Time Warp: The Gap Between Developers' Ideal vs Actual Workweeks in an AI-Driven Era

Sukrit Kumar*†, Drishti Goel§†, Thomas Zimmermann¶, Brian Houck§∥, B. Ashok¶∥, Chetan Bansal§∥
*Georgia Institute of Technology, §Microsoft, ¶Microsoft Research

Abstract—Software developers balance a variety of different tasks in a workweek, yet the allocation of time often differs from what they consider ideal. Identifying and addressing these deviations is crucial for organizations aiming to enhance the productivity and well-being of the developers. In this paper, we present the findings from a survey of 484 software developers at Microsoft, which aims to identify the key differences between how developers would like to allocate their time during an ideal workweek versus their actual workweek. Our analysis reveals significant deviations between a developer's ideal workweek and their actual workweek, with a clear correlation: as the gap between these two workweeks widens, we observe a decline in both productivity and satisfaction. By examining these deviations in specific activities, we assess their direct impact on the developers' satisfaction and productivity. Additionally, given the growing adoption of AI tools in software engineering, both in the industry and academia, we identify specific tasks and areas that could be strong candidates for automation. In this paper, we make three key contributions: 1) We quantify the impact of workweek deviations on developer productivity and satisfaction 2) We identify individual tasks that disproportionately affect satisfaction and productivity. 3) We provide actual data-driven insights to guide future AI automation efforts in software engineering, aligning them with the developers' requirements and ideal workflows for maximizing their productivity and satisfaction.

Index Terms—developer productivity, satisfaction, software development, ideal workweek, automation

I. INTRODUCTION

In software engineering, the productivity and satisfaction of developers are pivotal factors that influence both individual performance, customer experience and ultimately, organizational success [1], [2]. The day-to-day activities which define a developer's workweek encompass a broad spectrum of tasks; from coding and designing new systems, to preparing documents, attending meetings, on-boarding new employees, adhering to security and compliance tasks, etc [3]. Each of these tasks is integral to the software development life cycle. Ideally, developers would prefer to allocate their time across these tasks in a way that optimizes both productivity and satisfaction— this can be referred to as their 'ideal workweek'. However, in practice, their 'actual workweek', can vary significantly from their 'ideal' due to fluctuating workloads, shifting organizational priorities, dependencies on other teams, technical challenges, the influence of the work environment, etc [4], [5], [6].

This misalignment between the ideal and actual workweek raises several intriguing research questions. Key among them is how these deviations impact developers' productivity and satisfaction. By examining these deviations, we can uncover patterns that may not only reveal the activities where developers struggle the most but also where they feel most productive and satisfied. Understanding these dynamics provides actionable insights for engineers, managers and organizations to create more effective work environments.

Furthermore, gaining deeper insights into the relationship between time allocation across activities, productivity, and satisfaction could inform the development of future and current AI-driven automation tools. With the growing role of AI in software engineering, understanding which tasks developers are most eager to automate becomes crucial. For instance, certain tasks may seem to offer high productivity but are inherently dissatisfying for developers to engage with. Identifying these tasks can help guide future research and development of these tools to be better aligned with real-world needs of the developers, allowing them to focus on more satisfying work.

In this paper, we a present the findings from our study aimed at analyzing the differences between a developer's 'ideal' and 'actual' workweek. Our study builds upon existing research in this field, particularly those exploring developers' typical and optimal workdays [3]. By analyzing deviations at the weekly level, we capture the cumulative effects of task allocation, while providing a more fine-grained understanding of how various activities impact productivity and satisfaction. To the best of our knowledge, this study is the first work that focuses on how deviations, both at the overall workweek and individual activity levels, affect developers' productivity and satisfaction. Our research additionally quantifies how deviations across different activities impact self-reported developer productivity and satisfaction. We also explore how AI tool usage influences the productivity and satisfaction of developers. In addition to that, we systematically analyze and present the developers' feedback on which tasks they would prefer to see automated using AI, providing valuable insights for future AI tool development.

An outline of the key research questions driving this study are as follows:

RQ1: How do developers allocate their time during a typical workweek, and how does this compare to their perception of an **ideal workweek?**

RQ2: How are developer's satisfaction and productivity affected by **deviations** from their ideal workweek?

RQ3: For which tasks do developers prefer using **AI tools**, and how does the frequency of AI tool usage **influence** their satisfaction and productivity?

The remainder of the paper is structured as follows, In Section II, we describe our survey design and limitations. In Section III we perform an analysis on the developers' ideal and actual workweeks. In Section IV, we analyze the impact of deviation from an ideal workweek on developer productivity and satisfaction. Section V discusses the impact of AI tool usage on developers and identify activities that developers want to automate. In Section VI, we discuss the implications of our findings for developers, managers and organizations. In Section VII, we discuss related works and, we finally present the conclusion in Section VIII.

II. METHODOLOGY

In this section, we provide a detailed description of the methodology used to design and conduct the survey. We then discuss potential limitations of our study.

A. Survey Design

To gain insights into the types of activities developers engage in during a typical workweek, we first conducted a series of exploratory interviews with 12 randomly selected participants. These semi-structured exploratory interviews provided a qualitative foundation, allowing us to iteratively develop a comprehensive list of high-level activities that reflect what a developer does over the course of a workweek. The findings from these interviews were instrumental in refining our survey design.

The survey was distributed to software engineers working in Microsoft teams across India and the United States between June to July 2024. A total of 6000 randomly selected individual contributor (IC) developers were invited to participate in the survey via email across multiple batches. We also sent follow ups to people who had started the survey but had not finished it and also to those who hadn't started it. We framed it as a study aimed at boosting developer productivity by helping us understand how they allocate their time in a workweek. The survey received 484 complete responses (responses rate of 8.06%). In the invite the participants were also informed that they could enter a sweepstake to win one out of ten $50 Amazon.com Gift Cards after finishing the survey. The ethics for this survey were reviewed and approved by the Microsoft Research Institutional Review Board (MSR-IRB), which is an IRB federally registered with the United States Department of Health & Human Services.

From the exploratory interviews, we identified sixteen key high-level activities, which were subsequently used to quantify the developers' time allocation in the survey. The list of activities are listed out in Table I. We designed the survey questions to broadly cover the main research questions that we had in mind and also included additional questions which we

TABLE I
KEY ACTIVITIES IDENTIFIED DURING THE EXPLORATORY INTERVIEWS

Developer Activities
Architecting and designing new systems
Coding new features
Development Environment Setup
Documentation
Pull Requests/Code Review
Code Refactoring
Debugging during development
Setting-up Monitoring and Dashboards
Authoring Tests
Security and Compliance
Addressing Customer Support Tickets
Communication and Meetings
Task Creation and Management
Giving Technical Presentations
Mentoring and Onboarding
Learning New Technologies

thought could offer interesting new insights. A brief summary of the questions is given below :

- Their current role and years of experience in the industry & team
- The number of hours spent on various activities in the previous workweek
- The percentage of time they would want to allocate to each activity in an *ideal* workweek
- How productive and satisfied they felt in the past workweek
- Activities they found to be most cognitively challenging
- How often do they use AI tools to assist them in various activities
- Two open-ended questions about the tasks they would want to automate using AI tools, and advice for new hires to boost their productivity and satisfaction levels

B. Limitations

This study presents an exploratory survey, which, to the best of our knowledge, is the first of its kind to introduce the concept of a developer's perceived ideal workweek and assess how deviations from it impact productivity and satisfaction. Additionally, it explores the specific activities developers would prefer to automate using AI tools. While this research provides valuable insights, the following limitations should be addressed by future studies.

External Validity: Since the study was conducted within a single company, the results may not be generalizable to organizations of different sizes, structures, or roles. Additionally, the low response rate and the survey's confinement to specific teams further limit the generalizability of the findings to developers in other contexts.

Construct Validity: The survey relies on the developers to accurately self-report the time they spend on the key

activities, productivity and satisfaction levels, which may introduce inaccuracies and biases. Additionally, the sequence of questions could have influenced the responses (e.g. activities to automate)

Internal Validity: The key activities identified in the survey may vary across different developers and organizations. While we sought to minimize this variability through exploratory interviews, the findings may still differ for other developer populations. Additionally, the survey data was collected during a security push at Microsoft, which may have impacted the typical workweek. To mitigate this, we distributed the survey in batches over a month and asked developers to report time spent on activities over the last five workdays, helping to average out short-term fluctuations from the security push and feature releases.

III. COMPARING DEVELOPER'S ACTUAL VS. IDEAL WORKWEEK

To answer RQ1, we asked the developers to specify the number of hours that they spend on key activities during a typical workweek, as well as the percentage of time they would ideally allocate to these activities. We transform the actual workweek responses into the percentage scale by dividing the time spent on each activity by the total number of hours in the response. We present the developers' actual and ideal workweek responses in Fig. 1(a) and Fig. 1(b) respectively, and also highlight the key differences in the percentage of time spent between them in Fig. 2.

A. The Actual Week: How Developers Spend Their Time

In the actual workweek, as depicted in Fig. 1(a), there is a considerable variation in the time developers spend across activities, specifically for 'Coding', 'Architecting & designing new systems', 'Security & Compliance', 'Debugging', and 'Communication & Meetings', with wide inter-quartile ranges and the presence of many outliers. This could be attributed to the differences in the type of project, team structure, level of experience, or individual roles. Furthermore, developers dedicate the majority of their time on 'Communication & Meetings' ($\approx$12%), 'Coding' ($\approx$11%), 'Debugging' ($\approx$9%), 'Architecting and designing new systems' ($\approx$6%), and 'Pull Requests/Code Reviews' ($\approx$5%). Conversely, they spend the least time on 'Learning New Technologies', 'Mentoring and Onboarding', 'Giving Technical Presentations', and 'Setting-up Monitoring and Dashboards'.

B. The Ideal Week: How Developers Want to Spend Their Time

In contrast to the actual workweek, the ideal workweek shows a less varied distribution across most activities, as seen in Fig. 1(b). Developers prefer to allocate their time primarily to 'coding' ($\approx$ 20%) and 'Architecting & designing new systems' ($\approx$ 15%), while maintaining a more balanced distribution across the remaining activities. This desire for balance is echoed by individual developers in their personal experiences. For instance, developer D223 offers this advice to new employees: *"Maintain a good balance of all the activities*
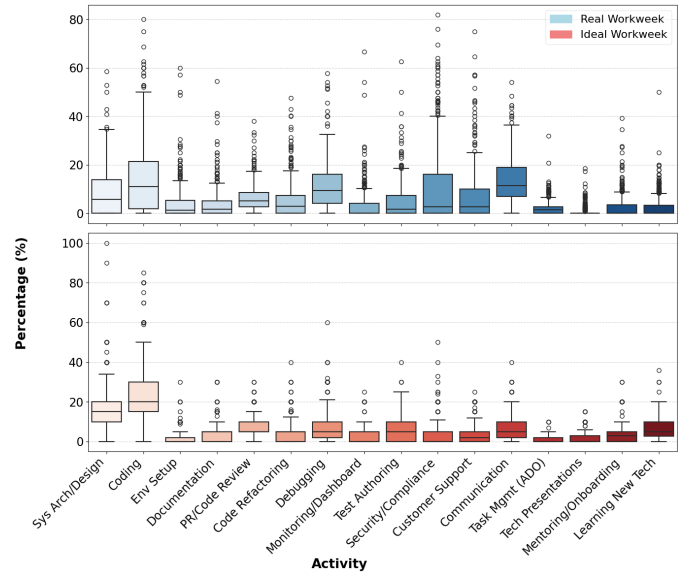

Fig. 1. Box Plot of Percentage Time Spent in an Actual Workweek and Ideal Workweek. (a) Top plot: Real Workweek. (b) Bottom plot: Ideal Workweek.
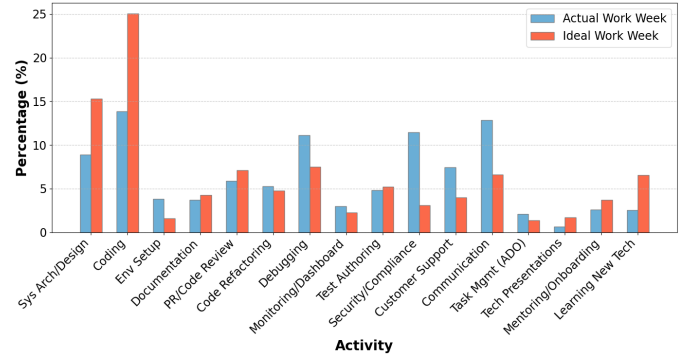

Fig. 2. Average percentage of time spent on the key activities in the Actual vs Ideal Workweek

needed to prevent burn out, and start with the activities that you find interesting."

The most striking differences are observed in 'Communication & Meetings' and 'Task Creation & Management', where developers show a strong preference for reducing the time spent. These activities are marked by high variability in the actual workweek, but are more contained in the ideal scenario, reflecting a widespread desire to minimize the time spent on these activities. Developer D342 emphasizes this sentiment: *"Optimize for operations. Automate EVERYTHING for Continuous Delivery. The word 'manual' should not be in your vocabulary unless referring to something to read. This way, you will have more time to focus on the creative process that is software engineering and be able to release often and with confidence."*. Similarly, D238 adds, *"More focus time and fewer meetings."*

In Fig. 2 we highlight several areas where there is a significant gap between the developers' ideal and actual workweek, shown by the percentage of time spent on each development

related activity. Developers want to spend more time on core activities such as 'Coding', 'Architecting and designing new systems', and less time on 'Communication', 'Addressing Customer Support Tickets', and 'Security & Compliance'. Additionally, the developers desire to spend less time on debugging code, and instead want to invest more time on preventative measures like 'PR/Code Reviews', 'Documentation', and 'Mentoring/Onboarding' that could potentially alleviate pains fixing issues due to poor documentation or code. This indicates a potential desire for improving code quality and reducing long-term maintenance costs and focusing and reducing technical debt [7]. Similarly, the preference to spend more time on learning new technologies, and engaging in technical presentations suggests that developers consider continuous learning and knowledge-sharing an essential part of their workweek. Developer D158 shares their personal experience regarding the importance of continuous learning in a workweek: *"Initially, my learning somewhat slowed as I was too caught up in the urgency to release features. The workweek was about completing as many ADO items as possible. However, my development as an engineer actually came from setting aside blocks of 2-3 hours whenever possible to go deep into certain topics. In my opinion, each team should have a mini boot-camp for onboarding engineers. The material does not need to be fully comprehensive but there should at least be a list of topics that the new member can deeply research. For example, if your job is to connect two internal systems together, you should make it your responsibility to deeply understand both systems even though it's not directly related to your task."*

> 💡 **Takeaway**: A developer's perception of an ideal work-week is not well aligned with their actual workweek. Developers want to spend more time on core development activities such as coding, designing new systems, learning new technologies, and knowledge sharing, while wanting to reduce the time spent on dealing with tasks related to security & compliance, communication, debugging, addressing incident/support tickets, and task management.

IV. IMPACT OF DEVIATIONS FROM THE IDEAL WORKWEEK ON DEVELOPER PRODUCTIVITY & SATISFACTION

In this section, we address RQ2 by evaluating how deviations from a developer's self-perceived ideal workweek influence their levels of satisfaction and productivity. Additionally, we identify the specific activity-level deviations and how they contribute to the observed effects.

A. Bridging the Gap: How Deviations from the Ideal Workweek affect Developers' Productivity and Satisfaction

In the survey, we asked the developers to rate their previous week on five different levels of productivity ('Very productive', 'Productive', 'Neither productive nor unproductive', 'Unproductive', and 'Very unproductive') and satisfaction ('Very satisfied', 'Satisfied', 'Neither satisfied nor dissatisfied', 'Dissatisfied', and 'Very Dissatisfied'). We split the developers based on each of these different categories and calculate and plot the Spearman rank correlation coefficient [8], and mean absolute error [9] between the actual and ideal workweek, which are defined by the time they spend on the 16 key activities and the time they would ideally allocate to each activity.
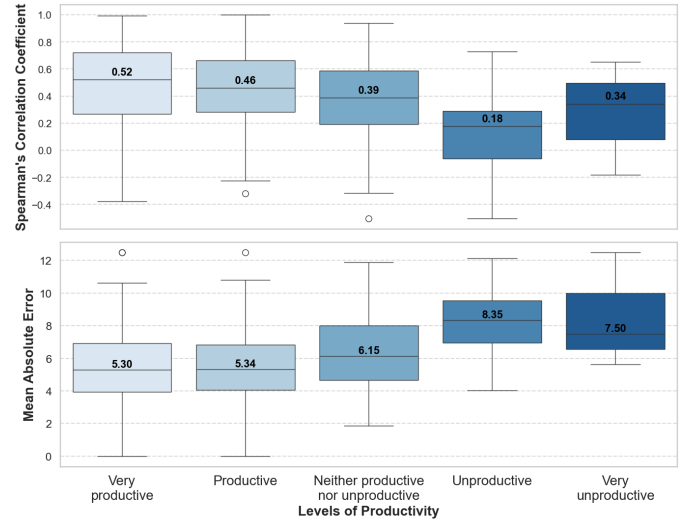

Fig. 3. (a) Spearman's Rank Correlation Coefficient for different categories of Productivity. (b) Mean Absolute Error for different levels of productivity
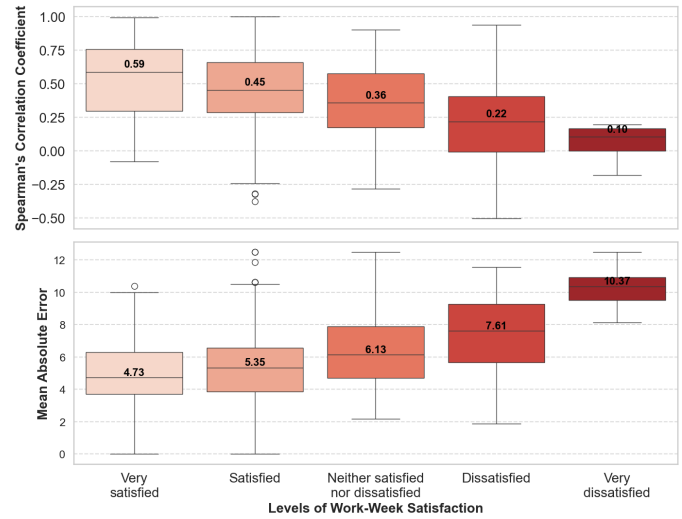

Fig. 4. (a) Spearman's Rank Correlation Coefficient for different categories of Satisfaction. (b) Mean Absolute Error for different levels of Satisfaction

Fig. 3(a) and Fig. 3(b), show how differences between actual and ideal workweeks affect developers' productivity. We observe two interesting trends in both the computed metrics. Developers who felt very productive had the highest median Spearman correlation (about 0.52) between their actual and ideal workweeks. This correlation decreased for less productive developers, reaching about 0.18 for unproductive and 0.34

for very unproductive developers. For the mean absolute error (MAE), the difference between actual and ideal workweeks increased as productivity decreased. The MAE rose from 5.3 for very productive developers to 7.5 for very unproductive ones. This shows that the deviation from the ideal work-week is relatively lower for developers who felt productive during a work-week.

A similar, but more pronounced pattern is observed in the satisfaction levels (Fig. 4(a) and Fig. 4(b)). The Spearman's rank correlation between their actual and ideal workweek was again highest for developers who felt very satisfied in that workweek (0.59) and it's lower at 0.10 for those who were very dissatisfied. Likewise, the MAE demonstrates a similar pattern, showing that developers who are more satisfied tend to be closer to their ideal workweek hence having lower MAE values (4.73 for very satisfied vs 10.37 for the very dissatisfied group).

> 💡 **Takeaway**: As developers' actual workweek diverges from their *perceived ideal workweek*, both productivity and satisfaction show a downward trend. This suggests that aligning with an ideal workweek is not merely a preference, but also crucial for maintaining high levels of performance. It also underscores the necessity of streamlining current research efforts on automation using AI to effectively bridge the gap between the actual and ideal workweek.

B. Impact of activity-level deviations on productivity & satisfaction

In the previous section, we highlighted the impact of deviation from a developer's ideal workweek on their productivity and satisfaction levels. To gain a more detailed understanding of these deviations, we calculate the differences between the time spent on these activities in the actual and ideal workweek (Actual - Ideal). Each variable in the Table II and Table III is the difference between the actual and ideal workweek for that particular activity. Positive difference values imply more time being spent on the activity than desired (over-allocation), while negative values imply less time being spent than desired (under-allocation). We apply an Ordinary Least Squares (OLS) regression model to identify specific activities that significantly influence developers' productivity and satisfaction. OLS was chosen for its ability to model linear relationships for continuous variables. Since the primary goal is to assess how incremental changes in activity allocation impact productivity and satisfaction, OLS provides a straightforward, and interpretable framework for the same.

We highlight activities having statistically significant positive relations in the color green and significant negative relations in red in both the tables. Conventionally, a p-value of < 0.05 indicates statistically significant results at a 95% confidence level, suggesting that the observed deviations in these activities are likely to have a meaningful impact on productivity or satisfaction.

TABLE II
OLS REGRESSION RESULTS - PRODUCTIVITY

Variable	coef	std err	t	**P> $|t|$**
const	4.1078	0.055	74.621	0.000
Arch. & Design	0.0009	0.003	0.355	0.723
Coding	0.0061	0.002	3.027	0.003
Dev. Environment	-0.0158	0.005	-3.422	0.001
Docs	0.0103	0.005	1.995	0.047
PR/Code Review	-0.0046	0.006	-0.803	0.423
Refactoring	0.0130	0.004	2.935	0.003
Debugging	-0.0021	0.003	-0.643	0.521
Monitoring & Dashboards	0.0063	0.005	1.197	0.232
Testing	0.0045	0.004	1.093	0.275
Security & Compliance	-0.0033	0.002	-1.431	0.153
Customer Support	-0.0051	0.003	-1.651	0.099
Communication	-0.0079	0.004	-2.120	0.034
Task Mgmt.	-0.0149	0.010	-1.451	0.147
Tech-Talks	0.0001	0.012	0.011	0.992
Mentoring/Onboarding	0.0007	0.006	0.119	0.905
Learning Tech.	0.0118	0.005	2.372	0.018

TABLE III
OLS REGRESSION RESULTS - SATISFACTION

	coef	std err	t	**P> $|t|$**
const	4.0436	0.061	66.061	0.000
Arch. & Design	0.0057	0.003	1.950	0.052
Coding	0.0043	0.002	1.907	0.057
Dev. Environment	-0.0151	0.005	-2.935	0.004
Docs	0.0161	0.006	2.803	0.005
PR/Code Review	-0.0133	0.006	-2.086	0.038
Refactoring	0.0128	0.005	2.584	0.010
Debugging	-0.0010	0.004	-0.267	0.789
Monitoring & Dashboards	-0.0098	0.006	-1.692	0.091
Testing	0.0031	0.005	0.684	0.494
Security & Compliance	-0.0090	0.003	-3.470	0.001
Customer Support	-0.0065	0.003	-1.900	0.058
Communication	-0.0129	0.004	-3.114	0.002
Task Mgmt.	0.0085	0.011	0.743	0.458
Tech-Talks	-0.0032	0.013	-0.243	0.808
Mentoring/Onboarding	0.0077	0.007	1.142	0.254
Learning Tech.	0.0126	0.006	2.279	0.023

In Table II we observe statistically significant positive relations between the time spent on coding, documentation, code refactoring and learning, on developer productivity, while a negative relation for maintaining development environment, and communication. Moreover, for satisfaction, Table III highlights a statistically significant positive relation of satisfaction with documentation and learning, while a negative relation for development environment, PR/Code reviews, security and compliance, and communication. As described by developer D141 and D18 respectively: *"Keeping organized documentation of project related knowledge will certainly improve productivity"*, *"S360 takes up the majority of my time, which means I spend almost no time creating useful features and responding to customer requests."*

C. Mapping Activity Time Across Productivity-Satisfaction Profiles: A Path Toward Smart Automation

In Fig. 5, we categorize developers into three distinct groups: 'High Productivity, High Satisfaction,' 'High Produc-

tivity, Low Satisfaction,' and 'Low Productivity, Low Satisfaction,' based on their self-reported productivity and satisfaction levels from the previous workweek (as detailed in IV-A). For the sake of simplicity, we merged the 'Very Productive' and 'Productive' groups into a single 'High Productivity' category, and similarly for other categories. The 'Low Productivity, High Satisfaction' group was excluded due to its small sample size of two developers. For each remaining group, we visualize the average hours spent on the 16 key activities, highlighting patterns in how time allocation varies across different productivity and satisfaction combinations.

The developers who felt highly productive and satisfied were the ones who spent the most time on activities such as 'Architecting & designing new systems', 'Coding', 'Debugging', 'Refactoring Code', and 'Learning New Technologies', while lesser time on 'Documentation', 'Security & Compliance', 'Addressing Customer Support Tickets', 'Communication', 'ADO Task Creation and Management' and 'Mentoring & Onboarding'. An opposite trend is observed for the developers from the 'Low Productivity, Low Satisfaction' group, as they tend to spend more time on 'Security & Compliance', 'Addressing Customer Support Tickets', and 'Communication'.
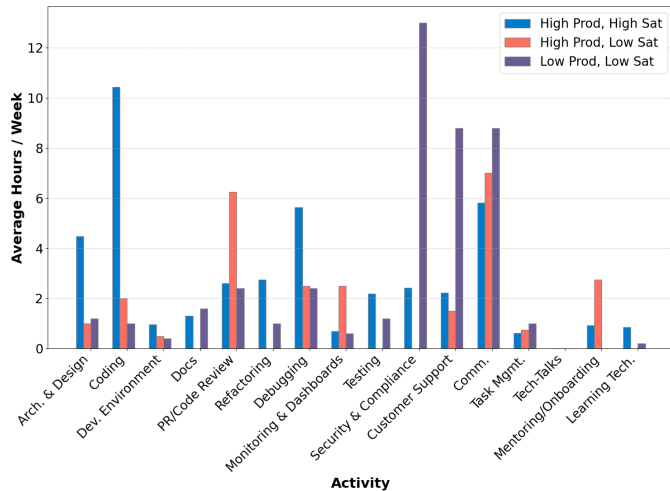

Fig. 5. Average actual Weekly Hours Allocated to Various Activities by Developers in High vs. Low Productivity-Satisfaction Groups

Developers in the 'High Productivity, Low Satisfaction' category dedicate most of their time to activities such as 'Pull Requests/Code Review,' 'Setting up Monitoring and Dashboards,' 'Communication,' and 'Mentoring and Onboarding.' Despite feeling highly productive during the week, these developers report lower satisfaction levels. This suggests that while they recognize the value of these tasks, they prefer spending less time on them, making these activities ideal candidates for automation.

> **♀ Takeaway**: Developers who experience a sense of productivity, yet feel dissatisfied with their workweek present an excellent opportunity for task automation. Our analysis reveals that this group invests significant time in activities such as 'PR/Code Reviews', 'Monitoring & Dashboards', 'Communication', and 'Mentoring/Onboarding'. Consequently, these tasks emerge as ideal candidates for automation.

V. IMPACT OF AI TOOL USAGE ON DEVELOPER PRODUCTIVITY AND SATISFACTION

The rapid advancement of artificial intelligence (AI) and its growing adoption across industries, particularly in automating development tasks, brings forth a crucial question: *Which tasks should be automated using AI to maximize both productivity and satisfaction?* In this section, we observe how the frequency of AI tool usage impacts the developers' productivity and satisfaction and analyze the responses gathered from developers, examining which tasks they believe are most suited for automation.

A. How does the Frequency of AI Tool Usage Influence Productivity & Satisfaction

In the survey, we asked the developers to indicate how frequently they incorporate AI tools during their workweek, ('Daily', 'Weekly', 'Monthly', 'Once every 6 months', or 'Once every year'). We again collapse the top 2 and bottom 2 categories for both productivity and satisfaction into a single category (Productive/Unproductive).

Fig. 6 reveals that developers who use AI tools daily report the highest levels of productivity, with 83.7% reporting to be "productive" However, as the frequency of AI tool usage decreases (e.g., from weekly to annually), the proportion of productive developers steadily declines, while the percentage of those reporting neutrality or even negative productivity outcomes increases.

Fig. 7 shows a similar trend between AI tool usage and satisfaction levels. Developers who utilize AI tools on a daily basis experience the highest levels of satisfaction (74.5%), with most of them reporting to be "satisfied." Again, as AI usage frequency decreases, dissatisfaction becomes more prevalent.

B. Which tasks do developers want to automate?

In the survey, we asked the developers to answer the question *"Which of your current tasks would you most like to see automated? Which processes do you think could be enhanced to minimize repetitive work?"*

We received 242 open-ended responses from the developers. To analyze the responses, we utilized the GPT-4 model to identify an initial set of task categories highlighted in the data. The model identified 17 distinct categories. To ensure comprehensive coverage, we uniformly sampled 60 responses based on their length. Four annotators independently reviewed these responses using a closed-coding approach.
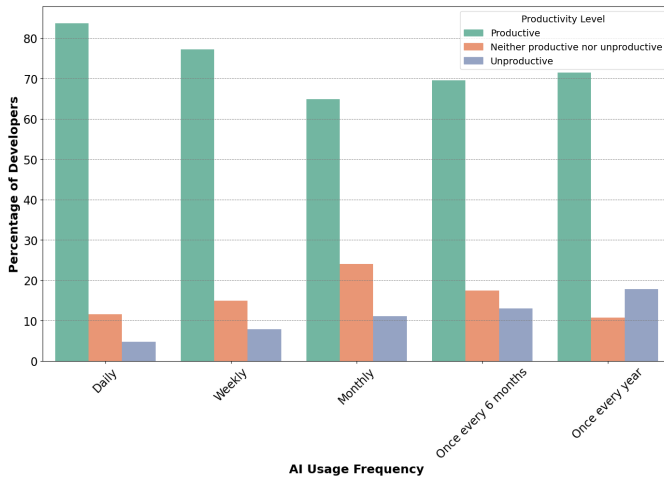
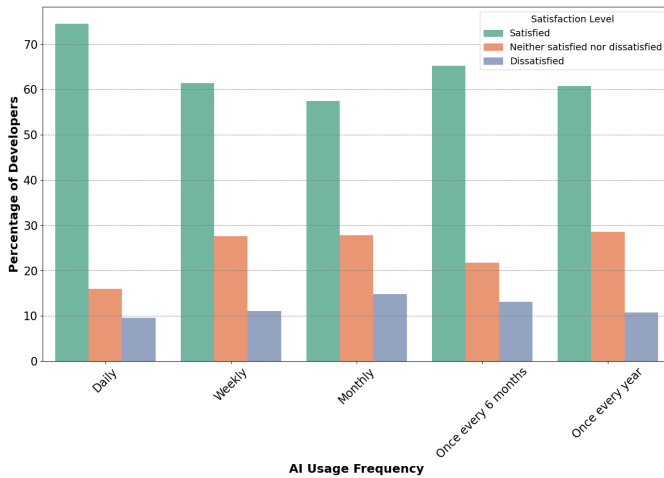Fig. 6. Effect of the frequency of AI tools usage on productivity

Fig. 7. Effect of the frequency of AI tools usage on satisfaction

They assigned labels to each response based on the initial categories suggested by GPT-4, while also identifying any new categories that emerged during the process. The inter-rater agreement, measured using the Jaccard similarity metric, was 81.5%, indicating a substantial level of agreement among annotators. Additionally, through discussions, the annotators agreed to introduce one new category ("Cloud Infrastructure Maintenance"), resulting in a final list of 18 categories of tasks that developers would like to automate using AI. The remaining 182 responses were distributed equally amongst the four annotators and manually labelled using the extended taxonomy to identify the frequency of each category.

The final categories are summarized in Table IV. For each category, we provide the name and a short description. The table also illustrates the results from the manual annotation process and the frequency of each category based on the manual annotation results.

Documentation emerged as the most frequent category in our analysis of tasks that could be automated with AI.

Developers suggested the use of AI to create and update documentation, maintain team knowledge bases, and also to efficiently understand existing code and systems from the available information. For example, D302 highlighted the importance of efficient documentation by saying: *"I would love to see automated documentation of infrastructure and code. A lot of time is lost trying to understand the current resources that go into securing, building, and deploying our products and trying to understand how the different parts of our code base interact and depend on each other. It'd really lessen the cognitive load if this was well-documented in a way that is automatically kept up-to-date."* Similarly, developer D165 expressed the growing need for automating code documentation to keep pace with the rapid development cycles: *"I would love it if there were some way to automatically generate architectural documentation or either a data flow or call stack diagram based on the structure of the actual code in the repository. While tools like CodeFlow exist for VSCode and Visual Studio, there's a learning curve and the lack of reward for better internal documentation means creating and updating documentation is on a volunteer basis. This, and the rapid pace of development means that onboarding any new engineer or existing engineer to a new area is a time-consuming manual process of last-minute documentation updates, or passing down tribal knowledge verbally. Would be nice if there were some way to tie internal documentation to a particular code base, so that every PR also 'refreshes' the internal documentation or diagrams."*

Following this, several developers emphasized the need for automating tasks associated with the setup and maintenance of development environments, which often involve complex processes such as configuring SSH keys, installing software dependencies, and initializing new development instances. Developer D81 points out the current challenges, stating, *"Setting up the Dev. environment could be way simpler. Right now it is somewhat involved, time consuming, and brittle."* Furthermore, D403 emphasizes the significant impact of automation on productivity, noting, *"Setting up a development environment takes up a huge amount of time in the development cycle, hence I feel that enhancing the same will greatly boost developer productivity."*

Developers also suggested the need to automate tasks surrounding the authoring, execution and monitoring of tests. For instance, D81 envisions a future where AI could generate comprehensive acceptance tests based on system usage logs: *"Dreaming wild, what I would like AI to do is to create all the (missing) acceptance tests of the system (maybe by looking at all the usage logs?) because currently any attempt to improve or refactor the code monolith is a risky venture. The only thing that would make it less risky is to have tests that validate that the system functionality is still there."*

Beyond the aforementioned technologies, developers also convey the importance of enhancing automation efforts to improve various aspects of project management. This includes the efficient creation and management of tasks in systems like Azure DevOps (ADO), which encompasses han-

dling sub-tasks and repeatable tasks, as well as tracking project status effectively. Developer D38 shares his insights on utilizing AI to generate work items in ADO using a structured prompt, such as: *"Create work item with title [Task][GIVE_TASK_DESCRIPTION] related to work [Feature_Item], work description [1 line description] - assigned to [UserName](Given the user name it path to correct team)."*

> ♀ **Takeaway**: Incorporating AI tools is crucial for enhancing developers' productivity, while also boosting their overall professional satisfaction. However, to maximize the benefits of automation, it is essential to align AI development efforts with the specific tasks and activities that developers most desire to automate. By focusing on these areas, research and automation initiatives can more effectively streamline workflows and address developers' real-world needs.

VI. DISCUSSION

Our study provides valuable insight into the complex dynamics of a software developers' workweek. We aim to understand how the deviations from ideal and actual workweeks impact productivity and satisfaction levels. Additionally, we explored the role of up and coming AI tools and how we can modify our approach towards building new tools to build more tools that developers want. In this sections we will discuss a short summary of our findings and discuss the implications of our findings and impact on developers, managers and organizations in the software development industry as a whole.

A. Time allotment vs productivity/satisfaction

Our results show most developers generally have very different workweeks from their ideal ones. This deviation has a measurable impact on both productivity and satisfaction levels. We quantitatively showed that as the gap between the actual and ideal workweek increases there is a substantial negative impact on both productivity and satisfaction levels. This finding highlights the importance of having a better balance between what a developer wants to do vs what they typically do in practice.

Unsurprisingly, activities that positively influence both productivity and satisfaction are those typically considered core development tasks, such as coding, documentation, and code refactoring. Conversely, time spent on maintaining and creating development environments, dealing with security and compliance issues, and excessive communication negatively impacts both metrics. These activity-level insights offer broader suggestions for how organizations can optimize their workflows to maximize time on core development activities while minimizing time on less preferred tasks.

These findings present several opportunities for software developers, teams, organizations and researchers working in this field. Organizations can potentially redesign or optimize existing workflows to better align with the needs of developers. There may also be opportunities to adjust team structures to distribute tasks more effectively, allowing developers to focus more on their preferred activities. Additionally, introducing new tools or improving existing ones could help streamline less desirable tasks, freeing up more time for core development activities.

A potential interesting direction for future research would be to explore the longer-term impact of better aligning developers' workweeks with their ideal ones. This could involve examining changes in code quality, project success metrics, and overall user experience over time. However, it's important to note that finding the right balance will likely require ongoing assessment and adjustment, as the ideal workweek may vary not only between individuals but also evolve over time for each developer.

B. Future impact of AI tools

We observed a strong correlation between how frequently developers used AI tools and their reported satisfaction and productivity levels. Developers who use these tools daily tended to report the highest level of both satisfaction and productivity. This underscores the importance of these tools in the modern software development landscape.

As discussed earlier, developers most want to see activities automated which we observed to correlate with lower satisfaction and productivity levels. These findings present a new exciting direction for future AI tools and automation development efforts. By focusing the development of AI tools on areas which can have dual benefit of boosting developer productivity and satisfaction whilst at the same time allowing them to focus on tasks they enjoy and find more fulfilling.

Currently a large number of AI tools focus specifically on helping developers write code [10]–[12]. However, our findings suggest a strong desire for automation in non-coding related tasks such as documentation, task management, security and compliance, and communication. This reveals a gap in the current AI tool landscape and presents a significant opportunity for innovation. Developing new AI tools that address these areas could offer substantial benefits not only to individual developers but also to organizations as a whole.

The potential impact of such targeted AI tool development is far-reaching. By automating tasks that developers find less productive or satisfying, we could see a shift in how developers allocate their time, potentially bringing their actual workweeks more in line with their ideal ones. This could lead to higher job satisfaction, increased productivity, and potentially even improvements in code quality and project outcomes as developers are able to focus more on core development tasks they find most engaging. However, as the adoption of these tools grow there might be a change in the what a developer considers to be an ideal workweek. Future research should not only focus on developing these tools but also on understanding their broader impacts on the software development as a whole.

VII. RELATED WORK

To our knowledge, this is the first work that compares a developer's ideal and typical workweek and assesses the

TABLE IV
WHICH TASKS DO DEVELOPERS WANT TO AUTOMATE?

Category	Category description	Count
Documentation	Creating & updating documentation, generating API docs from code comments, & maintaining team knowledge bases. Efficiently understanding existing code, APIs, & systems from documentation & reports.	████████ 82
Environment set-up/maintenance	Setting up SSH keys, installing software dependencies, syncing Git repositories, updating local branches, initializing new development instances/containers.	██████ 66
Write/maintain tests	Unit test creation for routine/similar coding tasks. Authoring, execution, & monitoring of tests.	█████ 60
Task Tracking & Backlog Management	Task creation & management in systems such as Azure DevOps, sub-tasks, repeatable tasks, or project status tracking.	████ 47
Security & Compliance	Reviews, tracking, addressing issues (S360 tasks), handling security wave items, & managing renewals or permissions.	███ 40
Incident/Customer Issue Management	Live site reporting, root cause analysis, incident correlation, triage, customer communication, and ICM handling.	███ 38
Communication	Summarizing long email threads, creating ADO tasks, answering frequent technical questions from different developers. Efficient ways to handle meeting minutes, schedules, notes, action items, etc.	███ 33
Deployment & Release Management	Deployments along with build-and-release pipelines often involve many manual steps and repetitive tasks. This includes validation checks, release notes creation, and understanding deployment health metrics.	██ 26
PR/Code Review/Change Management	Code review feedback & formatting, detection of anti-patterns, merge conflict resolutions, branch management, rerunning flaky jobs, merging development branches, handling pull request iterations, reviews, & descriptions.	██ 25
Infrastructure Monitoring & Alerts	The setup, maintenance, and interpretation of monitoring dashboards, alert rules, and other telemetry data.	██ 23
Debugging	Automation around debugging processes, particularly surrounding build failures, errors, and runtime exceptions.	█ 15
Code Refactoring	Automating parts of the refactoring process, enforcing code quality standards, and detecting anti-patterns.	█ 13
Build Automation	Faster and reliable build systems to swiftly integrate and manage new code changes into existing codebases.	█ 11
Maintenance of Internal Tools	Maintenance tasks related to the upkeep of existing tools and systems.	█ 10
Onboarding	Adding new team members, giving them walk-throughs of the best practices, resources, setting up their development environment, and access permissions are areas suggested for improvement.	▌ 7
Cloud Infrastructure Maintenance	Resource provisioning, scaling, performance monitoring, security patching, backups, cost optimization, network management, and incident response for efficient cloud operations.	▌ 5
Workflow Integration	Better integration between version control systems, issue trackers, & other development tools.	▏ 3
Reworking Legacy Code	Assistance in dealing with codebase legacies, updating old systems, or adapting to new systems.	▏ 3

impact of their deviations on productivity and satisfaction. We also explore the role of AI tools in the software development process and capture potential future directions for AI tool development. We give a brief summary of existing work in three areas: developer activities and time spent, developer productivity and satisfaction, and AI tools and automation.

A. Developer activities and time allotment

Understanding how developers spend their time across various activities has been a very well explored topic. Prior work analyzed how developers spend their time within the IDE by specifically tracking IDE usage, monitoring test and other refactoring feature usage [13], [14]. This work did not focus on other activities outside the IDE like collaboration, communication and planning.

More recent works have captured a more holistic picture of what a developer does and aimed to capture the time spent through various methods of capturing data like surveys [3], [15], [16], observations and interviews [17], and through tracking usage of the computer [13], [14]. While many studies focus on understanding the daily activities, fewer have looked at patterns across an entire workweek. Our study builds on this gap, extending our window to the past work week allowing us to get a more comprehensive understanding of how a developer spends their time and also reducing the impact of day to day variations. In addition, we compare the actual workweek with an ideal workweek.

B. Developer productivity and satisfaction

There exists a substantial amount of prior work on trying to understand the factors that influence developer productivity

and satisfaction [2], [18]–[20]. There is significant work in understanding how different factors like interruptions [21], [22], emails and communication [23] impact both productivity and satisfaction.

Prior work [24]–[27] showed also a link between developer happiness and increased productivity. Therefore, it is important to analyze how different activities affect a developers satisfaction and in-turn productivity. Masood et al. [28] analyzed how different tasks and activities affect developer happiness and productivity.

C. AI tools and automation in software development

The rapid development and adoption of AI tools and automation in software development has opened new avenues for understanding their impacts on the field. There have been works [29]–[32] that analyze how different AI tools impact developer productivity. There is also a body of work [33], [34] on understanding specific tasks that developers want to use AI tools.

We build upon these existing studies by analyzing the impact of AI tool usage on productivity and satisfaction. We further investigate whether developers want particular tasks to be automated. This can drive future development of AI tools and ensure that they are aligned with the actual requirements of the developers.

VIII. CONCLUSION

In this paper, we provide valuable insights on the complex relationship between software developers' workweek and how it's deviation from their ideal workweek affects their productivity and satisfaction levels. Here are our key findings:

1) There is a large deviation between the ideal and actual workweeks, with developers clearly wanting to spend more time on core developmental activities and less on communication/maintenance ones.
2) As the deviation between the ideal and actual workweek increases the productivity and satisfaction levels tend to fall.
3) Usage of AI tools by a software developer is positively correlated with higher levels of productivity and satisfaction. This effect is especially more pronounced when the frequency of usage increases.
4) Developers have strong preferences towards automation on activities like documentation, environment setup, testing, monitoring etc. These insights will inform the research and development of future AI tools.

The insights we present in this paper have important implications for developers, managers and companies as a whole. Teams and companies can potentially boost satisfaction and productivity levels by working on more closely aligning a typical workweek with a developer's ideal workweek. Furthermore, the development of future AI tools can take into account the needs of a developer as discussed in our paper and fill gaps in this area.

While our work is restricted to Microsoft, it would be interesting to see how our research fits into different organizational contexts and over longer intervals of time. Additionally, another interesting aspect to look at would be how AI-tools potentially change a typical and ideal workweek of a developer as the adoption and scope of AI increases.

In conclusion, we try to understand how time spent on different activities over a workweek and the deviation from a developers "ideal" workweek impact productivity and satisfaction levels. Our findings reveal significant potential to reshape, innovate, and streamline workflows, enhancing developer productivity and satisfaction. This will lead to more fulfilling work experiences and higher quality outcomes, benefiting both developers and the organizations.

REFERENCES

[1] D. Graziotin, F. Fagerholm, X. Wang, and P. Abrahamsson, "What happens when software developers are (un)happy," *Journal of Systems and Software*, vol. 140, pp. 32–47, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121218300323

[2] M.-A. Storey, T. Zimmermann, C. Bird, J. Czerwonka, B. Murphy, and E. Kalliamvakou, "Towards a theory of software developer job satisfaction and perceived productivity," *IEEE Transactions on Software Engineering*, September 2019. [Online]. Available: https://www.microsoft.com/en-us/research/publication/towards-a-theory-of-software-developer-job-satisfaction-and-perceived-productivity/

[3] A. N. Meyer, E. T. Barr, C. Bird, and T. Zimmermann, "Today was a good day: The daily life of software developers," *IEEE Transactions on Software Engineering*, vol. 47, no. 5, pp. 863–880, 2021.

[4] T. DeMarco and T. Lister, "Programmer performance and the effects of the workplace," in *Proceedings of the 8th International Conference on Software Engineering*, ser. ICSE '85. Washington, DC, USA: IEEE Computer Society Press, 1985, p. 268–272.

[5] J. Singer, T. Lethbridge, N. Vinson, and N. Anquetil, "An examination of software engineering work practices," in *Proceedings of the 1997 Conference of the Centre for Advanced Studies on Collaborative Research*, ser. CASCON '97. IBM Press, 1997, p. 21.

[6] D. Perry, N. Staudenmayer, and L. Votta, "People, organizations, and process improvement," *IEEE Software*, vol. 11, no. 4, pp. 36–45, 1994.

[7] M. Fowler, *Refactoring: improving the design of existing code*. Addison-Wesley Professional, 2018.

[8] C. Spearman, "The proof and measurement of association between two things." *International journal of epidemiology*, vol. 39 5, pp. 1137–50, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:14780428

[9] C. J. Willmott and K. Matsuura, "Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance," *Climate Research*, vol. 30, pp. 79–82, 2005. [Online]. Available: https://api.semanticscholar.org/CorpusID:120556606

[10] "Github copilot," https://github.com/features/copilot, accessed: 2024-10-10.

[11] "Cursor," https://www.cursor.com/, accessed: 2024-10-10.

[12] "Cody by sourcegraph," https://sourcegraph.com/cody, accessed: 2024-10-10.

[13] R. Minelli, A. Mocci, and M. Lanza, "I know what you did last summer-an investigation of how developers spend their time," in *2015 IEEE 23rd international conference on program comprehension*. IEEE, 2015, pp. 25–35.

[14] S. Amann, S. Proksch, S. Nadi, and M. Mezini, "A study of visual studio usage in practice," in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, vol. 1. IEEE, 2016, pp. 124–134.

[15] M. Beller, V. Orgovan, S. Buja, and T. Zimmermann, "Mind the gap: On the relationship between automatically measured and self-reported productivity," *IEEE Software*, vol. 38, no. 5, pp. 24–31, 2021.

[16] D. Russo, P. H. Hanel, S. Altnickel, and N. van Berkel, "The daily life of software engineers during the covid-19 pandemic," in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 2021, pp. 364–373.

[17] M. K. Gonçalves, C. R. de Souza, and V. M. González, "Collaboration, information seeking and communication: An observational study of software developers' work practices." *J. Univers. Comput. Sci.*, vol. 17, no. 14, pp. 1913–1930, 2011.

[18] 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, p. 20–48, Mar. 2021. [Online]. Available: https://doi.org/10.1145/3454122.3454124

[19] 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.

[20] C. Sadowski and T. Zimmermann, Eds., *Rethinking productivity in software engineering.* Apress, 2019.

[21] G. Mark, D. Gudith, and U. Klocke, "The cost of interrupted work: more speed and stress," in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, 2008, pp. 107–110.

[22] B. P. Bailey, J. A. Konstan, and J. V. Carlis, "The effects of interruptions on task performance, annoyance, and anxiety in the user interface." in *Interact*, vol. 1, 2001, pp. 593–601.

[23] B. Houck, H. Yelin, J. Butler, N. Forsgren, and A. McMartin, "The best of both worlds: Unlocking the potential of hybrid work for software engineers," May 2023. [Online]. Available: https://www.microsoft.com/en-us/research/publication/the-best-of-both-worlds-unlocking-the-potential-of-hybrid-work-for-software-engineers/

[24] 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.

[25] ——, "Software developers, moods, emotions, and performance," *IEEE Software*, vol. 31, no. 4, pp. 24–27, 2014.

[26] Y.-H. Kim, E. K. Choe, B. Lee, and J. Seo, "Understanding personal productivity: How knowledge workers define, evaluate, and reflect on their productivity," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019, pp. 1–12.

[27] D. Girardi, F. Lanubile, N. Novielli, and A. Serebrenik, "Emotions and perceived productivity of software developers at the workplace," *IEEE Transactions on Software Engineering*, vol. 48, no. 9, pp. 3326–3341, 2022.

[28] Z. Masood, R. Hoda, K. Blincoe, and D. Damian, "Like, dislike, or just do it? how developers approach software development tasks," *Information and Software Technology*, vol. 150, p. 106963, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584922001045

[29] A. Ziegler, E. Kalliamvakou, X. A. Li, A. Rice, D. Rifkin, S. Simister, G. Sittampalam, and E. Aftandilian, "Measuring github copilot's impact on productivity," *Commun. ACM*, vol. 67, no. 3, p. 54–63, Feb. 2024. [Online]. Available: https://doi.org/10.1145/3633453

[30] M. Borg, D. Hewett, D. Graham, N. Couderc, E. Söderberg, L. Church, and D. Farley, "Does co-development with ai assistants lead to more maintainable code? a registered report," *arXiv preprint arXiv:2408.10758*, 2024.

[31] R. Ulfsnes, N. B. Moe, V. Stray, and M. Skarpen, "Transforming software development with generative ai: Empirical insights on collaboration and workflow," in *Generative AI for Effective Software Development*. Springer, 2024, pp. 219–234.

[32] S. Peng, E. Kalliamvakou, P. Cihon, and M. Demirer, "The impact of ai on developer productivity: Evidence from github copilot," *arXiv preprint arXiv:2302.06590*, 2023.

[33] M. Khemka and B. Houck, "Towards effective ai support for developers: A survey of desires and concerns," *ACM Queue*, vol. 22, no. 3, July 2024. [Online]. Available: https://www.microsoft.com/en-us/research/publication/towards-effective-ai-support-for-developers-a-survey-of-desires-and-concerns/

[34] C. Bird, D. Ford, T. Zimmermann, N. Forsgren, E. Kalliamvakou, T. Lowdermilk, and I. Gazit, "Taking flight with copilot: Early insights and opportunities of ai-powered pair-programming tools," *Queue*, vol. 20, no. 6, pp. 35–57, 2022.