra, J. C. de Souza Filho, E. F. Coutinho, A. Gama, A. L. Ferreira, G. L. a. de Andrade, and C. E. Feitosa, "How human and organizational factors influence software teams productivity in covid-19 pandemic: A brazilian survey," in *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*, SBES '20, (New York, NY, USA), p. 606–615, Association for Computing Machinery, 2020.
[35] A. Rot, M. Sobinska, and P. Busch, "Programming teams in remote working environments: an analysis of performance and productivity," in *2023 13th International Conference on Advanced Computer Information Technologies (ACIT)*, pp. 376–381, 2023.
[36] B. Johnson, T. Zimmermann, and C. Bird, "The effect of work environments on productivity and satisfaction of software engineers," *IEEE transactions on software engineering*, vol. 47, no. 4, pp. 736–757, 2021.
[37] C. C. Martin and K. D. Locke, "What do peer evaluations represent? a study of rater consensus and target personality," *Frontiers in Education*, vol. Volume 7 - 2022, 2022.
[38] B. Rangel, "Convergence of multiple rating sources with objective measures of work performance: A meta-analysis," master's thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 2015.

APPENDIX

Survey Questions

Q1. How old are you?

Q2. Do you hold any previous degrees in a related field?
Options: Yes (list previous degree), No

Q3. What is your current academic year?
Options: 1st, 2nd, 3rd

Q4. How many years of programming experience do you have?
Options: ¡1, 1, 2, 3, 4, 5+

Q5. What was your primary role in the team this past 2 weeks?
Options: non-specific, managerial, coding, front-end design

Q6. How would you rate your productivity over the past 2 weeks, 1 is very low, 2 is below average, 3 is average, 4 is above average, and 5 is very high?
Options: 1, 2, 3, 4, 5

Q7. On a scale of 1-5, how satisfied are you with your performance this past 2 weeks?
Options: 1, 2, 3, 4, 5

Q8. Which tasks did you work on this past 2 weeks?
Open-ended

Q9. What kinds of activities do you consider as "being productive", in regards to this project?
Open-ended;
Explanation: "Productivity in the field of software engineering is considered to be the ratio between the value of the finished product, compared to the value of the input required to produce it, but how contributions towards the completion of the product are valued can be evaluated in different ways. In common frameworks and measures of productivity, factors such as code contributions, issue completion, participating in code reviews, and personal productivity satisfaction ratings are used to determine how many valuable contributions a certain team member has made to a project. When asking this question, we're not asking about particular practices or ways of working that might be more efficient. This question is in relation to what actions or activities have you observed yourself or others perform that contribute to the final product that you are developing. Any possible kind of contribution you view as having added value to the development of the final product is a valid answer."