

Agency, Expertise, and the Limits of Augmentation in LLM-Assisted Programming Workflows

Gabriel Hughes

UCL Interaction Centre
University College London
gabriel.hughes@ucl.ac.uk

Yvonne Rogers

UCL Interaction Centre
University College London
y.rogers@ucl.ac.uk

Abstract

Large language models (LLMs) are being rapidly adopted for complex knowledge work, yet little is known about how end users are adapting these tools to their needs, why their customised workflows take the shape they do, and what those adaptations reveal about unmet needs in human-AI interaction. This paper reports findings from seventeen semi-structured interviews with knowledge workers, primarily software engineers, investigating how complementary human-AI relationships develop in practice, and whether users perceive LLMs to be genuinely augmentative. Expert software engineers had developed workflows that preserved their agency over the parts of their work they valued. These were characterised by meticulous upfront specification and iterative agent configuration, which kept strategic and creative decisions in human hands. Even within these expert workflows, however, participants described new cognitive burdens, particularly regarding context management, output evaluation, and maintaining strategic oversight, all of which current tool design does little to mitigate by default. These findings suggest that complementarity is not a default property of LLM-based tools but something users must deliberately construct, often in ways those tools do not naturally support.

1. Introduction

Large language models (LLMs), which power popular chatbots like ChatGPT and Claude, have over a billion users worldwide, yet their design has not evolved significantly past chatbot-based paradigms since they entered mainstream usage. Prompting a conversational interface via natural language remains the standard for engaging with AI, despite arguments that it is a poor substitute for genuine natural-language interaction (Kraljic & Lahav, 2024; Meredith Ringel Morris, 2024). Even interactions with LLM-powered coding tools are organised as linear chat conversations, whose structure, combined with the quantity of output LLMs create, produces significant information processing challenges for complex knowledge work like software development (Alami & Ernst, 2025; Suh et al., 2023; Tang et al., 2024). While there is substantial evidence these tools can genuinely augment complex knowledge work (Dell’Acqua et al., 2023; Mohamed et al., 2026), the ways in which they are changing the nature of work can come with significant cognitive costs (Ranganathan & Ye, 2026; Simkute et al., 2024; Tang et al., 2024). What remains under-explored is how practitioners are adapting to this tension in potential, and what their adaptations reveal about the conditions under which genuine augmentation becomes possible.

Research on end-user development has long shown that people with domain expertise will adapt tools to meet their needs when default designs fall short (Ko et al., 2011). LLM users appear to be doing exactly this. Sarkar and Drosos (2025) conducted one of the first ethnographic studies of ‘vibe coding’ (a popularised term for LLM-assisted programming), in which they analysed over eight hours of recorded expert coding sessions. They found that experienced developers are adapting LLM-based tools in various ways that both preserve their agency over their work and mitigate LLMs’ shortcomings, notably the tendency for their quality to deteriorate as their context grows.

Each end-user adaptation reveals a potential new UX issue and, therefore, a design opportunity. Observational research like Sarkar and Drosos (2025) is therefore beneficial for understanding the details of an end-user-developed practice, but it leaves open questions regarding why those adaptations emerged and how attitudes and practices have developed over time. To that end, this research sought to answer the following research questions through seventeen semi-structured interviews with knowledge workers, primarily software engineers:

- **RQ1.** How do knowledge workers’ custom adaptations of AI tools for complex workflows reveal unmet needs in human-AI collaboration?
- **RQ2.** What is the perceived potential for LLMs to augment human abilities?

The data suggest that a genuinely complementary relationship with LLM-based tools is not something the tools provide by default but something users must construct. The most experienced software engineers had developed sophisticated workflows for preserving agency over the parts of their work they valued while offloading less rewarding jobs, but these came with significant cognitive costs. Those costs, and the strategies needed to manage them, are least accessible to less experienced users, who lack both the technical skill to evaluate LLM-authored code output reliably and the metacognitive resources to manage effective delegation.

The primary contribution this research makes is to provide empirical support for the importance of promoting user agency and the enablement of intrinsically rewarding, or meaningful, work as key components of a more human-centred, or complementary, relationship with AI. While the traditional view of complementarity, centering on balancing the relative strengths and weaknesses of humans and technology, often with the goal of quantifying complementarity (Steyvers et al., 2022; Wilder et al., 2020), is still crucial, these interviews suggest that whether a relationship with AI is experienced as genuinely augmentative depends less on the balance of capabilities than on whether it allows the user to protect and advance the work they value. This paper treats complementarity in this richer sense as the condition under which LLM use is experienced as genuinely augmentative; the two concepts are distinct, but the data support their connection inductively. This perspective tends to be missing from much of the literature and technological development around LLMs, but is an important framing for future design work.

2. Methodology

2.1. Participants & Recruitment

Participants (N=17) were professional knowledge workers, mostly software engineers, with the exception of one journalist, one philosophy graduate, and a design educator. Table 1 shows details of each participant’s profession and LLM expertise. Recruitment mostly followed a community-driven approach. Users of the open-source productivity software Obsidian were identified as a strong target, since a significant number already share their workflows online (e.g., on Discord, Reddit, and YouTube), and many were already using LLMs in conjunction with Obsidian. This initial group of Obsidian users contributed three participants. Two others were found on YouTube or X (f.k.a. Twitter) by searching for ‘agentic coding’ or ‘Claude Code’ (which was at the time, and still is, one of the most widely used LLM-based coding tools). These were developers who wrote about or made videos about their workflows. Nine participants were connections of other participants or UCL’s network and did not publish content about their work online.

2.2. Procedure

Semi-structured interviews were chosen as the research method as a primary focus of the study was to uncover participant attitudes towards various aspects of interacting with AI, as well as their motivations for using AI tools in specific ways. These interviews lasted approximately 60 minutes each and were adapted to each participant based on a discussion guide. The discussion guide, and subsequent reflexive thematic analysis (Braun & Clarke, 2019), were largely structured around **RQ1** and **RQ2** and focused on how and why participants’ LLM-assisted workflows took shape the way they did.

2.3. Analysis

The data consisted primarily of automated transcriptions via Microsoft Teams. Thirteen out of seventeen participants gave informed consent to have their interviews recorded. Data from the other four participants consisted of detailed notes taken by the interviewer. Data was analysed using reflexive thematic analysis (Braun & Clarke, 2019). This approach aims to understand implications emerging from qualitative data by identifying patterns and themes present across multiple participants. Themes were defined

PID	Profession	LLM Expertise*
P1	QA Engineer + Writer	Expert
P2	Software Engineer	Expert
P3	Software Engineer	Expert
P4	Software Engineer	Expert
P5	Journalist	Intermediate
P6	Higher Education	Proficient
P7	Software Engineer	Expert
P8	Philosophy Graduate	Intermediate
P9	Software Engineer	Expert
P10	Software Engineer	Expert
P11	Software Engineer	Proficient
P12	Software Engineer	Expert
P13	Software Engineer	Expert
P14	CS Professor	Expert
P15	CS PhD Student	Proficient
P16	ML Engineer	Expert
P17	Software Engineer	Expert

*Table 1 – This table shows each participant’s profession as well as their relative expertise using LLMs. *Note: Here ‘Expert’ is used solely to distinguish engineers who have experience using agents to execute tasks autonomously, some of whom utilise multi-agent architectures. ‘Proficient’ indicates domain expertise but limited experience with agentic workflows, i.e., these users may have experience with contextually grounded tools like Claude Cowork, but they primarily interface with LLMs via a chat interface. ‘Intermediate’ refers to users who primarily interface with LLMs via a chat interface.*

iteratively based on the research questions summarised above. A private, local LLM (deepseek-r1 via Ollama) was used as a reflexive prompt to surface alternative readings: the LLM was prompted, for each transcript, to identify points of deviation from the themes that could point to additional ones, or observations that could add nuance to or alter themes. For each excerpt surfaced, the researcher assessed whether it represented a genuine deviation from a theme or was consistent with it and revised or retained accordingly.

3. Findings

The strategies that software engineers have developed, individually and as a community, for working with LLMs and LLM-powered agents have the potential to protect their agency over the aspects of their work they find meaningful or rewarding (such as the strategic or creative) while effectively offloading rote or repetitive work. This same offloading, combined with the speed at which LLMs produce code, also opens up the potential to glean even more reward from ‘giving life’ to new ideas at much greater speed. However, despite that positive potential, the interviews revealed how LLMs also heavily shift the nature of software engineering work from execution towards management or orchestration (or from production to evaluation), and this shift introduces new cognitive burdens that developers are still learning how to manage. Thus, the augmentative capacities of LLMs are counterbalanced by their potential to produce burnout. These findings are broken down in greater detail across three sections:

- §3.1: Software Engineers’ Practices for Working with LLMs,
- §3.2: The Perceived Potential for Augmentation,
- §3.3: Tradeoffs between Augmentation and Burnout

3.1. Software Engineers' Practices for Working with LLMs

The most experienced software engineers in this study had developed ways of working with LLM-based agents that strived to establish complementary relationships in which the human's and the agent's roles were roughly divided between strategic decision-making and execution (e.g., code generation and debugging), respectively.

These practices were not derived from any explicit framework; varied theories still abound online regarding what exactly constitutes best practice for working with agentic coding tools. Participants instead described arriving at them through a combination of trial and error, learning through their own practice, and from tweets, blog posts, and videos shared by peers. Importantly, figuring out which tactics work for a specific context required sustained, deliberate effort that LLM-based systems don't necessarily support by default. At a high level, these practices are encapsulated in a few distinct but overlapping behaviours.

The first involves investing significant effort in specification and planning before any code is generated. This isn't a new practice for engineers, but its purpose has changed considerably in this new context. It is now as much about preserving agency as it is about defining the vision for a product. It is also a direct response to the default behaviour of LLMs, which is highly execution-driven, often at the expense of clarity and comprehensiveness (Laban et al., 2025). As P14 observed, "These models all want to go to code as soon as possible. Sooner or later it asks, "Should we code this?" Eventually I'll say, "OK, I think we have enough." The key word there is 'eventually.' By the time they hand anything to the agent, 'I've already gone through the major questions in the design phase, so I know what's going to happen.' Similarly, P9 described a process of extended collaborative planning with the LLM before allowing it to write a line of code: 'we are going to keep talking till we get to the level of specificity where it's appropriate for the handoff.' This front-loading keeps strategic and architectural decisions in human hands, while delegating the translation of those decisions into code to the model.

The second practice is the use of persistent agent configuration files (often referred to as AGENTS.md, CLAUDE.md, or SKILL.md files) that encode behavioural instructions, coding standards, and context that would otherwise need to be re-specified across sessions. P7 described their value: they can 'give very sloppy instructions' in the heat of a creative moment, knowing that the clarity encoded in the configuration file will compensate. Their key insight was that writing good agent instructions requires a different cognitive state than actively building something – "those are usually totally different sessions for me" – and that capturing instructions during moments of reflective clarity is what makes the system usable under less ideal conditions. This practice, similar to upfront specification, also appears to be a response to another limitation of LLMs: statelessness, or lack of actual memory (Dong et al., 2025; N. F. Liu et al., 2024). LLMs' statelessness, coupled with their frequent handling of large contexts, often forces the user to act as a memory store on their behalf, introducing new cognitive burdens (discussed more below).

The third practice is deliberate human-in-the-loop checkpointing, or structuring agent tasks so that humans review and approve frequently before an agent executes. This is enforced at varying levels and depends on the detail of a project's upfront specification work; however, some level of checkpointing is critical as a response to another limitation of LLMs, which is their tendency to 'drift.' Drifting really occurs because of their aforementioned statelessness; in other words, because they don't have memory, LLMs constantly need to summarise previous decisions and turns of conversation, and this compression (particularly the longer a session lasts) and resulting loss of detail can cause the LLM to 'drift' from its initial focused context (Laban et al., 2025; N. F. Liu et al., 2024). P10 describes responding to this behaviour: 'I usually start with "Let's make a plan," and make it actually write down a plan file and execute it step by step and confirm the steps with me as it goes along. That way it usually doesn't really drift.'

A practitioner perspective on why this matters came from P12, a prototyping architect who works with engineering teams trying to adopt AI. They described the problem with ceding too much control: 'Sometimes because you feel like it knows so well what it's doing, you end up giving it your full trust and letting

it take over... Then once it gets stuck, you're in a mess because you don't know all the things it created under the hood.' P12's preferred response of forcing the agent to work in ways they understand, even when it proposes something arguably better, reflects a broader shift they observed in how developers now make sense of their own codebases. Where the work was once to plan and then implement, it is now increasingly to specify, then verify what was built. Several other participants also described deliberately eliciting Socratic interaction (prompting the model to ask questions rather than generate answers) as a way of counteracting what P8 called the 'answer bias': LLMs' default tendency to produce confident responses rather than surface the assumptions that should be examined first. What unites these practices is that they keep the human in the strategic, creative, and architectural parts of the work while delegating execution to the model.

3.2. The Perceived Potential for Augmentation

Across participants and domains, negative perspectives on LLMs were most consistently expressed in response to the technology taking over work the user found rewarding, e.g., problem solving and strategic thinking for developers, ideation and craft for writers, critical thinking for a philosophy student. On the other hand, positive perspectives on LLMs were voiced by participants who had found ways to offload work they found tedious, repetitive, time-consuming, or cognitively burdensome (in an unrewarding way), while retaining agency and control over the parts of their work they found meaningful.

In sum, AI is experienced as augmentative when it handles jobs the user would rather not do, while leaving the user in control of the work they actually want to do. This adds nuance to traditional definitions of complementarity, i.e., a complementary relationship with AI is not one in which AI is simply delegated tasks it is 'better at.' Several participants noted that they were perfectly capable regarding the work they delegated, and in some cases could do it better than the LLM. P10, for instance, described delegating the database layer of a project: 'I know exactly how to do that; it just takes a week and it's annoying.'

When complementarity is achieved on these terms, participants described a qualitative shift in their experience of work. Participants with the most positive relationships with AI were not necessarily those producing the most output, though the ability to automate large amounts of rote work with LLMs was a factor. Where they perceived AI to be augmentative was in the renewed relationship with work it enabled. In several cases this was described as a reinvigoration of enjoyment that had been eroded by years of repetitive execution work. P3 described how encountering the same problem for the tenth time had gradually drained the joy from coding, and how LLMs had introduced the possibility of converting 'every wild idea' into a tangible proof of concept. P10, who described having 'a cemetery of dead side projects' before agentic coding tools, reported 'a renewed interest and enthusiasm' driven by being able to let the LLM handle the parts they found draining while retaining the problem-solving they find genuinely rewarding: 'my love was never the ability to write code. I think my love was for the ability to solve a problem.'

The same phenomenon explains why LLMs fail to feel augmenting in other contexts. P1, who writes and codes, was clear that they would not use AI for writing: 'I have the privilege of only doing things that I enjoy. Why would I pawn the things I enjoy off on AI?' P5 described the process of making connections between a quote and a piece of fact during journalism as something AI isn't capable of replicating: 'the logic and the connections that you make between either a quote that somebody's made or a piece of a fact that you've made – that I just think AI isn't capable of doing.' They argued that the spontaneous, chance-driven nature of ideation is fundamentally incompatible with the structured, prompt-based interface of a chatbot: 'I think a spontaneous process where you are trying to type something into the chatbot misses the point around the spontaneity of the idea process.'

P8, who experimented extensively with AI-assisted essay writing, found that it substituted a different and less satisfying kind of cognitive work for the kind they valued: 'I was spending a lot of time thinking about how to work with AI to produce a very effective essay... it always felt more scattered... everything was a lot shallower. That's not to say the output was shallower, but I wasn't responsible for anything that wasn't shallow.' This concern extended to the value of effort itself. P8 observed that intrinsic motivation

depends on recognising that work produces a commensurate reward, and that AI risks severing that connection: ‘I just don’t think it is realistic to expect that people can continually get the same amount of enjoyment out of doing something when they don’t have to work hard... You develop intrinsic motivation by recognising that X amount of work will give X reward.’ A similar intuition was expressed by P4, an early adopter of agentic tools, who, despite overall enthusiasm, noted that when everything is automated, it raises the question of where intrinsic value comes from at all: ‘If everything takes no effort, does it have any value?’

P13, a games developer who had used AI extensively for web development, highlighted what happens when AI takes over too much of the rewarding work. When they extended AI use to the parts of development they genuinely loved, the work became ‘very monotone... very dry... instead of being a sort of passion... it started to become more of a chore.’ They described this not as a productivity problem but as a loss of craft: ‘writing the code isn’t the sole goal of development. There is a craftsmanship part to it that is very undervalued if you offload all of that work.’

The intrinsic reward framing also helps explain a related phenomenon: the tendency to overestimate productivity gains from LLM use. P3 described what they called a ‘magic factor.’ This describes the situation in which the moments of surprise at what LLMs are capable of are so significant that they overshadow the frustrating hours spent in unsuccessful ‘doom loops,’ which are both less memorable and often lack negative professional consequences since experimentation is encouraged. The result is that people exit interactions with LLMs perceiving them to have been more helpful than objective metrics might support. Perceptions of augmentation and actual augmentation are therefore not the same thing, and intrinsic reward is partly what conflates them. When AI handles work the user finds unrewarding and leaves them with work they find rewarding, the experience genuinely does feel like augmentation, regardless of whether output quality or quantity has measurably improved.

3.3. Tradeoffs between Augmentation and Burnout

The dark side of this perception-vs-reality phenomenon is that some software engineers described coding with agents as having become addictive, suggesting that these interactions are tapping into extrinsic reward mechanisms as well.

P7 captured this shift without being prompted: ‘the work is shifting from writing the code to specifying the system. Not everybody enjoys that part of the work. I think it’s actually much more mentally draining... I am thinking harder than I have in a long time because the pace and volume of everything is just so fast.’ P10, despite their enthusiasm for what agentic coding has made possible, also noted: ‘a lot of what I’m doing now, it genuinely feels like I’m not doing software engineering anymore. It sometimes feels like I’m babysitting agents.’ P9 described the cognitive demands in terms of what managing multiple agents actually requires, likening the engineer’s new role to that of a startup founder: ‘The founder’s capability at describing a future vision is really important because they’re not going to be able to review all these decisions that need to happen with them out of the loop [i.e., the decisions made by agents].’ These interviews demonstrate that the new work is not only cognitively demanding but, for many participants, less intrinsically rewarding than the production work it partially replaces, which could at least partially explain why it is so ‘mentally draining.’

A related concern, expressed more tentatively across several interviews, is whether extensive LLM use is affecting skill development and output quality over time. P8 was most direct in expressing this concern: ‘AI does a lot of things you would otherwise spend time doing, which means you’re not spending time thinking about the material... my understanding went down. There’s no doubt about it, because I was just moving through far more information more quickly.’ P3 described observing a near-universal pattern among coworkers a few months into working intensively with LLMs: ‘almost universally everyone starts to lower their quality bar a bit... because of the sheer amount of outputs you have to consume and validate. That’s dangerous.’ P13 offered the most detailed account of skill atrophy from personal experience. Having used AI heavily for web development, they found that after several months, their intuitive grasp of the codebase had faded. Tasks that were once automatic became deliberate, and de-

bugging, which had previously felt instinctive, required conscious effort where it hadn't before. They described losing what they called their *Fingerspitzengefühl*, or the feeling in one's fingertips for where problems were located (in other words the intuition that comes with expertise) (Dreyfus, 2004). They attributed this directly to having offloaded the thinking: 'you initially would probably think that... I still do the thinking, it's just like the writing, the basic code. But I think you need to sort of keep your skills sharp, just like how athletes still do the basic drills.' These observations were not the focus of this study and should be treated as preliminary, but they point to questions that longitudinal research will need to address.

Several participants also raised questions for what implications the challenges they faced could have for those still developing expertise. The strategies that allow experienced developers to approximate complementarity have to be learned in tandem with all that the inexperienced don't know about programming. As P10 observed, 'the stuff you don't know you don't know. That's the gigantic problem... a junior won't necessarily know that what [the LLM] is doing is correct... you just don't have the experience.' This dynamic was echoed by P6, a design educator who observed the same pattern in students. The result is that LLMs, in their current default design, risk replacing cognitive processes that novices need to develop rather than supporting them, which means amplifying existing gaps rather than closing them. So while LLMs have the potential to maximise rewarding work, they also have the potential to amplify existing inequalities in expertise.

4. Discussion

This paper has described how some of the strategies that professional software engineers (SWEs) have developed to work more effectively with LLM-based agents can both have augmentative, rewarding effects while at the same time producing burnout. Throughout, emphasis has been placed on the potential for LLMs to afford SWEs more time for the intrinsically rewarding, or meaningful, aspects of their work while offloading the rote or tedious. It is important to note that this was an emergent theme, not one driven by the study's research questions; however, it serves as a productive framing when considering how to design AI-powered tools that are more complementary to their users.

Such a framing echoes discourses on human flourishing, which encourage a more holistic perspective on individual and collective well-being (VanderWeele, 2017). However, there is reason to think this reorientation in perspective requires deliberate effort. Brynjolfsson (2022) argues that technology developers and businesses face structural incentives that favour automation over augmentation. Substituting existing tasks is the path of least resistance, while genuine augmentation demands more radical organisational transformation. The result is what he called the 'Turing trap,' a systematic bias toward developing AI that emulates human performance at existing tasks instead of expanding what humans are capable of. In this view, complementarity is not treated as the default.

From a 'human flourishing' perspective, complementarity looks beyond enhancing human skills with the goal of improving productive output, and towards considerations of the consequences that such a myopic focus (i.e., on productivity) might have on greater well-being. Here that means designing to protect human agency by default instead of allowing it to be something that users have to work for. As this interview sample exhibited, even those users who are best positioned to architect their workflows to protect their agency (because they have the technical skills to steer LLM behaviour beyond simple prompting) still struggle to succeed in getting agents to work for them, risk confronting burnout, and occasionally still inadvertently get locked out of decision-making. This has been observed more broadly in corporate AI adoption, where the initial productivity gains from AI use tend to give way to workload creep, cognitive fatigue, and burnout as workers self-impose higher pace and expanded scope without being asked to (Gent, 2025; Ranganathan & Ye, 2026). The fatigue and burnout are likely, at least in part, a result of the human's role in software development work shifting from a production and execution focus to one that is more evaluative and managerial (Simkute et al., 2024). Another recent study suggests that this encroachment on agency could extend far beyond software engineering. In a large-scale survey with over 300 American workers across a diverse range of professions, Ranjit et al. (2026) found that

meaningful work – tasks which workers associate with a greater sense of agency or happiness – may be disproportionately exposed to intrusion by AI, as compared to ‘low-level’ work.

This is not to say that high-level domains should be exclusively reserved for humans and low-level tasks delegated disproportionately to artificial agents, but rather that there is still a ways to go towards designing truly complementary AI systems: ones that are capable of maximising the human user’s agency over what they find meaningful without introducing unnecessary cognitive burdens. Naturally, ‘meaningful’ differs from person to person, and current AI systems are not designed to uncover what that means for each user nor to structure interactions accordingly (Fischli et al., 2026). Current systems operate closer to what Fischli et al. (2026) call the ‘liberal’ approach to promoting autonomy in human-AI interactions, i.e., acting on what users explicitly ask for, without the ‘capability-boosting’ dimension that would require understanding and actively protecting each user’s needs over time.

Where the flourishing framework and Brynjolfsson’s case for augmentative AI highlight the importance of protecting agency at the systems level, self-determination theory (SDT) provides possible explanations for its consequence at the individual level. SDT is an empirical framework that studies how differing types of motivation predict outcomes such as psychological wellbeing, learning, and performance on creative problem-solving tasks (Deci & Ryan, 1985). Central to SDT is the distinction between autonomous motivation, in which people experience volition and self-endorsement of their actions, and controlled motivation, in which behaviour is driven by external contingencies or pressure. Importantly, autonomous motivation leads to better psychological health outcomes, greater persistence, and more effective performance on heuristic tasks, which are common in knowledge work (Deci & Ryan, 2008).

Work that is autonomously motivated is also more energising, while controlled motives drain energy, which could partially explain why multiple participants reported that AI’s potential to free up their time to work on personal projects felt so invigorating (Deci & Ryan, 2008). Even though software engineering in a work context can be considered extrinsically motivated, participants’ experiences indicate that it is very much possible to be motivated by, and to find enjoyment in, specific aspects of that work. This is perhaps why some react so strongly to AI ‘treading on their toes,’ so to speak, when it comes to those aspects referred to here as meaningful or rewarding. For many, strategic problem solving and decision-making encapsulate what they find rewarding about engineering. For some (arguably a minority), like P13, this extends to the act of writing code. However, even for P13, the importance of writing code was less about the actual act of writing code and more about the upkeep of the core skillset and intuition (the ‘craftmanship’) that made him a good engineer. As they said, ‘It’s just more fun to do things yourself.’ Whether or not people like P13 are right in believing that generative AI spells doom for software engineering as a craft, the design of future AI systems could better tilt the balance towards complementing, and away from replacing, humans by becoming more adaptable to individual users’ personal fulfillment. The industry-standard practice of keeping repository-level context files (e.g., ‘CLAUDE.md’) is evidence of, and a step towards, this. However, it is also indicative of something broken, particularly because they are not airtight solutions for addressing individual user needs due to LLMs’ limitations in managing context (N. F. Liu et al., 2024), nor do they reliably improve code quality (Gloaguen et al., 2026).

The findings for expert users raise the question of how less experienced users will be impacted. Will they have the same experiences as experienced software engineers or could their challenges be compounded? Recent work suggests that offloading tasks to AI denies users the experience of productive struggle through which accurate knowledge of one’s own capabilities develops, which results in reduced persistence even in the short term (Elizondo et al., 2024; Fleming & Daw, 2017; G. Liu et al., 2026; Yeung & Summerfield, 2012). Expert software engineers have already developed this self-knowledge, but novices haven’t, which could explain why the benefits of LLM-assisted coding are not shared equally across skill levels (Prather et al., 2024). Complicating this picture further are the new cognitive burdens discussed above. Participants’ accounts of the demanding nature of learning to work with LLMs are consistent with Simkute et al. (2024)’s analysis of the ‘ironies of generative AI,’ which documents

how the shift from production to evaluation introduced by automation does not simply reduce workload but redistributes it in ways that are often more taxing than anticipated. This is an effect of automation documented at least as far back as 1983 (Bainbridge, 1983). Sarkar and Drosos (2025) corroborate this phenomenon independently, describing it as programming expertise being ‘redistributed rather than eliminated’ in agentic workflows. Experienced developers may be better positioned to manage these new burdens, having already developed both the foundational technical skill to evaluate LLM outputs and the metacognitive strategies that effective delegation requires. Non-experts have neither.

Presented with system designs that require expert skill to protect agency and reward, novices risk of flooding too much, and, in the process, lose the opportunity to learn not only important technical skill but also what they find rewarding about coding, which in turn impacts their ability to develop successful delegation strategies. Zhang (2024) goes even further, arguing that certain human activities derive their value intrinsically, from the quality of one’s direct engagement, and cannot be meaningfully delegated to computational systems without loss. This again highlights the importance of exploration and the development of underlying skill, as some activity may not be able to be offloaded without impacting, at the very least, persistence and motivation. Two design directions for AI systems surface from these considerations: one that focuses on fostering skill development, and another that focuses on adaptability to what the individual user finds rewarding, ideally both part of the same system. These correspond to two of the autonomy-promoting strategies that Fischli et al. (2026) propose for personal AI agents: capability-boosting, which promotes autonomy by helping users build the skills they need to pursue their goals independently, and meta-autonomy, which gives users control over what kind of AI interaction they want across different domains, e.g., following direct instructions in one context, scaffolding skill development in another.

5. Conclusion

These findings suggest that the question of whether LLMs augment human capabilities is not a property of the technology itself, but of the relationship a user is able to construct with it. That relationship is shaped in part by the level of agency a user is able to exercise over the parts of their work they find rewarding or meaningful. Expert software engineers demonstrate that this is achievable to some extent, but only through sustained effort that current tool design neither scaffolds nor encourages. For less experienced users, the same tools that offer the promise of augmentation risk replacing the cognitive processes through which both technical skill and a sense of meaningful work develop. This reframes what it means to design for augmentation. Rather than optimising for output quality or task completion, systems that genuinely complement their users must account for what those users find rewarding, and actively work to preserve it. Two design directions follow from this work: systems that scaffold the skill development required for meaningful delegation, and systems that adapt over time to individual users’ sense of what constitutes rewarding work.

References

- Alami, A., & Ernst, N. A. (2025, January). Human and Machine: How Software Engineers Perceive and Engage with AI-Assisted Code Reviews Compared to Their Peers. <https://doi.org/10.48550/arXiv.2501.02092>
- Bainbridge, L. (1983). Ironies of automation. In *Analysis, design and evaluation of man-machine systems* (pp. 129–135). Elsevier.
- Braun, V., & Clarke, V. (2019). Reflecting on reflexive thematic analysis. *Qualitative Research in Sport, Exercise and Health*, 11(4), 589–597. <https://doi.org/10.1080/2159676X.2019.1628806>
- Brynjolfsson, E. (2022). The Turing Trap: The Promise & Peril of Human-Like Artificial Intelligence. *Daedalus*, 151(2), 272–287. https://doi.org/10.1162/daed_a_01915
- Deci, E. L., & Ryan, R. M. (1985). *Intrinsic Motivation and Self-Determination in Human Behavior*. Springer US. <https://doi.org/10.1007/978-1-4899-2271-7>

- Deci, E. L., & Ryan, R. M. (2008). Self-determination theory: A macrotheory of human motivation, development, and health. *Canadian Psychology / Psychologie canadienne*, 49(3), 182–185. <https://doi.org/10.1037/a0012801>
- Dell'Acqua, F., McFowland, E., Mollick, E. R., Lifshitz-Assaf, H., Kellogg, K., Rajendran, S., Kraye, L., Candelon, F., & Lakhani, K. R. (2023). Navigating the Jagged Technological Frontier: Field Experimental Evidence of the Effects of AI on Knowledge Worker Productivity and Quality. *SSRN Electronic Journal*. <https://doi.org/10.2139/ssrn.4573321>
- Dong, C. V., Lu, Q., Norman, K. A., & Michelmann, S. (2025). Towards large language models with human-like episodic memory. *Trends in Cognitive Sciences*, 29(10), 928–941. <https://doi.org/10.1016/j.tics.2025.06.016>
- Dreyfus, S. E. (2004). The Five-Stage Model of Adult Skill Acquisition. *Bulletin of Science, Technology & Society*, 24(3), 177–181. <https://doi.org/10.1177/0270467604264992>
- Elizondo, K., Valenzuela, R., Pestana, J. V., & Codina, N. (2024). Self-regulation and procrastination in college students: A tale of motivation, strategy, and perseverance. *Psychology in the Schools*, 61(3), 887–902. <https://doi.org/10.1002/pits.23088>
- Fischli, R., Franklin, M., Manzini, A., & Gabriel, I. (2026). Agents, Alignment, and the Many Faces of Autonomy. *Minds and Machines*, 36(3), 34. <https://doi.org/10.1007/s11023-026-09786-9>
- Fleming, S. M., & Daw, N. D. (2017). Self-Evaluation of Decision-Making: A General Bayesian Framework for Metacognitive Computation. *Psychological Review*, 124(1), 91–114. <https://doi.org/10.1037/rev0000045>
- Gent, E. (2025). AI coding is now everywhere. But not everyone is convinced. *MIT Technology Review*.
- Gloaguen, T., Mündler, N., Müller, M., Raychev, V., & Vechev, M. (2026, February). Evaluating AGENTS.md: Are Repository-Level Context Files Helpful for Coding Agents? [arXiv:2602.11988 [cs]]. <https://doi.org/10.48550/arXiv.2602.11988>
- Ko, A. J., Abraham, R., Beckwith, L., Blackwell, A., Burnett, M., Erwig, M., Scaffidi, C., Lawrance, J., Lieberman, H., Myers, B., Rosson, M. B., Rothermel, G., Shaw, M., & Wiedenbeck, S. (2011). The state of the art in end-user software engineering. *ACM Computing Surveys*, 43(3), 1–44. <https://doi.org/10.1145/1922649.1922658>
- Kraljic, T., & Lahav, M. (2024). From Prompt Engineering to Collaborating: A Human-Centered Approach to AI Interfaces. *Interactions*, 31(3), 30–35. <https://doi.org/10.1145/3652622>
- Laban, P., Hayashi, H., Zhou, Y., & Neville, J. (2025, May). LLMs Get Lost In Multi-Turn Conversation [arXiv:2505.06120 [cs]]. <https://doi.org/10.48550/arXiv.2505.06120>
- Liu, G., Christian, B., Dumbalska, T., Bakker, M. A., & Dubey, R. (2026, April). AI Assistance Reduces Persistence and Hurts Independent Performance [arXiv:2604.04721 [cs]]. <https://doi.org/10.48550/arXiv.2604.04721>
- Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., & Liang, P. (2024). Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics*, 12, 157–173. https://doi.org/10.1162/tacl_a_00638
- Meredith Ringel Morris. (2024, October). Prompting Considered Harmful – Communications of the ACM. Retrieved February 9, 2026, from <https://cacm.acm.org/opinion/prompting-considered-harmful/>
- Mohamed, A., Assi, M., & Guizani, M. (2026). The Impact of LLM-Assistants on Software Developer Productivity: A Systematic Review and Mapping Study [Just Accepted]. *ACM Trans. Softw. Eng. Methodol.* <https://doi.org/10.1145/3809494>
- Prather, J., Reeves, B. N., Leinonen, J., MacNeil, S., Randrianasolo, A. S., Becker, B. A., Kimmel, B., Wright, J., & Briggs, B. (2024). The Widening Gap: The Benefits and Harms of Generative AI for Novice Programmers. *Proceedings of the 2024 ACM Conference on International Computing Education Research - Volume 1*, 469–486. <https://doi.org/10.1145/3632620.3671116>
- Ranganathan, A., & Ye, X. M. (2026). AI doesn't reduce work—it intensifies it. *Harvard Business Review*. Retrieved March 12, 2026, from <https://hbr.org/2026/02/ai-doesnt-reduce-work-it-intensifies-it>

- Ranjit, J., Zhou, K., Swayamdipta, S., & Quercia, D. (2026, March). Are We Automating the Joy Out of Work? Designing AI to Augment Work, Not Meaning.
- Sarkar, A., & Drosos, I. (2025, October). Vibe coding: Programming through conversation with artificial intelligence [arXiv:2506.23253 [cs]]. <https://doi.org/10.48550/arXiv.2506.23253>
- Simkute, A., Tankelevitch, L., Kewenig, V., Scott, A. E., Sellen, A., & Rintel, S. (2024). Ironies of Generative AI: Understanding and Mitigating Productivity Loss in Human-AI Interaction. *International Journal of Human-Computer Interaction*. <https://doi.org/10.1080/10447318.2024.2405782>
- Steyvers, M., Tejeda, H., Kerrigan, G., & Smyth, P. (2022). Bayesian modeling of human-ai complementarity. *Proceedings of the National Academy of Sciences*, 119(11), e2111547119.
- Suh, S., Min, B., Palani, S., & Xia, H. (2023). Sensecape: Enabling Multilevel Exploration and Sensemaking with Large Language Models [arXiv: 2305.11483]. *UIST 2023 - Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. <https://doi.org/10.1145/3586183.3606756>
- Tang, N., Chen, M., Ning, Z., Bansal, A., Huang, Y., McMillan, C., & Li, T. J.-J. (2024, May). A Study on Developer Behaviors for Validating and Repairing LLM-Generated Code Using Eye Tracking and IDE Actions. <https://doi.org/10.48550/arXiv.2405.16081>
- VanderWeele, T. J. (2017). On the promotion of human flourishing. *Proceedings of the National Academy of Sciences*, 114(31), 8148–8156. <https://doi.org/10.1073/pnas.1702996114>
- Wilder, B., Horvitz, E., & Kamar, E. (2020, May). Learning to Complement Humans [arXiv:2005.00582 [cs]]. <https://doi.org/10.48550/arXiv.2005.00582>
- Yeung, N., & Summerfield, C. (2012). Metacognition in human decision-making: Confidence and error monitoring. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 367(1594), 1310–1321. <https://doi.org/10.1098/rstb.2011.0416>
- Zhang, H. (2024). Searching for the Non-Consequential: Dialectical Activities in HCI and the Limits of Computers. *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 1–13. <https://doi.org/10.1145/3613904.3641945>