Developer Productivity in the Age of Generative AI: A Psychological Perspective

Rob Edwards

Department of Psychology

Northumbria University

Word Count: 8,521

Author Note

An MSc thesis submitted in partial fulfilment of the requirements for the degree of MSc Psychology.

Abstract

Generative AI is profoundly reshaping developers' daily work, yet its psychological impact is still poorly understood. This study tackles that gap, exploring how senior engineers are renegotiating their professional identities as they integrate AI into their workflows. Through thematic analysis of interviews with 12 senior software and platform engineers, this research found a central theme of redefinition: developers are shifting from hands-on *coders* to high-level *conductors* of AI-driven systems. This transformation is driven by three core findings: (1) a fundamental re-architecting of cognitive focus from low-level implementation to high-level strategic thinking; (2) a consequent re-evaluation of productivity, shifting from metrics of output to measures of outcome and impact; and (3) the emergence of AI as a *double-edged tool* for agency, which balances the empowerment of new capabilities against profound anxieties about de-skilling and professional worth. The study concludes that navigating this new landscape requires more than just technical skill. Success now depends on a renewed focus on the uniquely human capacities for critical oversight and strategic thought.

Keywords: Generative AI, Software Engineering, Platform Engineering, Developer Productivity, Agency, Skill Development

Developer Productivity in the Age of Generative AI: A Psychological Perspective

Introduction

Generative artificial intelligence (AI) is not just another tool, it is fundamentally changing knowledge work, particularly software development (Dăniloaia & Turturean, 2024; V. C. Storey et al., 2025). Software development has always relied on complex problem-solving and tools designed to boost human capabilities (Meyer et al., 2017), however AI provides us with much more than a small evolution. It marks a genuine paradigm shift. With stunning speed this technology has jumped from a niche specialty to a ubiquitous assistant. Tools that write, explain, and check code are now embedded directly into developers' core workflows, from their Integrated Development Environments (IDEs) to their command-line interfaces (DeBellis et al., 2025; Nguyen-Duc et al., 2023).

This is not just an incremental update, it is a massive economic and behavioural shift. Framed as cognitive aids (Noda et al., 2023), these systems become a cognitive partner that actively replaces old work patterns (Görmez et al., 2024; Shihab et al., 2025). A systematic review covering 395 research articles confirms this is not just hype; it is a major well-documented transformation (Hou et al., 2024). However, the latest industry research suggests a more nuanced reality, framing AI not as a simple productivity panacea but as a powerful *amplifier* (DeBellis et al., 2025). The core message is that AI amplifies existing team dynamics and organisational cultures, for better or worse. High-performing teams with clear goals, strong communication, and effective workflows will likely see their success magnified. Conversely, teams struggling with technical debt or poor communication may find that AI exacerbates these dysfunctions. This amplifier effect makes it a critical catalyst, intensifying the very tensions this thesis will explore.

The technology industry is championing a narrative of huge productivity gains, but the actual evidence is still complex and contested. This optimism conveniently ignores a deep, long-standing tension in software development: a conflict between how developer productivity is measured and how it is experienced. For decades the industry's focus on simplistic, output-centric metrics has been in direct conflict with the developer's core psychological need for agency, their sense of ownership, mastery, and autonomy (Deci & Ryan, 2000; Meyer et al., 2014). AI is not creating this problem; it is acting as a powerful catalyst on this unresolved tension, forcing an urgent re-examination of both.

Consider the first half of this tension, the challenge of developer productivity (Jaspan & Green, 2023). For decades, the industry has struggled to measure this, often resorting to simplistic and misleading metrics like "lines of code" or "story points" (Forsgren et al., 2018; Meyer et al., 2014). These metrics proved deeply flawed. They incentivized quantity over quality (a classic example of Goodhart's Law), failed to capture the value of "invisible" work like mentoring or debugging, and ultimately demoralized developers (Fowler, 2003). This failure forced a shift towards more holistic, multidimensional, models like the SPACE framework (Forsgren et al., 2021) which mandates that metrics move beyond simple output to include *Satisfaction and Well-being*, *Performance*, *Activity*, *Communication and Collaboration*, and *Efficiency and Flow* (SPACE). This aligns with the broader psychological perspective where productivity is not just output, but the effective pursuit of valued goals while maintaining well-being and considers effectivness (Graziotin et al., 2015; Jaspan & Green, 2023; Zelenski et al., 2008). AI now threatens to upend this human-centric view, raising new questions. For instance, who really owns the code now? Is it the developer, the AI, or the team? We also do not know if developers will focus on their own contributions or on the team's overall success. This remains a critical, unexamined question.

Running in parallel to productivity, the AI-driven shift also strikes at the other half of the tension; a developer's fundamental sense of agency. This is not a peripheral concern, it is a core psychological requirement. Decades of research in Self-Determination Theory (SDT) establish that human motivation, performance, and well-being are founded on three basic psychological needs (Deci & Ryan, 2000):

1. Autonomy: The need to feel volitional and be the author of one's choices.

2. Competence: The need to feel effective and experience mastery.

3. Relatedness: The need to feel connected to and valued by others.

These factors are the established bedrock of developer motivation and job satisfaction (DeBellis et al., 2024; M.-A. Storey et al., 2021). AI introduces a direct conflict with each of these foundational needs. It challenges autonomy by blurring the lines of authorship. It threatens competence by automating fundamental tasks, raising a huge question: is AI a tool for leverage, or a substitute that makes personal accomplishments feel smaller (Debellis et al., 2025; Noda et al., 2023)? Finally, it has the potential to disrupt relatedness by altering the critical human-to-human collaboration and mentorship that defines software development. This forces a hard look at what it means to be a developer in an AI-augmented future.

This challenge to the developer's future is precisely where a major gap has opened between their daily reality and the research community's ability to study it. Because this technology is moving so fast, we need to capture first-hand accounts from the field to understand what is actually happening. While industry reports might show productivity numbers (DeBellis et al., 2025), they do not fully explain the psychological impact on the developers using these tools every day. This leaves a crucial blind spot; we do not understand how developers are making sense of their new workflow or what it means for their professional identity, job satisfaction, and well-being. That is the gap this study addresses. Using thematic analysis, this research explores how AI is reshaping developers' views on productivity and agency. Our approach is guided by critical realism, which means we are not just describing experiences but seeking to uncover their underlying causes (Wiltshire & Ronkainen, 2021). A qualitative study is essential for this work; numbers alone simply cannot capture the rich, contextual story that emerges from developers' own words.

To explore these issues, this study uses semi-structured interviews with software and platform developers, focusing on three key questions:

1. How do developers describe and evaluate their productivity (for both themselves and their teams) when using AI tools?

2. How does using AI affect a developer's sense of ownership, path to skill mastery, and autonomy?

3. How does a developer's primary focus (individual output vs. team output) shape their experience and perception of AI's impact?

If organisations want AI to deliver real value for their developers, they cannot ignore the human side of the equation. Understanding the psychological factors at play is not just an afterthought; it is central to making human-AI collaboration effective. This is the only way to properly leverage these new tools while maintaining team productivity and motivation (Ralph et al., 2020). This research investigates that human dimension, how AI adoption truly feels and functions for the developers on the ground.

Method

Study Design

I chose a qualitative research design grounded in a critical realist epistemology (Creswell & Creswell, 2017). This perspective assumes that while our understanding of reality is always filtered through personal interpretation, these interpretations point to real, underlying structures and causes.

This approach was perfect to meet my primary research goal: to understand developers' lived experiences as they integrated generative AI tools into their work. I focused specifically on how these tools affect developers' sense of productivity and agency.

I define a developer as any professional directly involved in the software development lifecycle. This includes both software engineers, who write application code, and platform (or DevOps) engineers, who build and maintain the infrastructure that code runs on.

My main data collection method was semi-structured interviews. This method is ideal for critical realism because it helps move beyond a person's surface-level experiences to uncover the deeper mechanisms that cause them. A survey might capture broad opinions, but interviews allowed for a flexible, in-depth conversation. This flexibility was crucial, it gave participants the space to focus on what they felt was important and allowed me to dig deeper into the why behind their experiences. This process was essential for gathering the rich, contextual data needed to understand the underlying nature of agency and productivity in this new landscape (Braun & Clarke, 2006).

Participants and Recruitment

I used a purposive sampling strategy to recruit senior-level software and platform engineers. To qualify, participants needed at least five years of professional experience and had to be active users of AI tools for development work.

The final sample included 12 participants. My initial target was 12 to 18 people, but the goal was always informational power, not just hitting a number. By the final interviews, the core themes were repeating, suggesting we had approached thematic saturation, where new interviews stop revealing significant new insights (Guest et al., 2006). Because the participants were all senior professionals, their feedback was incredibly rich and reflective, providing dense, high-quality data. I do acknowledge that while the core themes felt saturated, a larger sample might have uncovered more peripheral experiences.

Recruitment took place on professional networks like LinkedIn and in the DORA community (Google Cloud, n.d.) to ensure a diverse group. An initial call for participants brought in 20 responses. This pool was narrowed down to the final 12 after screening for experience and accommodating scheduling conflicts.

The final group was diverse. They used different AI tools, held various roles (front-end, back-end, platform, architect), worked in companies ranging from startups to large corporations, and came from different cultural backgrounds. This diversity was key to capturing a rich range of perspectives. A full demographic overview is available in Table **??**.

Procedure and Materials

After receiving ethical approval, I sent an information sheet to all potential participants. Those who agreed to join provided informed consent through a Google Form and then scheduled an interview. The interviews were all conducted between May and

August 2025.

I collected the data using one-on-one, semi-structured interviews on Google Meet. Each session lasted about 45 minutes and was recorded with the participant's permission. The interview guide (see **??**) was designed to be both focused and flexible, with three main parts:

1. Warm-up: I started with simple questions about the participant's role and how they used generative AI.

2. Core Questions: The main part of the interview centered on open-ended questions about productivity, agency (ownership, autonomy, and mastery), and the balance between individual and team focus.

3. Wrap-up: I left time at the end for participants to share any final thoughts or reflections.

This structure ensured all key topics were covered while still allowing me to explore unexpected ideas that came up. After each interview, I held a brief debriefing session with the participant and shared a debrief sheet.

Finally, I transcribed each interview recording verbatim. To ensure accuracy, I listened to the audio multiple times, making sure to capture the full context of the conversation, including significant pauses or other non-verbal cues.

Data Analysis

I analysed the interview data using inductive thematic analysis, following the six-phase process outlined by Braun and Clarke, 2006: (1) data familiarisation, (2) initial code generation, (3) searching for themes, (4) reviewing themes, (5) defining and naming themes, and (6) producing the report. While the initial stages were purely inductive and *bottom-up* to ground the analysis in the data, the full process involved a more nuanced, two-stage strategy to achieve interpretive depth, as detailed below.

This entire process was guided by my critical realist perspective. The goal was to move beyond the surface-level stories shared in the interviews to understand the deeper psychological forces shaping how developers experience AI at work.

Analytic Strategy

I used a deliberate two-stage process to analyse the data, which ensured the findings were both grounded in the interviews and informed by established theory.

The first stage was purely inductive, meaning I stayed as close to the participants' own words as possible. After reading and re-reading all 12 transcripts, I went through them line-by-line to generate over 300 descriptive codes. This process captured the explicit, surface-level meaning of what developers said, grounding the entire analysis in their direct accounts.

After establishing the descriptive codes, I did a second, *top-down* analysis to dig deeper. Why was this necessary? A purely descriptive approach is not enough when dealing with complex psychological concepts like agency and productivity. To truly understand the data, I needed to connect the participants' raw accounts with established psychological theories. In this stage, I used theories like self-determination theory, effort justification, and the automation paradox as a lens to interpret the descriptive codes. This allowed me to move from what participants said to why they might have said it, uncovering the deeper mechanisms shaping their experiences.

I recognise that this interpretive stage risks introducing researcher bias. I managed this in two ways. First, I made sure every theoretical interpretation was directly tied to the descriptive codes from stage one. Second, I engaged in a process of critical reflection to track and question my analytical assumptions throughout the process.

This two-stage approach produced a final thematic map that is both built from the ground up and sharpened by established theory, providing a robust foundation for my findings.

Analytic Rigour and Trustworthiness

I built credibility (how believable the findings are) by spending a significant amount of time immersed in the data and using the systematic two-stage analysis described previously. To support transferability (how applicable the findings might be to other contexts), I provide rich, detailed descriptions of the participants' experiences in the discussion section. This allows you, the reader, to judge for yourself how these insights might apply elsewhere.

For dependability and confirmability (how consistent and objective the process was), I kept an audit trail that documents the steps of the analysis, from initial codes to the final thematic map. Finally, as mentioned, I engaged in a process of critical reflection to critically examine my assumptions and potential biases throughout the research. A full account of this process is available in Appendix **??**.

Ethics

This study received full ethical approval from the Northumbria University Research Ethics Committee. Participation was entirely voluntary, and I obtained informed consent from every participant before their interview. The consent form clearly explained the study's purpose, what participation involved, the right to withdraw at any time without penalty, and how all data would be kept confidential. To protect anonymity, I removed all identifying details from the interview transcripts and have used pseudonyms throughout this report. The original recordings were stored securely and kept separate from the anonymised data.

I had a plan in place to manage any potential distress. If a participant had become upset, I would have immediately paused the interview and offered them information for support services, like their company's Employee Assistance Programme (EAP) or relevant mental health charities.

Results and Discussion

The analysis of interviews with 12 senior developers revealed three core themes that address the central research questions: How is generative AI *really* changing their work, specifically productivity, agency, and professional focus? Eleven of the developers had already woven AI deep into their daily workflows, with at least two years of professional use under their belts. The twelfth offered a crucial and valuable divergent counterpoint, a skeptical, minimal-use perspective that helps sharpen the analysis. To maintain confidentiality, all participants have been given pseudonyms (see Appendix **??** for a full overview).

A central concept emerged from our conversations: the developer's role is being redefined from *coder* to *conductor*. But even that metaphor is not perfect. It implies a top-down control that does not quite capture the collaborative relationship developers described. They did not see AI as a subordinate to command but as a cognitive partner. As David put it, it's a "non-judgemental senior advisor."

This complex shift in professional identity boils down to three core themes:

1. The Re-Architecting of Developer Focus
2. The Shifting Definition of Productivity
3. The Double-Edged Tool of Agency

Everyone's journey looks a bit different. While the themes were common, the developers used AI in surprisingly diverse ways. We saw several archetypes emerge: the *Planner* (Alex), using AI for high-level strategy; the *Learner* (Leo), using it to master new skills; the *Systems Thinker* (Ben), who offloaded syntax to focus on architecture; and the *Force Multiplier* (George), who used AI to single-handedly fill the roles of an entire team. This shows the shift from coder to conductor is anything but a monolithic experience.

This is not just a story about technology changing a worker. Recent industry research already frames AI as a powerful *amplifier*, not a silver bullet (DeBellis et al., 2025), and our interviews hammered this point home. Charlie articulated this perfectly, framing the AI as a mirror for its user's own abilities:

> I think it makes good developers a thousand times better. It makes poor developers a thousand times poorer. Um, so I think it is a, you know, it's a mirror of its operator.

Charlie's *mirror* analogy is the perfect lens for understanding our findings. The AI does not dictate outcomes; it magnifies what the developer brings to the table—their skills, their mindset, and their context. Each of the three themes that follow is filtered through this powerful amplifier effect.

Table 1

Thematic Framework Summary

Main Theme	**Brief Definition**	**Core Analytical Tension**
The Re-Architecting of Developer Focus	Characterises the shift in a developer's cognitive focus, prompted by AI, away from low-level implementation and toward higher-level strategic thinking and systems architecture.	The tension between a focus on **Individual Output** and the need for **System-Level Impact**.
The Shifting Definition of Productivity	Describes the evolution in how developers perceive productivity, moving from tangible metrics of output to qualitative concepts of outcome, such as value, impact, and a sustainable state of flow.	The tension between measures of **Raw Output** (e.g., speed, volume) and the desire for **Meaningful Outcome** (e.g., value, quality).
The Double-Edged Tool of Agency	Explores the dual impact of AI on developer agency, wherein the empowerment from new capabilities is simultaneous with anxieties stemming from potential deskilling and over-reliance.	The tension between the **Empowerment** of new capabilities and the anxieties of **De-skilling and Risk**.

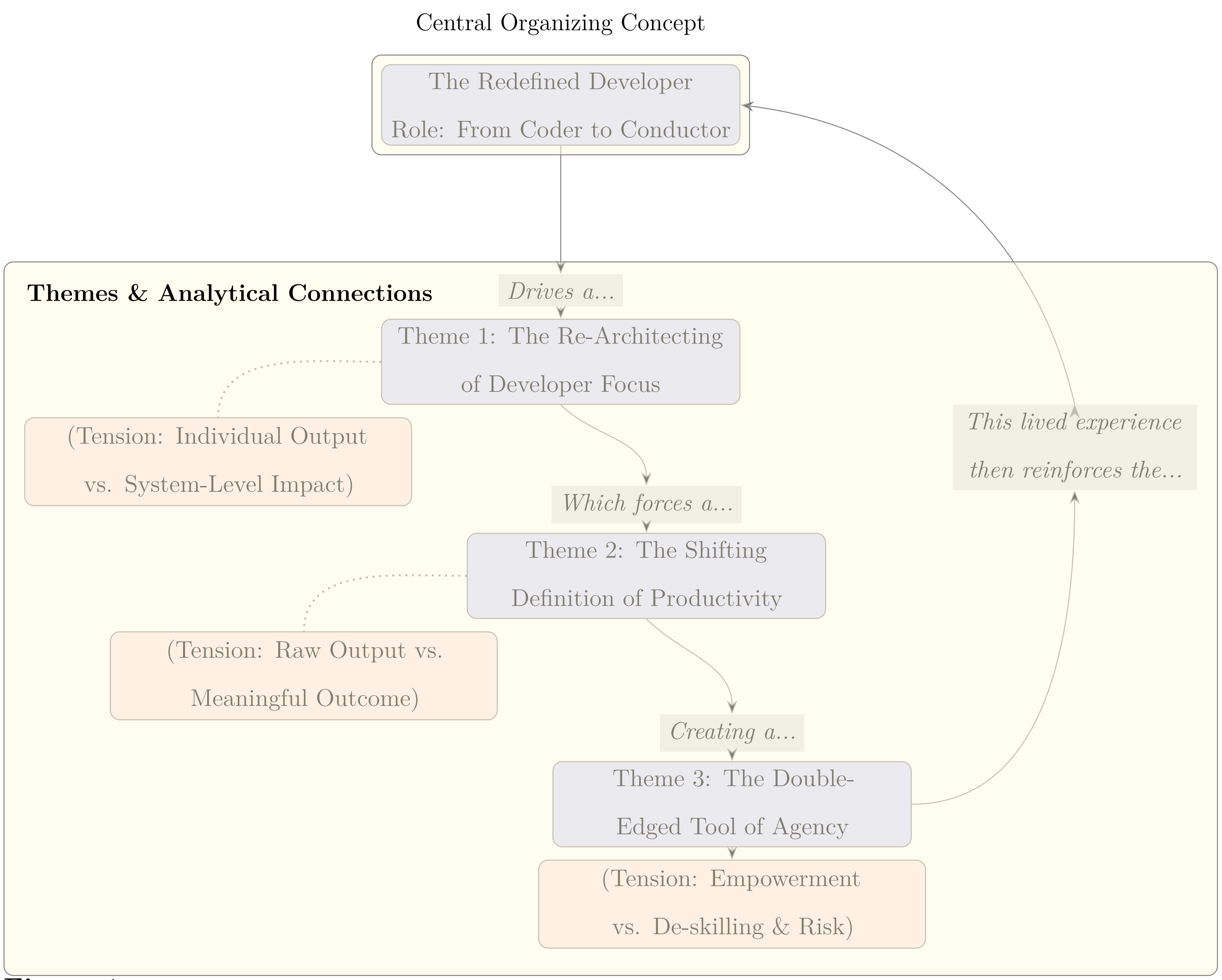
Central Organizing Concept
The Redefined Developer Role: From Coder to Conductor
Themes & Analytical Connections
Drives a...
Theme 1: The Re-Architecting of Developer Focus
(Tension: Individual Output vs. System-Level Impact)
Which forces a...
Theme 2: The Shifting Definition of Productivity
(Tension: Raw Output vs. Meaningful Outcome)
Creating a...
Theme 3: The Double-Edged Tool of Agency
(Tension: Empowerment vs. De-skilling & Risk)
This lived experience then reinforces the...

Figure 1

A Critical-Narrative Model of the Thematic Cycle of Redefinition.

The Re-Architecting of Developer Focus

This theme is about a fundamental shift in what it means to be a developer, their professional identity. Generative AI is pushing engineers to move their focus from tangible, individual output (like writing lines of code) to more intangible, system-level thinking. For most participants, AI acted as a catalyst, accelerating their shift from line-by-line coding towards higher-level architectural thinking.

This transition, however, is not universal. The experience of Harry, a deep specialist in a niche, high-stakes field, provides a crucial counterpoint. He fundamentally did not trust the AI's output in his domain, which prevented him from deeply integrating it into his work. His scepticism was born from repeatedly receiving factually incorrect answers in his area of expertise, leading to a blunt conclusion:

> every time I ask AI a topic or a little bit trickier more like difficult question about a topic I'm really well versed in. It's complete bullshit.

Harry's experience is more than just an exception, it reveals the absolute prerequisite for the coder to conductor shift: trust. Without a baseline faith in the tool's accuracy, developers will not use it for anything other than trivial tasks.

For the eleven developers who did cross that threshold, AI triggered a major cognitive shift. They consistently described this change in two ways: first, as an elevation to more strategic, systems-level thinking, and second, as the adoption of the AI as a collaborative, cognitive partner.

Elevation to Strategic, Systems-Level Thinking

Offloading the cognitive burden of syntax to AI frees developers to concentrate on system architecture and strategy, a shift that is reshaping their professional identity. Ben, for example, now sees himself as "much more of a systems level thinker" because the act of coding, once a specialised craft, now "seems like... anyone could do this." His comment gets to a core finding: implementation is becoming a commodity.

The developer's value is shifting from the tangible act of writing code to the intangible act of ensuring a system's architectural integrity. While AI can generate the bricks (code), the developer must be the architect who designs the blueprint and guarantees the entire structure is sound. This cognitive reallocation was vividly captured by Charlie, who described the automation of "grunt work" as a "forcing function" that pushed him to become a better architect. To effectively direct the AI, he realised he needed to deepen his own knowledge, concluding, "the more I knew about what I was trying to instruct the models to do, the more effective I can become."

This new focus on high-level thinking led participants to flatly reject simplistic, quantitative views of productivity. Alex dismissed the idea of writing “code by the kilogram,” stating, “I’m a lot more interested in if a tool could help me plan or understand a document.” David articulated the same principle, defining productivity not by volume but by its value to others:

> Um making impact would be would be being productive. So not the amount of lines written or anything, but actually building something of like substance that that someone else could readily consume or use or help someone with.

Together, these statements signal a move away from a factory-floor model of software development towards an ethic focused on intellectual engagement and human-centric value.

A key part of this architectural role is better planning, and AI acts as a forcing function for it. To delegate a task to an AI, a developer must first clarify the desired outcome. As Charlie noted, the process forces a more thoughtful approach:

> it actually forces me to think through what I want... I actually was very thoughtful about how I wrote down what the prompt was that I wanted it to create.

This highlights a powerful side effect: AI forces metacognition. The need to craft an effective prompt compels a developer to engage in metacognition, or *thinking about thinking*, the higher-order processes used to monitor and regulate one’s own problem-solving (Flavell, 1979). The AI becomes a partner in a metacognitive loop, prompting the developer to translate a vague idea into a clear, explicit set of requirements.

By offloading the mental details of implementation, developers are not simply working faster; they are working at a higher level of abstraction. This transition, from a craftsperson focused on code to an architect focused on a system’s design, is a redefinition of the developer’s role. However, this cognitive offloading is a strategic trade-off that introduces clear risks, creating a direct link to the anxieties around de-skilling that will be explored in the next theme.

AI as Mentor, Collaborator, and Cognitive Partner

Beyond being a simple tool, AI was consistently framed by participants as a cognitive partner that plays two distinct roles. At times, it acts as a “non-judgemental mentor” for top-down guidance. At other times, it is a peer-level collaborator and “rubber duck”, a term for verbalising a problem to an inanimate object to find a solution (Hunt & Thomas, 1999).

This “non-judgemental” quality is one of the AI’s most powerful psychological features. It creates a profound sense of psychological safety which is the belief that an environment is safe for interpersonal risk-taking (Edmondson, 1999). This private space for learning, free from social judgement, removes the fear of embarrassment that can inhibit creativity and experimentation. David captured this perfectly, framing the AI as a trusted confidant:

> I use it as a non-judgemental, um, you know, senior advisor, I think. That’s how I see it. ...you don’t want to be the guy that’s constantly tapping someone on the shoulder going, ’how can you just show me how to do this?’... You can, you know, question it as much as you want. It’s not going to get angry or annoyed with you, you know?

This psychological safety encourages developers to ask the “stupid questions” they might hesitate to voice to human colleagues. Charlie, for instance, explained he is comfortable asking the AI basic questions about testing that he would never ask a person for fear they would “think I’m stupid.” The AI partner, he concluded, creates a space where “I can ask it anything that I would have been afraid to ask another human.”

This private interaction also has a clear strategic purpose: acting as a partner for rehearsal before social engagement. This behaviour ranges from a quick tactical check to a deep refinement of an argument. At the tactical level, Frank uses the AI to gauge his position’s viability before a team debate:

> ...get a sense for how much should I... fight for my position... if the AI is like, 'Oh, that's a good idea,' then I'll fight for it more, but if the AI is like, 'That's a terrible idea,' then I'll like back off...

This rehearsal can evolve into a more sophisticated workflow. Jeff uses the AI not just for answers but as a sparring partner to "critique" his own knowledge and "give me feedback." This process transforms the AI from an information-retrieval tool into a key part of his metacognitive loop, allowing him to strengthen his arguments, anticipate counter-arguments, and enter discussions with a more robust position.

This dynamic also acts as a powerful "equaliser" in team settings, where an idea's influence can be tied to the assertiveness of the person proposing it. The AI shifts the basis of influence from persuasive rhetoric to demonstrable results. As George explained:

> sometimes it's the most introverted and quiet people that have the best ideas... if you can create something that actually works and show it to people, then that speaks, that speaks much louder than you can.

The ability to privately validate an idea or build a prototype provides a new, more meritocratic medium for communication. The working prototype becomes the argument, allowing the most compelling demonstration, rather than the loudest voice, to drive technical direction.

The sophistication of this human-AI partnership is clear in the way developers differentiate their use of tools. Ben described his workflow:

> if I wanna like get an understanding of a thing or like challenge an idea... I'll do Command Shift I for the chat pane... if I want it to just like kind of fix something up... I'll just do like Command I for the inline.

This is a clear example of distributed cognition (Grinschgl & Neubauer, 2022; Hollan et al., 2000) in action. The developer and AI tools form a single, hybrid problem-solving system where cognitive tasks are strategically divided: the conversational chat pane is a partner for high-level ideation, while the inline assistant is a subordinate for mechanical tasks.

AI's role, therefore, extends beyond that of a simple tool, functioning instead as a key component in a distributed cognitive system that both enhances problem-solving and fosters the psychological safety required for more inclusive and creative collaboration.

This re-architecting of focus represents a fundamental shift. By offloading low-level implementation and fostering a collaborative partnership, AI elevates the developer's role from a craftsperson of code to an architect of systems. This is not just a change in workflow; it is a redefinition of the developer's core value. The main implication is that career progression may need to shift away from pure coding skill toward architectural and strategic capabilities. But if a developer's value is no longer in the tangible code they produce, how should their productivity be measured? This foundational change directly leads to the question of productivity, explored in the following theme.

The Shifting Definition of Productivity

This theme addresses how developers are rethinking productivity in the age of AI. A critical tension emerges between traditional measures of output (speed and volume) and emerging definitions of outcome (impact and value).While most participants are shifting towards valuing outcomes, this transition is not seamless. For some, like the sceptical user Harry, productivity remains tied to implementation speed. For others, the promise of AI-driven efficiency was often undermined when the tool itself hindered progress. This theme, therefore, explores both the aspirational redefinition of productivity and the pragmatic challenges of achieving it.

Recent, large-scale industry research supports this shift from volume to value. The 2025 DORA report on AI-assisted software development uncovered a critical paradox: developers who heavily use generative AI report higher job satisfaction and less burnout, even while spending less time on the work they deem *valuable* (Storer, 2025). The resolution to this paradox is simple: the very definition of *valuable work* is in flux. The DORA research suggests developers are renegotiating their contributions, moving from hands-on implementation towards a more holistic view of value that encompasses utilitarian, reputational, and intrinsic factors. The report's findings provide compelling quantitative support for the qualitative shift observed in this study, where productivity is being redefined not by the volume of code, but by the strategic impact of the work delivered.

From Volume to Value and Impact

The most significant shift in how participants view productivity is the move away from valuing code volume toward valuing the tangible impact of their work. This re-prioritisation reflects a deeper understanding of a developer's role, where the ultimate measure of success is not output quantity, but its quality, efficiency, and usefulness to others.

One participant, for example, explicitly rejected traditional metrics, instead framing productivity as the "total cost of build." As Alex stated:

> For me, it's... a total cost of build. How long does it take me to build a proper, uh, quality, uh, product?

Alex's metaphor of a "total cost of build" reframes productivity from a short-term measure of output to a holistic, long-term process of value creation. The concept of *cost* extends beyond the immediate effort of writing code; it incorporates the future expenses of maintenance, debugging, and user friction—all aspects of technical debt. By focusing on building a "proper, quality, product," Alex aligns himself with a craftsman ethic, where pride is derived not from production speed but from the final artefact's durability and integrity. This perspective defines a productive developer not as someone who writes code quickly, but as someone who thoughtfully architects a sustainable and valuable solution.

This focus on impact also reveals a social and collaborative dimension to productivity, where value is defined by how useful the work is to others. David, for instance, defined being productive not by the "amount of lines written", but by "building something of like substance that that someone else could readily consume or use or help someone with."

David's contrast between "lines written" and building something of "substance" that can "help someone with" exemplifies Organisational Citizenship Behaviour (OCB), defined as discretionary behaviour that promotes organisational effectiveness (Organ, 1988). He draws a clear line between quantifiable output, his formal job duties, and the discretionary, altruistic acts that are a core dimension of OCB (Fan et al., 2023; Felix & Eboka, 2024). His goal is not merely to complete a task but to improve the team's social and psychological context by creating something his colleagues can "readily consume," which reflects a more evolved view of OCB (Felix & Eboka, 2024). This focus on his voluntary, positive impact on the people around him, rather than simply fulfilling his role's requirements, is the very heart of OCB.

From Raw Speed to Sustainable Flow

While AI is often marketed as a tool for raw speed, participants described a more nuanced ambition: achieving a sustainable state of *flow.* Their goal was not simply to accelerate tasks but to use AI to eliminate the cognitive friction that disrupts momentum. Leo described this unproductive friction as "spinning my wheels, searching through Stack Overflow and documentation sites," an experience that "doesn't feel productive." Participants therefore saw the reduction of such tasks as a direct enhancement of productivity, reframing the primary benefit of AI from mere acceleration to the preservation of focus.

This reframes AI as a tool for managing cognitive resources, not just increasing output. Participants aimed to reach a "good enough" state more quickly, conserving mental energy for complex challenges. The impact was transformative for some, like Frank, who described the change not as an incremental improvement but as a subjective shift of an "order of magnitude." This metaphor signals a qualitative change in the nature of his work; the AI has not just made him faster but has fundamentally altered his experience of what is possible.

As George put it:

> I think it's getting something that's good enough out the door as quickly as you

> can so you can move on to the next thing, right? Um, and I think, I think AI accelerates that.

This pragmatic approach, prioritising momentum over perfection, aligns with cognitive load theory (Sweller, 2024; Sweller et al., 2019) and the importance of flow states (Nakamura & Csikszentmihalyi, 2014). Getting something "good enough" out the door aligns directly with achieving a flow state, where a key component is the seamless progression between tasks. The AI enables this by rapidly generating a baseline solution, as Leo explained:

> it's getting it to the point where it works is the piece that I get accelerated and then I have that, it doesn't change the fact that I still have this whole other phase of work I do on it afterwards.

By accelerating the journey to a workable first draft, the AI helps conserve a developer's finite cognitive resources.This allows them to bypass the initial, often paralysing, stages of a task and immediately engage with tangible work, facilitating a more sustainable and productive state of flow.

However, this AI-assisted speed also creates new pressures. As one participant noted, their personal baseline for a productive day has shifted upwards, recalibrating expectations in a way that affects well-being. Frank stated:

> my baseline for productivity has definitely changed... that's now the new baseline of like low productivity.

This reveals a potential productivity paradox (Brynjolfsson, 1993; Brynjolfsson et al., 2020), where tools designed for efficiency also raise expectations. Frank's phrase, "new baseline of like low productivity," shows that his internal benchmark for performance has been permanently raised; what was once a standard day's work now feels insufficient. This is the double-edged sword of AI-driven efficiency: the same tool that removes friction can create a new anxiety, a feeling of perpetually falling short of a new, machine-augmented standard. This recalibration of expectations has significant implications for developer well-being, as the gap between a normal and an AI-assisted day can become a major source of stress and a potential pathway to burnout.

The Contested Evidence on AI-Driven Speed

While the participants experienced a profound shift in their sense of speed and flow, these qualitative findings must be placed within the contested landscape of quantitative research. The narrative of AI-driven productivity is not straightforward. This complexity is reflected in the DORA annual reports; the 2024 edition found AI adoption correlated with a decrease in throughput, a finding reversed in the 2025 report, which showed a positive relationship (DeBellis et al., 2024, 2025). Furthermore, a landmark study by Brynjolfsson et al. (2025), often cited for its 14% productivity boost, did not study developers but rather customer support agents. The gains were almost entirely among novice workers, with minimal impact on experienced staff.

This nuance is critical, as this study's sample consists exclusively of senior developers. The domain-specific literature here presents an even more contradictory picture. A randomised controlled trial by Becker et al. (2025), involving experienced developers on complex codebases, found that access to modern AI tools actually *increased* task completion time by 19%.

The experiences of David helps explain this friction in the quantitative data. Leveraging AI in a high-stakes context introduces new cognitive overhead. As David, a platform engineer, learned, blindly trusting an AI's confident output can lead to significant reputational damage:

> I typed into ChatGPT, I got an answer, which was very confidently stated... I went into the AWS console... and that feature did not exist and I would have ended up, you know, very embarrassed.

This experience forced David to develop a rigorous, time-consuming verification workflow. This suggests a potential explanation for the findings of Becker et al. (2025): for experienced developers, the time saved by AI may be offset by the time required to critically evaluate and safely integrate its output. The perceived order of magnitude speed increase may be potent in the initial, generative phases of a task, but this benefit is likely offset by the cognitive costs of verification and risk mitigation.

Ultimately, this theme documents a fundamental shift in how developers conceptualise productivity. AI integration is pushing them to abandon crude metrics of volume in favour of more meaningful, outcome-driven concepts of value, impact, and sustainable flow. This is not a simple transition but a complex negotiation fraught with new pressures and paradoxes. The clear implication for management is that traditional productivity metrics, such as lines of code or story points, are becoming obsolete and may even incentivise the wrong behaviours. This profound re-evaluation of a developer's contribution raises a critical question that leads to the next theme: If the very definition of their work is changing, what is the impact on their sense of agency, control, and professional self-worth?

The Double-Edged Tool of Agency

This theme examines the contradictory effects of AI on developer agency, exploring the central tension between profound empowerment and significant anxieties about de-skilling and risk. The double-edged nature of this relationship emerged clearly from the participants' accounts. For George, AI was a force multiplier that enabled him to single-handedly accomplish a project that would have otherwise required a full team:

> I ended up building it on my own by using AI to fill in the gaps. So it was my business analyst, it was my developer.

Yet, for others, this empowerment came at a direct cost to their fundamental skills. As Ben explained, his reliance on AI has eroded his basic knowledge:

> I've actually started forgetting syntax for for languages that I know... I kind of forgotten how to do um, even like kind of basic things like a for loop.

This theme, therefore, focuses on how developers navigate the competing psychological forces of empowerment and erosion to maintain their sense of control.

Empowerment and Skill Enhancement

A consistent narrative among participants was the profound empowerment they felt when using AI as a learning and development partner. This was particularly transformative, as the interactive, non-linear nature of AI-driven exploration offered a powerful alternative to traditional learning models. One participant, for instance, described how AI helped them to “re-restructure” their own learning, putting them back in control of a process that had been a source of difficulty.

> so I have ADHD... I did terrible in school... through these tools, um they have helped me to kind of re-restructure like my own learning. Um so it’s put the control back into my own hands. (Ben)

This experience connects directly to Self-Determination Theory, which holds that autonomy,the feeling of control over one’s actions is a fundamental psychological need for well-being and motivation (Deci & Ryan, 2000). Ben’s description of his past struggles suggests an educational history that failed to support this need. The AI, in contrast, becomes a tool for reclaiming this control. His phrase “put the control back into my own hands” directly articulates this restored sense of autonomy. By allowing him to “re-restructure” his learning to fit his cognitive style, the AI creates an environment that satisfies this core psychological need and fosters a renewed sense of empowerment.

This empowerment also extended to increased confidence in tackling new challenges. Another participant, new to the Python language, initially felt like a fraud but found that using AI as a constant dialogic partner accelerated their learning.

> I suppose when I first started using it on the Python stuff, I felt a little bit fraudulent... I also fire up, uh, Claude so that I can constantly ask questions. (Leo)

Vygotsky's theory of the Zone of Proximal Development (ZPD) and scaffolding helps explain this experience (Al-Hamadi & Yousif, 2025). The ZPD is the gap between what a learner can do alone and what they can achieve with guidance. Leo's feeling of being "fraudulent" suggests he was operating in this zone. The AI acts as a *more knowledgeable other*, providing the scaffolding to bridge the gap. By having an on-demand partner to "constantly ask questions," he could safely navigate the ZPD, building both confidence and competence. This just-in-time, dialogic support is the hallmark of effective scaffolding, enabling the learner to internalise new knowledge and grow.

Risk, De-skilling and Skill Atrophy

A pervasive anxiety about skill atrophy counterbalanced this sense of empowerment. This concern reflects the risk of accumulating cognitive debt, where the short-term cognitive ease from a tool comes at the long-term cost of deep, effortful thinking and skill retention (Stadler et al., 2024). This tension was a central theme in participants' experiences, and recent empirical research strongly supports it. Studies have found that while self-confidence is associated with more critical thinking in AI-assisted tasks, confidence in the GenAI tool itself is associated with less critical thinking (Lee et al., 2025). These anxieties also mirror findings that using an LLM is correlated with impaired memory of the produced text and a lower sense of ownership (Kosmyna et al., 2025).

This phenomenon was clear in the account from Ben, who admitted to forgetting basic syntax because the AI always generated it for them:

> I've actually started forgetting syntax for for languages that I know... I kind of forgotten how to do um, even like kind of basic things like a for loop.

This experience exemplifies the cognitive principle of *use it or lose it*, where procedural knowledge becomes less accessible if not regularly retrieved (Faust et al., 2021). Offloading procedural knowledge, like the syntax for a "for loop," means a developer no longer exercises the cognitive pathways for retrieving it. This creates a tension between a short-term productivity gain and long-term skill atrophy. The immediate benefit is a reduction in cognitive load, but the potential cost is an erosion of fundamental skills, creating a dependency that could leave a developer vulnerable. The developer, therefore, engages in a constant, implicit cost-benefit analysis between offloading and de-skilling.

This anxiety extends beyond syntax to a developer's core sense of value, manifesting as what some in public discourse are calling *AI Impostor Syndrome* (Nosta, 2025). Ivan described a profound dissatisfaction after completing a complex task with a single instruction to an AI, asking, "what had I added as a value." This mirrors Leo's experience of feeling "a little bit fraudulent" when first using AI in an unfamiliar language.

> I developed a no-code agent. I was not happy, honestly, because I wrote one line of instruction and it worked. And I thought, what had I added as a value, right? Uh, so it made me feel low. (Ivan)

This dissatisfaction is a direct expression of cognitive debt. The outcome was achieved without the mental effort that, according to theories of intrinsic motivation, is essential for developing a sense of competence and mastery (Deci & Ryan, 2000). The AI, in effect, paid the cognitive price, leaving the developer with the result but not the feeling of accomplishment. This aligns with the psychological principle of Effort Justification, which holds that people value outcomes more when they have put more effort into achieving them (Festinger, 1957). Recent research adds a crucial nuance, demonstrating this effect occurs primarily when individuals have a high degree of perceived control over the outcome (Harmon-Jones et al., 2024). Ivan's question, "what had I added as a value," reveals a crisis of contribution where he felt responsible for the outcome (high control) but exerted no meaningful effort. This highlights a paradox: while AI can dramatically reduce effort, this very reduction can strip an activity of its intrinsic value, undermining a developer's sense of competence.

A perceived learning paradox further compounded this anxiety. Grace described being caught between accelerated learning and an ever-expanding knowledge gap.

> it both accelerates my, um, growth in learning but also, like, accelerates the huge gap in my knowledge... I feel like the gap of things that I feel like I need to know now is getting wider and wider um because of because of AI.

This paradox of accelerated learning leading to a heightened sense of inadequacy exemplifies cognitive overload (Lahlou, 2025; Sweller, 2011). By making vast amounts of information instantly accessible, the AI simultaneously accelerates learning while revealing the sheer scale of what remains unknown. The result is a feeling of being perpetually behind. The “huge gap in my knowledge” is not a static deficit but a dynamic, expanding chasm. This suggests AI’s broader psychological impact may be to create a new form of intellectual anxiety: a constant awareness of the vast, expanding universe of information just beyond one’s grasp.

Some participants, however, contextualised this growing knowledge gap as a natural consequence of a new, higher level of abstraction in engineering. Ben, for instance, framed this evolution not as a simple loss of skill but as the next logical step in programming history:

> I’m kind of seeing it as like yet another higher level higher level programming language abstraction like we keep stepping up in languages over time all the way from, you know, COBOL and all that. So it’s seems like that, maybe the next iteration of that kind of thing.

Ben’s metaphor provides a technical grounding for the anxiety Grace describes. By framing AI as another layer of abstraction, he suggests the skills being "lost" are becoming the new fundamentals, just as modern developers no longer need to manage memory manually. While this signals progress, it also explains the anxiety: the very definition of foundational knowledge is shifting, widening the gap between what one knows and what one feels they need to know.

This anxiety was not universal. Charlie offered a compelling counter-narrative, arguing that AI amplifies the need for critical thinking rather than diminishing it.

> I think that's probably the opposite of what a lot of people will tell you is that they're worried about it removing critical thinking... I actually am taking the perspective that I I want to know what I'm asking it to do more innately than I did before.

Charlie's perspective reframes the AI-developer relationship as one that demands a higher level of human expertise. Where others see a tool that replaces knowledge, Charlie sees one that demands deeper, more intuitive mastery. To effectively command an AI on complex tasks, a superficial understanding is insufficient; one must know the subject matter *more innately.* This metacognitive stance is not just about knowing facts but about understanding underlying principles so thoroughly that one can critically evaluate, guide, and correct the AI's output. In this model, the AI elevates the developer into the role of a critical director, whose value lies not in rote knowledge but in the deep expertise required to wield the tool effectively.

The adoption of generative AI, then, creates a profound psychological tension. A developer's agency is not a simple story of enhancement or erosion but a continuous negotiation between the tool's empowering possibilities and the risks of de-skilling and dependency. This double-edged experience is the core psychological reality of the shift from coder to conductor. The critical implication for training is the need to cultivate metacognitive skills; critical evaluation, prompt engineering, strategic delegation, to ensure AI serves as an amplifier, not a crutch. To conduct an orchestra of AI agents is to wield immense power, but it is also to bear the responsibility of critical oversight and the anxiety that one's own skills may be fading into the background.

The relationship between AI and developer agency creates a significant theoretical conflict within the framework of Self-Determination Theory. While AI clearly supports the psychological need for *autonomy* by giving developers greater control over their learning and problem-solving, it simultaneously threatens their need for *competence.* This threat manifests through anxieties about skill atrophy and the hollow feeling of accomplishment that comes from effortless achievement. This tension suggests a potential hierarchy of needs in this context, where the immediate satisfaction of autonomy may be prioritised over the longer-term, and perhaps more fundamental, need for competence. The unresolved nature of this conflict has profound implications for long-term well-being; if the daily use of a primary work tool consistently undermines a core psychological need, it could foster a sustained sense of professional inadequacy, potentially leading to burnout even as surface-level productivity appears to increase.

Conclusion

The integration of AI into senior software and platform engineering developers is shaped by a core psychological tension: AI acts as a *double-edged tool* for developer agency. This central conflict—the balance between empowerment and the anxiety of de-skilling, drives a fundamental redefinition of the developer's role. This tension directly reshapes the other core themes. The *re-architecting of developer focus* is a behavioural adaptation, as developers offload commoditised skills to elevate their cognitive work to the safer ground of high-level strategy. In turn, this shift necessitates a *shifting definition of productivity*, a psychological re-evaluation where professional value is deliberately relocated from automatable outputs to uniquely human outcomes.

This study contributes to the psychological literature on knowledge work in several ways. The empowerment described by participants validates Self-Determination Theory, showing AI can satisfy the need for autonomy. However, the dissatisfaction with *easy* achievements lends weight to the theory of Effort Justification, suggesting that for experts, cognitive struggle is vital for job satisfaction. This creates a potential theoretical tension, where the autonomy-supportive nature of AI may paradoxically undermine the need for competence by conflicting with the principle of effort justification. Finally, the study extends Cognitive Load Theory, illustrating how developers use AI to offload extraneous load, while also highlighting the cognitive costs of verification and trust calibration.

While much of the initial industry hype focused on productivity gains from generating more code, a more nuanced reality has emerged. These findings align with recent industry-level research, which shows that significant value is found not by automating the fraction of a developer's time spent coding, but in augmenting the majority of their time spent on complex, collaborative tasks like architectural discussions, code reviews, and planning (DeBellis et al., 2025). This represents an evolution in AI adoption, moving beyond an initial *J-curve* of learning towards mature usage patterns that support the holistic practice of software engineering. This implies that training and management should shift focus from maximising coding output to cultivating the critical metacognitive skills required to effectively direct and validate AI-generated work. This trend is likely to continue as both AI capabilities and developers' skills in wielding them improve.

The primary strength of this study is its rich, qualitative exploration of the lived experiences of senior, professional users. However, the small, purposive sample means the findings are transferable, not generalisable. This study also represents a snapshot in time; the experiences captured here will undoubtedly shift as AI capabilities, particularly in the agentic space, become more advanced.

Several crucial avenues for future research emerge from these findings. A longitudinal study is essential to track how developers' skills and identities evolve as AI technology matures. The potent but subjective anxieties around skill atrophy require a large-scale quantitative study to test the correlations between AI usage and validated scales for skill decay, job satisfaction, and burnout. As AI becomes more autonomous, research must investigate its impact on team-level dynamics, such as the development of shared mental models. Finally, while this study focused on senior engineers, future work must investigate the novice-expert divide to understand how these tools affect skill acquisition for junior developers.

These findings also point to a deeper, more philosophical challenge. The shift towards developers using natural language to direct AI runs headfirst into a critical, unresolved tension. Some software engineering experts argue that the very nature of programming languages is their hard-won precision, a quality fundamentally at odds with the inherent ambiguity of human language. Programming languages are designed to be simple and unambiguous; natural languages are anything but. What does it mean for the future of the craft if the primary interface for creating rigorously precise instructions is, by its very nature, imprecise? This suggests a critical challenge: how can the industry reconcile the expressive power of natural language with the absolute need for deterministic, reliable code? It is a question that this research brings into sharp focus, but one the field has yet to answer.

The story of the developer in the age of AI reveals a clear truth: software development is not dying, it is changing. The transition is not one of simple replacement but a profound, often challenging, renegotiation of professional identity. These tools are not just making the old work faster; they are forcing the creation of new work, demanding a shift from the tangible craft of writing code to the intangible art of conducting systems. Successfully navigating this transition requires more than just technical skill. It demands metacognitive awareness, a comfort with ambiguity, and a renewed focus on the uniquely human capacity for strategic thought. Ultimately, the future of software engineering will be defined not by the power of the AI, but by the wisdom developers use to wield it.

In reflecting on the central metaphors used in this analysis, it becomes clear that while the shift from coder to conductor metaphor captures the developer's strategic focus, it is the cognitive partner metaphor more accurately reflects their lived experience. The former implies a hierarchy, but the latter speaks to the nuanced, collaborative, and peer-like relationship that defines the day-to-day reality of AI-augmented development. This partnership, however, extends beyond the individual. The findings of this thesis suggest that generative AI acts as both a mirror and a multiplier for an organisation's existing culture. The critical question for leadership, therefore, is not whether AI, but whether the organisation is ready for what the tool will reveal and reflect. Success is not rooted in the technology itself, but in the individual and team capabilities that the AI will inevitably amplify, for better or for worse.

References

Al-Hamadi, D., & Yousif, J. H. (2025). Artificial intelligence revolution for enhancing modern education using zone of proximal development approach. *Applied Computing Journal*. https://api.semanticscholar.org/CorpusID:277479920

Becker, J., Rush, N., Barnes, E., & Rein, D. (2025). Measuring the impact of early-2025 ai on experienced open-source developer productivity. https://arxiv.org/abs/2507.09089

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

Brynjolfsson, E. (1993). The productivity paradox of information technology. *Commun. ACM*, *36*, 66–77. https://api.semanticscholar.org/CorpusID:15074120

Brynjolfsson, E., Li, D., & Raymond, L. (2025). Generative AI at Work. *The Quarterly Journal of Economics*, *140*(2), 889–942. https://doi.org/10.1093/qje/qjae044

Brynjolfsson, E., Rock, D., & Syverson, C. (2020, February). Unpacking the ai–productivity paradox. In *How ai is transforming the organization*. The MIT Press. https://doi.org/10.7551/mitpress/12588.003.0014

Creswell, J. W., & Creswell, J. D. (2017). *Research design: Qualitative, quantitative, and mixed methods approaches* (5th). Sage publications.

Dăniloaia, D., & Turturean, E. (2024). Knowledge Workers and the Rise of Artificial Intelligence: Navigating New Challenges. *SEA: Practical Application of Science*, *12*(35), 111–121. https://doi.org/10.70147/s35111121

DeBellis, D., Storer, K., Harvey, N., Beane, M., Edwards, R., Fraser, E., Good, B., Kalliamvakou, E., Kim, G., Maxwell, E., D'Angelo, S., Inman, S., Murillo, A., & Villalba, D. (2025, September). *Dora 2025 state of ai-assisted software development report* (tech. rep.). Google. https://cloud.google.com/resources/content/2025-dora-ai-assisted-software-development-report?e=48754805

DeBellis, D., Storer, K. M., Lewis, A., Good, B., Villalba, D., Maxwell, E., Castillo, K., Irvine, M., & Harvey, N. (2024). *2024 accelerate state of devops* (tech. rep.). Google

LLC. https://dora.dev/research/2024/dora-report/2024-dora-accelerate-state-of-devops-report.pdf

Debellis, D., Storer, K. M., Villalba, D., Harvey, N., D'Angelo, S., & Brown, A. (2025). DORA: Impact of Generative AI in Software Development.

Deci, E. L., & Ryan, R. M. (2000). The" what" and" why" of goal pursuits: Human needs and the self-determination of behavior. *Psychological inquiry*, *11*(4), 227–268.

Edmondson, A. (1999). Psychological safety and learning behavior in work teams. *Administrative Science Quarterly*, *44*(2), 350–383. https://doi.org/10.2307/2666999

Fan, Q., Wider, W., & Chan, C. K. (2023). The brief introduction to organizational citizenship behaviors and counterproductive work behaviors: A literature review. *Frontiers in Psychology*, *14*. https://api.semanticscholar.org/CorpusID:261849188

Faust, T. E., Gunner, G., & Schafer, D. P. (2021). Mechanisms governing activity-dependent synaptic pruning in the developing mammalian cns. *Nature Reviews Neuroscience*, *22*(11), 657–673.

Felix, O., & Eboka, I. Z. (2024). Citizenship behaviour and organizational performance: A review of extant literature. *International Journal of Management & Entrepreneurship Research*. https://api.semanticscholar.org/CorpusID:266850683

Festinger, L. (1957). A theory of cognitive dissonance. redwood city: Stanford university press. *Organization Science*, *23*(4), 1077–1099.

Flavell, J. H. (1979). Metacognition and cognitive monitoring: A new area of cognitive–developmental inquiry. *American Psychologist*, *34*(10), 906–911. https://doi.org/10.1037/0003-066X.34.10.906

Forsgren, N., Humble, J., & Kim, G. (2018). *Accelerate: The science of lean software and devops: Building and scaling high performing technology organizations*. IT Revolution Press.

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

Fowler, M. (2003). Who Needs an Architect? *IEEE Software*, *20*(5), 11–13. https://doi.org/10.1109/MS.2003.1231144

Google Cloud. (n.d.). Dora community of practice [Accessed: 2025-08-19].

Görmez, M. K., Yılmaz, M., & Clarke, P. M. (2024). Large language models for software engineering: A systematic mapping study. In M. Yilmaz, P. Clarke, A. Riel, R. Messnarz, C. Greiner, & T. Peisl (Eds.), *Systems, software and services process improvement* (pp. 64–79). Springer Nature Switzerland.

Graziotin, D., Wang, X., & Abrahamsson, P. (2015). Do feelings matter? On the correlation of affects and the self-assessed productivity in software engineering. *Journal of Software: Evolution and Process*, *27*, 467–487. https://doi.org/10.1002/smr.1673

Grinschgl, S., & Neubauer, A. C. (2022). Supporting cognition with modern technology: Distributed cognition today and in an ai-enhanced future. *Frontiers in Artificial Intelligence*, *5*. https://api.semanticscholar.org/CorpusID:250497580

Guest, G., Bunce, A., & Johnson, L. (2006). How Many Interviews Are Enough? *Field Methods*, *18*(1), 59–82. https://doi.org/10.1177/1525822X05279903

Harmon-Jones, E., Matis, S., Angus, D. J., & Harmon-Jones, C. (2024). Does effort increase or decrease reward valuation? considerations from cognitive dissonance theory. *Psychophysiology*, e14536. https://api.semanticscholar.org/CorpusID:267522374

Hollan, J., Hutchins, E. L., & Kirsh, D. (2000). Distributed cognition. *ACM Transactions on Computer-Human Interaction (TOCHI)*, *7*, 174–196. https://api.semanticscholar.org/CorpusID:1490533

Hou, X., Zhao, Y., Liu, Y., Yang, Z., Wang, K., Li, L., Luo, X., Lo, D., Grundy, J., & Wang, H. (2024). Large language models for software engineering: A systematic literature review. *ACM Transactions on Software Engineering and Methodology*, *33*(8). https://doi.org/10.1145/3695988

Hunt, A., & Thomas, D. (1999). *The pragmatic programmer: From journeyman to master*. Addison-Wesley Professional.

Jaspan, C., & Green, C. (2023). Defining, Measuring, and Managing Technical Debt. *IEEE Software*, *40*(3), 15–19. https://doi.org/10.1109/ms.2023.3242137

Kosmyna, N., Hauptmann, E., Yuan, Y. T., Situ, J., Liao, X.-H., Beresnitzky, A. V., Braunstein, I., & Maes, P. (2025). Your brain on chatgpt: Accumulation of cognitive debt when using an ai assistant for essay writing task. *arXiv preprint arXiv:2506.08872*.

Lahlou, S. (2025). Mitigating societal cognitive overload in the age of ai: Challenges and directions. *ArXiv*, *abs/2504.19990*. https://api.semanticscholar.org/CorpusID:278165081

Lee, H.-P. (, Sarkar, A., Tankelevitch, L., Drosos, I., Rintel, S., Banks, R., & Wilson, N. (2025). The impact of generative ai on critical thinking: Self-reported reductions in cognitive effort and confidence effects from a survey of knowledge workers. *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. https://doi.org/10.1145/3706598.3713778

Meyer, A. N., Fritz, T., Murphy, G. C., & Zimmermann, T. (2014). Software developers' perceptions of productivity. *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 19–29. https://doi.org/10.1145/2635868.2635892

Meyer, A. N., Murphy, G. C., Zimmermann, T., & Fritz, T. (2017). Design Recommendations for Self-Monitoring in the Workplace. *Proceedings of the ACM on Human-Computer Interaction*, *1*(CSCW), 1–24. https://doi.org/10.1145/3134714

Nakamura, J., & Csikszentmihalyi, M. (2014). The concept of flow. In *Flow and the foundations of positive psychology: The collected works of mihaly csikszentmihalyi* (pp. 239–263). Springer.

Nguyen-Duc, A., Cabrero-Daniel, B., Przybylek, A., Arora, C., Khanna, D., Herda, T., Rafiq, U., Melegati, J., Guerra, E., Kemell, K.-K., et al. (2023). Generative artificial intelligence for software engineering - a research agenda. *Software: Practice and Experience*.

Noda, A., Storey, M.-A., Forsgren, N., & Greiler, M. (2023). Devex: What actually drives productivity: The developer-centric approach to measuring and improving productivity. *Queue*, *21*(2), 35–53.

Nosta, J. (2025). Ai and the new impostor syndrome [Accessed: 2025-09-22]. *Psychology Today*. https://www.psychologytoday.com/ca/blog/the-digital-self/202503/ai-and-the-new-impostor-syndrome

Organ, D. W. (1988). Organizational citizenship behavior: The good soldier syndrome. *Administrative Science Quarterly*, *33*, 331. https://api.semanticscholar.org/CorpusID:149987707

Ralph, P., Baltes, S., Adisaputri, G., Torkar, R., Kovalenko, V., Kalinowski, M., Novielli, N., Yoo, S., Devroey, X., Tan, X., Zhou, M., Turhan, B., Hoda, R., Hata, H., Robles, G., Milani Fard, A., & Alkadhi, R. (2020). Pandemic programming. *Empirical Software Engineering*, *25*(6). https://doi.org/10.1007/s10664-020-09875-y

Shihab, M. I. H., Hundhausen, C., Tariq, A., Haque, S., Qiao, Y., & Mulanda, B. (2025). The Effects of GitHub Copilot on Computing Students' Programming Effectiveness, Efficiency, and Processes in Brownfield Programming Tasks.

Stadler, M., Bannert, M., & Sailer, M. (2024). Cognitive ease at a cost: LLMs reduce mental effort but compromise depth in student scientific inquiry. *Computers in Human Behavior*, *160*, 108386.

Storer, K. M. (2025, March). How gen ai affects the value of development work [Accessed: 2025-09-22]. https://dora.dev/research/ai/value-of-development-work/

Storey, M.-A., Zimmermann, T., Bird, C., Czerwonka, J., Murphy, B., & Kalliamvakou, E. (2021). Towards a Theory of Software Developer Job Satisfaction and Perceived Productivity. *IEEE Transactions on Software Engineering*, *47*(10), 2125–2142. https://doi.org/10.1109/tse.2019.2944354

Storey, V. C., Yue, W. T., Zhao, J. L., & Lukyanenko, R. (2025). Generative Artificial Intelligence: Evolving Technology, Growing Societal Impact, and Opportunities for Information Systems Research. *Information Systems Frontiers*, 1–22. https://doi.org/10.1007/s10796-025-10581-7

Sweller, J. (2011). Cognitive load theory. In *Psychology of learning and motivation* (pp. 37–76, Vol. 55). Elsevier.

Sweller, J. (2024). Cognitive load theory and individual differences. *Learning and Individual Differences*. https://api.semanticscholar.org/CorpusID:267955574

Sweller, J., van Merriënboer, J. J. G., & Paas, F. (2019). Cognitive architecture and instructional design: 20 years later. *Educational Psychology Review*, *31*, 261–292. https://api.semanticscholar.org/CorpusID:150705146

Wiltshire, G., & Ronkainen, N. (2021). A realist approach to thematic analysis: Making sense of qualitative data through experiential, inferential and dispositional themes. *Journal of Critical Realism*, *20*(2), 159–180. https://doi.org/10.1080/14767430.2021.1894909

Zelenski, J. M., Murphy, S. A., & Jenkins, D. A. (2008). The Happy-Productive Worker Thesis Revisited. *Journal of Happiness Studies*, *9*(4), 521–537. https://doi.org/10.1007/s10902-008-9087-4