

“Maybe We Need Some More Examples:” Individual and Team Drivers of Developer GenAI Tool Use

Courtney Miller,¹ Rudrajit Choudhuri,² Mara Ulloa,³ Sankeerti Haniyur,⁴ Robert DeLine,⁴ Margaret-Anne Storey,⁵

Emerson Murphy-Hill,⁴ Christian Bird,⁴ Jenna L. Butler⁴

¹Carnegie Mellon University, PA, USA. Email: courtneymiller@cmu.edu

²Oregon State University, OR, USA. Email: choudhru@oregonstate.edu

³Northwestern University, IL, USA. Email: mara.ulloa@u.northwestern.edu

⁴Microsoft, WA, USA. Email: sahaniyur,rdeline,emerson.rex,cbird,jennbu@microsoft.com

⁵University of Victoria, BC, Canada. Email: mstorey@uvic.ca

Abstract

Despite the widespread availability of generative AI tools in software engineering, developer adoption remains uneven. This unevenness is problematic because it hampers productivity efforts, frustrates management’s expectations, and creates uncertainty around the future roles of developers. Through paired interviews with 54 developers across 27 teams – one frequent and one infrequent user per team – we demonstrate that differences in usage result primarily from how developers perceive the tool (as a collaborator vs. feature), their engagement approach (experimental vs. conservative), and how they respond when encountering challenges (with adaptive persistence vs. quick abandonment). Our findings imply that widespread organizational expectations for rapid productivity gains without sufficient investment in learning support creates a “Productivity Pressure Paradox,” undermining the very productivity benefits that motivate adoption.

1 Introduction

The field of Artificial Intelligence (AI) has experienced several historic hype cycles where failed returns on investment led to AI Winters [42]. During the expert systems boom of the 1980s, global investments exceeded \$2.5 billion [30, 68, 78], but when these systems failed to deliver the expected returns, funding was dramatically cut, precipitating the second AI Winter [78, 87]. Today, AI experts and some economists argue that the transformational promise of Generative AI (GenAI) will create significant lasting impact across industry sectors [19, 48]. GenAI’s ability to augment processes that were previously difficult to digitize has led to widespread optimism and the rapid deployment of GenAI tooling by many organizations, often with minimal strategic planning and high expectations regarding productivity gains and subsequent cost savings [60, 63, 64, 88].

Like many others, the software engineering (SE) domain has been profoundly impacted by the introduction of GenAI tooling [22, 79, 83]. GenAI tools in SE show great potential, with some projections estimating productivity increases between 20% to 55% [19, 49, 73]. Yet despite these projections, studies report strikingly mixed outcomes in real-world usage [8, 40, 61]. While some developers achieve the promised productivity gains [82], others experience decreased efficiency when attempting to integrate these tools [13].

In the face of these challenges, a rapidly-evolving body of research focused on the adoption, usage, and integration of GenAI development tools has emerged in the past several years cataloging both enablers and barriers [5, 9, 54, 61]. Much of this work relies on individual-level models such as the Technology Acceptance Model

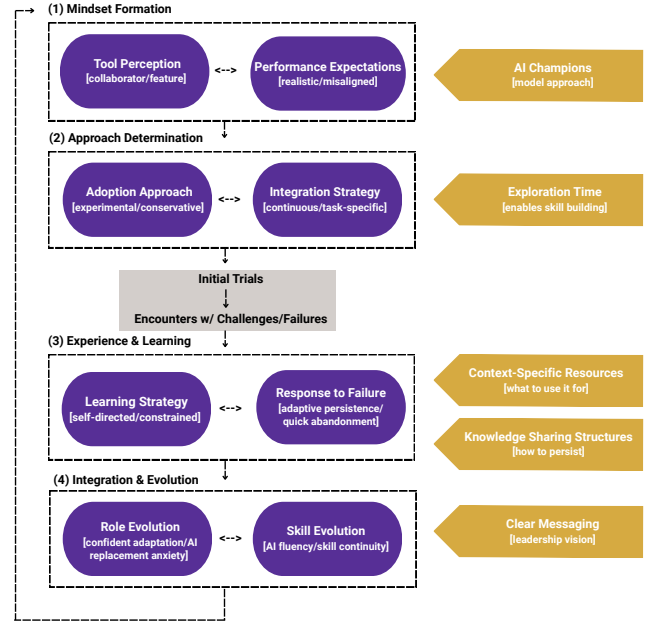


Figure 1: Conceptual theoretical framework. Purple ovals == individual factors, yellow arrows == external factors.

(TAM) [23] and Unified Theory of Acceptance and Use of Technology (UTAUT) [90]. While such lenses surface beliefs like perceived usefulness, they rarely hold constant the external realities - such as codebase complexity and management - that strongly shape tool adoption [13, 34, 57]. With the goal of understanding what differentiates developers who are frequent versus infrequent GenAI development tool users within the same team contexts, we begin by exploring the research question (RQ):

RQ1 What individual factors distinguish frequent and infrequent users of Generative AI development tools?

Early discussions revealed that team and organizational factors appeared to shape individual factors for some developers (cf. Figure 1), leading us to explore these emergent results with a second RQ:

RQ2 How do individuals report team and organizational factors influence their Generative AI development tool usage?

We utilize a paired interview design, conducting sequential semi-structured interviews with 54 developers representing 27 pairs from the same team matched on key factors—primary programming language, role, and seniority- but who exhibit contrasting

usage patterns (one frequent user, one infrequent), identified using telemetry data from a large multinational software company. This matching strategy ensures that both developers share the same team-level context—codebase, manager, policies—so we can directly compare the context-specific blockers an infrequent user faces with the work-arounds their frequent teammate has discovered. In doing so we trace how organizational conditions are interpreted and acted on rather than assuming purely individual causes, providing an empirically-grounded sociotechnical account of GenAI tool usage.

We identified key differences in approaches frequent and infrequent users (cf. Figure 1): frequent users often conceptualized the tooling as a collaborative partner, adopted continuous and experimental integration approaches, and leveraged adaptive persistence when facing challenges, while infrequent users more often viewed it as a utility feature, maintained conservative integration patterns, and more often quickly abandoned tools when facing challenges. Our analysis also identified key commonalities between the two groups: for many developers, organizational factors can actively shape these individual factors through an amplification effect, e.g., team-specific demonstrations of applying GenAI tools on common development tasks can transform developers' perception of tool usefulness. Most critically, we identified a phenomenon which we refer to as the *Productivity Pressure Paradox*: increased productivity expectations from management without corresponding support often create a paradoxical effect, where developers lack the time necessary to develop the skills that would save time. This finding challenges the prevailing GenAI development tool deployment strategy across the software industry, which frames the challenge of figuring out how to use these tools to yield the expected productivity gains as the responsibility of individual developers [60, 63, 64, 88]. Like historical technological transformations in manufacturing and software engineering, we argue systematic productivity gains require systematic organizational change, not individual workflow optimization [12, 25]. The “deploy and figure it out” strategy where companies invest billions into GenAI tools without investing in the corresponding support structures necessary to enable the systematic evolution of development workflows that would potentially yield a return on investment, are largely driven by competitive pressures and repeat past failures to recognize that transformative technologies demand transformative organizational approaches [25, 43, 86].

In summary, this paper makes the following contributions: (1) a conceptual theoretical framework detailing the differences in individual factors between frequent and infrequent GenAI tool users; (2) a taxonomy of the ways different team and organizational factors can shape individual factors that influence usage; and (3) the concept of the *Productivity Pressure Paradox* and evidence-based strategies organizations can adopt to overcome it.

2 Background and Related Work

Given the rapid permeation of these tools in development work, the factors that affect developers' adoption and integration of GenAI tools have become a focal topic in the SE research community.

2.1 Individual Factors Affecting Adoption

Research has explored why developers use GenAI tools or not, often by adapting or extending classic technology acceptance theories

(e.g., UTAUT [90, 91]). Early evidence suggests that traditional adoption factors do not fully apply in developer-GenAI tool interaction contexts. For example, Russo [79] found that GenAI tool compatibility with a developers' workflow drives early adoption, outweighing traditional UTAUT factors such as effort expectancy.

Trust has emerged as a critical factor in the design and adoption of these tools [46, 92]. Choudhuri et al. [17] found that system quality, functional value, and goal alignment increase developers' trust in GenAI tools, which—combined with risk tolerance, self-efficacy, and intrinsic motivations significantly influences adoption.

Additionally, empirical studies highlight the significant challenges developers face with current GenAI tooling support including inconsistent performance, poor code/output quality, suggestions that fail to meet contextual requirements, alongside interaction friction and sensemaking difficulties [16, 18, 54, 61]. These issues force developers into a cycle of constant re-prompting and result verification, placing a heavy burden on the user [13, 18]. Developers must often invest high cognitive effort into crafting prompts and frequently edit or debug AI-generated code; actions which disrupt their “flow” state and can negate potential productivity gains [8, 82, 84]. They also raise ethical and legal concerns, including uncertainty about code provenance or licensing for AI-generated code, as well as concerns about biases or security vulnerabilities introduced by AI contributions [5, 9].

Task differences can further complicate adoption [45]. Lambiasi et al. [57] found that adoption drivers are not one-size-fits-all; they vary by task and can even negatively impact adoption in certain cases. For instance, peer opinions correlate with higher GenAI tool usage for decision-making tasks. Yet, strong social influence can backfire: developers might develop high expectations that the tool then fails to meet [57]. Moreover, some developers fear that over-reliance on AI could erode their own skills or even threaten job security in the long run [54]. Indeed, expectations play a crucial role in guiding appropriate tool usage. How developers conceptualize the AI's capabilities can influence how they integrate it in their work [44, 94]. Prior work indicates that fostering an accurate understanding of an AI's capabilities (and limitations) improves user trust and helps foster appropriate usage [3, 94].

Overall, the literature paints a narrative in which developers adopt GenAI tools when it demonstrates value, integrates smoothly into workflows, and earns user trust. However, existing studies largely treat developers as a relatively uniform group or focus on formative adoption intentions. This framing limits the understanding of the presence of substantial environmental confounds such as team culture, codebase complexity, managerial attitudes, tool availability, and organizational norms, among others, which often vary across developers, making it difficult to disentangle individual attitudes from environmental influences. As a result, it remains unclear whether observed adoption patterns stem from personal disposition or from differences in the surrounding environment.

Our study addresses this gap directly through its paired interview design. By interviewing matched pairs of developers; each working on the same team and project, and with access to the same GenAI tools; we can directly compare the blockers an infrequent user faces with the work-arounds their frequent teammate has discovered. This design allows us to isolate the individual-level

mindsets, strategies, and responses that shape divergent usage patterns, while revealing the influences of team and organizational scaffolding. We move beyond the individualistic lens of prior work and offer a sociotechnical account of GenAI usage that is both empirically grounded and practically actionable.

2.2 Environmental Factors Affecting Adoption

Beyond individual attitudes, the environment in which a developer works (their team and organization) plays a critical role in GenAI adoption [53, 55, 75]. Decades of socio-constructivist research have shown that technology uptake is not merely a function of individual utility but a deeply sociotechnical process shaped by social context, organizational norms, and collective sensemaking [47, 70, 77]. Fulk's theory of the social construction of technology, for instance, emphasizes how coworkers' beliefs and organizational cues shape perceptions of technological value [34].

In the SE domain, emerging studies echo these insights. Peer influence and leadership support have been shown to shape developers' trust in GenAI tools [15, 53]. For example, while top-down mandates alone do not guarantee adoption, strong managerial support (e.g., providing resources and training) can significantly boost trust [53]. Similarly, Cheng et al. [15] show how developers in online communities collectively interpret GenAI tooling through shared experiences, with success stories boosting confidence and widely reported failures fostering caution.

However, some recent empirical work has also found mixed results when examining these dynamics [56, 79]. For instance, Russo [79] found limited support for the influence of organizational stance (external encouragement), a result that contrasts with broader theoretical expectations in organizational communication [34, 35, 80]. One explanation is that these studies are able to capture static slices of environmental influence through individual attitudes (e.g., survey item on management support), failing to reflect how multifaceted team dynamics and organizational support influence adoption. This limitation can lead to counterintuitive findings (e.g., a lack of significant evidence for role of organizational stance) that conflict with broader theoretical expectations (e.g., Fulk's work showing co-worker beliefs strongly guide technology perceptions [34]). Also, many studies focus on aggregate trends or initial intentions [56, 57].

We address this gap by taking a comprehensive, in-situ approach. By interviewing matched pairs of developers who share the same team and context but have contrasting usage patterns, we can directly ask frequent users how they overcame the context-specific barriers faced by their peers. In doing so, we extend beyond affirming that "environment matters" to unpacking which aspects of the environment—organizational/peer support, learning resources, etc.—make a difference to developers. This nuanced understanding helps explain the divergence in usage trajectories even among developers in similar contexts. Identifying these differences can guide teams and organizations in creating environments that support sustained, effective GenAI tool use in software development.

3 Research Design

To understand what distinguishes frequent and infrequent GenAI development tools users within similar organizational contexts, we performed semi-structured interviews with 54 developers representing 27 pairs from different teams across a large multinational

software company. Below, we detail ethical considerations, study design, analysis strategy, and limitations.

3.1 Ethical Considerations

We recognize that studying technology adoption in contexts with strong organizational pressure requires exceptional and up front care to protect participants' wellbeing and privacy. Given the company's culture of heavily promoting GenAI tool adoption, we implemented multiple safeguards to protect all participants.

Telemetry Data and Privacy. Tool usage telemetry is collected on an opt-out basis, with developers able to disable collection at any time. In sensitive contexts, opting out is explicitly encouraged.

Protecting Within-Team Anonymity. To ensure participants from small teams were not identifiable to colleagues (especially infrequent users), we did not disclose that we interviewed other team members. We further framed the study as exploring "diverse developer experiences with GenAI tools."

Ensuring Non-Judgmental Interview Environments. We aimed to create psychological safety for participants whose usage or beliefs might diverge from organizational expectations, by vetting all questions for neutrality in terms of tool usage, reviewed by our team and external researchers. The first author, who is not affiliated with any tooling teams, conducted all interviews and emphasized confidentiality and ensured a thorough anonymization process.

3.2 Identifying and Recruiting Participants

Our study employed a paired interview design, recruiting developers from the same teams who exhibited contrasting usage patterns. This approach minimized the influence of team-specific factors—codebase complexity, development practices, and organizational culture—that have confounded previous studies [5, 13, 50, 57, 61, 79]. By comparing developers in closely similar environments, we isolated individual-level factors affecting usage patterns.

Identifying Pairs of Developers. To identify developer pairs with contrasting usage patterns but similar environments across the company, we extracted data about developers (that had not opted out) from internal telemetry databases containing demographic attributes and tool usage. Across the company, Github Copilot is the officially sanctioned and freely available GenAI development tool [1]. The telemetry usage data for each developer indicates the number of days each developer used GitHub copilot during an eight-week observation window (from April 1st to May 31st 2025). We also collected their job title, career stage, employee level, geographic area code (i.e., country), team within organizational hierarchy (i.e., their management chain from the CEO down to their direct manager), and primary programming language.

Using this data set, we implemented a multi round pairing algorithm using the Hungarian method for optimal assignment and `scipy.optimize` [66, 81]. First, the algorithm grouped developers by career stage, job title, employee level, country, and direct manager—creating groups with similar professional contexts. Within each group, the algorithm split developers into two subgroups at the median usage level before applying the assignment algorithm to maximize usage difference between pairs.

Once the pool of pairs had been identified, we filtered it to ensure each pair shared relevant characteristics while exhibiting significant

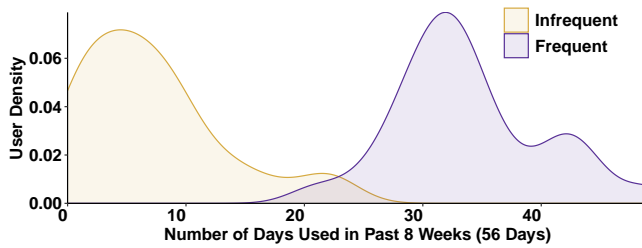


Figure 2: GenAI Tool Usage Distributions

differences in usage. Specifically, We selected pairs that had same primary programming language and a usage gap in the top third across all pairs (measured by the difference in days of GenAI tool usage between the pair’s frequent and infrequent user).

Note our definitions of frequent and infrequent users are relative to their teams’ usage (not to all developers across the company). We use the terms “infrequent-AI users” and “frequent-AI users” to refer to the two groups of developers that represent the less frequent and more frequent users from each pair respectively. The median usage frequency for the infrequent and frequent groups was 5.5 days and 33 days during our observation window (cf. Figure 2).

Participant Recruitment. We recruited candidate pairs from engineering teams globally through an internal company chat, due to regulatory reasons we excluded pairs from Germany and Norway. We reached out to each developer individually, only interviewing developers from pairs where both agreed to participate.

3.3 Interview Protocol

Our interview protocol strategically leveraged the paired design. For each pair, we first interviewed the infrequent user to understand their perspectives, any barriers, and specific challenges they face. We then interviewed the corresponding frequent user from the same team, exploring how they may have navigated similar challenges and their broader experiences. This sequencing allowed us to probe the frequent users about context-specific barriers their teammates faced, revealing divergent responses and strategies they may have followed to overcome shared obstacles. The interview protocol evolved iteratively while maintaining consistency in core questions to ensure comparability across participants.¹

Our interview protocol initially drew from the UTAUT framework [90], a well-established lens in prior technology adoption research [57, 79, 90]. UTAUT’s four core constructs—performance expectancy, effort expectancy, social influence, and facilitating conditions—provided a systematic foundation for understanding usage behaviors in organizational contexts.

We began with 30-minute interviews exploring these constructs through questions on perceived tool benefits (e.g., “What are your current expectations about how GitHub Copilot can help with your development work?”), ease-of-use and integration challenges, alongside team influences, and organizational support. However, early interviews revealed patterns beyond traditional adoption factors—differences in how the two groups conceptualized and approached

AI tool usage. To explore these emergent themes, we extended interviews to 60 minutes and supplemented UTAUT questions with new probes about participants' tool perceptions, integration strategies, learning approaches, and evolving skill perceptions.

3.4 Data Collection and Analysis

Table 1: Participant Demographics

Location		Role		Copilot Use	
United States	36 (66.7%)	Senior SWE	32 (59.3%)	<6 months	11 (20.4%)
China	6 (11.1%)	SWE II	18 (33.3%)	6-12 months	9 (16.7%)
Kenya	4 (7.4%)	SWE	4 (7.4%)	1-2 years	21 (38.9%)
Canada	2 (3.7%)			2-3 years	13 (24.1%)
Estonia	2 (3.7%)				
Ireland	2 (3.7%)				
Romania	2 (3.7%)				

The interviews took place over video conferencing. From 670 invitations, 315 developers agreed to participate. Of those that agreed, 152 were from 76 pairs where both members agreed. We scheduled interviews with pairs on a first-come, first-served basis once both developers had agreed to participate, but were limited by protocol scheduling logistics, busy developer schedules, and time zones. In total, we conducted 56 interviews (labeled PID1-56), with 54 participants (27 matched pairs) included in the final analysis. Two participants were excluded when their pair partners dropped out. We summarize the participant demographics in Table 1.

We qualitatively analyzed interview transcript data using iterative thematic analysis [10]. Our process was guided by Lincoln and Guba’s trustworthiness criteria [39], as described by Nowell et al [69]. During this process, we followed a commonly recommended strategy [59]: throughout the process we switched between the different stages of analysis – jumping between exploring the rich transcripts, engaging with and analytically memoing the data [65], coding, searching for themes, and refining the codes and coding framework. While UTAUT guided our interview questions and early memoing to organize emerging barriers, our actual coding framework was developed through iterative inductive analysis.²

To leverage contextual richness, we coded both interviews from each pair in the same session, following the interview sequence (infrequent user first, then the frequent user). This paired approach allowed us to track how developers from closely similar team contexts responded differently to similar challenges. The analysis began with the first author performing open-ended inductive coding of each interview as data collection progressed. After eight interviews, all authors came together and performed an in-depth analysis of the codes and coding framework. Iterative adjustments to the coding framework and interview guide were made as necessary. Once the framework stabilized, the first author re-coded all transcripts, with uncertain cases being reviewed with another author’s assistance. We used HeyMarvin for qualitative analysis, facilitating code organization and pattern evolution tracking [2]. We ended the interviewing process when we reached our saturation criterion [32], which we defined as two consecutive interview pairs without learning any new major insights (i.e., 4 interviews). For higher-level theme

¹The complete interview guide is available in the supplementary material.

²The complete coding framework and manual are available in the supplementary material.

identification, we shifted focus from individual pairs to patterns distinguishing the two broader groups—frequent and infrequent users. This analytical pivot allowed us to identify systematic differences in how developers perceive, approach, and respond to AI tools despite shared organizational contexts, revealing individual factors that transcend team-specific circumstances. We report qualitative findings organized into emergent categories. We avoid discussing frequency counts or percentages in line with established guidance against quantifying qualitative data [27].

3.5 Member Checking

Following data analysis, we validated our findings through member checking with interviewees to ensure our interpretations accurately reflected their perspectives and experiences [20]. We sent all interviewees a draft of the paper’s introduction, methods, results and discussion along with a list of questions asking for their extent of agreement with and/or the relevance of the findings, alongside additional thoughts and feedback. Participant feedback confirmed our findings, with some providing additional clarifications, but no new insights or disagreements emerged e.g., *“I think it accurately reflects my experiences”* (PID34).

3.6 Limitations

Our interview study is affected by several limitations commonly experienced in interview research. The transferability of our findings may be influenced by self-selection bias among participants [62, 76], as there could be differences between those invited and those who participated. All participants were engineers at a large software company vocally committed to AI adoption, with GitHub Copilot as the only officially sanctioned tool. This limits generalizability to engineers at smaller or larger companies without AI promotion, or to open source development contexts. Despite this limitation, we argue our findings are still of value to the broader community because historically, it has been shown that single-case case studies can provide meaningful contributions to scientific discovery [31], and are an essential type of research in addition to research that studies broader populations as championed by Basili [6]. A limitation to construct validity is that we measured AI tool usage through GitHub Copilot invocations only. Engineers using general-purpose AI tools (e.g., internal OpenAI models) or non-compliant tools (e.g., Cursor, Windsurf) could be classified as infrequent users despite potentially heavy AI usage elsewhere. As discussed in Section 3.1, the company’s strong pro-AI culture may have influenced participants’ responses. Finally, throughout this discussion, we occasionally use causal language (e.g., ‘leads to’) for clarity and readability. These phrasings should not be interpreted as statistical causal claims, as our qualitative design does not support formal causal inference.

4 RQ1: What Distinguishes Frequent and Infrequent GenAI Development Tool Users

During our analysis, we identified distinct patterns in how frequent and infrequent users perceive, approach working with, and respond to challenges using GenAI development tools— even among developers in similar organizational contexts (cf. Figure 3). Figure 1 illustrates our conceptual framework showing how individual tool integration progresses through distinct stages, with external factors influencing this process throughout (note that while arrows

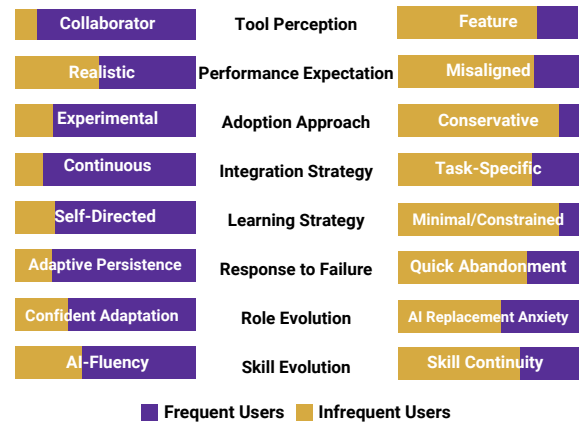


Figure 3: Distributions of user type proportions by factor.

connect to specific stages for visual clarity, external factors impact the entire adoption journey).

4.1 Mindset Formation: Initial Perceptions Shape Usage Patterns

We observed two primary diverging patterns in tool perception that appeared to influence subsequent usage behaviors.

4.1.1 Tool Perception: Collaborator vs. Feature. Many frequent-AI users described their interactions with the tooling using collaborative language and partnership framing, using terms like “teammate,” “collaborator,” or “assistant.” e.g., *“I use GitHub as a colleague to discuss ‘OK, this is what I’m trying to look at, this is the high level task, these are the my plans to go about implementing this feature, Do you see any gaps in that?’”* (PID2). This tool as a collaborator perception appeared to encourage deeper engagement as they invested time understanding its capabilities and strengths, as they would when onboarding a new team member. Infrequent-AI users more often described these tools as enhanced features such as advanced code completion or search functionality e.g., *“I’m not going to say that I find it completely without value, but it’s about as useful to me as a stack overflow search.”* (PID7). This utility framing appeared to limit engagement patterns— these developers tended to approach the tool with transactional expectations rather than collaborative ones. The limiting nature of the utility framing is evident when comparing PID7 with their teammate PID23, who despite facing similar technical challenges, developed a more engaged usage pattern e.g., *“we have to write a lot of thorough tests that we can run on our gates periodically. So setting up those tests, writing the basic framework for them is just a lot of boilerplate code that GitHub Copilot helps me come up with... So that’s one of the places I use it.”* (PID23).

4.1.2 Performance Expectations: Realistic vs. Misaligned. Perception also appeared to influence performance expectations. Frequent-AI users who viewed the tool as a collaborator often expressed realistic expectations, expecting assistance rather than replacement, similar to a junior colleague who provides value yet requires oversight: *“I just have to type what I’m trying to do and then it gets me like 90% of the way there and I have to just change a couple things.”* (PID16). Infrequent-AI users who viewed it as a feature often held misaligned expectations, either anticipating minimal

value or expecting near-perfect functionality: *"If I write a function, I would expect it to be able to write a perfect unit test for it."* (PID48). Both types of misaligned expectations appeared to lead to limited exploration or early abandonment when initial attempts failed.

Key Insights: Frequent-AI users view GenAI as a collaborator. Infrequent users see it as a more or less useful developer tool.

4.2 Approach Determination: How Developers Choose to Engage

Following initial mindset formation, developers exhibited different interconnected approaches to tool usage (Figure 1), with experimental mindsets seemingly leading to broader usage patterns and conservative mindsets to isolated task-specific patterns.

4.2.1 Usage Approach: Experimental vs. Conservative Patterns. We observed two distinct approaches that shaped usage depth. Often frequent-AI users took an experimental approach, proactively trying new features, pushing tool boundaries, and experimenting to identify effective use cases e.g., *"at first actually I just give every [one] of my tasks to the to the agent mode and ask him to write the code."* (PID55). This experimental approach by PID55 contrasts with their teammate PID54's more conservative usage pattern, illustrating how risk tolerance and experimentation vary even among developers working on the same codebase. Frequent-AI users tended to treat the tool as having undefined potential, actively seeking new applications and unconventional use cases. *"I started using images as well for doing CSS. Like, I'll open Paint and draw the layout I want, screenshot it, and feed that to Copilot to generate the CSS. It's very good."* (PID38). This experimental approach led to continual usage with tools becoming deeply embedded in their daily workflows, reflecting a view of the tool as a general-purpose collaborator. The willingness to experiment created a positive feedback loop— with successful use cases encouraging the exploration of others.

Alternatively, infrequent-AI users were more likely to take a conservative approach, limiting tool usage to conventional well-vetted applications e.g., *"Something I've heard from a lot of people who've been using it is the first thing they tried out is writing tests using GitHub Copilot and I think that's probably the first thing that I actually tried out."* (PID8). A conservative approach appeared to couple naturally with task-specific usage limited to particular well-defined use cases, e.g., *"It is potentially useful for generating what I would describe as boilerplate code, and I find it frankly dangerous to use for anything beyond that."* (PID7).

Key Insights: Tool perception shapes use: collaborative framing by frequent-AI users drives exploration, utility framing by infrequent-AI users limits use to specific development tasks.

4.3 Encounters with Challenges: Universal Experiences, Divergent Responses

All participants reported encountering challenges with tool usability, particularly around prompt engineering and identifying effective use cases. The divergence emerged not in whether they faced difficulties, but rather in how they interpreted them and responded.

4.3.1 Response to Failure: Adaptive Persistence vs. Quick Abandonment. Many frequent-AI users demonstrated adaptive persistence when facing limitations, continuing with modified approaches and treating failures as solvable puzzles requiring refinement strategies—such as breaking tasks into simpler subtasks or leveraging AI to improve their prompts e.g., *"If I give a big enough task to Copilot, it would hallucinate a lot. So it often needs to be broken down into smaller components. And once I do that, at times it does a pretty decent job."* (PID27). In the face of poor tool performance for a particular task, they were also more likely to strategically select tasks for tooling assistance based on the tool's strengths and weaknesses, rather than abandoning or writing off the tooling entirely e.g., *"It's more just working with it, not exactly that I found a way to optimize it or make it better, but more like, 'OK, these are the limitations, how do I work with it?'... These are some tasks that it can do. So OK, let me use it for those specific purposes."* (PID2).

Many infrequent-AI users showed different response patterns to identical challenges. Compared to frequent-AI users who rarely exhibited quick abandonment when facing a challenge, infrequent-AI users were much more likely to do so (cf. Figure 3) e.g., *"I did something recently with the agent... It involved changing a lot of files in a similar manner. And it did the the change, let's say changing 15 files, but I had some compile errors and then I gave up on this approach and I took the slow and methodic way."* (PID24). This quick abandonment is particularly striking when contrasted with PID24's own teammate PID26, who faced similar prompt engineering challenges but evolved their approach: *"At the beginning I was using more the Google style. Write a sentence or a few words about what I want to search but... when using GitHub Copilot you need to tell a short story. Here is what I want to do. Here is what I tried so far. I failed with this. What would be another option [to go] about it?"* (PID26). Particularly among infrequent-AI users with pessimistic tool expectations, many appeared to interpret these failures as confirmation of tool limitations, rather than as learning opportunities.

4.3.2 Learning Strategies: Self-Directed vs. Constrained Engagement. Patterns in response to failure appeared to directly connect to learning strategies. Many frequent-AI users were self-directed learners pursuing adaptive persistence who naturally sought multiple learning resources. *"I spent time reading blog posts, watching YouTube videos, even looking at how other people structure their prompts on GitHub. There's actually a lot to learn about how to use these tools effectively."* (PID8). This multi-modal approach—combining formal documentation, community resources, and peer examples—accelerated skill development. As with any tool, the learning curve to proficiency takes time and a common pattern among frequent-AI users was the recognition that carving out the time to do so, even when impending deadlines made it difficult, was worthwhile.

Conversely, infrequent-AI users were more likely to engaged in minimal or constrained learning often due to time, resource, or motivational constraints e.g., *"Exploration wise it's been minimal. It kind of just whatever is obvious from the chat window and then mainly just that or completion. So stuff that's kind of like very low friction."* (PID6). Without dedicated learning efforts, some infrequent-AI users remained stuck in limited usage patterns, reinforcing their perception of minimal tool value within their development workflows.

Key Insights: When challenged, frequent-AI users persist and engage in self-directed learning, while infrequent-AI users disengage with AI tools more quickly during development tasks.

4.4 Integration and Evolution: Long-term Professional Impact

Developers exhibited different visions of GenAI's impact on their professional identity and skills, which appeared to shape their integration efforts and skill development priorities.

4.4.1 Role Conceptualization: Evolution and Competitive Pressure. Both frequent and infrequent users recognized GenAI's potential to transform the developer role, though their responses differed. For many infrequent-AI users the impact of this paradigm shift manifested as generalized concerns regarding GenAI making certain developer roles obsolete: *"I think it will take entry level jobs now [that] it has reached the level where you can assign GitHub Copilot a ticket on Azure DevOps and it can create a pull request for you itself."* (PID44). The introduction of GenAI has created uncertainty regarding career progression especially for some early-career developers: *"I am a junior developer right now, what does that mean in three years, four years? I do not know. I don't know if it's still coding and being technically viable or is it a different skill? So I think it has introduced a very important conversation on yeah, maybe pivoting."* (PID25).

For frequent-AI users, concerns regarding the disruptive potential of GenAI often manifested as motivation to stay as in-the-know as possible. Many were motivated to improve their proficiency with the tooling so they could stay competitive with other developers—whomst they viewed as the immediate threat e.g., *"At the end of the day everyone is using it and everyone is being more productive and you wanna be one of those [people] otherwise [you're] gonna be left behind."* (PID20). Frequent-AI users recognized the same threat as infrequent-AI users but often channeled it into motivation for skill development. They often exhibited confident adaptation, believing GenAI would enhance rather than replace their role, with some actively reframing their mindset positively given the perceived inevitability of tool integration *"Viewing it as an extension of myself and trying to make the most of that versus trying to not be replaced... I think using it that way has been encouraging because if I view it as just amplifying me, multiplying me, not replace me, that gives me more encouragement, more energy to pursue it freely and not having that fear in the back of my head or these negative thoughts."* (PID6). This belief also appeared to suggest a full circle impact back onto tool perception as visualized in Figure 1 *"I would think of it as a helper to make work efficient and easier and faster. So I'll say the job description would be software developer helper or assistant."* (PID29).

4.4.2 Skill Evolution: AI-Fluency vs. Continuity. The different responses to competitive pressure manifested in contrasting approaches to skill development. Most frequent-AI users recognized AI fluency as an essential new competency requiring active cultivation e.g., *"I think it's really important that you know how to prompt well and ask the right questions. I think that's gonna be very useful for the future."* (PID12). They invested in developing prompt engineering abilities, context management skills, and judgment about when and

how to leverage AI assistance. Infrequent-AI users often prioritized traditional skills while acknowledging potential changes.

Key Insights: Developers see AI as transformative, but frequent users pursue fluency and advantage, while infrequent-AI users focus on job security and preserving traditional skills.

5 RQ2: How Team and Organizational Factors Shape Individual GenAI Use by Developers

While our paired interview design revealed why developers in closely similar contexts showed different usage patterns by exploring context-specific barriers, these external factors do not affect all team members uniformly. Emergent findings from our analysis revealed that team and organizational factors do not just support or constrain usage— they appear to actively shape individual factors for some developers. Yet in every pair we studied, one developer was a frequent tool user, the other was not. Initially, we expected that developers within similar contexts might show more uniform responses to these shared conditions. Instead, this emerging finding illuminated something more nuanced: These shared organizational factors interact with the individual differences identified in RQ1 in complex, emergent ways. Social construction theory demonstrates that individuals within the same work group systematically interpret organizational influences differently depending on factors such as personal characteristics and social positions [34, 72]. Our results mirror this: we see how similar team contexts produce divergent responses through differential sensemaking processes [72, 90, 93]. This sociotechnical dynamic manifested through several interconnected mechanisms that appear to influence developers' usage journeys from initial perception through long-term integration.

5.1 The Impact of Leadership Communication

Clear encouragement and communication from leadership regarding tool usage and value often provided developers with the confidence, permission, and motivation to invest resources into learning and exploration. As one frequent-AI user explained: *"Encouragements at the org level itself to share examples of how AI is making you better at work I think that was like a big thing"* (PID2). When managers shared concrete examples and tips, it provided practical value that motivated exploration for some developers: *"I mean, they've shared tips and tricks that they found useful, which honestly have been helpful at times."* (PID23). The regularity and consistency of messaging also appeared to matter in some cases. After stalling in their usage after experiencing challenges, one developer found renewed motivation through recent communications from management: *"there's been more messaging in the last month to try it out so I'm going to get back into it and see if I can find a way of using it that actually makes me more productive"* (PID48). Conversely, the absence of clear messaging left some developers uncertain about whether to integrate AI tools into workflows and what the value add would be. Several infrequent-AI users described environments where they received minimal guidance which appeared to reinforce limited usage patterns: *"Just it's a tool, [if] we can use then use it. But [leadership] haven't really been pushing... So we haven't really done much with that"* (PID37).

Key Insights: Consistent leadership messaging encourages AI exploration. Without it, some developers are hesitant to use AI.

5.2 Providing Context-Specific Resources

Many developers struggled with filtering through the large volume of ever-changing learning resources available to identify relevant resources worth investing their limited time into e.g., *“Some weeks I have 10+ meetings show up on my calendar about AI learning from all sorts of different places. And I can’t go to all 10 so it becomes almost a little overwhelming where you just don’t want to do any of them.”* (PID33). While some found resources on general topics like the basics of prompt engineering helpful, especially in early learning stages, such resources often failed to help support many developers through the transition between tool experimentation and meaningful effective integration into their workflows in practice. Many frequent and infrequent users expressed the need for context-specific guidance, calling for team-specific resources identifying suitable tasks and step-by-step implementation instructions, which one participant described as ‘AI-onboarding’ e.g., *“I mean like magic wand would be a comprehensive guide of how to use it for the specific projects that we work in with like example tasks that you could accomplish with it.”* (PID48).

A lack of context-specific guidance created uncertainty about tool capabilities and appropriate use cases even among highly-motivated developers e.g., *“everyone wants to try GitHub Copilot, but right now they’re still confused [about] what Copilot can do and how they can achieve their goals through [it].”* (PID52). Particularly when combined with strong encouragement to use the tool, a lack of concrete relevant guidance lead to frustration for some developers e.g., *“So this is where I get a bit jaded. A lot of the time what we hear from leadership is just ‘use AI, use AI, and AI is going to solve everything.’... I see a lot of talking about ‘we should use AI’, I don’t see a lot of useful collaboration.”* (PID17).

Team-specific demonstrations were perceived as quite valuable and informative, with both participants from one team discussing a demonstration from a colleague who presented a shared-screen style step-by-step demonstration on how they used the tooling to accomplish a specific task commonly performed by the team *“there’s a very common pattern and structure that we have and he was showing how to do the prompts to use agent mode to make a brand new UX component and it saves a lot of time and so it was a very helpful demo.”* (PID6). The specificity of these demonstrations—showing exactly how to accomplish common team tasks—made the value immediately apparent and addressed the critical question of which tasks to use AI tools for and how.

Key Insights: Developers struggle to connect generic AI guidance to their work; team-specific demos show how AI may apply to their work.

5.3 Building Social Learning Structures

Social learning structures helped create environments where individual discoveries benefited entire teams. When teams had robust sharing practices, individual discoveries could create momentum for broader adoption. Effective knowledge sharing took various

forms, from formal sessions to informal exchanges, and these structures built collective competence and resilience in multiple ways.

Different structures served complimentary purposes. AI evangelists and champions helped some developers set realistic expectations about tool capabilities. By openly sharing both successes and limitations, they helped calibrate expectations appropriately.

Formal sessions provided structured learning and helped support skill development for some developers: *“I would say [the] majority of my learning is from attending those sessions because most of the work we do is very similar in nature and it’s quite applicable”* (PID43). When teams created spaces for shared learning, it transformed individual struggles into collective problem-solving opportunities. However, peer influence operates differently even within teams. For example, PID48 found peer recommendations highly motivating: *“Just hearing the recommendations from one of my co-workers is very motivating because if we’re working as a good team, then we’re all like having similar processes.”* (PID48). Yet their own teammate PID35 remained an infrequent user despite this collaborative environment.

Informal knowledge sharing through daily interactions also shaped adoption journeys and enabled continuous micro-learning: *“Sometimes you just get stuck, you go to somebody else and they’re like, ‘Have you tried this feature?’ And you’re like, ‘I have not. Did it work for you?’ And they’re like, ‘Yes.’ And that’s how you end up trying [the feature], because somebody else tried it.”* (PID18). Micro-interactions accumulated into substantial support over time, creating an environment where exploration was socially reinforced. However, developers without access to such structures faced different experiences. Some described learning in relative isolation: *“I never share with anyone and I never received anything from my colleagues as well. I guess we all just kind of sit here and try to think that everyone else can use it.”* (PID10). Without contextual examples or peer support, the learning burden fell entirely on individuals e.g., *“I won’t ask question on my team because I feel like using Copilot is more personal. It’s not something my team can help me [with]”* (PID56).

Key Insights: Social learning structures can turn individual struggles into shared competencies; while knowledge isolation reinforces limited usage patterns.

5.4 Unexpected Interactions: Time, Expectations, and Developing Proficiency

Sufficient learning time appeared crucial for both initial adoption and advanced skill development. Many benefited from formal time allocations: *“Our org has dedicated days to learn. We have around three to five days per semester where we can spend time on learning”* (PID22). This legitimized exploration and removed guilt about skill development. Others leveraged scheduling flexibility to carve out their own time: *“After seeing how powerful it is, I wanted to try it out myself. So one day I just thought OK... I will give one day [to] just focus on it. Try to learn what it is.”* (PID21). Time during day-to-day work without deadline pressures proved essential for effective skill development and identifying effective use cases for many developers. However, this individual initiative varies dramatically within teams. PID21 proactively carved out a full day for exploration, while their teammate PID8 took a more conservative approach, limiting exploration to use cases that others had validated.

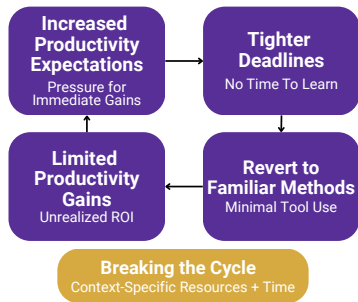


Figure 4: The Productivity Pressure Paradox.

A common theme among participants was experiencing increased productivity expectations from management because they have access to these tools e.g., *“I know that management is expecting us to produce code faster. And I know that the bar is gonna be raised every year because now, even though it’s true or not true, [they] think that we should be much much more productive because we have these tools.”* (PID20). Some managers appeared to assume tool access automatically enabled rapid effective integration into existing workflows and immediate productivity gains, overlooking the significant learning effort and time investments required. Many developers reported experiencing a circular challenge: *“There’s a skill gap among all of us about prompt crafting. The impression is that if I get better at prompt crafting then the Copilot could increase my productivity. But how do I get to that? How much productivity do I need to put into prompt crafting in order to increase productivity in my workflow? So there’s a chicken or egg problem.”* (PID46).

Increased productivity expectations which often translated into tighter deadlines prohibited many developers from doing the exploration necessary to learn how to use the tools effectively, instead falling back on their tried-and-true methods e.g., *“I do really feel the pressure of deadlines and trying to get stuff out and so feel less of the freedom to explore... The few times I’ve tried it, being discouraged, I’m like, OK, well, let me just get this done and then I’ll get to it. That’s been the hindrance.”* (PID6). Without time for exploration, many developers felt they couldn’t understand the tool’s full capabilities or how to use it most effectively: *“we don’t have so much time to learn about it because we are already overburdened with the tasks that we need to do”* (PID44).

These contradictory experiences—heightened productivity expectations, significant learning time requirements, pressure-induced reversion to familiar methods, and limited support—formed a consistent pattern across many developers, suggesting systemic challenges in how some organizations approach GenAI tool integration.

Key Insights: When organizations expect immediate gains from AI without learning support, the resulting pressure blocks the desired productivity benefits.

6 Discussion and Implications

6.1 The Productivity Pressure Paradox

Many organizations invest in GenAI development tools expecting productivity returns [4, 19] while simultaneously treating adoption and integration into existing systems as an individual responsibility [60, 63, 64, 88]. Yet within the context of knowledge work,

the evolving capabilities of these tools create a “jagged technological frontier,” where they improve certain workflow tasks but can harm performance when applied to other seemingly comparable tasks [24]. Many organizations are implicitly asking developers to navigate this jagged edge on their own while maintaining or increasing their productivity.

This contradictory set of experiences—where developers must simultaneously maintain velocity, learn new tools, determine integration strategies, and navigate tool capability boundaries without systematic support—creates what we term the *Productivity Pressure Paradox*. This paradox creates a self-perpetuating organizational dynamic (cf. Figure 4): increased productivity expectations hinder the skill development investments needed to achieve those very gains through effective workflow integration, a challenge many participants faced (Section 5.4). Cognitive science reveals the underlying mechanism: time pressure fundamentally shifts individuals from flexible exploration to rigid exploitation patterns [96], leading many developers to default to familiar approaches rather than discovering optimal integration strategies (Section 4.3).

The paradox appears to operate bidirectionally: first, productivity pressure prevents the temporary productivity dip necessary for learning; second, this approach places burden on individuals to solve organizational optimization challenges.

Increasing Productivity: From Systems to Individual Burdens.

The *Productivity Pressure Paradox* exemplifies how the knowledge sector uniquely places “onto the individual worker the burden of improving output produced per unit of input” [67, 74]—a departure from established principles regarding productivity optimization which demonstrate that productivity improvements come from optimizing systems, not individual heroics [25, 43, 86]. Henry Ford did not expect workers to individually discover how to build cars faster, he systematically redesigned the entire production process, investing significant time and effort into pioneering a reimagined system and building new tools to make it happen. The continuous-motion assembly line wasn’t an incremental improvement from worker tinkering but a comprehensive system redesign that allowed individuals to focus on execution within an optimized workflow. Workers were largely uninvolved in the optimization process; their productivity increased as a consequence of systemic change.

While the labor involved in pre-industrial automotive factories differs significantly from knowledge work like software engineering, this same principle is echoed in canonical software engineering literature. In the *Mythical Man Month*, Fred Brooks argues that adding developers to a late project does not speed it up but rather slows it down by adding layers of complexity [12]. To gain efficiency, better systems are needed. Brooks recognized that productivity improvements on a systematic level require systematic approaches rather than simply adding resources. The parallels become clearer when examining how many organizations approach GenAI integration today. Just as adding developers to a late project increases complexity without corresponding capability, adding GenAI without systematic support creates new coordination challenges and learning requirements (Section 4.3). Yet many organizations expect individual developers to navigate this complexity while maintaining delivery velocity, precisely the individualized approach to productivity that has proven ineffective across domains [14, 41, 52, 58]. We

provide evidence for and echo the argument that the obligation to optimize productivity through GenAI integration should be shifted away from individuals and back toward systems.

6.2 Implications for Organizations

Our findings show how organizational factors can amplify individual usage patterns—reinforcing success when present or exacerbating difficulties when absent. These organizational approaches influence many developers' ability to navigate usage challenges.

Additionally, participant challenges align with key *technostress* factors [11, 89]. *Techno-overload* manifested as increased productivity expectations regardless of actual proficiency. *Techno-complexity* emerged through evolving tools creating steep learning curves. *Techno-uncertainty* appeared as developers struggled to integrate tools into workflows without clear guidance. Research demonstrates these factors harm productivity but can be mitigated through organizational support [33, 36, 89]. Task-technology fit theory shows that aligning technology capabilities, task requirements, and individual abilities requires systematic organizational intervention rather than individual adaptation alone [38].

Organizations should reconceptualize how they assign responsibility for GenAI productivity optimization. The gap between what organizations expect and the support they provide can create a counterproductive cycle, where the pressure for productivity gains risks undermining the desired results. History offers a cautionary tale: the expert systems boom collapsed in part because organizations failed to provide adequate support infrastructure despite investing \$2.5 billion back in the 1980s, leading to the second AI winter when maintenance proved costlier than anticipated benefits [21, 68, 78]. Our findings echo this pattern, which suggests that organizations investing billions into GenAI technology might better realize returns by actively shaping systematic tool integration and investing in context-specific scaffolding that fosters usage and integration [87].

Increasing Productivity by Increasing Support.

Based on our findings, we identify interventions that helped participants navigate AI tool usage challenges. Table 2 summarizes these evidence-based approaches. Effective organizational messaging proved foundational for shaping tool perception and adoption. When tools were framed as collaborative 'partners', developers often reported more successful engagement patterns. Champions proved crucial, translating abstract capabilities into practical applications. These informal leaders created communities of practice where developers shared failures and successes without productivity judgment. *"We share everything—the prompts that work, the ones that fail spectacularly, and most importantly, why"* (PID47).

Context-specific resources resonated more than generic documentation. Developers benefited from domain-specific patterns, use case repositories tailored to their tech stack, and debugging strategies relevant to their codebase. Protected experimentation time—'AI Fridays' or temporarily suspended velocity expectations—enabled pressure free exploration without deadline pressures. These interventions reinforce each other. The productivity pressure paradox emerges when support fails at any level, forcing individuals to bear organizational optimization burdens.

6.3 Implications for Research

Our findings raise important questions about temporal dynamics in GenAI adoption. While technology adoption is inherently dynamic [77, 91], longitudinal research could reveal whether initial struggles give way to fluency as expertise theory suggests [29], or if rapid AI innovation disrupts traditional learning curves. Does the productivity pressure paradox diminish as developers internalize usage patterns, or does it transform as tools evolve and expectations escalate? Research on technostress suggests pressures may compound rather than resolve [85], indicating the paradox may evolve rather than disappear.

The mechanisms through which team factors shape individual approaches warrant deeper investigation. While team support amplifies adoption success or failure, specific mechanisms remain underexplored. Team influences alter individual behaviors through social learning [51], and organizational values cascade through levels [28], yet how this shapes AI tool usage remains unclear. Teams interpret mandates through local contexts [71]—understanding these dynamics could inform more nuanced adoption strategies.

The distinction between viewing GenAI as "collaborator" versus "feature" reveals how mental models profoundly impact adoption trajectories. Research should explore how these models form through prior experiences, initial interactions, or social influences [37], and whether interventions like training or mentorship can shape productive mental models to prevent abandonment patterns [95].

Finally, our findings about shifting roles, from "code producers" to "AI orchestrators", highlight the need to rethink developer education. Successful AI adopters developed meta-cognitive skills: knowing when to use AI, crafting prompts, and critically evaluating outputs. Yet educational programs remain focused on traditional coding [7]. How should we teach students to view AI as a partner rather than merely a tool [26]? Research on these pedagogical shifts could prepare developers to navigate both the productivity pressure paradox and evolving development work.

7 Conclusion

Our study challenges the prevailing narrative that genAI tool adoption is solely driven by individual initiative, revealing instead that successful adoption equally depends on the sociotechnical scaffolding provided by teams and organizations. Through interviews with 54 software developers, we show that developers thrive not just through personal mindset and learning strategies, but because their environments support experimentation, normalize failure, and offer recurring, context-specific guidance. We identify a *Productivity Pressure Paradox*, where organizations expect gains from AI without investing in the necessary scaffolding infrastructure. Indeed, to realize genAI's potential, we argue for a systemic shift in responsibility—from individuals to organizations; calling for environments that foster experimental learning through clear messaging and managerial support. Ultimately, the future of AI in software development will not be determined by tool capabilities alone, but by how well we design our support infrastructures around them.



8 Data Availability

Interview guide, codebook, and codebook manual are available in the supplemental materials on HotCRP. We will post the final supplementary materials publicly on Zenodo with the camera ready.

Table 2: Evidence-Based Team Interventions for AI Tool Adoption

Intervention	Target Stage	Mechanism	Implementation Examples
Clear Organizational Messaging	Mindset Formation	Sets expectations; legitimizes exploration	Leadership demos; documented value propositions; success metrics
AI Tool Champions	Approach Determination	Models usage; provides guidance; reduces risk	1–2 designated team experts with 10% time allocation for peer support
Context-Specific Use Cases	Experience & Learning	Translates capabilities to team workflows	Team wiki with domain examples from actual repositories
Knowledge Sharing Infrastructure	Experience & Learning	Accelerates learning; normalizes challenges	Dedicated channel; “AI wins/fails” in standups; pair programming
Protected Learning Time	Throughout	Enables pressure-free experimentation	“AI Fridays”; sprint time allocation; hackathons
Collaborative Sessions	Experience & Learning	Transforms individual struggles to team learning	Weekly demos; monthly challenges; brown bags

Acknowledgments

Heartfelt gratitude is bestowed upon Chanel  for braving a cross-country summer adventure in the name of continuing her ongoing work as a world-class canine researcher. Her brilliance, curiosity, and natural sense of wonder continue to propel the team toward progress. Meet Pudding , the canine researcher. He has had an ever-enthusiastic presence during this project and is super proud of the work. We thank the interview participants for generously sharing their expertise, perspective, and time with us, all insights in this work originated from them, and this work would not have been possible without them. Courtney Miller, Rudrajit Choudhuri, and Mara Ulloa performed this work during a summer internship at Microsoft. Margaret-Anne Storey contributed to this work while being a visiting researcher at Microsoft. Special thanks are given to Brian Houck, Ben Hanrahan, and Travis Lowdermilk for providing valuable guidance on the interview protocol and exploring the emergent results. We thank Eirini Nathan for providing valuable insights and resources that helped shape the discussion section and the key message of the paper. We also thank Nancy Baym and Syboney Biwa for helping us contextualize our findings in the broader context of related fields of study and find valuable connections between them that helped strengthen this work.

References

- [1] [n. d.]. GitHub Copilot. <https://github.com/features/copilot>. Accessed Jun. 2025.
- [2] [n. d.]. HeyMarvin. <https://heymarvin.com/>. Accessed Jun. 2025.
- [3] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N Bennett, Kori Inkpen, et al. 2019. Guidelines for human-AI interaction. In *2019 CHI Conference on Human Factors in Computing Systems*. 1–13.
- [4] Jessica Apotheker et al. [n. d.]. From Potential to Profit with GenAI. <https://www.bcg.com/publications/2024/from-potential-to-profit-with-genai> Accessed: 2025-05-17.
- [5] Leonardo Banh, Florian Holldack, and Gero Strobel. 2025. Copiloting the future: How generative AI transforms Software Engineering. *Information and Software Technology* 183 (2025), 107751.
- [6] Victor R Basili, Forrest Shull, and Filippo Lanubile. 2002. Building knowledge through families of experiments. *IEEE transactions on software engineering* 25, 4 (2002), 456–473.
- [7] Brett A Becker, Paul Denny, James Finnie-Ansley, Andrew Luxton-Reilly, James Prather, and Eddie Antonio Santos. 2023. Programming is hard-or at least it used to be: Educational opportunities and challenges of ai code generation. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. 500–506.
- [8] Joel Becker, Nate Rush, Elizabeth Barnes, and David Rein. 2025. Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity. *arXiv preprint arXiv:2507.09089* (2025).
- [9] Josiah D Boucher, Gillian Smith, and Yunus Doğan Telliel. 2024. Is resistance futile?: Early career game developers, generative ai, and ethical skepticism. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. 1–13.
- [10] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative research in psychology* 3, 2 (2006), 77–101.
- [11] Craig Brod. 1984. Technostress: The human cost of the computer revolution. (*No Title*) (1984).
- [12] Frederick P Brooks. 1974. The mythical man-month. *Datamation* 20, 12 (1974), 44–52.
- [13] Jenna Butler, Jina Suh, Sankeerti Haniyur, and Constance Hadley. 2025. Dear Diary: A randomized controlled trial of Generative AI coding tools in the workplace. In *Proc. Int’l Conf. Software Engineering (ICSE)*.
- [14] Sue Cantrell and Corrie Comisso. [n. d.]. Outcomes over outputs: Why productivity is no longer the metric that matters most. <https://www.deloitte.com/us/en/insights/topics/talent/measuring-productivity.html>
- [15] Ruijia Cheng, Ruotong Wang, Thomas Zimmermann, and Denae Ford. 2024. “It would work for me too”: How online communities shape software developers’ trust in AI-powered code generation tools. *ACM Transactions on Interactive Intelligent Systems* 14, 2 (2024), 1–39.
- [16] Rudrajit Choudhuri, Dylan Liu, Igor Steinmacher, Marco Gerosa, and Anita Sarma. 2024. How far are we? the triumphs and trials of generative ai in learning software engineering. In *Proceedings of the IEEE/ACM 46th international conference on software engineering*. 1–13.
- [17] Rudrajit Choudhuri, Bianca Trinkenreich, Rahul Pandita, Eirini Kalliamvakou, Igor Steinmacher, Marco Gerosa, Christopher Sanchez, and Anita Sarma. 2024. What Guides Our Choices? Modeling Developers’ Trust and Behavioral Intentions Towards GenAI. *arXiv preprint arXiv:2409.04099* (2024).
- [18] Rudrajit Choudhuri, Bianca Trinkenreich, Rahul Pandita, Eirini Kalliamvakou, Igor Steinmacher, Marco Gerosa, Christopher Sanchez, and Anita Sarma. 2025. What Needs Attention? Prioritizing Drivers of Developers’ Trust and Adoption of Generative AI. *arXiv preprint arXiv:2505.17418* (2025).
- [19] Michael Chui, Eric Hazan, Roger Roberts, Alex Singla, and Kate Smaje. 2023. The economic potential of generative AI. (2023).
- [20] Juliet Corbin and Anselm Strauss. 2014. *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Sage publications.
- [21] Daniel Crevier. 1993. *AI: the tumultuous history of the search for artificial intelligence*. Basic Books, Inc.
- [22] Kyle Daigle and GitHub Staff. 2024. *Octoverse: The state of open source and rise of AI in 2023*. Technical Report. GitHub. <https://github.blog/news-insights/research/the-state-of-open-source-and-ai/>
- [23] Fred D Davis, Richard P Bagozzi, and Paul R Warshaw. 1989. Technology acceptance model. *J Manag Sci* 35, 8 (1989), 982–1003.
- [24] Fabrizio Dell’Acqua, Edward McFowland III, Ethan R Mollick, Hila Lifshitz-Assaf, Katherine Kellogg, Saran Rajendran, Lisa Kray, François Candelon, and Karim R Lakhani. 2023. Navigating the jagged technological frontier: Field experimental evidence of the effects of AI on knowledge worker productivity and quality. *Harvard Business School Technology & Operations Mgt. Unit Working Paper* 24-013 (2023).
- [25] W Edwards Deming. 2018. *Out of the Crisis, reissue*. MIT press.
- [26] Paul Denny, James Prather, Brett A Becker, James Finnie-Ansley, Arto Hellas, Juho Leinonen, Andrew Luxton-Reilly, Brent N Reeves, Eddie Antonio Santos, and Sami Sarsa. 2024. Computing education in the era of generative AI. *Commun. ACM* 67, 2 (2024), 56–67.
- [27] Norman K Denzin and Yvonna S Lincoln. 2011. *The Sage handbook of qualitative research*. sage.
- [28] James R Detert, Roger G Schroeder, and John J Mauriel. 2000. A framework for linking culture and improvement initiatives in organizations. *Academy of management Review* 25, 4 (2000), 850–863.

- [29] K Anders Ericsson, Ralf T Krampe, and Clemens Tesch-Römer. 1993. The role of deliberate practice in the acquisition of expert performance. *Psychological review* 100, 3 (1993), 363.
- [30] Edward Feigenbaum and Howard Shrobe. 1993. The Japanese national Fifth Generation project: introduction, survey, and evaluation. *Future Generation Computer Systems* 9, 2 (1993), 105–117.
- [31] Bent Flyvbjerg. 2006. Five misunderstandings about case-study research. *Qualitative inquiry* 12, 2 (2006), 219–245.
- [32] Jill J Francis et al. 2010. What is an adequate sample size? Operationalising data saturation for theory-based interview studies. *Psychology and Health* (2010).
- [33] Anna Mette Fuglseth and Øystein Sorebø. 2014. The effects of technostress within the context of employee use of ICT. *Computers in human behavior* 40 (2014), 161–170.
- [34] Janet Fulk. 1993. Social construction of communication technology. *Academy of Management journal* 36, 5 (1993), 921–950.
- [35] Janet Fulk, Joseph Schmitz, and Charles W Steinfield. 1990. A social influence model of technology use. In *Organizations and communication technology*. SAGE Publications, Inc., 117–140.
- [36] Fulvio Gaudioso, Ofir Turel, and Carlo Galimberti. 2017. The mediating roles of strain facets and coping strategies in translating techno-stressors into adverse job outcomes. *Computers in Human Behavior* 69 (2017), 189–196.
- [37] Katy Ilonka Gero, Zahra Ashktorab, Casey Dugan, Qian Pan, James Johnson, Werner Geyer, Maria Ruiz, Sarah Miller, David R Millen, Murray Campbell, et al. 2020. Mental models of AI agents in a cooperative game setting. In *Proceedings of the 2020 chi conference on human factors in computing systems*. 1–12.
- [38] Dale L Goodhue and Ronald L Thompson. 1995. Task-technology fit and individual performance. *MIS quarterly* (1995), 213–236.
- [39] Egon Guba. 1979. Naturalistic inquiry. *Improving Human Performance Qtrly.* (1979).
- [40] William Harding and Matthew Kloster. 2024. Coding on copilot: 2023 data suggests downward pressure on code quality. https://www.gitclear.com/coding_on_copilot_data_shows_ais_downward_pressure_on_code_quality/ (2024).
- [41] Douglas H Harris. 1994. *Organizational linkages: Understanding the productivity paradox*. National Academies Press.
- [42] John Haugeland. 1997. *Mind Design II*. MIT Press.
- [43] Heather A Haveman and Rachel Wetts. 2019. Organizational theory: From classical sociology to the 1970s. *Sociology Compass* 13, 3 (2019), e12627.
- [44] Robert R Hoffman, Shane T Mueller, Gary Klein, and Jordan Litman. 2023. Measures for explainable AI: Explanation goodness, user satisfaction, mental models, curiosity, trust, and human-AI performance. *Frontiers in Computer Science* 5 (2023), 1096257.
- [45] Le Van Huy, Hien TT Nguyen, Tan Vo-Thanh, Nguyen Huu Thai Thinh, Tran Thi Thu Dung, et al. 2024. Generative AI, why, how, and outcomes: A user adoption study. *AIS Transactions on Human-Computer Interaction* 16, 1 (2024), 1–27.
- [46] Brittany Johnson, Christian Bird, Denae Ford, Nicole Forsgren, and Thomas Zimmermann. 2023. Make your tools sparkle with trust: The PICSE framework for trust in software tools. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 409–419.
- [47] Matthew R Jones and Helena Karsten. 2008. Giddens's structuration theory and information systems research. *MIS quarterly* (2008), 127–157.
- [48] Mladan Jovanovic and Mark Campbell. 2022. Generative artificial intelligence: Trends and prospects. *Computer* 55, 10 (2022), 107–112.
- [49] Eirini Kalliamvakou. 2024. A developer's second brain: Reducing complexity through partnership with AI.
- [50] Ranim Khojah, Mazen Mohamad, Philipp Leitner, and Francisco Gomes de Oliveira Neto. 2024. Beyond code generation: An observational study of chatgpt usage in software engineering practice. *Proceedings of the ACM on Software Engineering* 1, FSE (2024), 1819–1840.
- [51] Katherine J Klein and Steve WJ Kozlowski. 2000. A multilevel approach to theory and research in organizations: Contextual, temporal, and emergent processes. *Multilevel theory, research, and methods in organizations: Foundations, extensions, and new directions* (2000), 3–90.
- [52] Amy J Ko. 2019. Why we should not measure productivity. In *Rethinking Productivity in Software Engineering*. Springer.
- [53] Paweł Korzyński, Susana Costa e Silva, Anna Maria Górka, and Grzegorz Mazurek. 2024. Trust in AI and top management support in generative-AI adoption. *Journal of Computer Information Systems* (2024), 1–15.
- [54] Mohammad Amin Kuhail, Sujith Samuel Mathew, Ashraf Khalil, Jose Berengueres, and Syed Jawad Hussain Shah. 2024. "Will I be replaced?" Assessing ChatGPT's effect on software development and programmer perceptions of AI tools. *Science of Computer Programming* 235 (2024), 103111.
- [55] Sreejith Kurup and Vivek Gupta. 2022. Factors influencing the AI adoption in organizations. *Metamorphosis* 21, 2 (2022), 129–139.
- [56] Stefano Lambiasi, Gemma Catolino, Fabio Palomba, Filomena Ferrucci, and Daniel Russo. 2024. Investigating the role of cultural values in adopting large language models for software engineering. *ACM Transactions on Software Engineering and Methodology* (2024).
- [57] Stefano Lambiasi, Gemma Catolino, Fabio Palomba, Filomena Ferrucci, and Daniel Russo. 2025. Exploring Individual Factors in the Adoption of LLMs for Specific Software Engineering Tasks. *arXiv preprint arXiv:2504.02553* (2025).
- [58] Shoo K Lee, Sukhy K Mahl, and Brian H Rowe. 2021. The Induced Productivity Decline hypothesis: more physicians, higher compensation and fewer services. *Healthcare Policy* (2021).
- [59] Sarah Lewis. 2015. Qualitative inquiry and research design: Choosing among five approaches. *Health promotion practice* 16, 4 (2015), 473–475.
- [60] Ze Shi Li, Nowshin Nawar Arony, Ahmed Musa Awon, Daniela Damian, and Bowen Xu. 2024. AI tool use and adoption in software development by individuals and organizations: a grounded theory study. *arXiv preprint arXiv:2406.17325* (2024).
- [61] Jenny T Liang, Chenyang Yang, and Brad A Myers. 2024. A large-scale survey on the usability of ai programming assistants: Successes and challenges. In *Proceedings of the 46th IEEE/ACM international conference on software engineering*. 1–13.
- [62] Bernd Marcus and Astrid Schütz. 2005. Who are the people reluctant to participate in research? Personality correlates of four different types of nonresponse as inferred from self-and observer ratings. *Journal of personality* (2005).
- [63] Hannah Mayer, Michael Chui, and Roger Roberts. 2025. *Superagency in the workplace: Empowering people to unlock AI's full potential*. Technical Report. McKinsey Digital. <https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/superagency-in-the-workplace-empowering-people-to-unlock-ais-full-potential-at-work>
- [64] Microsoft. [n.d.]. AI at Work Is Here. Now Comes the Hard Part. <https://www.microsoft.com/en-us/worklab/work-trend-index/ai-at-work-is-here-now-comes-the-hard-part>. Accessed Mar. 2025.
- [65] Matthew B Miles, A Michael Huberman, and Johnny Saldana. 2014. *Fundamentals of Qualitative Data Analysis*. Sage Los Angeles, CA.
- [66] G Ayorkor Mills-Tettey, Anthony Stentz, and M Bernardine Dias. 2007. The dynamic hungarian algorithm for the assignment problem with changing costs. *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-07-27* 7 (2007).
- [67] Cal Newport. 2021. The Frustration with Productivity Culture. <https://www.newyorker.com/culture/office-space/the-frustration-with-productivity-culture>. Accessed: 2025-06-04.
- [68] Harvey Newquist. 1994. *Brain Makers*. Editors & Engineers, Limited.
- [69] Lorelli S Nowell, Jill M Norris, Deborah E White, and Nancy J Moules. 2017. Thematic analysis: Striving to meet the trustworthiness criteria. *International journal of qualitative methods* 16, 1 (2017), 1609406917733847.
- [70] Wanda Janina Orlikowski. 1999. Technologies-in-practice: an enacted lens for studying technology in organizations. (1999).
- [71] Wanda J Orlikowski. 2000. Using technology and constituting structures: A practice lens for studying technology in organizations. *Organization science* 11, 4 (2000), 404–428.
- [72] Wanda J Orlikowski and Debra C Gash. 1994. Technological frames: making sense of information technology in organizations. *ACM Transactions on Information Systems (TOIS)* 12, 2 (1994), 174–207.
- [73] Sida Peng, Eirini Kalliamvakou, Peter Cihon, and Mert Demirel. 2023. The impact of ai on developer productivity: Evidence from github copilot. *arXiv preprint arXiv:2302.06590* (2023).
- [74] Matt Perman. [n.d.]. The Six Major Factors that Determine Knowledge Worker Productivity. <https://www.whatsbestnext.com/2010/02/the-six-major-factors-that-determine-knowledge-worker-productivity/>. Accessed Mar. 2025.
- [75] Kalyan Prasad Agrawal. 2024. Towards adoption of generative AI in organizational settings. *Journal of Computer Information Systems* 64, 5 (2024), 636–651.
- [76] Steven G Rogelberg et al. 2003. Profiling active and passive nonrespondents to an organizational survey. *Jrnl. of Applied Psych.* (2003).
- [77] Everett M Rogers, Arvind Singhal, and Margaret M Quinlan. 2014. Diffusion of innovations. In *An integrated approach to communication theory and research*. Routledge, 432–448.
- [78] Alex Roland and Philip Shiman. 2002. *Strategic computing: DARPA and the quest for machine intelligence, 1983-1993*. MIT Press.
- [79] Daniel Russo. 2024. Navigating the complexity of generative ai adoption in software engineering. *ACM Transactions on Software Engineering and Methodology* 33, 5 (2024), 1–50.
- [80] Joseph Schmitz and Janet Fulk. 1991. Organizational colleagues, media richness, and electronic mail: A test of the social influence model of technology use. *Communication research* 18, 4 (1991), 487–523.
- [81] SciPy. [n.d.]. Optimization (scipy.optimize). <https://docs.scipy.org/doc/scipy/tutorial/optimize.html>. Accessed Apr. 2025.
- [82] Agnia Sergeyuk, Ilya Zakharov, Ekaterina Koshchenko, and Maliheh Izadi. 2025. Human-AI Experience in Integrated Development Environments: A Systematic Literature Review. *arXiv preprint arXiv:2503.06195* (2025).
- [83] Anastasiya Sichkarenko. 2024. The State of Developer Ecosystem 2024: The Unstoppable Rise of AI, Leading Languages, and Impact on Developer Experience. <https://blog.jetbrains.com/team/2024/12/11/the-state-of-developer-ecosystem-2024-unveiling-current-developer-trends-the-unstoppable-rise-of-ai-adoption-leading-languages-and-impact-on-developer-experience/>.

- [84] Ningzhi Tang, Meng Chen, Zheng Ning, Aakash Bansal, Yu Huang, Collin McMillan, and T Li. 2023. An empirical study of developer behaviors for validating and repairing ai-generated code. In *13th Workshop on the Intersection of HCI and PL*.
- [85] Monideepa Tarafdar, Qiang Tu, Bhanu S Ragu-Nathan, and TS Ragu-Nathan. 2007. The impact of technostress on role stress and productivity. *Journal of management information systems* 24, 1 (2007), 301–328.
- [86] Frederick Winslow Taylor. 2004. *Scientific management*. Routledge.
- [87] Amirhosein Toosi, Andrea G Bottino, Babak Saboury, Eliot Siegel, and Arman Rahmim. 2021. A brief history of AI: how to prevent another winter (a critical review). *PET clinics* 16, 4 (2021), 449–469.
- [88] Uplevel. 2024. *AI Won’t Solve Your Developer Productivity Problems for You*. Technical Report. Uplevel. <https://uplevelteam.com/blog/ai-for-developer-productivity>
- [89] Nelda Vendramin, Giulia Nardelli, and Christine Ipsen. 2021. Task-Technology Fit Theory: An approach for mitigating technostress. In *A handbook of theories on designing alignment between people and the office environment*. Routledge, 39–53.
- [90] Viswanath Venkatesh, Michael G Morris, Gordon B Davis, and Fred D Davis. 2003. User acceptance of information technology: Toward a unified view. *MIS quarterly* (2003), 425–478.
- [91] Viswanath Venkatesh, James YL Thong, and Xin Xu. 2012. Consumer acceptance and use of information technology: extending the unified theory of acceptance and use of technology. *MIS quarterly* (2012), 157–178.
- [92] Ruotong Wang, Ruijia Cheng, Denae Ford, and Thomas Zimmermann. 2024. Investigating and designing for trust in ai-powered code generation tools. In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*. 1475–1493.
- [93] Karl E Weick. 1990. Technology as equivoque: Sensemaking in new technologies. (1990).
- [94] Justin D Weisz, Michael Muller, Jessica He, and Stephanie Houde. 2023. Toward general design principles for generative AI applications. *arXiv preprint arXiv:2301.05578* (2023).
- [95] Justin D Weisz, Michael Muller, Stephanie Houde, John Richards, Steven I Ross, Fernando Martinez, Mayank Agarwal, and Kartik Talamadupula. 2021. Perfection not required? Human-AI partnerships in code translation. In *Proceedings of the 26th International Conference on Intelligent User Interfaces*. 402–412.
- [96] Charley M Wu, Eric Schulz, Timothy J Pleskac, and Maarten Speekenbrink. 2022. Time pressure changes how people explore and respond to uncertainty. *Scientific reports* (2022).