arXiv:2611.12646v1 [cs.SE] 9 Nov 2026

Surrendering and Reclaiming Judgment: How Software Engineering Students Experience AI Reliance

NATHAN CASSEE, University of Victoria, Canada
UMIT AKIRMAK, University of Victoria, Canada
LUIZ GUERRA, University of Victoria, Canada
MARGARET-ANNE STOREY, University of Victoria, Canada
NEIL ERNST, University of Victoria, Canada

Software architecture is one of the skills novice developers must master to succeed in industry, and as AI disrupts software development, it is also transforming how students learn architecture. Existing work on how students and novices work with AI has mostly focused on small programming exercises or on students' perceptions of AI, so little is known about how students experience AI as they learn software architecture in larger team projects. We report on an exploratory case study of 43 senior software engineering students who used AI throughout a semester-long, team-based software architecture project. We combined surveys with focus groups and analyzed the focus group transcripts and the open survey responses through Braun & Clarke's reflexive thematic analysis. We constructed two themes. The first, **Surrendering Judgment to AI**, traces how participants' reliance on AI deepened as they used it to architect and implement their systems, until they surrendered their judgment and became stuck in a codebase they had created but could not understand, which made them rely on AI even more. Participants described feeling ashamed, frustrated, and emotionally disconnected from their work, a toll that, in our reading, left them with even less reason to break their reliance. The second, **Applying Human Judgment**, centers on participants who kept or reclaimed their judgment: the knowledge and experience it rested on, and the strategies they used to apply it. Participants' accounts suggest that relaxed deadlines and a lecture and lab on working with AI gave them room to acquire knowledge and reclaim judgment. Given the expectations placed on junior developers and the widespread adoption of AI, we believe these first-hand accounts can help software engineering education and industry adopt AI in ways that preserve novice developers' judgment, ownership, and motivation.

Additional Key Words and Phrases: Software Engineering Education, Experience of Generative AI Reliance, Emotional Disconnect

ACM Reference Format:
Nathan Cassee, Umit Akirmak, Luiz Guerra, Margaret-Anne Storey, and Neil Ernst. 2026. Surrendering and Reclaiming Judgment: How Software Engineering Students Experience AI Reliance. 1, 1 (November 2026), 43 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Competent and productive developers are expected to master a variety of higher-order skills. One of the most important skills is software architecture, or the ability to design software architectures [31]. Mastering architecture is particularly

Authors' Contact Information: Nathan Cassee, nathancassee@uvic.ca, University of Victoria, Canada; Umit Akirmak, uakirmak@uvic.ca, University of Victoria, Canada; Luiz Guerra, luizguerra@uvic.ca, University of Victoria, Canada; Margaret-Anne Storey, mstorey@uvic.ca, University of Victoria, Canada; Neil Ernst, nernst@uvic.ca, University of Victoria, Canada.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.
Manuscript submitted to ACM

difficult for novices, because its open-ended and complex nature makes it hard to learn [25]. Therefore, novices need sustained practice: repeated engagement with realistic, difficult software architecture problems [18, 38].

The widespread adoption of generative AI in software engineering[1] might disrupt how novices master software architecture, as AI can reason about software architecture [28] and even suggest entire architectures [20]. These capabilities make AI very useful for experienced developers and may increase their productivity [57, 64]. However, how these benefits extend to novices is less clear, because a developer's experience affects how they use AI for development [32], and the capabilities that enable experienced developers to use AI for architecture might not serve novices in the same way. In fact, it might be attractive for novices who have not yet mastered architecture to offload the demanding parts of architectural work to AI [16].

At the same time, software engineering education is rapidly exploring how to incorporate generative AI [21, 51]. Educators are redesigning their courses, architecture courses in particular, with the expectation that AI can take over routine implementation work. This should free students to spend their time on the higher-order architecture skills these courses are meant to teach [27, 56]. Because AI is new and rapidly evolving, there are no established guidelines on how it should be adopted. Existing work that might inform guidelines has focused mostly on foundational skills like programming [21, 26, 47, 51], in individual, short-lived exercises, and it characterizes primarily *how* novices use AI [29, 39, 53, 56] through self-reported perceptions that may not match what happens during development [9, 13]. However, this work does not focus on what it's like for novice developers to work with AI while learning challenging skills like software architecture in a semester-long, team-based project. We therefore ask: **How do senior software engineering students experience AI as they learn software architecture in a team project?**

We conducted an exploratory case study of 43 senior software engineering students who used AI throughout a semester-long, team-based software architecture project. We combined three surveys with four focus groups, and we analyzed the focus group transcripts and the open survey responses through reflexive thematic analysis [12].

In this paper, we report on the two themes we constructed through reflexive thematic analysis. Both are about what it was like for students to work with AI, and both center on participants' judgment over what AI produced. Our first theme, **Surrendering Judgment to AI**, is constructed around participants' accounts of an extensive reliance on AI, of surrendering their judgment, first over architectural decisions and then over code they could no longer evaluate, and of the emotional toll that followed: they felt ashamed that they had to rely on AI, frustrated when incorrect information from AI left them stuck, and increasingly disconnected from their work as their reliance deepened. One participant described AI as an "intelligent gaslighter", and another as "that friend that you meet and think he's really smart [...] And he's been manipulating you".

Our second theme, **Applying Human Judgment**, centers on the accounts of participants who spoke about applying their judgment while working with AI, and on participants who reclaimed it during the course. What stood out to us is that their judgment was anchored in software engineering knowledge: some brought it into the project, and others recognized during the project that they lacked it and acquired it. Participants who reclaimed their judgment also described changing how they worked with AI. One of them put it like this: "AI was like milk and a cup of tea. A little bit would enhance the drink, but too much could ruin it and require you to remake it."

To us, these results suggest that the current generation of students and novice developers may be particularly vulnerable. In participants' accounts, three conditions pushed them toward surrender: an unfamiliar technology stack, delivery pressure outside their control, and an expectation that they use AI. None of these is unique to a course project,

[1] https://survey.stackoverflow.co/2025

as junior developers entering industry experience all three [10, 41, 69]. We believe, these conditions can contribute to a sustained reliance on AI that erodes the motivation to learn, and without that motivation there is little reason to break the reliance. Our second theme, applying human judgment, suggests this outcome is not inevitable: participants told us how relaxed deadlines and a lecture and lab on working with AI helped some of them reclaim their judgment. These changes hint at interventions that can help developers and organizations adopt AI sustainably, in ways that preserve novice developers' judgment, ownership, and motivation.

This paper makes three contributions. First, we show that what participants surrendered to AI was their judgment over what AI was doing. Storey [62] describes cognitive surrender by what developers stop doing (Section 2); our participants' accounts help us understand what they gave up. In participants' own words, we also document what that surrender felt like, including an emotional toll that, to our knowledge, prior work has not reported. Read through Self-Determination Theory [55], these accounts suggest why surrender left participants disconnected from their work (Section 5.1). Deadlines they could not control undermined their sense of autonomy, and working beside an AI they felt knew more than they did undermined their sense of competence; together, we think this contributed to their work feeling hollow.

Second, we show what the judgment of participants who kept or reclaimed it rested on: knowledge of the technologies AI worked with and strategies for working with AI that they discovered during the project. Participants started a new chat when AI went off track, kept each chat scoped to the problem at hand, wrote custom instructions, had AI work in small steps they could test, edited the plans AI proposed before it wrote any code, and asked AI to explain the code it produced. Each strategy helped participants apply judgment, gave them the chance to notice when AI made mistakes, and helped preserve their own understanding of the code.

Third, we argue that instruments that can detect surrender are necessary. Existing closed survey items did not allow us to detect the surrender; only participants' own words, in the open survey responses and the focus groups, allowed us to describe what surrender was like for these participants. We believe these accounts can be used by educators and researchers to help identify surrender in their own courses, or to develop instruments that can help detect it.

2 Related Work

Our study focuses on the experiences of senior software engineering students as they adopt generative AI on a software architecture project. We position our contributions in light of research on generative AI in software engineering education, organized by what that literature asks: what students *do* with AI, what they *think* of it, what relying on it *costs* them, and, in a much smaller body of work, what using it is *like* for them. Almost none of this work concerns software architecture education.

Introductory programming. Most of what is known about students and generative AI comes from foundational programming courses [21, 26, 27, 29, 39, 51, 53]. These studies catalog how students use AI on their coursework and consistently report wide variation: some students have AI produce entire assignments, while others use it only to look things up or make small changes. Their participants, however, are first- and second-year students working individually on short, well-specified exercises [21, 29, 51], a poor proxy for the open-ended, team-based work of software architecture that we study.

Team projects. Fewer studies observe students while they're working on longer team projects, and the studies that do catalog what changed in how students work, staying close to participants' explicit, surface-level accounts rather than interpreting what the experience was like for them. Salomon et al. [56] surveyed 349 students and interviewed 24 of

them in a project-based course where teams built three versions of a system with unrestricted access to AI. They find that students used AI not only to generate code but to validate work retrospectively against requirements and design, consulted AI to find information about software design, and that they turned to AI before consulting teammates, which reduced direct collaboration. Mircea et al. [42] surveyed 150 students across 18 capstone teams, finding that voluntary AI use was near-universal across the lifecycle, and that students articulated their own limits, recommending verification and warning against losing independent understanding. Wang et al. [66] examine a setting much like ours: a term-long, team-based greenfield project in which students directed their own use of generative AI. Asking which skills should be taught as generative AI becomes common in software development, they find that non-programming skills, such as working out requirements and specifications, matter more for effective human–AI collaboration than they used to.

Software architecture. Software architecture itself remains barely studied. Ferino et al. [23] reviewed 80 studies of novices adopting LLMs, and found only three that report novices using LLMs for architectural work: one on architecture definition, and two on diagram generation. They call this small number a research gap, and name architecture as a direction for future research. Mitas and Sachweh [43] address that gap from the tool side, proposing an AI-driven tutor for cloud-based software architectures.

Perceptions of AI. Beyond how students use AI, related work studies what they think of it, and most of these studies report a mix of positive and negative views [23]. Views split most clearly on learning. Most students in an undergraduate Java course judged ChatGPT suitable for learning programming concepts, but some avoided it for fear of not developing their own proficiency [30]. First-year engineering students listed over-reliance that impairs learning among their concerns [44], and computing students at one university grew more worried about becoming over-reliant on AI across three survey rounds [35]. Views also split by task: students judge AI more useful for coding and debugging than for analysis and design [6], the work an architecture course centers on. This literature records feeling as well as opinion: Ferino et al. [23] report that 56 of the 80 studies they reviewed capture novices' emotions towards LLM tools, including satisfaction, surprise, frustration, and fear. What it captures, however, is sentiment aimed at the tools and belief about them, not what students encountered in doing the work, which is the focus of our study.

Cognitive engagement. Closest to our study is work that focuses on the consequences of AI use and how those consequences are experienced by students. Applying structural equation modeling to survey data from 299 STEM students across five North American universities, Choudhuri et al. [16] find that students who trusted and routinely used generative AI reported having significantly lower cognitive engagement, and that neither prior experience with AI nor time in academia protected them. However, this work measures engagement through closed survey items, so it cannot capture, in students' own words, what working with AI was like. Storey [62] has begun to describe a related pattern among developers, "cognitive surrender" to AI: adopting its output with minimal scrutiny, as opposed to cognitive offloading, the strategic delegation of a discrete task to a tool. Cognitive surrender describes what developers stop doing when they surrender. It does not say what they give up, or what surrender is like for the developer who experiences it.

Students' experience of AI use. Three studies report the encounter of students with AI more directly. Choudhuri et al. [14] ran a between-subjects study in which 22 students completed three short software engineering tasks. Students using ChatGPT were no more productive, and felt no more capable, than students using traditional resources, but they were significantly more frustrated; tracing five AI faults to seven consequences, the authors report uncertainty about correctness and self-doubt among them. Anderson et al. [4] had ten advanced students make a non-trivial change to an unfamiliar codebase over three hours, half of them with access to ChatGPT, and nine of the ten became

stuck at least once. They contribute a typology of seven kinds of stuck moment and of how AI helped, hindered, or caused each impasse; several of these mechanisms recur in our data. They also find that task success and system comprehension come apart, as AI allowed participants with only a medium level of comprehension to complete a task that comparable participants without AI did not, and they close by calling for study of the long-term implications of working with a reduced understanding of the system. Alakmeh et al. [3] used a design similar to that of Anderson et al. [4], and proposes a taxonomy of personas that describes how novice programmers rely on AI: self-sufficient developers, understanding-gated developers, and AI-steered developers. These personas distinguish modes of working with AI, but not what it was like for students to program with AI, or to rely on it.

Choudhuri et al. [15] used reflexive thematic analysis on interviews with sixteen students drawn from senior and capstone software engineering courses. They find that students use AI across a wide range of learning and implementation tasks, and that some describe careful practices, such as cross-checking AI answers against documentation and Stack Overflow, though the authors stress that over-reliance and initial blind trust remain common among beginners. Several of the challenges they report echo what we find: students struggling to make sense of misleading responses, and struggling to get AI to do what they intend. Their account, however, organizes those challenges by where AI helps and hurts across task phases and traces them to properties of the tools, rather than grounding them in how a reliance on AI made participants feel about themselves and their work.

Research gap. Taken together, prior work tells us what students do with AI, what they believe about it, and what relying on it might cost them. It does not focus as much on what working with AI is like for the students themselves while they learn a skill as demanding as software architecture. This question is important because courses are rapidly adopting AI, based on the expectation that students will use AI [27, 56], yet there are no established guidelines on how they should. Students in these courses must learn a demanding skill while working with a tool that can do much of that work for them. We therefore argue that it is essential to capture not only how students use AI, but also how they experience working with it under these conditions: what relying on it does to their judgment over their work and to their motivation for it, and how they work with it to preserve their judgment.

Existing work rarely asks this. Studies with experienced developers measure productivity [57, 64], and studies in education characterize how novices use AI, mostly through self-reported perceptions [29, 39, 53, 56]. Studies that focus on the experience of working with AI compress it into at most a few hours of individual work on a prescribed task [3, 4, 14]. Those that span a whole project ask about workflow, collaboration, or skills rather than experience [42, 56, 66], and none study software architecture [23]. The work that does focus on experiential accounts [15] does not situate those accounts in the context of a single project, nor does it explore the emotional impact of AI reliance. Ferino et al. [23] arrive at a similar gap from the other direction: only 19 of their 80 studies consider motivation, and those studies report why novices take up an LLM tool or refuse it, not what a sustained reliance does to their motivation for the work. They also name the effect of LLM tools on novice developers' mental health as an open research gap.

We address this gap by following a single cohort through one semester-long, team-based software architecture project, combining surveys administered as the project unfolded with focus groups held after its final phase. We construct an interpretive account of how these senior software engineering students experienced a deepening reliance on AI, including the emptiness and loss of motivation some described, and of how other students kept or reclaimed their judgment.

3 Methodology

Our research question is: **How do senior software engineering students experience working with AI on software architecture projects?** To answer it, we studied student development teams in a third-year undergraduate architecture course at *University of Victoria*, in which students were encouraged to use AI while they designed and implemented a modern full-stack web application. We position this work as an **exploratory case study** that uses surveys and focus groups to describe how student teams experienced the project. In line with case study guidelines [68], we first describe the course, then the instruments we used to collect data, then the reflexive thematic analysis we applied to it, and finally how we established the trustworthiness of that analysis.

3.1 Context (The Course)

A case study requires a *real-life context* in which data collection took place [54], and we describe ours in detail because a detailed description of the context establishes the credibility of the findings and lets readers judge how far they transfer to their own setting [46]. Our context was a semester-long, third-year undergraduate course in the software engineering program of *University of Victoria*, taught in the fall semester of the academic year '25–'26 and taken by students from both the software engineering and the computer science program. The learning goals of the course include basic software design and architecture skills.

The course combined lectures, lab exercises, and a team project, and the project is what this study focuses on. Teams of four to five students had to design and implement a software system from the ground up, applying modern web-development principles and technologies, starting from a list of high-level requirements the teaching team provided. The teaching team assigned students to teams using the same start-of-course survey we use as a self-efficacy instrument (Table 1), ensuring diversity in programming capabilities and balancing teams by gender; students could nominate one preferred teammate from within their own lab section.

The project ran in two phases with two deadlines each. In Phase I, teams first delivered a set of design documents describing a software design that satisfied the project requirements, and then a fully functional implementation of that design. In Phase II, the teaching team added requirements, and teams delivered an updated design and an updated implementation. Phase II was not a repeat of Phase I, however. Teams primarily refactored and reworked the code they wrote in Phase I: they had to identify at least three technical debt items, record them in their backlog, and address them. They extended their architecture documentation with a module view and a component-and-connector view, each accompanied by an element catalog, interface descriptions, rationale, and a behavior sketch, and they revised or deprecated the architecture decision records they had written in Phase I. Two new capabilities were required as well: a Model Context Protocol server through which an AI agent could book rooms, and an external API endpoint. The two phases also ran on different schedules: Phase I allocated two weeks each to design and implementation, whereas Phase II allocated one week to design and three weeks to implementation.

For this offering, the course was redesigned to account for the widespread adoption of generative AI in industry [22, 41]. Its learning objectives were expanded to include developing software with generative AI tools, and students were encouraged from the start of the project to experiment with generative AI, especially for coding and implementation. The teaching team expected that this would let students spend more of their time on the system's high-level design and specification, instead of on the boilerplate coding that had required considerable effort from students in previous years.

When the course was redesigned, in early summer 2025, there were few or no established guidelines for incorporating generative AI into software design course projects like this one, so the teaching team had no materials from which

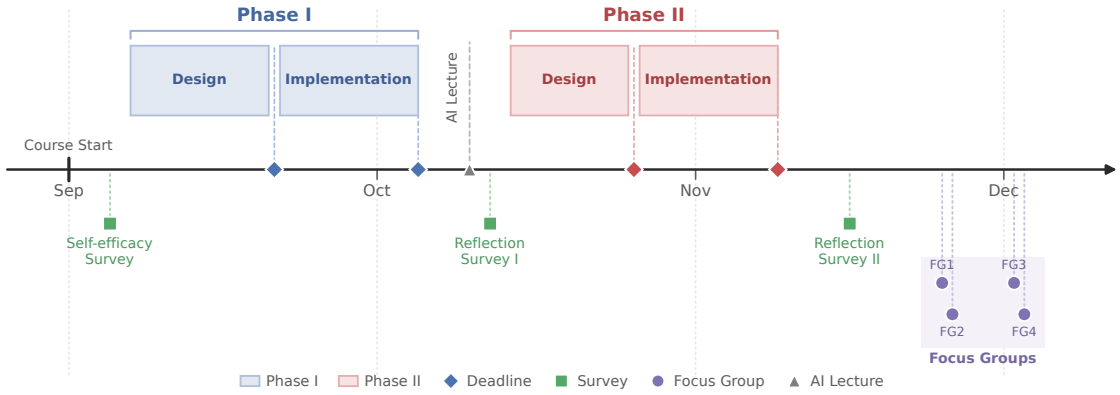

Fig. 1. Study timeline showing both the course deadlines and activities, and the data collection.

to instruct students. With no established practice to teach from, the teaching team instead encouraged students to experiment with various tools for themselves, using student packs of popular generative AI coding tools (Copilot, Google Gemini, etc.). Halfway through the course, between the two phases, the teaching team delivered a short lecture and a lab covering emergent best practices for AI usage. Both were built around a single three-phase workflow (research, planning, and implementation): students first use AI to understand the problem, then have it write an implementation plan that the team reviews before any code exists, and only then let it generate code in small, individually verified increments. The lecture presented this workflow as the alternative to "vibe coding" and also covered context engineering, project-wide custom instruction files, retrieval-augmented generation, evaluating model output, and the Model Context Protocol. The lab applied the same workflow to a small web application, to shows students how to work with GitHub Copilot. The materials of these sessions are made available on Figshare.[2]

3.2 Data collection

We collected data using two methods at several points during the project, as Wohlin and Rainer [68] recommend, to build a corpus that describes students' experiences from more than one angle. Three surveys were administered before, during, and after the project, and after Phase II we ran focus groups with the student teams to collect richer accounts of their experiences in an interactive group setting. Figure 1 shows the course events (above the timeline) and the data collection activities (below it).

The ethical review board of *University of Victoria* approved the study.[3] Participation was voluntary, and by consenting, students also allowed us to analyze the coursework they submitted. To ensure students did not feel coerced, a volunteer not involved in the course or the research distributed the consent forms, and the teaching team could not access the list of consenting students during the course.

3.2.1 Surveys. We administered three surveys: a *self-efficacy survey* at the start of the course and a *reflection questionnaire* after each project phase. The self-efficacy survey characterizes participants' at the start of the course, which helps

[2] https://figshare.com/s/7f25856966525523ea37
[3] Protocol number 25-0298

Section	Format	Content
`Prior Experience`	3 questions with mixed formats: 2 frequency sliders (Never Used — Use Daily) and 1 four-option multiple choice	Asking participants to report their familiarity with version control tools and prior experience with web application development.
`Programming Self-Efficacy`	7 questions on a 7-point Likert scale (1 - not at all confident, 7 - very confident), adapted from Ramalingam and Wiedenbeck [52]	Asking participants to rate their confidence in a range of programming tasks, from writing a simple Python program to comprehending complex multi-module object-oriented code.
`Software Design Self-Efficacy`	9 questions on a 7-point Likert scale (1 - not at all confident, 7 - very confident), inspired by Bass et al. [8]	Asking participants to rate their confidence in software design tasks, from describing design patterns and architectural views to creating UML diagrams and designing an architecture from user stories.
`Prior AI Usage`	3 closed multi-select questions, with an optional other field	Asking participants whether they previously used generative AI, and if so, which tools and for which software engineering tasks.
`Generative AI Self-Efficacy`	5 questions on a 7-point Likert scale (1 - not at all confident, 7 - very confident), selected from Wang and Chuang [67]	Asking participants to rate their confidence in AI-assisted programming tasks, from fixing bugs with AI support to generating entire functional programs.
`AI Attitude`	3 questions on a 4-point Likert scale (1 - strongly disagree, 4 - strongly agree), inspired by Miller et al. [41]	Asking participants to rate their agreement with statements on AI usage orientations, from persistence when AI fails to verifying and adjusting AI outputs.
`Industry Confidence`	3 questions on a 7-point Likert scale (1 - not at all confident, 7 - very confident)	Asking participants to rate their confidence in obtaining a junior software engineering position, contributing to a development team, and using generative AI in a professional role.

Table 1. The sections present in the self-efficacy instrument, together with a description of the questions included in each section.

interpret variation in how students adopted and used generative AI during the project [54]. The reflection questionnaires capture participants' self-reported AI usage and their perceptions of its impact after each phase.

Self-efficacy survey. Table 1 summarizes the sections of the self-efficacy survey; the full verbatim instrument is in the replication package.[4] The survey captures two types of pre-existing participant characteristics: factual prior experience and self-efficacy. `Prior Experience` covers distributed version control and modern web technologies, both of which the course required, and `Prior AI Usage` covers earlier use of AI for software development. Self-efficacy, the belief in one's own capability to execute a task [5], is an important predictor of learning behavior and performance, and the three self-efficacy sections cover the core competency domains of the course: the assumed prior knowledge (programming), the learning objectives (software design), and the phenomenon under study (AI usage). To ensure construct validity,

[4] https://figshare.com/s/7f25856966525523ea37

Section	Format	Content
`AI Usage`	3–4 questions with mixed formats: 1 binary (Yes/No) and 2–3 open-text fields	Asking participants whether they used generative AI during the phase, which tools they used, for which tasks, and (Phase I only) how they would describe the overall role of generative AI within their team.
`AI Outcomes`	5 statements on a 5-point Likert scale (Strongly Disagree — Strongly Agree)	Asking participants to evaluate five outcomes of their AI use: time savings, support for learning about software architecture, contribution to team goals, the overhead of verifying and correcting AI outputs, and the degree to which AI influenced architectural decisions.
`Open Reflection`	2–3 open-text questions (partially conditional on AI use)	Asking participants, in Phase I, who found AI useful to describe why and which strategies they applied, and who did not find it useful to explain why and what they would change to make it more useful; and, in Phase II, to describe the challenges they encountered and how they addressed them. Participants who did not use AI are asked to explain why not.
	Phase II only	
`Longitudinal Change`	3 questions with mixed formats: 1 binary (Yes/No) and 2 open-text fields (one conditional)	Asking participants whether their AI usage changed between Phase I and Phase II, and if so, how (e.g., changes in reliance or prompting strategy). Designed to capture the influence of the AI-skills lecture delivered between phases on students' usage.

Table 2. The sections present in the reflection questionnaires, together with a description of the questions included in each section. The bottom section is exclusive to the Phase II questionnaire.

their items were adapted from Ramalingam and Wiedenbeck [52] for programming, inspired by the knowledge and skill framework of Bass et al. [8] for software design, and selected from Wang and Chuang [67] for generative AI.

Reflection questionnaires. Table 2 summarizes the sections of the reflection questionnaires; the full verbatim instruments are in the replication package. `AI Usage` records whether, with which tools, and for which tasks students used generative AI in each phase. `AI Outcomes` asks students to rate the impact of their AI use on productivity, learning, and architectural decisions, and `Open Reflection` lets them describe strategies and challenges that Likert items alone cannot capture. The Phase II questionnaire adds `Longitudinal Change`, which tracks whether and how students' AI usage changed between phases, and so whether the lecture delivered between them affected their practice.

3.2.2 Focus Groups. To complement the survey data with richer, qualitative accounts, we ran four focus groups with eight student teams after Phase II. Focus groups suited the group-based nature of the project. Compared to

individual interviews, they let participants build on and challenge each other's accounts, surface a team's consensus and disagreements, and reach a shared sense-making that individual interviews cannot access [61].

Team recruitment. The teaching team announced the focus groups to the whole class and explained their purpose. Participating teams received cookies and soda, and their focus group replaced a group reflection assignment. Of the 16 teams, 11 expressed interest, and the teaching team helped select eight of them. Because focus groups are group-based, we did not want them to re-trigger interpersonal conflicts or place undue stress on participants. Therefore, one reason not to select all teams is to exclude teams that had experienced interpersonal conflict during the project or had members who did not contribute. This might bias the focus groups toward more cohesive teams, a risk we accepted to avoid confrontational or uncomfortable sessions for the participants.

The eight teams were distributed across four sessions, two teams per session, yielding 8 to 10 participants per session. This kept sessions within the recommended range of 8 to 12 participants [61]: below it, one or two voices tend to dominate, and above it, active participation by all members becomes difficult to manage. Four sessions across eight of the 16 teams also gave a wide variety of perspectives on the project [61].

Script Design. The focus group script follows the guidelines of Braun and Clarke [12] and Stewart and Shamdasani [61] and has five sections (Table 3). It orders questions from general to specific, as Stewart and Shamdasani [61] recommend, and moves participants from concrete, shared recollection toward deeper individual reflection. Each session opened with an anchoring exercise in which participants recalled an architectural decision they had made, which was meant to minimize retrospective bias [24], and then moved to a group discussion on team dynamics, AI use, and motivation. Each session closed with a synthesis exercise in which participants wrote down one or two words, or a metaphor, characterizing the role generative AI had played for them during the project, and then shared and responded to each other's characterizations. We refer to these as participants' *closing metaphors* where we draw on them in Section 4.

Because each session combined two established teams, participants knew their own teammates but not the other team, which risks within-team clustering during the discussion [61]. To counteract this, discussion prompts explicitly invited participants from different teams to react to and build on each other's accounts, and follow-up probes were directed across teams to surface differences and similarities in how the two teams approached the project. This cross-team dynamic also invited disagreement about experiences the two teams did not share.

The script was piloted once with junior researchers, which led to minor adjustments so that all five sections fit within the allotted 90 minutes. Stewart and Shamdasani [61] recommend piloting with respondents representative of the target population, but we had limited access to students outside the study population.

Procedure. Participants gave written informed consent before each session. The moderator then explained the purpose and scope of the session, invited participants to respectfully disagree with each other, and reiterated that there were no wrong opinions. Sessions lasted 90 minutes and took place in a large, well-lit lab room where participants and the moderator sat around a square table. To encourage interaction across the two teams, participants were assigned seats alternating by team.

Each session was run by a moderator and a facilitator [61]. The moderator led the discussion, kept it on track, ensured all participants were called on, and managed the time. The facilitator sat to the side of the room, took notes, and was occasionally called on to ask summarizing questions. All sessions were audio- and video-recorded; the video captures

Section	Format	Content
`Anchoring Exercise`	Individual recollection of an architectural decision, followed by a round-robin sharing.	Participants individually jot down one key architectural decision or feature from their project, and then go round and share their decisions. Sharing grounds the subsequent group discussion in specific, concrete recalled experiences rather than abstract perceptions of the project and AI.
`Unpacking Decisions`	Structured group discussion centered around the previously shared decisions, with moderator probes directed across both teams.	Explores team dynamics and AI's role in decision-making from two angles. The first covers decisions that generated notable disagreement (broadly interpreted as difficult, time-consuming, or revisited), probing team process, shared AI norms, and whether AI contributed to or complicated conflict. The second covers decisions for which the team relied heavily on AI, probing how AI was used for architectural reasoning, how outputs were accepted or rejected, and whether AI use for design differed from use for coding. Probes actively invited the other team to compare or contrast their own experience.
`Evolution of AI Use`	Group discussion with moderator probes	Participants describe changes in how they used generative AI between Phase I and Phase II: what prompted those changes, whether outputs improved, and any adjustments to prompting strategies or task allocation.
`Motivation`	Group discussion with moderator probes	Participants reflect on whether they genuinely cared about the quality of their work and what drove or undermined their investment in the project, providing motivational context for AI adoption choices.
`Synthesis Exercise`	Individual reflection followed by brief group sharing, and a final open question	Each participant writes one or two words, or a metaphor, characterizing the role AI played in their project, shares with the group, and briefly responds to each other's characterizations. A final open question invites participants to raise anything not covered in the discussion.

Table 3. The sections of the focus group script, together with a description of their format and content. The script was designed to move participants from concrete recollection (anchoring exercise) toward progressively deeper reflection on decisions, longitudinal change, and motivation, exploring the experience of AI usage from three complementary angles: teamwork dynamics, decision-making, and motivation.

non-verbal cues, such as gestures and reactions, that supplement verbal responses [61]. These recordings were the primary data source for the analysis, and Section 3.3 describes their transcription.

Researcher Positioning. In line with Braun and Clarke [12] and Stewart and Shamdasani [61], we describe the positions and backgrounds of the moderator and facilitator. The first author moderated all sessions and later coded the data; they are a software engineering researcher with extensive experience teaching software engineering courses and working with software architecture in industry. This background helped them interact with participants, for instance by letting students know they could use domain-specific terminology, and it informed their coding of the transcripts (Section 3.3). The second author facilitated the sessions and is an experienced qualitative researcher. Neither was involved in the course, so there was no prior relationship between them and the participants.

3.3 Data Analysis

We analyzed the data along two strands. The closed-response survey items were analyzed descriptively, to characterize the participants and the context they worked in. The qualitative data (the focus group transcripts and the responses to the open survey questions) were analyzed through reflexive thematic analysis, to construct an account of what the project was like for participants.

3.3.1 Descriptive Analysis of Closed-Response Data. The closed-response items of the three surveys (the Likert-scale ratings of programming, software design, and generative AI self-efficacy, the AI outcome perceptions, and the binary and multi-select items on prior and in-course AI usage) are summarized with descriptive statistics. We report them after the themes, in Section 4.3, to situate participants' accounts in relation to their self-efficacy and self-reported AI usage.

3.3.2 Thematic Analysis of Qualitative Data. We analyzed the open survey questions and the focus group transcripts using reflexive thematic analysis, following the framework of Braun and Clarke [11, 12] as applied in comparable software engineering education studies [2, 48]. We believe its inductive and reflexive approach suits our interest in the lived experiences of students. Because the study is exploratory, thematic analysis lets us identify emerging patterns of meaning in what students share, and it lets us draw on our experience as software engineering researchers to interpret their accounts.

Analytic Orientation and Reflexivity. Our coding and theme development were primarily inductive: given the exploratory nature of the study, we started from participants' accounts rather than a pre-selected theory or framework, and we constructed the two themes from the patterns of meaning we saw across those accounts. We selected self-determination theory [55] only when we noticed that some statements related to its constructs: that participants had little say over the pace of the project, that they did not feel able to do the work the project asked of them, and that they had stopped caring about what they were building. Braun and Clarke [12] note that a theoretical lens need not be settled in advance and may be adopted during interpretation where it affords a richer account. Our analysis is therefore inductive in its coding and theme development and theoretically informed only later: we report the themes in Section 4 in participants' own terms and use self-determination theory in Section 5 as a lens to interpret them.

Because the setting is educational, we take a more contextualist view. To interpret how students experienced the course, we relied on information from the teaching team about what participants went through and how they constructed working with AI. The first author, who coded the data, also drew on their own teaching and industry experience (see Researcher Positioning above) while coding the transcripts.

Pre-analysis: Transcription and Preparation. The audio and video recordings of the focus groups were transcribed with Whisper [50], run locally because the recordings contain personally identifiable information and the consent forms did not allow the recordings or transcripts to be shared with external services. The first author then corrected

the automated transcripts by hand and annotated them with contextual information visible only in the video, such as participants nodding in agreement, raising their hands, or reacting non-verbally to others' statements. In a second pass, all data collected for this study (surveys and focus groups) were pseudonymized: each participant received a unique identifier that replaced their name and identifies them across the surveys and focus groups.

The six stages of Braun and Clarke. Braun and Clarke [12] identify six phases of reflexive thematic analysis. The phases are described in order but are not linear; as the analysis developed, we moved between them several times to refine codes and to find and name themes. Below we describe what we did in each phase.

- **Phase 1: Dataset familiarization.** The first author read all survey responses and transcripts and rewatched each focus group recording. This happened a few weeks after the focus groups were held and transcribed, and so served as an effective reminder of the sessions. During this process, the first author wrote free-form memos recording initial observations, impressions, and emerging questions about the data. Prolonged engagement with the data, and documenting reflective thoughts about potential codes while becoming familiar with it, are the means Nowell et al. [46] describe for establishing trustworthiness in this phase. The memos already contain observations that became part of the final two themes, for instance the notion that students felt gaslighted by AI: *AI will suggest changes to everything. As one participant says: "AI gaslights you".*
- **Phase 2: Data coding (two passes).** The first author coded the data in two passes. The first pass produced descriptive (semantic) codes that stayed close to the data, and a selection of relevant excerpts from the transcripts. The first and second author then discussed the more interesting codes and excerpts. Because our coding is reflexive, the goal of this discussion was not to measure agreement [12] but to consider alternative explanations and deepen our understanding of what participants were saying; discussions of this kind are the peer debriefing Nowell et al. [46] recommend throughout coding. The second pass served as a completeness check and an opportunity to extract latent meaning not apparent in the first pass, for instance by coding more of the emotions participants expressed and by identifying every passage in which they talked about AI reliance. Working through the dataset twice also follows Nowell et al. [46], who recommend that coding be scrutinized at least twice and that every data item receive full and equal attention before themes are treated as settled. The second pass resulted in 1,775 coded excerpts with 1,501 unique codes.
- **Phase 3: Initial theme generation.** The first author grouped the codes into seven candidate themes, each a recurring pattern across the data that we believed relevant to the research question, such as how working with AI made participants feel, whether they controlled AI, and how the course structure influenced their AI usage. The first two authors discussed these candidate themes and their excerpts repeatedly, to work out how each should be defined and what united the codes within it. They were not themes yet, but they captured experiences participants repeatedly shared and were instrumental in constructing the final two themes.
- **Phase 4: Theme development and review.** To assess whether the candidate themes held up against the full dataset, the first author created Miro boards for the seven candidate themes, grouping similar codes, annotating the groups with interpretations, and visualizing the relations between them. Nowell et al. [46] recommend diagrams of this kind to make sense of the connections between codes and themes and to keep a record of how concepts and themes developed. The boards also made clear which focus groups or participants each code came from, and it was through them, and through a final board, that a structure of two contrasting themes centered on AI reliance was constructed. Each board was discussed extensively with the second author to verify that the underlying data supported the interpretation, and in later iterations with all authors to agree on a shared

interpretation of the data. Returning to the raw data in this way is the test of referential adequacy Nowell et al. [46] describe for this phase, and vetting the boards with the other authors follows their recommendation that themes be reviewed by the wider team.

- **Phase 5: Theme refining, defining, and naming.** Starting from the Miro boards, the first author wrote several draft summaries of the two emerging themes. Through these drafts, together with the visual summaries, the definitions of the final two themes emerged, and we settled on the insights we combine to form each theme. Keeping the drafts documents how the themes were defined and named, which Nowell et al. [46] list among the means of establishing trustworthiness in this phase.
- **Phase 6: Writing up.** For each theme, we selected representative quotes and combined them into a narrative that conveys both the theme's meaning and its significance for understanding students' experiences with generative AI during the project.

On the usage of generative AI for reflexive thematic analysis. A recent mapping study shows that generative AI is being applied to thematic analysis in software engineering [7], and custom-made tools exist to support this [60]. However, Montes et al. [45] found several limitations: generative AI is not suitable for latent interpretation, and when it constructs themes, there are no clear boundaries between them. Ernst and Treude [19] likewise conclude that generative AI lacks the analytical depth reflexive thematic analysis requires. Braun and Clarke themselves reject its use for reflexive thematic analysis, arguing that the meaning-making the method requires is not something generative AI is capable of [33]. In light of these findings and critique, we did not use generative AI for any of the coding or construction of the themes. In fact, we only used AI to evaluate and critique early drafts of our prose, including the descriptions of the themes, to help improve the flow and clarity of our writing.

3.4 Trustworthiness of this study

Case study research in software engineering is commonly evaluated against construct validity, internal validity, external validity, and reliability, or, for flexible design studies, against credibility, transferability, dependability, and confirmability; Runeson et al. [54] treat the choice between the two as one of terminology. Both frameworks assume that a stronger analysis is one that depends less on the researcher who produced it. Reflexive thematic analysis makes the opposite assumption: the researcher's subjectivity is a resource for the analysis, and a second analyst working the same data would produce a different account, and not necessarily a better or worse one [12]. For this reason we do not compute inter-rater reliability, as there was only one coder, and we make no claim of saturation, a concept Braun and Clarke [12] consider incompatible with reflexive thematic analysis and one we never tried to reach.

Instead, we evaluate this study against the criteria Braun and Clarke [12] define for reflexive thematic analysis, and we follow the practical guidelines of Nowell et al. [46]. From Nowell et al. [46] we take the phase-by-phase means of establishing trustworthiness, and Section 3.3 reports where each was applied: prolonged engagement with the data, triangulation of the data collection modes, memo writing during familiarization, peer debriefing between the first two authors, diagramming how codes and themes connect, a return to the raw data to test referential adequacy, and documentation of how the themes were defined and named. The criteria of Braun and Clarke [12] ask that transcription be thorough and checked against the recordings, and that coding give repeated attention to every data item rather than build themes from a few vivid examples (Section 3.3); that themes cohere around a central organizing concept, that data be interpreted rather than summarized, and that the extracts evidence the claims made (Section 4); and that the

approach and its theoretical assumptions be stated explicitly, that what was claimed match what was done, and that the researcher be positioned as active in the analysis (Section 6).

Section 4 reports the findings, first the two themes constructed through reflexive thematic analysis and then the descriptive statistics.

4 Results

From the focus groups and the open survey responses, we constructed two themes about how students experienced working with AI, both centered on participants' judgment over what AI produced. We first give an overview of both themes and each theme's **central organizing concept**: the core idea that unites a theme as a pattern of shared meaning [12].

The first theme, **Surrendering Judgment to AI**, is organized around the accounts of participants who spoke about their extensive and deepening reliance on AI, which we interpret as participants eventually surrendering their judgment to it. Participants who said they felt unprepared for the project described ceding decision-making to AI, no longer verifying its output, and eventually becoming stuck on bugs in a codebase they did not understand. To get unstuck, they often relied on AI even more instead of repairing their understanding, so the surrender only deepened their reliance. We believe an important element of this theme is how participants spoke about their work after that surrender: they described feeling ashamed, frustrated, and emotionally disconnected from work that had started to feel pointless and empty. In our reading, this emotional toll left them with even less reason to break the cycle of AI reliance.

In contrast to this surrender, the second theme, **Applying Human Judgment**, is organized around the accounts of participants who never surrendered their judgment to AI, and of participants who reclaimed it from AI. The core observation in this theme is how this judgment was anchored in the knowledge and experience of the participants. We observe how some participants refer to knowledge they brought into the course, while others said they recognized during the project that they lacked knowledge, and acquired it. In this theme, participants shared how they felt changes to the course gave them the space to acquire knowledge, and how their judgment afforded a different way of working with AI. One in which they did not surrender their judgment to it.

Because we constructed the themes inductively, surrender as a construct was not discussed with participants during the focus groups. Therefore, accurately counting the number of participants who surrendered during the course, or grouping participants into groups based on whether they surrendered, is therefore not possible, nor was that our goal. Reflexive thematic analysis discourages counting how often codes occur [12], which is another reason why we do not definitely attempt to assign participants to either theme. The transcripts do show that participants' experiences varied: some surrendered their judgment and later reclaimed it, some never surrendered it, and some surrendered it and never reclaimed it. Several participants, among them P5, P6, P9, P19, P32, P37, P44, and P45, speak in both themes. P6, for instance, described their reliance on AI as a "slippery slope" in the first theme, and in the second theme described "wanting to write more of the code by my own hands".

Figure 2 shows both themes, the insights that make up each, and the insight that connects them. In the remainder of this section, we quote the participants to explain how each theme is constructed.

4.1 Theme 1: Surrendering Judgment to AI — a deepening reliance on AI that sustained itself

This theme is constructed around three insights that follow participants from the conditions at the start of the project, to being stuck in a codebase they could no longer judge, to the emotional toll they described after that surrender, which we read as leaving them with even less reason to break the cycle.

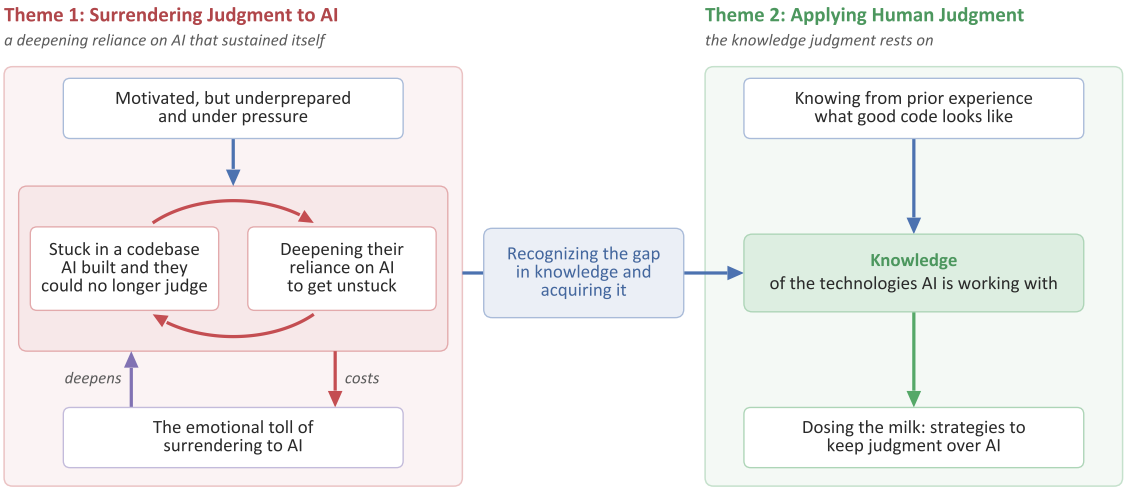

Fig. 2. Overview of the two themes and their insights. Theme 1 is a cycle: participants who arrived motivated but underprepared and under pressure surrendered their judgment to AI and became stuck in a codebase they could no longer judge, and what they described feeling after that surrender left them, in our reading, with even less reason to break the cycle. Theme 2 builds on knowledge: judgment over AI rests on knowing the technologies AI works with, and participants discussed the strategies that helped them apply it. The insight that connects the two themes, recognizing the gap in knowledge and acquiring it, is how some participants said they broke the cycle; others brought that knowledge into the course with them.

4.1.1 Motivated, but underprepared and under pressure. In these accounts, participants describe three conditions that we believe are relevant for understanding what later led them to surrender to AI: the motivation they started with, the gap they felt between their own skills and what the project asked of them, and the pressure of the deadlines.

> *I think yeah I was excited to learn new things. That was my motivation to try out uh like typescript [at] first like I wanted to learn uh because nowadays typescript is more widely used so i was willing to give [it] a chance and learn as much as i could over the span of this project.* P18 – Focus Group

Referring to the start of the project, some participants described being motivated to learn new skills. For P18, the project was a chance to learn skills they considered relevant to their future work. Moreover, we think that the recent widespread adoption of AI by industry and expectations that developers can use AI for software development will have contributed to this motivation, as some participants state that they were motivated because they were allowed and encouraged to use AI. *"But also, I was just going to say that being encouraged to use AI made me more motivated because it was the first course that I was allowed to use AI."* P38 – Focus Group Both quotes point to participants who initially felt motivated to work on the project. For different reasons, we read each as describing an interest in the work itself.

While talking about the start of the project, participants also mentioned their experience and confidence. To enroll in the course, participants were required to have passed prior programming courses and, in the first week of the course, almost all participants reported confidence in their own programming skills (Figure 4). When designing the course, the teaching team expected these prior programming skills to transfer to the web development skills required for this project. However, despite this experience and initial confidence, participants reported not knowing how to start their projects. *"Some of us have never even started like a JavaScript like project or something like that. So just like getting that*

started is even just a big barrier." P9 – Focus Group For this participant to refer to starting as a "big barrier" suggests that, initially, they felt their existing programming knowledge did not transfer to web development, and that the project sat outside their comfort zone. We read the gap between what participants brought and what they felt the project asked of them as a sign that they did not feel confident in their own abilities, nor able to do the work the project asked of them. It also hints at a mismatch between what the course team expected of participants and what participants felt they could do. P5, who was themselves already experienced with this stack (as we return to in the second theme), observed how this shortfall was shared across the class, and how important AI became:

> *we didn't know anything about the software architecture a lot of the people in the class didn't have any experience with like TypeScript or Docker or like any of the technology we were using [...] and then AI was like the only glue that would like somehow progress forward* P5 – Focus Group

We read P5's observation as showing how a shortfall of experience translated directly into an early reliance on AI. With unfamiliarity in TypeScript and Docker, and the software architecture itself something the class did not yet know, AI became the "glue" holding the work of the team together, the single thing that let the class "somehow progress forward".

Finally, we think the course structure, especially with respect to the deadlines, further contributed to how participants felt at the start of the project. *"I felt like we had a very short amount of time to make a lot of really big decisions about the system and then had to jump right into implementation."* P37 – Focus Group It is this feeling of having limited time that we think may also have left participants with little say over how they worked: describing "a very short amount of time" for "really big decisions", and then having to "jump right into implementation", P37 hints at a pace that was set for them rather than by them. Combined with the accounts of missing experience, this makes us believe participants experienced considerable stress at the start of the project. *"I feel like in the context of the course, it was like a bit, I feel like a word came to mind like a bit overwhelming."* P9 – Focus Group P9 goes as far as to describe the start of the course as "overwhelming", which reinforces our sense that at least some participants felt they had neither the experience the project required nor much say over how they worked on it.

The course was designed with a clear idea around AI usage: participants could use AI for the implementation, especially to work with a new programming language, while they were encouraged to use the course material for the software architecture. For some participants, this was part of what made the project motivating at the start. However, the way these participants describe starting the project suggests that this was not a boundary they found easy to hold. Coupled with how participants experienced the start of the course, with them describing not feeling competent in the technologies the project required, and not feeling much control over its pace, we think the motivation participants started with was already under strain, a shift we return to in the third insight.

4.1.2 A reliance on AI that deepened itself. Based on how participants described using AI once they were working on the project, we construct an account of how their initial reliance on it pulled them into a self-reinforcing cycle.

> *I used it[AI] quite extensively at the start of the project, trying to get all the different tools configured and work together to get a baseline project.* P36 – Focus Group

Given how some of the participants experienced the start of the project, unfamiliar with the stack, and under deadlines that left them little room, any tool that gets a baseline project running is valuable, and that is what P36 describes using AI for. *"I think my biggest reason for preferring to go to AI over, let's say, asking a teammate or a TA or prof is it's just, well,*

there's a huge time constraint. There's due dates and we all felt that sometimes it was too short of a due date or too short of a time working on the project." FG47 – Focus Group This quote from FG47[5] illustrates how participants experienced the conditions at the start of the course as a reason to use AI. Crucially, though, while this initial turn to AI might have helped some of the participants get started, for some it also became a deepening cycle of reliance on AI. Across the various groups, participants described becoming reliant on a tool they found useful and powerful, but at a cost. The closing metaphor from P22 (Table 3) names the ambiguous role AI came to occupy:

> *Um, I said I felt completely dependent on it[AI], and that it was my best friend and greatest enemy.* P22 – Focus Group

To us, holding "best friend" and "greatest enemy" together in a single sentence, alongside feeling "completely dependent", names the bind that runs through this insight: the tool participants could not work without was also the one causing them trouble, and being dependent on it meant they could not simply put it down.

To understand what this reliance was like, we begin with how aware participants were of it. Probes in the focus group asked participants to describe how they used AI while making architectural decisions, and specifically whether they allowed AI to make those decisions for them. Across the focus groups, participants repeatedly denied allowing AI to make architectural decisions; P25, for instance, describes using AI as an information source rather than ceding decision-making to it.

> *So I think in that way, we kind of used AI to do some research on some differences we might experience based on our choice. But I think overall, the actual decisions were more based on our own knowledge of them.* P25 – Focus Group

However, when we prompted participants to give examples of their AI usage, those examples complicate this account: *"[I used AI] to find out what files are needed to get a working backend structure, and what to fill into those files"* P34 – Survey The initial structure of a backend, in terms of files and modules, determines the architecture, especially for a TypeScript project. We do not read this as participants misrepresenting their own process; rather, under the pressure of an unfamiliar project and tight deadlines, relying on AI to scaffold the backend implicitly cedes these impactful architectural decisions to it, perhaps without participants recognizing how important those decisions were. In this case, the initial design of the backend also specifies the interface between the backend and the frontend, and therefore core components of the system's architecture.

As some of the participants worked with and relied on AI for their projects, they eventually found themselves unable to make sense of the AI-generated information. *"It[AI] was extremely helpful yet confusing cause I still didn't understand what we were exactly doing, it was just creating code for us!"* P21 – Survey We read the combination of "extremely helpful" and "still didn't understand" in a single sentence as showing how far the reliance went. We do not read this as the participant being unaware of the gap, as they name it themselves, but as this participant pushing on without fully understanding what AI did: AI "was just creating code for us", and the work continued. This is where, in our reading, ceding decision-making shades into surrendering judgment, because code a participant cannot follow is also code they cannot evaluate. P21 is not the only participant who did not fully evaluate what AI produced: *"I'd actually say that I probably don't register how often it[AI] is incorrect."* P32 – Focus Group

Continuing while suspecting AI might be wrong was not something only P32 described. When we asked the focus groups what they did with information they could not check for themselves, this is what participants shared:

[5]Identifiers starting with FG denote participants who consented to take part in the focus groups, but not to the use of their survey data.

> **Moderator** *If you prompted AI with a task description, and it gave you back some information that you maybe didn't really know how to validate or how to qualify, what did you do in those cases?*
>
> **P14** *I would try to keep going.*
>
> **P22** *Cross your fingers*
>
> **Comment** *P16, P11, P12 and P19 all nod in agreement.*

This interaction, and the nods that accompanied it, suggest that several participants in this session chose to continue working with information they were not able to fully process: they were aware that AI might give them incorrect information, yet still accepted that possibility.

These quotes indicate that, at least initially, participants were able to get away with not verifying, or only very shallowly verifying, AI-generated information. However, this behavior eventually led to issues for participants. *"But if you tell the AI to do it, it'll just go. And then it'll complete it. And you don't actually see the issues with that design because it's just done it for you."* **FG49 – Focus Group** What FG49 describes is not a suspicion that something is wrong, but the absence of any occasion to notice: because AI would "just go" and complete the task, the problems it introduced into a design were never surfaced for the participant to see. Speaking about it afterwards, FG49 can say the issues were there.

Those problems still had to be dealt with eventually. However, given how some of these participants were using AI, and how little experience they had with the stack, we think they were working with a codebase they had produced but could not fully understand. This is the tension in how participants used AI: the reliance that let them create the codebase at the start of the project was also what began to cause their problems. Dealing with these problems was challenging: *"A lot of like the debugging is like, oh, here's this error that I'm facing. And then you make a change, and then you get a new error. And I don't really know if I'm stepping further or closer to my end goal, I guess."* **P44 – Focus Group** In this case, the participant describes being unable to make sense of the AI-generated information, but maybe more importantly, also not knowing whether relying on AI was helping them solve their problem. This situation, of being stuck without knowing whether they were making progress, was experienced by several participants during the project: *"[I would do] whatever docker would allow. Like it would give you an error we try and fix it and then it give a different error try and fix that"* **FG52 – Focus Group**.

For some participants, working through errors with AI did not merely stall progress but escalated the problem itself, as FG50 recalls another Docker issue: *"we couldn't give all of the information so maybe AI got confused [...] so that's where it[AI] started hallucinating stuff and that like kind of led to a lot of bad answers through that and then it just kept spiraling and we ended up in like a much bigger problem than what we started with"* **FG50 – Focus Group**. Looking back, FG50 can name what went wrong, though even in hindsight the account stays hedged: it is "maybe" that AI got confused. What we take from this is that the recognition came late. By their own telling, the answers "kept spiraling" into "a much bigger problem than what we started with" before the hallucination was caught, which suggests there was no point during the incident where the participant could have caught it.

We read these accounts as showing what that reliance cost participants: having lost their grip on the code, they became even more dependent on AI to solve the bugs and issues they ran into *"we relied heavily on AI to figure out how to do any of the development. No one in our group had development experience to the level that this project required so generative AI was a massive crutch for us."* **P22 – Survey** The phrase "massive crutch" names the dependence: a crutch is

what you need in order to move at all, and P22 describes a group that could not have progressed, or resolved the bugs they hit, without it.

When participants were stuck and relying on AI like this, escaping rarely meant repairing their own understanding; instead, it often meant deepening their reliance on AI. When we asked participants whether they checked or validated the information AI gave them while stuck, 3FG50 described their first response: *"I think one of the first things that I did, because of the time constraints, was just use a different AI model. But it kept giving similar answers [as the first model]."* FG50 – Focus Group This answer reveals how deep the reliance ran: prompted about verification, the participant describes consulting another AI model. Verification itself was delegated to AI, and, in this case, without payoff, as the models "kept giving similar answers". Even when switching models did resolve the problem, participants' own understanding was not repaired:

> *There is a problem I was having where I was trying to make a request to the database and it would fail because of the time was in the wrong format, and I could not figure out why it was in the wrong format [...] until I switched which agent I was using and then it immediately figured it out. The first model kept giving information about why it could be wrong and then implementing the fixes and it was just wrong every time [...] and then it just gave a very concise reason of why exactly it was wrong.* P25 – Focus Group

These two accounts are why we consider the reliance self-reinforcing. Neither participant adjusted their AI usage or repaired their understanding when they got stuck; both reached further into AI. For P25 that worked, but the resolution stayed entirely within the AI: the new model gave "a very concise reason of why exactly it was wrong", while, by their own account, the participant had not been able to work out the problem themselves. Other times, reaching further into AI reliance resolved nothing at all: *"Once I started using it[AI] to program aspects of the project I had never done before (backend, database), it[AI] started changing a ton of things at once and it was difficult to know if it[AI] was on the right track. It[AI] never reached the desired functionality."* P26 – Survey P26 describes the cycle running its full course without any payoff: the parts of the project they knew least about were the parts they handed over most completely, AI "changing a ton of things at once" left them less able to tell whether it was "on the right track", and at the end of it the functionality was still not there. Where P25 at least got a working answer they did not understand, P26 got neither.

To us, these accounts also show the central organizing concept of this theme: what participants surrendered to AI was their judgment. They stopped evaluating what AI produced, whether information or code, and once they had surrendered their judgment, their reliance on AI only deepened.

4.1.3 An emotional toll that left little reason to break the cycle. To understand what surrender left participants with, we must consider not only how they used AI, but also what they told us it felt like. After surrendering, participants described feeling ashamed that they relied on AI rather than on their own knowledge, and frustrated with an AI that was confidently incorrect, which some of them experienced as gaslighting. They also described an emotional disconnect from work that had come to feel pointless and empty. In our reading, this emotional toll left them with less reason to break their reliance on AI.

Some of the participants appeared to be very aware of their own reliance on AI, and as P19 discusses, for them it triggered a sense of shame: *"From my perspective, shame I have to rely on it [AI] rather than my own knowledge."* P19 – Focus Group For this participant, a novice developer, the issue is not that the tool was unhelpful but that leaning on it "rather than my own knowledge" was something to be ashamed of: what they say about AI is really a statement about themselves. P22 describes the same relation between AI's capabilities and participants' own knowledge,

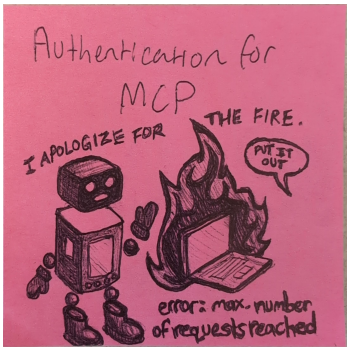

Fig. 3. Participants of the focus groups were also given sticky notes to take notes; FG53 also drew the following picture in addition to sharing their metaphor. The top half of the sticky note contains some unrelated text related to a task this participant worked on during the project.

but at the level of a team: *"None of us felt like we knew more than the AI did, so it was like, why would we not just trust it?"* P22 – Focus Group The trust here rests on a comparison, and it is a comparison P22 describes their whole team losing: no one in the team *felt* they knew more. Together, the shame P19 names and the deference P22 describes show how some participants who were deeply reliant on AI felt they did not possess the knowledge required to do some of the work themselves.

Some of the participants who relied on AI also trusted it, and that trust ran into a well-documented property of LLMs: their tendency to be confidently incorrect [70]. Based on some of the closing metaphors, we think these instances where AI was confidently incorrect were particularly challenging for some of the participants. In fact, FG51 used the word "gaslighter" to describe AI:

> *The term I started with [to describe AI] was like intelligent gaslighter because honestly it does feel like it then knows a couple of things it does give you it could be very helpful it gives you the answer you sort of want but it just sort of says like I'm right you're wrong trust me on this it should work the way I tell you to.*
> FG51 – Focus Group

We believe the notion of *gaslighting*[6] captures what FG51, and some of the other participants, were reaching for. FG53, who joined a different focus group, described AI in similar terms, and even drew a pictogram related to this idea (Figure 3):

> *I said Copilot is like that friend that you meet and think he's really smart. So you talk to him a lot, but then you get sick of him because you realize he doesn't actually know what he's talking about. And he's been manipulating you and spending a lot of time with you.* FG53 – Focus Group

While AI does not have any malicious intent, which is why the definition does not fully apply, we do believe that it felt like gaslighting for some of the inexperienced participants when the AI they trusted was confidently incorrect and made them doubt their own knowledge. The following interaction with FG52 shows what that could look like in practice:

[6]Defined by Merriam-Webster: "Gaslighting is a form of psychological and emotional abuse where a manipulator makes someone question their own memory, perception, or sanity. The goal is to establish power and control by forcing the victim to depend entirely on the abuser's version of reality."

> **FG52** *I think there were definitely times it[AI] wouldn't listen. It[AI] would say this[change to the backend] is necessary, which would make me consider it, but ultimately I wouldn't change the backend.*
>
> **Moderator** *So when you say AI would suggest changing the backend, you said you would consider it. Why was this?*
>
> **FG52** *Just because I didn't have perfect knowledge of what I was trying to do in the front end. You know, or what was actually needed. Obviously, I don't trust it[AI] 100% but it has better knowledge than me, and it's telling me that I need to do something.*

In this interaction, FG52 explains how AI kept insisting a change to the backend was needed, even though they thought it was not. Notably, this participant held their ground: "ultimately I wouldn't change the backend". What the interaction shows, then, is not that AI overrode their judgment, but what it cost them to keep it. Because they did not have "perfect knowledge" of what was needed, a tool they granted "better knowledge than me" could still make them reopen a decision they had already settled, and FG52 describes this happening repeatedly, as something AI "wouldn't listen" to. We take this to suggest that even where judgment survived, participants had to spend something to defend it against a confident AI.

For these participants, relying on AI did not just leave them doubting what they could do themselves; it could also be frustrating: *"It [AI] was useful but also frustrating as it would often spew out code that did not work correctly."* **P29 – Survey** When we consider these elements together, the frustration, the shame expressed by P19, and the sense of working with a tool that felt like it was gaslighting them, we can see why some participants described feeling disconnected from the project:

> **P5** *But prompting for four months is, like, pointless.*
>
> **P16** *That's a really good point. I feel like I just didn't care. It just was, like, very emotionally disconnected from the work. Like, I was, like, had no pride in it. I was just, like, just...*
>
> **P22** *Yeah, I don't really take much ownership over it either. Like, it doesn't feel like my work.*
>
> **P16** *True.*

This disconnect recurred among participants who described relying heavily on AI, though they each put it differently. For some, the work itself came to feel hollow: *"SENG-wise, like, making this, I think I agree with the sentiment that it just felt kind of empty"* **P16 – Focus Group**. However these participants described it, whether as the project being "pointless", as feeling empty, as taking no pride in the work, or as not taking ownership of it, the common thread we see is a loss of motivation for the project. We read these accounts as describing participants who no longer saw value in the work they were doing, or a reason to apply themselves to it.

We should be careful not to attribute all of this to AI. Some participants commented that the project itself felt purposeless, and a project a student sees no point in would dampen motivation regardless of how its code got written. Still, we believe the reliance amplified this disengagement, because of what participants attach their disengagement to when they describe it: not the brief or the topic, but "prompting for four months", not taking ownership of work that "doesn't feel like my work", and the shame and deference discussed above.

One way this showed itself, as P45 recounts, is that participants appear to have stopped caring about the quality of what they were building, letting suboptimal code stack on suboptimal code:

> *I found there was a lot of tech debt that you didn't really know was there until you ran into it. But AI kind of just kept on adding to it and found its own workaround instead of actually fixing the tech debt. So then it never got done the right way. It was just a fix on top of a fix on top of a fix on top of a fix.* P45 – Focus Group

This participant does appear to be aware of quality issues in the code, though they also describe technical debt they did not know was there until they ran into it. That AI kept adding to that debt without repairing it, and that the participant let it, reads as a sign that this participant had stopped caring about the project they were working on. This not caring may also connect to the shallow verification discussed by some of the participants in the second insight, as we think the two might reinforce one another. Once a participant no longer cared about the project, why would they carefully verify the AI-generated code or information being added to it?

Another element we consider part of this insight is the feeling some of the participants had that they were not learning anything. *"It[AI] is useful for implementing this project, but didn't help learning."* P11 – Survey We think this feeling, that AI did not help them learn, may have further contributed to some of them disengaging from the project. It matters especially because learning was part of what drew participants in: they told us they were motivated by the chance to pick up new languages and to work with AI. What the project delivered instead, on their accounts, was dependence on a powerful tool that could confidently feed them incorrect information and leave them stuck.

These various experiences that participants shared, of how their AI reliance left them feeling disconnected from the project, do more than describe an unpleasant experience. We think they also help explain why it became even more difficult for participants to break out of their reliance. A participant who finds the project "pointless" and takes no ownership of it has little reason to do the harder work of rebuilding their own understanding; it is easier to keep asking AI to make the code. The way P19 talks about the absence of accomplishment hints at this:

> *The most of it [excitement] came from just saying, you know, like, magic, make me this code. There is no, there's no sense of ownership, there's no, oh, I am able to create this thing, it's, there's no real sense of pride there.* P19 – Focus Group

Because "magic, make me this code" produced the project, these participants were denied the experience of "I am able to create this thing", the sense of mastery that might have rebuilt their confidence. This loss of motivation, then, did not only feel bad; it gave participants little reason to break the cycle, leaving the self-doubt that first pushed them towards AI in place. Some participants described this momentum directly, as a slope they could not get off: *"yeah I guess slippery slope almost like once we started it's like it's already like it's already some areas chopped like you're just gonna keep going."* P6 – Focus Group For us, this is why it matters to consider how the reliance made participants feel. It was not only uncomfortable; it was self-worsening: the less participants cared about the project, the fewer reasons they had to reclaim it from AI, and the more of it they left to AI to produce.

Across its three insights, this theme traces how participants spoke about a reliance on AI that deepened until they surrendered their judgment, and about the emotional toll of that surrender, which, in our reading, made the cycle even more difficult to break. The second theme begins with what these participants lacked: the knowledge required to judge what AI was doing.

4.2 Theme 2: Applying human judgment and the knowledge judgment rests on

The experiences of participants were diverse, and in addition to talking about surrender, participants also spoke about what it was like for them to apply their judgment. This second theme is centered around these accounts. We constructed this theme around three things participants described: applying their own judgment; the knowledge and experience it rested on, whether brought into the course or acquired during it; and the strategies that kept that judgment in their hands while they worked with AI. One participant states the connection between knowledge and judgment directly:

> *Yeah. I think that just points to the importance of you need to actually understand the underlying technologies, tools that you're dealing with, because it just helps with being able to give the right context and problem achieving, as well as understanding if it's somewhat in the realm of a good answer or not.* P36 – Focus Group

For us, this participant summarizes what we construct in the remainder of this theme: understanding the underlying technologies is what allowed participants to give AI the right context and to judge whether what came back was, in the words of P36, "somewhat in the realm of a good answer". As mundane as it sounds, a great deal of implicit prior knowledge is required to form such a judgment. The way this participant talks about a "good" answer implies a taste that goes beyond asking whether the response achieves the goal, and it speaks to how they cared about the quality of that response.

The first two insights of this theme describe where this understanding came from, pre-existing knowledge or knowledge that participants acquired during the course; the third insight describes the strategies participants used to keep it.

4.2.1 Knowing from prior experience what good code looks like. Some of the participants who spoke about applying judgment over AI, also referred to the experience they already had before starting the project. Their experiences reveal to us what it felt to lean on your own knowledge to judge what AI was doing.

> *I think I would like all the code that the AI made I reviewed and like make sure that I understood and if there was a problem I knew how to debug it and like fix any problem that there was yeah because I had a co-op in TypeScript and like basically this whole tech stack so I actually kind of knew what was going on* P5 – Focus Group

P5, whom we quoted in the first theme describing AI as "the only glue" that let the class progress, gives a different account of their own work when asked whether they vibe-coded their entire project. That they treat their familiarity with the tech stack as the reason they did not vibe-code it reveals how important they felt their own knowledge was. Moreover, the way the participant states they were able to debug any problems reveals not just familiarity, but also confidence in their own abilities.

We noticed similar ideas in how P3 discussed their pre-course familiarity with software diagrams and TypeScript: *"even in like previous classes before taking this class I've seen stuff like that [software diagram and typescript] so it was a lot easier to not have to rely on it[AI]"* P3 – Focus Group P3 constructs it quite clearly: they did not have to rely on AI because of their pre-existing knowledge, both in software design and in TypeScript, the language required for the project.

> *I don't know if I'd call it a challenge but part of the procedure for having GenAI create good code requires several iterations and revisions. Everything I had it do for me were tasks I knew how to do myself, so it just accelerated my working speed. When it makes mistakes, I catch them and tell it to fix them.* P43 – Survey

While talking about using AI for the project, P43 also talks about "good code", which suggests that participants like P43 had some ideal, or reference point, for what good code looks like. Developing this kind of taste reveals how this participant used prior knowledge to form judgments about the code AI was producing. This participant only used AI to write code they could judge themselves.

> *I kept prompting it[AI] to do something again and again. I would usually look through what they had given me and see if I can find any problems like what are the specific errors that I'm encountering or what do I not like what about the first way that they did it and then at least for copilot there's a way to restart the context like go back on the context and then I would probably like rewrite the prompt to include what I didn't like or liked about the first one or what I thought I had missed context wise.* P44 – Focus Group

While working on the project, we believe this participant had a very specific idea about how the issue should be solved. P44's idea of what should be done went beyond functional correctness: alongside checking for errors, they asked themselves "what do I not like [...] about the first way that they[AI] did it", which implies some taste, or subjective ideal, they wanted the code to meet. To make this kind of judgment about the code AI writes, we believe participants must have drawn on their prior knowledge and skills; it is only through prior experience as a developer, we think, that one develops the taste required to be critical of the code AI generates. We found the same in how P37 talks about working with AI: *"Sometimes without doing that [specifically adding context], it[AI] would start trying to make changes in different places that you didn't really want it to."* P37 – Focus Group Both quotes show how involved these participants were with what AI was doing, so involved that they were judging how AI was solving issues. In addition to requiring prior knowledge, these quotes also suggest care about the quality of the project that contrasts with the experiences described in the first theme.

The prior experience commented on by some of these participants might imply there was a clear dividing line between the participants, with the group without prior experience surrendering and the group with experience preserving their judgment. However, we do not believe there is such a clear separation between these groups; there are also accounts of participants who initially surrendered judgment but, throughout the course, restored it. The second insight describes how the latter group did so.

4.2.2 Recognizing the gap in knowledge and repairing it. Unlike the participants quoted in the first insight, some participants spoke about the steps they took during the project to acquire the knowledge needed to judge what AI was doing.

Building that knowledge required space the project did not initially afford participants, and part of that space came from changes to the course. Before the first-phase deadline, students expressed concerns to the course team, explaining they felt they could not meet the deadlines. This prompted the course team to adjust both deadlines, offering a grace period for the first and lessening the requirements for the second. Then, at the halfway point of the course project, immediately after the Phase I deadline, a lecture and a lab were given to students. The material covered concepts like

context engineering and how to apply custom instructions, and it was based on a combination of gray literature articles collected by the course team. Both were organized around the three-phase workflow described in Section 3.1.

Participants commented on these changes in the focus groups and in the surveys, which is why we believe they were so important: *"I relied on it[AI] a bit less because of the reduce[d] workload so I was able to do a bit of research about things rather than just asking it[AI] to do things "* P10 – Survey This quote shows how important it was for participants to feel like they had the space to research "things". In this case, a relatively small change in the conditions was perceived by this participant to afford a different way of working with AI. What P10 says they did is relevant: they were able to research and restore the understanding they had lost during the first phase, and it was through building that understanding that they developed the knowledge needed to apply their own judgment.

Participants also describe becoming aware that the way they had been working with AI was not right, and explain how the course material showed them they could work with AI differently.

> P32 *I'd say how to use AI properly is definitely emergent through the course.*
>
> FG49 *Yeah*
>
> P32 *The professor talking about how to maybe scaffold a way to use it properly actually got me thinking about how to use it properly*

P32's use of the word "properly" implies that, at some point during the first phase, working with AI felt improper, suggesting a moment of self-awareness. While this quote does not tell us whether the lecture triggered this realization or whether it had happened before the lecture, it does suggest that becoming aware of their AI usage was an important element in moving from surrender to applying judgment, as participants who became aware then began to improve their understanding and knowledge.

When it comes to realizing that they had to improve their knowledge, one team's experience stood out in particular. Just before the first deadline, there was a catastrophic failure that appeared to trigger a moment of realization shared by the entire team.

> *Before, we just vibe coded to its most extreme level. Didn't care about outputs, just took it and ran. Once that failed spectacularly, we switched to using it much more sparsely and cautiously.* P41 – Survey

Based on what the members of this team told us, it is apparent they were unable to continue working on their project after the first deadline. It shows how the structure of the course was instrumental for some students; without the initial tight deadlines, the participants from this team might not have ended up surrendering. While P41 did not refer to the change in course deadline, we think the change must have played a role in helping these participants recover their judgment. For P41 to refer to this switch as "using it[AI] much more sparsely and cautiously" again implies that this participant was actively aware of how the initial reliance had been problematic. The failure is hinted at as a catalyst, showing how this participant constructed boundaries around their AI usage in response, leading to an internalized notion of what it means to use AI sparingly and cautiously.

Recognizing the problem was not by itself enough to work differently. What this team did next is the clearest example in our data of participants deliberately acquiring the knowledge required to apply judgment, which this insight rests on:

> P24 *uh for team [redacted] for implementation II we all spent three hours out of our day and watched a video on nestjs and came to the conclusion that AI is not good for helping you with coding*
>
> ...
>
> P24 *And then we came to the conclusion that the AI cannot do NestJS complete.*
>
> P41 *Yeah. It needs a bit more structure.*

In the middle of a project with demanding deadlines, all team members committed to watching the same series of YouTube tutorial videos on the framework used in their project, spending "three hours out of our day" on it. What makes this notable is what the knowledge was for: it was only after learning the framework themselves that the team "came to the conclusion that the AI cannot do NestJS complete". Until then, we think they were unable to judge the code AI produced for this framework. That a relatively short series of videos was enough also reveals the importance of the prior software engineering knowledge these participants already held.

Two further things stand out to us. For this team, acquiring the knowledge was not an individual action: making it a group activity shows the whole team thought it important that everyone could apply judgment to what AI was doing. And the way participants speak about the realization implies that they recognized not only that AI has limitations, but, as P41 alludes to, that it is their own responsibility to give the models the structure they need.

The same acquisition of knowledge and experience happened more quietly for other participants. Before starting the project, many participants did not feel very familiar with web development; throughout the project, however, this started to change: *"I tried to use it[AI] less in phase 2 as I felt slightly more confident with the TypeScript basics"* P8 – Survey This quote shows how becoming more familiar with the web technologies required for the project was instrumental for this participant in reducing their AI reliance. However, it also suggests that, during Phase I, this participant started to build some of the experience required to judge AI output.

In addition to becoming more experienced with web development, some participants also commented on the familiarity with AI they developed as the course progressed. Even though participants mentioned they had worked with AI tools before the course, for most of them using AI this intensively for software engineering was a first.

> *I got more proficient using [AI]. I was able to manage the scope of change much more fluently, and move slow with AI, but with higher quality codes. I found myself using less [AI], and wanting to write more of the code by my own hands* P6 – Survey

P6 highlights how the feeling of becoming more proficient with AI gave the participant the space to "move slow". Like the quote from P10, we think this shows how the participant constructed working with AI: there is a trade-off where uncontrolled AI usage was quicker, helping participants achieve goals for the project (a functional system), but at a cost to their own ability to judge, and at a cost to quality. As we have seen in the first theme, some participants hinted at being aware of quality problems in their code, so it makes sense that participants felt moving more slowly allowed them to improve quality. What this felt like for P6 is worth noting: as they became more proficient, they describe "wanting to write more of the code by my own hands", which reads as the opposite of the emotional distance from their own project that participants described in the first theme. It suggests engagement and enjoyment in doing the work themselves, instead of surrendering their judgment to AI.

> *So I said, AI was like milk and a cup of tea. A little bit would enhance the drink, but too much could ruin it and require you to remake it.* P41 – Focus Group

In their closing metaphor, P41 describes what we consider to be a core element of this realization participants had. Used purposefully and carefully, AI enhances their work, but overuse has serious, even irreversible, consequences. Knowing there is a "too much" implies that participants discovered a boundary around their AI usage, and it shows how participants of this team appeared to care about the quality of their code, actively sought information to restore their judgment, and were able to dose the AI.

4.2.3 Dosing the milk — strategies that kept judgment in participants' hands. When participants applied their own judgment, they did so through strategies for working with AI that they discovered during the project, or in the words of P41, they dosed the milk. These strategies are how participants put the judgment described in the previous insights into practice. What they have in common is what they protect: the chance to notice when AI was going the wrong way, a unit of work small enough to check, the decision about what to build, and the participant's own understanding of the code. We believe that, beyond their practical value, discovering these strategies also restored some of the confidence in their own work that participants described losing in the first theme.

We consider these strategies noteworthy because they are what worked for this cohort. Unlike senior developers, these participants could not draw on years of knowledge and experience, so the ways senior developers offload large parts of an implementation to AI are unlikely to have worked for them. Instead, they discovered strategies that suited their own knowledge and experience, and their mastery of working with AI.

For some participants, how they had learned to work with AI was difficult to put into words:

> FG48 *Like, whenever it would start to go off track, I'd just kill that chat and start a new one.*
>
> Comment *P45 and P42 nod.*
>
> Moderator *And when you say off track, you mean?*
>
> FG48 *You just kind of can feel that it's going the wrong way. I don't know how to sort of explain that, but do you?*
>
> P45 *For me, it just keeps on giving the same answer without actually proposing a fix.*
>
> FG47 *It's, like, poisoning its own context.*
>
> FG46 *Or, like, when you're trying new solutions and it just, like, doesn't work. Yeah, the same reason, maybe.*

To us, this focus group interaction illustrates how important a sense of intuition was for the participants. Several of the participants appear to find it hard to name the signals they were paying attention to, instead only commenting that they could "feel" AI get off track. When prompted to explain this, participants not only refer to what we would consider intuition; they also appear to have discovered terminology, such as "poisoning its own context" (FG47), to describe the AI behavior they had become familiar with. Moreover, for P45, an important signal of AI going off track is it giving the same answer without proposing a fix. This is even more notable because, in the first theme, a participant encountered the same problem but did not interpret it as a signal to adjust their use.

> *There was still a heavy reliance on the use of gen AI due to the necessity to produce so much code in such a short amount of time, however, I found it more effective to maintain smaller chats and having only the necessary context for the immediate problem I tried to solve. This helped generate better code results. As well, using instructions helped better control the AI in providing some of this necessary context.* P37 – Survey

This quote is a response by P37 to a prompt asking whether anything changed in their AI usage from Phase I of the project to Phase II. It shows two things. First, even though the deadlines for Phase II were changed, the participant still experienced pressure, as evidenced by how they speak about having to produce so much code in a short amount of time. Second, the quote shows that this participant appeared to feel more capable after discovering strategies for working with AI. Both of these strategies, maintaining smaller chats and using custom instructions, are very specific approaches to context engineering[7]. We think this quote shows how the participant experienced a sense of control while working with AI.

We want to name P37 as a case that does not fit this theme cleanly. The strategies restored a sense of capability, but on this participant's own account neither the deadline pressure nor the "heavy reliance" itself went away. Judgment, here, is something the participant regained *within* a continued reliance on AI, rather than something that brought that reliance to an end. We return to what this implies in Section 5.

> P4 *i normally uh would use the co-pilot like built into vs code and i would make sure that it looks through documentation like i pointed towards the documentation requirements i need for a feature um i'll tell it to do an incremental steps so i can test it in between each like small portion of this uh requirement it needs to do*
>
> P9 *i was gonna say like as well like on top of that i feel like i did a similar thing like asking to do incremental and then also kind of like explaining or like making a plan first so it's like asking it to make a plan and also like explaining the reasoning and then i could say like okay actually like let's maybe do this instead like i'm kind of editing the plan if that makes sense*

Similarly, the two participants in this interaction refer to several specific strategies when asked how their AI usage changed. P4 describes instructing the AI models to consult the project documentation, another example of context engineering, but also having AI work in incremental steps that can be tested one at a time. P9 elaborates, describing how they had AI generate a plan first, which they could then manually tweak. The two strategies protect different things: the incremental steps keep the work small enough for the participant to check, whereas editing the plan keeps the decision about what to build with the participant. We read these strategies as showing that, throughout the course, participants not only became more familiar with web development; they also learned to use AI in a way that let them exercise their judgment: small, testable steps and editable plans gave them exactly the room to notice and correct what AI got wrong.

Not all strategies were aimed at controlling what AI produced. In the following exchange, P12 and P19 describe strategies aimed at understanding the code produced with AI:

[7] https://www.promptingguide.ai/guides/context-engineering-guide

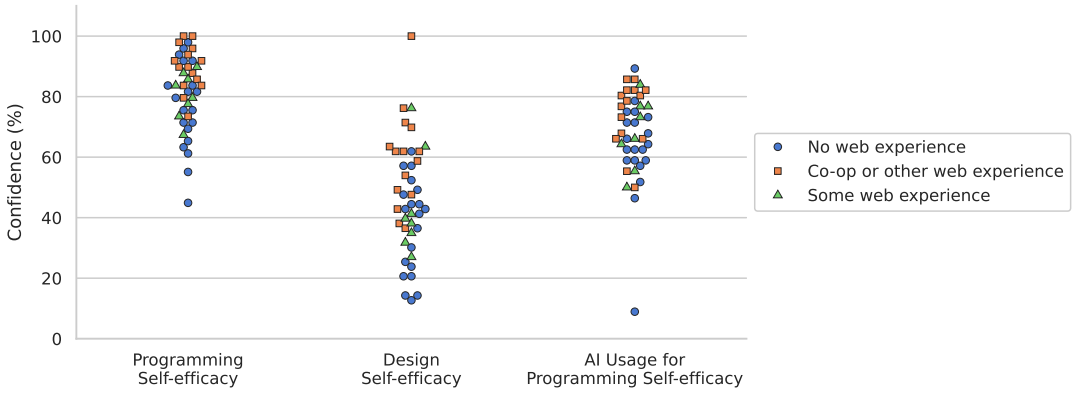

Fig. 4. The self-efficacy scores of the participants for the three knowledge areas (Programming, Design, AI usage for programming) divided by the prior experience of participants.

> **P12** *and I wanted to know what each line of code was doing for that specific feature.*
>
> **P19** *what he [Points to P12] said. I was just asking [AI] to write more comments in the code and explain every step rather than just "write me this thing".*

Both P12 and P19 refer to using AI in ways that helped them preserve their understanding of the code. While they do not comment on it explicitly, we believe this preserved understanding is exactly what participants needed to keep exercising the judgment described in the previous insights. Where the first two insights describe knowledge participants brought or acquired, this strategy is about not losing it again.

Both themes come from what participants told us in the focus groups and in their open survey responses, and together they describe two ways of working with AI: one in which participants judged what AI produced, and one in which they could not, or did not. Participants' answers to the closed survey questions, given before and during the project, give little sign of either experience.

4.3 Self-efficacy, and AI perceptions of the participants

In addition to the focus groups, we also deployed three surveys to participants throughout the project. The surveys contained a set of closed questions covering prior experience, self-efficacy, and the AI perceptions of participants. In this subsection we discuss those results, and how they relate to the themes we constructed from the focus groups and the open survey responses.

In total, 43 students participated in the self-efficacy survey at the start of the course. Their prior experience varies widely: 20 participants reported no experience with web development, 8 reported some experience, and 15 reported more extensive experience through co-ops or other web development roles. The survey item offered four response options. We merged two of these, having worked on a web application during a co-op or other term project, and free-text "other" responses describing comparable web development work, into the single "more extensive experience" level reported here.

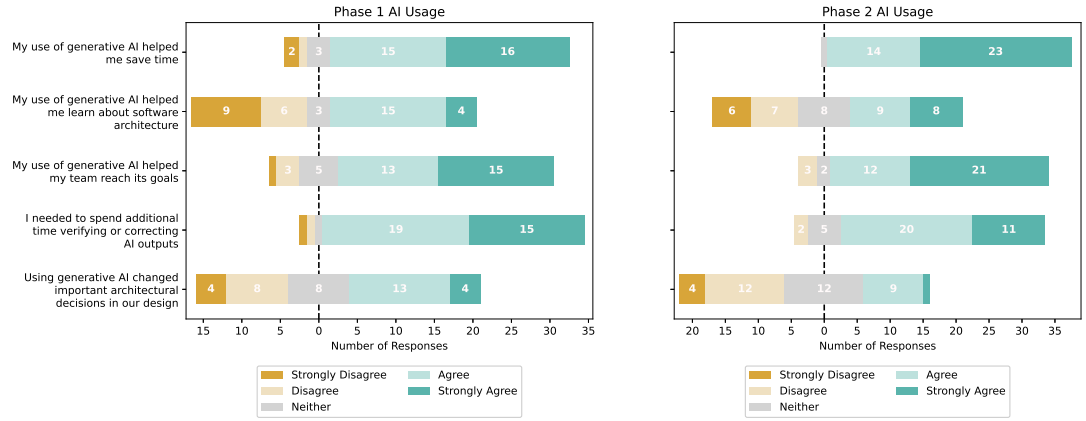

Fig. 5. Participant perceptions of AI after Phase I and Phase II.

In terms of self-efficacy, or the confidence participants reported in their own skills, Figure 4 shows the scores reported by each participant for the three categories (Programming, Design, and Programming with AI). A score of 100 indicates that, before the start of the course, a participant was very confident they could solve various tasks in each category, without AI assistance.

Taken together, these prior experiences and self-efficacy scores show that almost all participants reported high confidence in their independent programming skills, even those with limited prior web development experience. They also show that most participants felt confident in their software design skills, likely from prior courses on design patterns. Based on these prior experiences and self-efficacy scores, it appears that participants had the skills and confidence the teaching team expected. We do note, however, that this confidence concerned participants' general programming skills, not their confidence in the specific web development stack the project required, which nearly half of them reported no experience with.

After each development phase, participants completed a survey on their AI usage and perceptions (37 and 38 responses, respectively). Except for one student, who reported after both Phase I and Phase II that they had not used any generative AI,[8] all students indicated that they used generative AI, similar to other courses where students are allowed to use AI [56, 66].

Figure 5 shows participants' perceptions after each phase. Almost all participants reported that generative AI helped them and their teams reach their goals or save time. These perceptions suggest that, after each deadline, participants believed AI was effective and that it helped them complete the course project. This near-universal adoption and perceived usefulness is consistent with what large-scale surveys in comparable project-based software engineering courses report [56]. At the same time, it is notable that participants were less positive about how generative AI affected their learning, with 15 and 14 participants, respectively, disagreeing with the statement that AI helped them learn software architecture.

[8]This participant gave two responses when asked why they did not use AI: "*I'm in this program because I love coding and software and understanding how it all works. Generative AI is an easy way to get some code to exist (in whatever quality the AI spits out) but it does nothing to help me actually understand and learn. I think it's unfair to essentially require students to use a gen' AI to complete a project on time. If some students want to use AI to make it easier, let them, but don't expect everyone to use it. I'm here to learn the material myself. Once I understand what's going on then I can benefit from gen' AI. Thank you :)*" & "*I want to understand my teams code. I do not understand code without working through it myself. I want the documentation I write for my team to reflect my understanding, even if I could fake that understanding with genAI. GenAI makes me work faster at the expense of being able to understand what I am doing.*"

What stands out to us is that these closed survey responses give little sign of the surrender, or of the emotional toll that came with it; at most, the doubts about learning echo P11's sense that AI "didn't help learning". This suggests that surveys like ours are not suited to detect when students surrender their judgment, or what that surrender costs them. This is why we think an experiential account of AI adoption matters: it shows what working with AI is like for software engineering students, an experience that perception surveys have so far missed.

5 Discussion

In the focus groups, participants shared what it was like to work with AI. Some accounts revealed a disheartening experience: a deepening cycle in which judgment was surrendered to AI (*Surrendering Judgment to AI*). While speaking about surrender, participants also spoke of shame, frustration, and an emotional disconnect from their projects, a toll that, in our interpretation, made the cycle even harder to break. At the same time, participants also spoke about what it was like to apply their judgment to AI, how that judgment rested on their knowledge and experience, and how they cared about what they built (*Applying Human Judgment*). We believe our findings help uncover what the next generation of novice developers needs in order to adopt AI sustainably, and what must be done to preserve the motivation and well-being of junior developers.

In the remainder of this section, we use Self-Determination Theory as a lens to explain why the reliance on AI left participants feeling disconnected and demotivated. We then discuss what our findings imply for software engineering education and for junior developers in industry, and argue that both settings need diagnostics that detect when students and developers surrender their judgment to AI.

5.1 Why surrender felt empty: interpreting the findings through the lens of Self-Determination Theory

We constructed the themes inductively, based on the excerpts and codes. However, as encouraged by Braun and Clarke [12], we were open to theories that could help interpret and contextualize our themes. Specifically, we believe Self-Determination Theory (SDT) [55] helps explain how participants came to describe their work as empty and hollow. We also considered Flow [17] and Cognitive Load Theory [63], but found SDT the most useful for explaining the emotional disconnect that participants repeatedly described.

SDT is an established theory centered on human motivation and the conditions that enable humans to thrive. According to SDT, people have an inherent tendency to grow, but this tendency needs supportive conditions, and can be readily disrupted without them [55]. SDT further posits that motivation is not one quantity but a continuum of regulatory styles, ordered by how far the reason for acting lies within the person themselves. At one end sits amotivation, the state of lacking the intention to act, where people either do not act at all or go through the motions. At the other end sits intrinsic motivation, doing an activity for its inherent satisfaction. Extrinsic motivation covers the range between the two: acting not for the activity itself, but for an external reward. Its regulatory styles differ in how autonomous they are, from external regulation, where someone acts to obtain a reward or to satisfy a demand, through introjected regulation, where someone acts to avoid guilt or to protect their self-esteem, to identified and integrated regulation, where someone performs a task not because it is fun, but because they hold it to be personally important. SDT has been applied to a variety of domains, including education [55], and it has been used to understand how developers experience their work [69].

SDT states that three basic needs must be satisfied for someone to stay motivated and to thrive: **Autonomy**, **Competence**, and **Relatedness** [55]. Autonomy is the need to experience one's actions as volitional and as one's own. Competence is the need to feel effective and capable of meeting the demands of a situation. Relatedness is the need to

belong: to feel connected to others and cared for by them. Based on the experiences participants shared in Theme 1, we believe both their autonomy and their competence were thwarted.

According to SDT, thwarting these needs is likely to undermine an individual's motivation. Ryan and Deci [55] state this as a property of the environment rather than of the person. SDT places an individual at amotivation, the lowest level of motivation, when they do not feel competent to do an activity, do not value it, or do not expect it to produce the outcome they want. For instance, an individual might take a course because they want to learn a skill. However, if the course is too difficult, and the individual does not feel competent, they might come to work at just passing the deadlines of the course rather than at learning the skill (external regulation), and, if that continues, stop seeing any point in the work for the course at all (amotivation).

Determining the precise extent to which participants in this study felt motivated is not possible. The focus groups asked participants about their motivation (Table 3), but we did not use any validated instrument to measure it. However, some participants' quotes about the start of the course suggest that participants began with a relatively high level of motivation. For instance, P18 said they were motivated to learn web technologies. We do not read this as intrinsic motivation. P18 wanted to learn TypeScript because it is widely used in industry, which does not suggest the only motivation for this participant was learning typescript for the sake of TypeScript itself. SDT calls this identified regulation, where someone does the work because they believe it is important [55]. We think this fits a course project, as Ryan and Deci [55] note that people are only intrinsically motivated for activities they find interesting in themselves. As the course continued, participants describe conditions that, in SDT terms, would thwart their needs, and we hypothesize this undermined their motivation.

For autonomy, we think the structure of the project deadlines left some participants feeling like they had little control over their own actions. Ryan and Deci [55] already present deadlines and imposed goals as factors that undermine motivation, because they move the reason for acting outside the person. Several participants referred to the pressure they felt from the deadlines directly (P9, P37). Deadlines alone are not sufficient to thwart this need, but feeling overwhelmed, as some participants described being, would. On top of autonomy, we also believe these participants did not feel competent. P22 called AI a "massive crutch", and said their group could not have progressed without it. They also told us that "None of us felt like we knew more than the AI did", which suggests that some participants did not feel competent enough to complete the work without AI.

In light of SDT, we think how participants experienced their sense of autonomy and competence might explain why some participants came to describe their work as hollow, empty, or felt emotionally disconnected. We believe these accounts hint at a move towards external regulation, where the only reason left to act is to pass the project rather than to learn or to grow, and for some participants as a move towards amotivation, where they no longer saw a reason to act at all. We should be careful here, because participants kept working and all teams delivered a system. We therefore do not claim that participants fell into complete amotivation, but we do think our findings suggest that participants that surrendered to AI started to experience reduced motivation.

Our instruments did not cover relatedness, but prior work suggests that AI use can affect it too. Students who use AI report feeling less connected to their peers [16, 56] and to their instructors [56]. Our data hints at the same substitution. FG47 preferred going to AI over "asking a teammate or a TA or prof" because of the time constraints (Section 4). When students turn to AI instead of to their peers and instructors, we think this may also thwart their need for relatedness and further undermine their motivation.

Because we adopted SDT as a lens after we constructed the themes, Braun and Clarke [12] also suggest discussing how the theoretical lens might be limiting. There are two aspects we discuss. The first is relatedness, the sense of

belonging, which is one of the three needs identified in SDT, but which we did not extensively cover in the focus groups. Of the three needs, autonomy and competence are the ones SDT ties most directly to motivation. Ryan and Deci [55] report strong links between intrinsic motivation and the satisfaction of these two needs, and describe relatedness as bearing on motivation in a more distal way. They also note that people happily do intrinsically motivated work on their own. We therefore think our account remains consistent with SDT, as the emptiness described by participants might be explained by their lack of autonomy or competence. The second is the role of the team. Several accounts discuss the team: P5 describes AI as the only "glue" that let their team move forward, and the team of P24 and P41 reclaimed their judgment collectively, by all spending three hours watching tutorials on the framework they were using (Section 4). Our findings do not extend to, or discuss, the team dynamics.

Regardless of these limitations, we believe SDT is a good lens for understanding how the course and the deepening cycle of AI reliance affected participants' motivation. It helps explain why participants came to describe their work as hollow, and it grounds the implications we draw from this study.

5.2 Implications for software engineering education

The first set of implications that follows from this work concerns education. Existing literature links AI overreliance to cognitive debt, which Kosmyna et al. [36] measure as reduced neural engagement during AI-assisted tasks and Storey [62] describes as a team's eroding understanding of its system. Others speculate that overreliance might lead to skill atrophy [16]. To prevent these outcomes, prior work proposes AI literacy: making students aware of the limitations of AI and teaching them to work with it [4, 15, 16, 29, 56, 59]. Our findings suggest that AI literacy is only part of the picture: participants whose accounts we report in Theme 1 knew AI could give them incorrect information and accepted that risk anyway (Section 4), so awareness alone did not prevent the reliance that left them emotionally disconnected. Likewise, agent skills or custom prompts that claim to adapt AI models to support learning do not address any of the conditions that we believe led students to surrender their judgment. Participants described working with a technology they felt knew more than they did, and that at times felt like it was gaslighting them; teaching them how to work with it, we think, only partly addresses that experience. Two further conditions matter: the knowledge students have before they use AI, and the environment in which they learn.

First, students need relevant knowledge before they work with AI, because without it they have nothing to ground their judgment on. The strategies participants described in Theme 2, such as starting fresh chats, writing custom instructions, editing the plans AI generated, and working in small, testable steps, are worth teaching, but each rested on knowledge of the stack that participants brought or acquired. Our participants were senior students, yet many worked with a stack they did not know (20 of 43 reported no web development experience), and the ones who reclaimed their judgment did so by first building the knowledge they lacked (Section 4). We therefore think it may be better to withhold AI from students, even in senior software engineering courses, at least until they have built the knowledge a project requires. Restricting AI is not a new idea: some instructors have banned generative AI in their classes, and while students disagree with outright bans, a majority support some restrictions on its use [51]. Choudhuri et al. [16] propose a milder form, in which students alternate between AI-assisted work and supervised work without AI, so that the cost of delegating their thinking becomes visible.

Second, the environment in which students learn matters. Participants described deadlines that left them little control over how they worked, and some told us the relaxed deadlines gave them room to research rather than delegate (Section 4). We think students whose sense of autonomy is undermined in this way are more likely to surrender their judgment to AI, and because AI can complete much of their coursework, that surrender may go unnoticed. Abbas et al.

[1] argue that an environment in which students derive satisfaction from mastering concepts independently mitigates overreliance.

Read through SDT, restricting AI and shaping the learning environment protect the same thing: the competence and autonomy students need before AI can support their work rather than replace their judgment. It is therefore imperative that software engineering education is designed so that students never reach the point where they surrender their judgment, because the slide towards amotivation that follows frustrates the development of the skills they need in their degree and future careers. Ryan and Deci [55] report that more autonomous forms of motivation are associated with higher-quality learning, better performance, and more engagement. That is what students who surrender stand to lose.

5.3 The vulnerable position of junior developers

This research project studied the experiences of students in a software engineering degree program. However, based on reports of AI adoption in industry, we think junior developers might be at risk of the amotivation the students described. Three of the conditions we observed in this study also hold for many junior developers entering industry: an unfamiliar stack, delivery pressure outside of their control, and an expectation to use AI to enable productivity gains.

The first two conditions are long-standing. Junior developers are often asked to work in languages or frameworks they do not yet master, and the demanding deadlines in industry have been extensively documented [69]. The third condition is more recent. Miller et al. [41] describe a so-called "productivity paradox", where developers who received access to AI tools were expected to be more productive. Not all of these developers were immediately able to use AI effectively, and they described feeling more pressure to be productive. Read through the lens of SDT, these accounts suggest to us that the expectations around AI might undermine the autonomy of these developers in much the same way. The three conditions also bear on competence: junior developers starting their first job in industry have long reported feeling insecure when working with unfamiliar technologies [10]. Interpreted through SDT, this insecurity might be a sign that their need for competence is thwarted.

How these conditions might affect junior developers is visible in how the participants came to describe AI. At the end of each focus group, we asked participants for one or two words, or a metaphor, for the role AI had played in their project (Table 3). Table 4 lists all metaphors. Many are positive, and those that cast AI as a solution or a time saver echo the survey responses (Figure 5). Several, however, cast AI as knowing more than the participant: a mentor, a subject expert, a "god" (P12, P16). Others turn on its confidence: "an overconfident friend" (P8), a friend who has "been manipulating you" (FG53), an "intelligent gaslighter" (FG51). Three participants cast AI as a junior developer, but P45's junior developer knows everything.

Together, these metaphors reveal how the participants, who were still novices, came to view AI throughout the project. They show how participants made sense of the characteristics of AI, such as it being confidently incorrect [70], and how some came to see AI as an all-powerful knowledge base. To us, these metaphors further suggest that the participants who surrendered did not feel effective while working with AI. On top of the amotivation we described, this suggests that after they have surrendered their judgment to AI, juniors are left working with a tool they do not feel able to judge. We believe this is not a desirable situation, and therefore we pose the following question, which we believe is most important for the future of junior developers:

> *How do we help developers and organizations adopt AI sustainably, to preserve junior developers' sense of craft and ownership?*

A solution P3	*A "god", an "all-powerful subject matter expert"; dulling, and overpowered* P16	*With great power comes great responsibility* P26	*"like milk and a cup of tea. A little bit would enhance the drink, but too much could ruin it and require you to remake it"* P41	*Hasty: it accelerates development without regard for quality; and high level: it let us focus on the broader design while it did the fine things* FG47
A taxi driver from a small town on the west coast, thrown into downtown New York P4	*A junior developer: as if I'm a senior developer overlooking a junior's changes to see whether they are correct, and reminding them what we need* P18	*A double-edged sword: sometimes it saved me time, other times I felt like I was wasting my time* P29	*Helpful, and manual labor: it took away some of the busy work, filling in the things you didn't really want to do* P42	*A time saver, and a funnel: give it something broad and it narrows it down to what you want* FG48
Trying to cross the ocean without a boat or a plane; AI is what helps you cross, and the project would not have been possible without it P5	*Terrifyingly effective instant gratification: you immediately have a working thing, but get no benefit from it and you lose your brain* P19	*A gateway, giving quicker access to information; and a bridge that you're crossing in a 10-ton truck without knowing the bridge's capacity* P32	*Two steps forward, one step back: part of why I could do anything in this project, but I haven't gained as much as I could; and swimming with water wings for the first time: they make it difficult to swim, can be dangerous when you flip over, and using it feels like a bit of a disservice* P44	*An actual assistant, like an unpaid intern* FG50
"an overconfident friend [...] who sometimes gives you a wrong answer" P8	*"I felt completely dependent on it[AI], and that it was my best friend and greatest enemy"* P22	*A junior developer and research assistant that needs a lot of hand-holding* P36		*An "intelligent gaslighter": "I'm right you're wrong trust me on this it should work the way I tell you to"* FG51
A gateway to information; help, for the most part, but sometimes it trips up and is wrong P9	*Good at small tasks, but always has to be monitored* P24	*My tutor, teacher, expert, architect, programmer, brainstorm partner; not something to rely on solely, but it sped up the process* P37	*A junior developer, because I was managing what code I wanted; but a junior developer that knows everything* P45	*A guiding assistant: it had more knowledge of the content than me, but it was still an assistant* FG52
It did everything; a 100x software engineer P11	*A coworker: like a code session with somebody more experienced, who you can bounce ideas off of, or who corrects you* P25	*A mentor, because "he" knew more than me* P38	*A subject expert, because it just knows everything and you can ask it everything* FG46	*"that friend that you meet and think he's really smart [...] you realize he doesn't actually know what he's talking about. And he's been manipulating you"* FG53
A "god" P12				
Incredibly helpful P14				

Table 4. The closing metaphors of the 31 participants who shared one (Table 3), ordered by participant identifier down each column. Darker notes mark the metaphors we discuss in the text. Wording in quotation marks is verbatim; the remaining entries are condensed from the transcripts.

Existing reports about AI adoption in industry already reveal how developers might be at risk. In a longitudinal study, Vella and Blincoe [65] find that over time, 25% of developers reported that AI was reducing their developer experience. In August 2026, a video in which Brett Codes discusses why they stopped using AI for coding went viral on YouTube.[9] Responding to this video, Gergely Orosz, author of *The Pragmatic Engineer*, reported that they had also heard signals of developers burning out because of AI.[10]

One of the reasons Brett Codes gives for stopping is very similar to some of the accounts given by the participants in our study: "Using these AI agents for coding creates this apathy towards the work and the software. At least that's what I felt. Where I just care significantly less about things the AI generated." Unlike our participants, Brett Codes has 19 years of programming experience. At the same time, we believe they describe an experience very similar to what participants in this study went through. In their account, Brett Codes also describes external pressure: they felt pushed to be more productive with AI, and were mandated to use it. That even an experienced developer reports such an emotional disconnect shows us that the surrender we observed might not be limited to this study.

As relevant and important as we consider this question, we think that the tools and methods currently proposed to help junior developers adopt AI sustainably fall short. In Section 5.2, we argued that AI literacy, which prior work proposes for students [15], addresses only part of what led participants to surrender. Proposals aimed at industry share this focus on competence: Kam et al. [34] state that professional developers need to master more than working with AI, and also need software engineering skills. This recommendation is relevant. However, our findings, interpreted through the lens of SDT, may suggest that a focus on competence alone is not sufficient to prevent amotivation. The findings of Choudhuri et al. [16] already hint at this, as they conclude that for some students, an active awareness of the limitations of AI does not in itself deter a reliance on AI.

The participants in our study who reclaimed their judgment did so after the conditions of the project changed. The course team relaxed the deadlines and taught a lab on working with AI, and participants' accounts suggest that both mattered: some told us the course material showed them they could work with AI differently, and P10 told us the reduced workload was what gave them room to research rather than to delegate (Section 4). P6 describes that room from the other side, as being able to "move slow with AI, but with higher quality codes". In SDT terms, the two changes addressed different needs: the lab supported competence, and the relaxed deadlines supported autonomy. Both might therefore have changed how participants regulated their motivation while they worked with AI. Our results suggest how important these two different interventions were in helping participants reclaim their judgment. What we take away from this is that teaching juniors to evaluate AI output might not be enough to help them reclaim their judgment, if the conditions they work under leave them without a sense of autonomy. Of the three conditions we described in this study, delivery pressure and AI usage mandates are the ones an employer controls most directly.

We hope the accounts we report help organizations and junior developers adopt and work with AI in a way that preserves the craft and motivation of junior developers. As junior developers are expected to acquire new skills through their work, it is important they work in environments that facilitate that. Whether developers will still write code by hand is an open question in industry,[11] but we do not think the answer removes the need to learn on the job. Our participants worked with the AI tools available in 2025, and those who had AI generate their code still got stuck; those who got unstuck on their own terms first built the knowledge they lacked. We therefore think junior developers will still

[9] https://www.youtube.com/watch?v=2ZU3j4GQ4K8
[10] https://x.com/GergelyOrosz/status/2088567020623151475?s=20
[11] https://newsletter.pragmaticengineer.com/p/when-ai-writes-almost-all-code-what

need to build knowledge and experience through their work, whatever direction software engineering takes. Like Miles [40], we believe juniors must keep developing their craft, because that craft is what allows them to apply judgment.

We worry that juniors who surrender their judgment to AI lose both the ability and the will to develop that craft. Once they surrender, AI makes the decisions for them, they stop evaluating its output, and they start to feel disconnected from the work. Because more controlled forms of motivation are associated with worse learning outcomes [37], we think these juniors will also acquire new skills less effectively. That is why we think the question we pose matters: answering it can help keep the next generation of developers from becoming dependent on AI, losing their motivation, and never building the knowledge and craft they need to judge AI and to grow into independent, senior developers.

5.4 Developing the ability to detect surrender

Both sets of implications depend on being able to notice when students or junior developers surrender their judgment to AI, and in this study the surveys, and especially the closed questions within those surveys, were not the right tool to detect this. Before the project began, almost all participants reported high confidence in their own programming skills (Figure 4). After each phase, almost all of them reported that AI helped them and their teams reach their goals or save time (Figure 5). With similar perception-based instruments, Salomon et al. [56] report much the same surface picture at a larger scale, where 84% of students used generative AI and over 90% of them found it useful. None of these measures surfaced what participants told us in the focus groups and the open survey responses.

We therefore argue that educators and organizations need better diagnostics to help identify whether students and developers are in the process of surrendering their judgment. In particular when it comes to the emotional disconnect that participants spoke about when they spoke about surrender. In fact, the account shared by Brett Codes suggests this may already be happening to developers outside of this course, and the work on cognitive surrender in software engineering education further emphasizes how this might be a widespread problem [16]. Therefore, we believe that the accounts we report contribute a starting point: the words participants used to describe how surrender felt could seed instruments to further understand the concept of surrender, and afford us the ability to detect it early.

Until such instruments exist, the design of our focus groups offers one way to surface surrender (Table 3). Each session anchored the discussion in a concrete architectural decision or feature from the project, invited two teams to build on and challenge each other's accounts, and closed by asking each participant for a metaphor for the role AI had played. Key accounts in this paper came from both the exchanges and the closing exercise: the disconnect P16 and P22 described surfaced as they built on P5's remark that prompting for four months was "pointless", and the metaphors of AI as a gaslighter came from the metaphor question (Section 4, Table 4). Educators and researchers can reuse the elements from our design in their own courses, together with the survey instruments and course materials that are made available in our replication package.

Diagnostics are also the first step to answering the question we posed in Section 5.3. In our course, the course team relaxed the deadlines and taught a lecture and a lab on working with AI partway through the project, and participants' accounts suggest these changes helped some of them reclaim their judgment. Diagnostics could show educators and organizations when such changes are needed, and for whom.

6 Limitations

Reflexive thematic analysis does not look for the one true account in a dataset. Instead, we used it to build patterns of meaning from data collected in one context [12]. Our themes are therefore not statistically generalizable, and Braun and Clarke [12] do not treat that as a limitation. They ask instead that we explain what may have shaped our reading,

so readers can judge whether our findings apply to their own context [46]. Section 3.4 reports how we established trustworthiness during the analysis, and Section 5 says what our theoretical lens gains and loses. In this section we discuss the setting of this study, and what this setting might have obscured.

We collected data from a single course, and the context of that course will have shaped what participants told us. The course expected AI use, participants knew that and handled accordingly. The conditions also changed mid-project: the teaching team relaxed the deadlines and taught a lecture and a lab on working with AI. Those changes, and their timing, are part of what participants describe, and how they came to reclaim judgment.

We also heard from only some of those teams, and only after completion of the project. Eight teams took part across four sessions, selected from the eleven that expressed interest. We excluded teams that reported interpersonal conflict or non-contributing members, to avoid re-triggering that conflict (Section 3). Our accounts therefore come from teams that stayed intact, and we cannot say how teams whose collaboration broke down experienced AI. Participants also spoke in front of their own teammates and a second team, and that will have shaped what they said. The sessions ran after the project ended, so participants worked from memory. We opened the script with a concrete recollection exercise to reduce this effect [24, 61], but what we have are recollections.

Our data are what participants told us. Our analysis is therefore an account of how they made sense of the project. The two can differ: developers' perceptions of their own AI-assisted development do not always match their measured behavior [9, 13]. This matters most for our second theme. Where participants described understanding AI-generated code, we can only report that they said so. Prather et al. [49] show that students using AI can be overconfident about their own understanding, and because our data consists of recollections we can not verify their actual understanding.

Another analyst could read our data differently, and Braun and Clarke [12] consider that a property of reflextive thematic analysis, and not a flaw. One alternative reading matters for our first theme. The project was large and open-ended, and novices struggle with such tasks even when they have the skills [58]. What we read as a surrender to AI may therefore be a response to the task's openness rather than to AI itself. We do not think this replaces our reading, since participants tie the two together and describe reaching for AI when the task overwhelmed them. But the openness of the project and the expectation to use AI were connected, and we cannot separate them.

Our two contrasting themes invite a reader to sort participants into two groups, but it's important to note that participants do not sort that cleanly. Looking for patterns across a dataset can lose the contradictions within one person's account [12]. The same participant could describe surrendering judgment and reclaiming it, and there are instances in our data of this happening (Section 4). P37 is the clearest case: they regained a sense of capability while both their reliance on AI and the pressure they worked under continued. These instances are important, as they show how closely connected these two experiences were, and how some participants experienced both.

In reflexive thematic analysis, the researcher's subjectivity is a resource rather than a source of bias to be controlled [12]. The first author, who coded the data, teaches software engineering and has worked on software architecture in industry (Section 3). This likely made them alert to the loss of understanding participants described, and to what it means to care about a system's quality. It may also have made them quicker to read a hollowed-out account as a loss, and slower to see handing work to AI as a reasonable response to a demanding project. The second author facilitated the focus groups; their experience as a qualitative researcher in psychology and the social sciences helped them probe for detail. During the analysis, they and the first author discussed the coded excerpts to test alternative explanations (Section 3.3).

Through a factual description of the course, the stack, the deadlines, and the expectation to use AI (Section 3.1), we hope to help readers judge whether our account applies to their own setting.

7 Conclusion

In this study we asked how senior software engineering students experience AI as they learn software architecture in a team project. Architecture is a higher-order skill that novices master only through sustained practice, and prior work has focused little on what it is like for students to work with AI while learning it. We studied 43 students on a semester-long team project, and analyzed their accounts through reflexive thematic analysis [12].

We constructed two themes. The first, **Surrendering Judgment to AI**, is an account of how participants spoke about their extensive and deepening reliance on AI, which we interpret as participants eventually surrendering their judgment to it. Participants who felt unprepared ceded decision-making to AI, and that surrender only increased their reliance on AI. They stopped verifying code they could not evaluate, became stuck in a codebase they did not understand, and answered being stuck with more AI rather than with understanding. Prior work has established that novices over-rely on AI and trust it blindly [15]. We contribute an account of what that over-reliance was like for the students themselves: how participants ceded decision-making to AI, and the emotional toll they described afterwards, which we have not seen reported before. Participants spoke of shame and frustration, of feeling emotionally disconnected from their work, and of work they called pointless and empty, and we read these accounts as leaving them with even less reason to break the cycle. Read through Self-Determination Theory [55], these accounts suggest how a reliance on AI can come to erode the motivation novices need to keep learning.

Our second theme, **Applying Human Judgment**, is an account of participants who never surrendered their judgment to AI, and of participants who reclaimed it from AI. It centers on the knowledge and experience that their judgment rested on. It shows that surrender is not inevitable. What separates the themes is not the pressure, which participants worked under in both, but whether participants held the knowledge to judge what AI produced. Some brought that knowledge into the project. Others recognized that they lacked it, and acquired it once the course team relaxed the deadlines and taught a lecture and a lab on working with AI, and they then changed how they worked with AI and kept decisions in their own hands.

To us, these first-hand accounts of what surrendering judgment and applying it felt like to participants, and of how some participants moved from one to the other, are an important contribution. Judgment is what these participants surrendered, and we suspect it is what other developers surrender to AI as well. Studies in other settings, including with experienced developers, could test that. The conditions that pushed participants to surrender, an unfamiliar stack, delivery pressure outside their control, and an expectation to use AI, are conditions junior developers also meet in industry [10, 41, 69]. That is why we ask what is needed to help developers and organizations adopt AI sustainably. Current proposals focus on competence [34], and our findings point to something more attainable: the participants who recovered did so once the conditions around them changed, and those conditions are controlled by educators and employers.

However, to know when to change the environment, we need the ability to notice when AI adoption goes wrong. Existing instruments do not capture the nuances and subtleties of the accounts we report. And we believe that the words participants gave us, a “massive crutch”, an “intelligent gaslighter”, and four months of “pointless” prompting, can be used to diagnose such issues. With diagnostics that capture these experiences, we think the judgment, ownership, and motivation of the next generation of developers can be preserved.

References

[1] Muhammad Abbas, Farooq Ahmed Jam, and Tariq Iqbal Khan. 2024. Is it harmful or helpful? Examining the causes and consequences of generative AI usage among university students. *International Journal of Educational Technology in Higher Education* 21 (2 2024), 10. Issue 1.

doi:10.1186/s41239-024-00444-7

[2] Sirwan Khalid Ahmed, Ribwar Arsalan Mohammed, Abdulqadir J. Nashwan, Radhwan Hussein Ibrahim, Araz Qadir Abdalla, Barzan Mohammed M. Ameen, and Renas Mohammed Khdhir. 2025. Using thematic analysis in qualitative research. *Journal of Medicine, Surgery, and Public Health* 6 (8 2025), 100198. doi:10.1016/j.glmedi.2025.100198

[3] Tarek Alakmeh, Norman Anderson, Victoria Jackson, Guilherme Vaz Pereira, Umit Akirmak, Anthony Estey, Rafael Prikladnicki, André van der Hoek, Margaret-Anne Storey, and Thomas Fritz. 2026. Grasping AI Reliance in Program Comprehension and Coding through the AIRELI Persona Taxonomy. In *Proceedings of the 2026 34th IEEE/ACM International Conference on Program Comprehension* (Rio de Janeiro, Brazil) *(ICPC '26)*. Association for Computing Machinery, New York, NY, USA, 123–134. doi:10.1145/3794763.3794804

[4] Norman Anderson, Tarek Alakmeh, Victoria Jackson, Guilherme Vaz Pereira, Umit Akirmak, Anthony Estey, Rafael Prikladnicki, Thomas Fritz, André van der Hoek, and Margaret-Anne Storey. 2026. ChatGPT: Friend or Foe When Comprehending and Changing Unfamiliar Code. arXiv:2605.10702 [cs.SE] https://arxiv.org/abs/2605.10702

[5] Albert. Bandura. 1998. *Self-efficacy: the Exercise of Control.* W.H. Freeman. 604 pages.

[6] Luciano Baresi, Andrea De Lucia, Antinisca Di Marco, Massimiliano Di Penta, Davide Di Ruscio, Leonardo Mariani, Daniela Micucci, Fabio Palomba, Maria Teresa Rossi, and Fiorella Zampetti. 2025. Students' Perception of ChatGPT in Software Engineering: Lessons Learned from Five Courses. In *2025 IEEE/ACM 37th International Conference on Software Engineering Education and Training (CSEE&T)*. IEEE, 158–169. doi:10.1109/CSEET66350.2025.00023

[7] Cauã Ferreira Barros, Bruna Borges Azevedo, Valdemar Vicente Graciano Neto, Mohamad Kassab, Marcos Kalinowski, Hugo Alexandre D. Do Nascimento, and Michelle C.G.S.P. Bandeira. 2025. Large Language Model for Qualitative Research: A Systematic Mapping Study. In *2025 IEEE/ACM International Workshop on Methodological Issues with Empirical Studies in Software Engineering (WSESE)*. IEEE, 48–55. doi:10.1109/WSESE66602.2025.00015

[8] Len Bass, Paul Clements, and Rick Kazman. 2021. *Software Architecture in Practice* (4th ed.).

[9] Joel Becker, Nate Rush, Beth Barnes, and David Rein. 2025. Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity. https://metr.org/blog/2025-07-10-early-2025-ai-experienced-os-dev-study/.

[10] Andrew Begel and Beth Simon. 2008. Novice software developers, all over again. In *Proceedings of the Fourth international Workshop on Computing Education Research* (New York, NY, USA). ACM, 3–14. doi:10.1145/1404520.1404522

[11] Virginia. Braun and Victoria. Clarke. 2013. *Successful qualitative research: a practical guide for beginners.* SAGE Publications Ltd. 382 pages.

[12] Virginia Braun and Victoria Clarke. 2022. *Thematic Analysis: a practical guide.* SAGE. 338 pages.

[13] Jenna Butler, Jina Suh, Sankeerti Haniyur, and Constance Hadley. 2025. Dear Diary: A Randomized Controlled Trial of Generative AI Coding Tools in the Workplace. In *2025 IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 319–329. doi:10.1109/ICSE-SEIP66354.2025.00034

[14] Rudrajit Choudhuri, Dylan Liu, Igor Steinmacher, Marco Gerosa, and Anita Sarma. 2024. How Far Are We? The Triumphs and Trials of Generative AI in Learning Software Engineering. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering* (New York, NY, USA). ACM, 1–13. doi:10.1145/3597503.3639201

[15] Rudrajit Choudhuri, Ambareesh Ramakrishnan, Amreeta Chatterjee, Bianca Trinkenreich, Igor Steinmacher, Marco Gerosa, and Anita Sarma. 2025. Insights from the Frontline: GenAI Utilization Among Software Engineering Students. In *2025 IEEE/ACM 37th International Conference on Software Engineering Education and Training (CSEE&T)*. IEEE, 1–12. doi:10.1109/CSEET66350.2025.00007

[16] Rudrajit Choudhuri, Christopher Sanchez, Margaret Burnett, and Anita Sarma. 2026. Thinking Less, Trusting More: GenAI's Impacts on Students' Cognitive Habits. arXiv:2601.22430 [cs.HC] https://arxiv.org/abs/2601.22430

[17] Mihaly Csikszentmihalyi. 1990. *Flow: The Psychology of Optimal Experience.* Harper & Row, New York.

[18] Arie Van Deursen, Maurício Aniche, Joop Aué, Rogier Slag, Michael De Jong, Alex Nederlof, and Eric Bouwers. 2017. A Collaborative Approach to Teaching Software Architecture. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (New York, NY, USA). ACM, 591–596. doi:10.1145/3017680.3017737

[19] Neil A. Ernst and Christoph Treude. 2026. GenAI Is No Silver Bullet for Qualitative Research in Software Engineering. *ACM Transactions on Software Engineering and Methodology* (7 2026). doi:10.1145/3837087

[20] Matteo Esposito, Xiaozhou Li, Sergio Moreschini, Noman Ahmad, Tomas Cerny, Karthik Vaidhyanathan, Valentina Lenarduzzi, and Davide Taibi. 2026. Generative AI for software architecture. Applications, challenges, and future directions. *Journal of Systems and Software* 231 (1 2026), 112607. doi:10.1016/j.jss.2025.112607

[21] Gianni Fenu, Roberta Galici, Mirko Marras, and Diego Reforgiato. 2024. Exploring Student Interactions with AI in Programming Training. In *Adjunct Proceedings of the 32nd ACM Conference on User Modeling, Adaptation and Personalization* (New York, NY, USA). ACM, 555–560. doi:10.1145/3631700.3665227

[22] Samuel Ferino, Rashina Hoda, John Grundy, and Christoph Treude. 2025. Walking the Tightrope of LLMs for Software Development: A Practitioners' Perspective. arXiv:2511.06428 [cs.SE] https://arxiv.org/abs/2511.06428

[23] Samuel Ferino, Rashina Hoda, John Grundy, and Christoph Treude. 2026. Novice Developers' Perspectives on Adopting LLMs for Software Development: A Systematic Literature Review. *ACM Transactions on Software Engineering and Methodology* (3 2026). doi:10.1145/3800580

[24] Uwe Flick. 2000. *Episodic Interviewing.* SAGE Publications Ltd, 76–92. doi:10.4135/9781849209731.n5

[25] Matthias Galster and Samuil Angelov. 2016. What makes teaching software architecture difficult?. In *Proceedings of the 38th International Conference on Software Engineering Companion* (New York, NY, USA). ACM, 356–359. doi:10.1145/2889160.2889187

[26] Aashish Ghimire and John Edwards. 2024. *Coding with AI: How Are Tools Like ChatGPT Being Used by Students in Foundational Programming Courses.* 259–267. doi:10.1007/978-3-031-64299-9_20

[27] Suzanne Groothuijsen, Antoine van den Beemt, Joris C. Remmers, and Ludo W. van Meeuwen. 2024. AI chatbots in programming education: Students' use in a scientific computing course and consequences for learning. *Computers and Education: Artificial Intelligence* 7 (12 2024). doi:10.1016/j.caeai.2024.100290

[28] Luiz Franciscatto Guerra and Neil Ernst. 2025. Assessing LLMs for Front-end Software Architecture Knowledge. arXiv:2502.19518 [cs.SE] https://arxiv.org/abs/2502.19518

[29] Hacer Güner and Erkan Er. 2025. AI in the classroom: Exploring students' interaction with ChatGPT in programming learning. *Education and Information Technologies* 30 (6 2025), 12681–12707. Issue 9. doi:10.1007/s10639-025-13337-7

[30] Philipp Haindl and Gerald Weinberger. 2024. Students' Experiences of Using ChatGPT in an Undergraduate Programming Course. *IEEE Access* 12 (2024), 43519–43529. doi:10.1109/ACCESS.2024.3380909

[31] Wilhelm Hasselbring. 2018. *Software Architecture: Past, Present, Future.* Springer International Publishing, 169–184. doi:10.1007/978-3-319-73897-0_10

[32] Md Gulam Mashud Jaman, Abdul Mouhit, and Georgios Koutsopoulos. 2026. *Understanding Generative AI Adoption in Development Work: The Role of Developer Experience.* 316–332. doi:10.1007/978-3-032-30863-4_20

[33] Tanisha Jowsey, Virginia Braun, Victoria Clarke, Deborah Lupton, and Michelle Fine. 2025. We Reject the Use of Generative Artificial Intelligence for Reflexive Qualitative Research. *Qualitative Inquiry* (12 2025). doi:10.1177/10778004251401851

[34] Matthew Kam, Cody Miller, Miaoxin Wang, Abey Tidwell, Irene A. Lee, Joyce Malyn-Smith, Beatriz Perret, Vikram Tiwari, Joshua Kenitzer, Andrew Macvean, and Erin Barrar. 2025. What do professional software developers need to know to succeed in an age of Artificial Intelligence?. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering* (New York, NY, USA). ACM, 947–958. doi:10.1145/3696630.3727251

[35] Hieke Keuning, Isaac Alpizar-Chacon, Ioanna Lykourentzou, Lauren Beehler, Christian Köppe, Imke de Jong, and Sergey Sosnovsky. 2024. Students' Perceptions and Use of Generative AI Tools for Programming Across Different Computing Courses. In *Proceedings of the 24th Koli Calling International Conference on Computing Education Research* (New York, NY, USA). ACM, 1–12. doi:10.1145/3699538.3699546

[36] Nataliya Kosmyna, Eugene Hauptmann, Ye Tong Yuan, Jessica Situ, Xian-Hao Liao, Ashly Vivian Beresnitzky, Iris Braunstein, and Pattie Maes. 2025. Your Brain on ChatGPT: Accumulation of Cognitive Debt when Using an AI Assistant for Essay Writing Task. arXiv:2506.08872 [cs.AI] https://arxiv.org/abs/2506.08872

[37] R. A. Kusurkar, Th. J. Ten Cate, C. M. P. Vos, P. Westers, and G. Croiset. 2013. How motivation affects academic performance: a structural equation modelling analysis. *Advances in Health Sciences Education* 18 (3 2013), 57–69. Issue 1. doi:10.1007/s10459-012-9354-3

[38] Dastyni Loksa, Amy J. Ko, Will Jernigan, Alannah Oleson, Christopher J. Mendez, and Margaret M. Burnett. 2016. Programming, Problem Solving, and Self-Awareness. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA). ACM, 1449–1461. doi:10.1145/2858036.2858252

[39] Lauren E. Margulieux, James Prather, Brent N. Reeves, Brett A. Becker, Gozde Cetin Uzun, Dastyni Loksa, Juho Leinonen, and Paul Denny. 2024. Self-Regulation, Self-Efficacy, and Fear of Failure Interactions with How Novices Use LLMs to Solve Programming Problems. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1* (New York, NY, USA). ACM, 276–282. doi:10.1145/3649217.3653621

[40] Russ Miles. 2026. *The Sovereign Apprentice: AI Literacy for New Engineers.* Leanpub. https://leanpub.com/thesovereignapprentice

[41] Courtney Miller, Rudrajit Choudhuri, Mara Ulloa, Sankeerti Haniyur, Robert DeLine, Margaret-Anne Storey, Emerson Murphy-Hill, Christian Bird, and Jenna L. Butler. 2025. "Maybe We Need Some More Examples:" Individual and Team Drivers of Developer GenAI Tool Use. arXiv:2507.21280 [cs.SE]

[42] Michael Mircea, Elisa Schmid, Jakob Droste, and Kurt Schneider. 2026. How Do Software Engineering Students Use Generative AI in Real-World Capstone Projects? An Empirical Baseline Study. arXiv:2604.24521 [cs.SE] https://arxiv.org/abs/2604.24521

[43] Marcel Mitas and Sabine Sachweh. 2025. Lifelong Learning in Software Engineering: Towards an AI-driven Tutor for Cloud-Based Software Architectures. In *2025 IEEE European Technology and Engineering Management Summit (E-TEMS).* IEEE, 92–97. doi:10.1109/E-TEMS64751.2025.11239087

[44] Crista Mohammed and Sean Rocke. 2025. Electrical and Computer Engineering Freshmen and Generative AI: Awareness, Attitudes, and Ethics. In *2025 IEEE Global Engineering Education Conference (EDUCON).* IEEE, 1–8. doi:10.1109/EDUCON62633.2025.11016545

[45] Cristina Martinez Montes, Robert Feldt, Cristina Miguel Martos, Sofia Ouhbi, Shweta Premanandan, and Daniel Graziotin. 2025. Large Language Models in Thematic Analysis: Prompt Engineering, Evaluation, and Guidelines for Qualitative Software Engineering Research. arXiv:2510.18456 [cs.SE] https://arxiv.org/abs/2510.18456

[46] Lorelli S. Nowell, Jill M. Norris, Deborah E. White, and Nancy J. Moules. 2017. Thematic Analysis. *International Journal of Qualitative Methods* 16 (12 2017). Issue 1. doi:10.1177/1609406917733847

[47] Eduardo Carneiro Oliveira, Hieke Keuning, and Johan Jeuring. 2025. Uncovering Behavioral Patterns in Student–LLM Conversations during Code Refactoring Tasks. In *Proceedings of the 25th Koli Calling International Conference on Computing Education Research* (New York, NY, USA). ACM, 1–11. doi:10.1145/3769994.3770000

[48] Helen Pearson, Michelle Myall, Anne-Sophie Darlington, and Faith Gibson. 2025. The approach and application of analysing inductive and deductive datasets: a worked example using reflexive thematic analysis. *Qualitative Research in Psychology* 22 (10 2025), 842–886. Issue 4. doi:10.1080/14780887.2025.2499265

[49] James Prather, Brent N Reeves, Juho Leinonen, Stephen MacNeil, Arisoa S Randrianasolo, Brett A. Becker, Bailey Kimmel, Jared Wright, and Ben Briggs. 2024. The Widening Gap: The Benefits and Harms of Generative AI for Novice Programmers. In *Proceedings of the 2024 ACM Conference on International Computing Education Research - Volume 1* (New York, NY, USA). ACM, 469–486. doi:10.1145/3632620.3671116

[50] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2023. Robust speech recognition via large-scale weak supervision. In *Proceedings of the 40th International Conference on Machine Learning* (Honolulu, Hawaii, USA) *(ICML'23)*. JMLR.org, Article 1182, 27 pages.

[51] Christian Rahe and Walid Maalej. 2025. How Do Programming Students Use Generative AI? *Proceedings of the ACM on Software Engineering* 2 (6 2025), 978–1000. Issue FSE. doi:10.1145/3715762

[52] Vennila Ramalingam and Susan Wiedenbeck. 1998. Development and Validation of Scores on a Computer Programming Self-Efficacy Scale and Group Analyses of Novice Programmer Self-Efficacy. *Journal of Educational Computing Research* 19 (12 1998), 367–381. Issue 4. doi:10.2190/C670-Y3C8-LTJ1-CT3P

[53] Sergio Rojas-Galeano, Julian Tejada, and Fernando Marmolejo-Ramos. 2025. Between Tool and Trouble: Student Attitudes Toward AI in Programming Education. arXiv:2508.05999 [cs.ET] https://arxiv.org/abs/2508.05999

[54] Per Runeson, Martin Host, Austen Rainer, and Bjorn Regnell. 2012. *Case Study Research in Software Engineering*. 241 pages.

[55] Richard M. Ryan and Edward L. Deci. 2000. Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American Psychologist* 55 (2000), 68–78. Issue 1. doi:10.1037/0003-066X.55.1.68

[56] Marie Salomon, Kyle D. Chin, Reid Holmes, Thomas Fritz, and Gail C. Murphy. 2025. An Exploration of How Generative AI Affects Workflow and Collaboration in a Software Engineering Course. In *Proceedings of the 2025 ACM SIGPLAN International Symposium on SPLASH-E* (New York, NY, USA). ACM, 42–54. doi:10.1145/3758317.3759680

[57] Larissa Schmid, Tobias Hey, Martin Armbruster, Sophie Corallo, Dominik Fuchß, Jan Keim, Haoyu Liu, and Anne Koziolek. 2025. Software Architecture Meets LLMs: A Systematic Literature Review. arXiv:2505.16697 [cs.SE] https://arxiv.org/abs/2505.16697

[58] Alan H. Schoenfeld. 2016. Learning to Think Mathematically: Problem Solving, Metacognition, and Sense Making in Mathematics (Reprint). *Journal of Education* 196 (4 2016), 1–38. Issue 2. doi:10.1177/002205741619600202

[59] Cigdem Sengul, Rumyana Neykova, and Giuseppe Destefanis. 2024. Software engineering education in the era of conversational AI: current trends and future directions. *Frontiers in Artificial Intelligence* 7 (8 2024). doi:10.3389/frai.2024.1436350

[60] Ansh Sharma and James R Wallace. 2025. DeTAILS: Deep Thematic Analysis with Iterative LLM Support. In *Proceedings of the 7th ACM Conference on Conversational User Interfaces* (New York, NY, USA). ACM, 1–7. doi:10.1145/3719160.3735657

[61] David W. Stewart and Prem N. Shamdasani. 2015. *Focus Groups: Theory and Practice* (3 ed.). Sage, London.

[62] Margaret-Anne Storey. 2026. From technical debt to cognitive and intent debt. *ACM Queue* 24, 2 (2026).

[63] John Sweller. 1994. Cognitive load theory, learning difficulty, and instructional design. *Learning and instruction* 4, 4 (1994), 295–312.

[64] Jonathan Ullrich, Matthias Koch, and Andreas Vogelsang. 2025. From Requirements to Code: Understanding Developer Practices in LLM-Assisted Software Engineering. arXiv:2507.07548 [cs.SE] https://arxiv.org/abs/2507.07548

[65] Annie Vella and Kelly Blincoe. 2026. The Impact of AI Coding Assistants on Software Engineering: A Longitudinal Study. arXiv:2605.23135 [cs.SE] https://arxiv.org/abs/2605.23135

[66] Yingying Wang, Masih Beigi Rizi, Fatemeh Khashei, and Julia Rubin. 2026. From Specifications to Implementation in the Gen-AI Era: Lessons from a Project-Based Software Engineering Course. *Proceedings of the ACM on Software Engineering* 3 (6 2026), 1424–1447. Issue FSE. doi:10.1145/3797092

[67] Yu-Yin Wang and Yu-Wei Chuang. 2024. Artificial intelligence self-efficacy: Scale development and validation. *Education and Information Technologies* 29 (3 2024), 4785–4808. Issue 4. doi:10.1007/s10639-023-12015-w

[68] Claes Wohlin and Austen Rainer. 2022. Is it a case study?—A critical analysis and guidance. *Journal of Systems and Software* 192 (10 2022), 111395. doi:10.1016/j.jss.2022.111395

[69] Xin Zhao and Narissa Tsuboi. 2024. Early Career Software Developers - Are You Sinking or Swimming?. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Society* (New York, NY, USA). ACM, 166–176. doi:10.1145/3639475.3640106

[70] Kaitlyn Zhou, Jena Hwang, Xiang Ren, and Maarten Sap. 2024. Relying on the Unreliable: The Impact of Language Models' Reluctance to Express Uncertainty. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (Stroudsburg, PA, USA). Association for Computational Linguistics, 3623–3643. doi:10.18653/v1/2024.acl-long.198