Harnessing Hype to Teach Empirical Thinking: An Experience With AI Coding Assistants

Marvin Wyrich*
wyrich@cs.uni-saarland.de
Saarland University
Saarbrücken, Germany

Norman Peitek*
peitek@cs.uni-saarland.de
Saarland University
Saarbrücken, Germany

Kallistos Weis
kallistos@cs.uni-saarland.de
Saarland University
Saarbrücken, Germany

Sven Apel
apel@cs.uni-saarland.de
Saarland University
Saarbrücken, Germany

Abstract

Software engineering students often struggle to appreciate empirical methods and hypothesis-driven inquiry, especially when taught in theoretical terms. This experience report explores whether grounding empirical learning in hype-driven technologies can make these concepts more accessible and engaging. We conducted a one-semester seminar framed around the currently popular topic of AI coding assistants, which attracted unusually high student interest. The course combined hands-on sessions using AI coding assistants with small, student-designed empirical studies.

Classroom observations and survey responses suggest that the hype topic sparked curiosity and critical thinking. Students engaged with the AI coding assistants while questioning their limitations—developing the kind of empirical thinking needed to assess claims about emerging technologies. Key lessons: (1) Hype-driven topics can lower barriers to abstract concepts like empirical research; (2) authentic hands-on development tasks combined with ownership of inquiry foster critical engagement; and (3) a single seminar can effectively teach both technical and research skills.

CCS Concepts

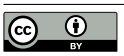
• **Social and professional topics** → **Computing education**.

Keywords

Empirical Software Engineering, Inquiry-Based Learning, Critical Thinking, Technology Hype, Experience Report

ACM Reference Format:
Marvin Wyrich, Norman Peitek, Kallistos Weis, and Sven Apel. 2026. Harnessing Hype to Teach Empirical Thinking: An Experience With AI Coding Assistants. In *34th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (FSE Companion '26), July 05–09, 2026, Montreal, QC, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3803437.3805785

*Both authors contributed equally to this work.



FSE Companion '26, Montreal, QC, Canada

ACM ISBN 979-8-4007-2636-1/2026/07
https://doi.org/10.1145/3803437.3805785

1 Introduction

"*I just wanted to try Copilot, and now I'm doing a user study,*" one student laughs. We cannot help but smile—because in that moment, we feel a little caught. Our primary goal was to teach empirical thinking: how to form hypotheses, design studies, and reason soundly from data. That goal has not changed from previous seminars offered by our chair. What changed was the framing. Instead of asking students to empirically investigate well-established tools or practices, we invited them to explore an emerging technology they were already eager to try. A bit of sugar on the broccoli, one might say. The result? Record application numbers.

Understanding the appeal of this seminar starts with the hype surrounding AI coding assistants. AI coding assistants—tools that use large language models (LLMs) to suggest code and complete programming tasks—have rapidly entered the software industry. GitHub Copilot, one of the most prominent examples, integrates seamlessly into popular development environments and provides real-time code suggestions based on natural-language prompts or partially written code [15]. Recent advances in LLM research have fueled interest in these tools and their potential to support everyday development work. But the excitement surrounding AI coding assistants is part of a familiar pattern: In software engineering, trending technologies have always had strong appeal [1, 6]. Sometimes that appeal stems from genuine curiosity or the hope for increased productivity, sometimes from the belief that being familiar with popular tools signals employability. The phenomenon of *Résumé-Driven Development* illustrates this tendency: developers and employers alike are drawn to what is perceived as current and in demand, even when those choices are not always the most suitable for the task at hand [12, 13].

As with many industry trends, this one is quickly making its way into the classroom, where educators face the challenge of deciding whether and how to incorporate AI coding assistants into the curriculum [2]. In just the past three years, the topic has attracted so much academic attention that several secondary studies have already begun synthesizing the emerging body of research [34, 35]. Research in this space primarily explores the opportunities, challenges, and risks of integrating generative AI into higher education [3, 8, 14, 18, 36, 38], discusses institutional strategies for responding to these developments [4, 19], documents early classroom observations [17, 27, 44], and provides initial guidelines on how

to include AI coding assistants in teaching [26, 41]. Some contributions even argue from an industry perspective that such tools should be actively taught in university settings [7].

While AI coding assistants are just one facet of the broader generative AI movement in education, they offer a particularly tangible entry point—one that invites technical exploration and also raises important questions about how such tools should be evaluated in practice. For example, many claims surrounding tools such as GitHub Copilot center on increased productivity or improved developer experience [15]. But how can we know whether such claims hold up under scrutiny, and whether it is worth investing time and resources into these tools? Teaching students how to answer such questions remains a fundamental but challenging task. Empirical methods are central to software engineering research [30] and practice [43], yet often perceived as abstract or dry, especially when taught in isolation from concrete problems.

We wondered: Can we harness the students' excitement around hype-driven technologies such as AI as a hook—while fostering critical, evidence-based thinking? In this experience report, we describe a seminar that set out to do exactly that. Students explored AI coding assistants not just as users, but as investigators, learning how to design studies, formulate hypotheses, and interpret results along the way. Through this seminar, we contribute empirical evidence for the pedagogical value of hype-driven topics in fostering critical reflection on scientific methodology. We also share three publicly available teaching artifacts: hands-on materials for learning to use GitHub Copilot, resources for evaluating students' experience and learning, and the raw data from our seminar evaluation.[1]

In the following, we outline the seminar's design and structure (Section 2), share impressions from our teachers' perspective to illustrate student engagement and learning during the course (Section 3), present the students' perspective from a post-seminar survey exploring how the hype topic shaped their experiences and perceptions (Section 4), reflect on broader lessons and implications for teaching empirical methods in fast-moving technological landscapes (Section 5), and conclude with a short summary (Section 6).

2 Course Design

Seminars play a central role in both the bachelor's and master's computer science programs at our university. They are semester-long courses, typically limited to 12–20 participants, with most chairs offering one or two seminars each term. As part of their degree requirements, students must complete, at least, two seminars in the bachelor's program and, at least, one in the master's program. While general learning objectives are defined at the departmental level, the detailed course design is left to the offering chair, resulting in substantial variation across seminars.

Students select from a department-wide list of available seminars, each announced with a title and description outlining the scope and expected deliverables. Applications include a brief motivation letter, which is reviewed by the responsible chairs to assess fit. A central allocation system then assigns students to seminars based on their preferences, evaluations, and global balancing criteria.

[1] https://github.com/brains-on-code/coding-assistants-empirical-thinking

In what follows, we describe the specific design of our seminar on AI coding assistants, including its overall structure, learning objectives, participants, and the activities that shaped the semester.

2.1 Overall Structure

We provide an overview of the seminar structure in Figure 1. The seminar comprised ten sessions of 90 minutes each. The first session introduced the overall structure of the seminar, explained all deliverables in detail, included a short lecture on designing empirical studies, and demonstrated how to use GitHub Copilot in Visual Studio Code. Sessions 2–4 were hands-on exercises in which students worked on programming tasks with and without GitHub Copilot. Each exercise emphasized a different use case—code generation, code explanation and refactoring, and debugging—allowing students to directly experience the tool's strengths and limitations in varied contexts. At the same time, we designed these sessions to spark ideas for students' own research questions by encouraging them to critically reflect on their observations. Sessions 5 and 7 focused on students' own planned studies: Each student tested their study design with a peer and reciprocally participated in a peer's study. Session 6 covered principles of scientific presentations, followed by a Q&A session focused on refining study designs after the first pilot. In sessions 8–10, students presented their research questions, study designs, initial findings and insights of their pilot study, as well as reflections on their empirical work.

After the in-class phase, students had 9.5 weeks to write a seminar paper. The paper outlines the motivation for their research question and study, reviews related work, and documents refinements based on the two pilot iterations, including preliminary results. We encouraged students to critically engage with their own study design, for instance, by reflecting on validity aspects and documenting potential limitations alongside their findings. This element of *active learning* has been shown to help in deepening students' understanding of empirical research [28].

2.2 Learning Objectives and Principles

We designed the seminar according to the principles of constructive alignment [5], ensuring that teaching activities, assessments, and learning objectives were mutually supportive. By the end of the course, students were expected to be able to independently design, refine, and execute an empirical study, reflecting critically on their methodological choices and potential validity threats based on recommendations of teaching empirical studies [31]. They were also expected to demonstrate ownership of their research by exploring a current, relevant topic and making independent design choices rather than replicating established studies. Finally, students were expected to communicate their findings clearly through structured scientific writing, integrating feedback from staged drafts.

We aligned our grading with these objectives and based it on the seminar paper, oral presentation, and active participation. All seminar instructors jointly reviewed and discussed each part, reaching consensus to ensure consistent evaluation and alignment with the intended learning outcomes.

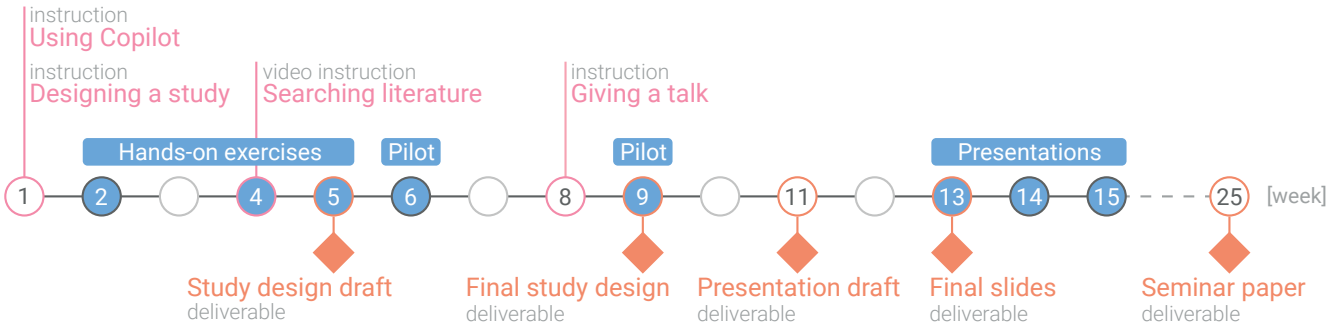


Figure 1: Schematic overview of the seminar structure, showing instructional sessions where content was taught to students (pink), sessions involving student activities (blue), and milestones with deliverables due at each point (orange). The timeline is organized by week; numbers indicate weeks with an on-site session or a milestone deliverable.

2.3 Participants and Instructors

At our department, seminars limit enrollment to maintain close supervision. In our chair, we aim for a staff-to-student ratio of approximately 1:1 to 1:2. Of 168 applicants for our seminar, 18 students were selected (by the departments' distribution system), of whom 13 concluded the seminar. We paired these 13 students with nine instructors (doctoral and postdoctoral researchers) from our chair, each student receiving one-to-one feedback across key milestones, including guidance on study design and presentation. We consider this close supervision highly beneficial for fostering student learning, though we recognize that not all chairs can allocate such a high number of instructors for legitimate reasons (cf. Section 5.5).

2.4 Hands-On Sessions

We designed three hands-on sessions to prepare our students for designing a small-scale empirical study. Each session focused on a distinct programming activity that features prominently in current research on AI coding assistants: code generation, code explanation and refactoring, and debugging. Students worked on targeted tasks with and without GitHub Copilot, reflected on their experiences, and discussed takeaways to surface potential research questions. We summarize each session's setup below.

Session 1: Code Generation. The first session was concerned with the correctness and quality of AI-generated code [23, 24], highlighting challenges such as the need for multi-turn prompting [24] and the risk of code smells in generated output [37]. We contrasted AI-assisted solutions with manually authored code. Students were split into two groups and given the same specification: One group implemented it with assistance from GitHub Copilot, the other without. After 30 minutes, students completed a short post-task questionnaire. The groups then crossed over to a second specification for another 30 minutes, reversing the availability of AI assistance. Both specifications came with a test suite to verify baseline conformance. A final questionnaire concluded the activity. We treated this programming exercise as a small-scale experiment itself: The questionnaires provided comparative data between the two groups, which we analyzed together with the students. The session closed with a meta-level discussion of what had been measured, what research questions such data could answer, and how such setups illustrate the structure of an empirical study.

Session 2: Code Explanation and Refactoring. AI coding assistants support tasks beyond simple code generation, such as code explanation and refactoring [20, 33]. We again divided students into two groups and provided an already implemented project. Their task was to understand how the project worked; one group could use an AI assistant, the other could not. After 30 minutes, students completed a short questionnaire. The groups then switched conditions on a second project for another 30 minutes, followed by another questionnaire. As in the first session, we framed the activity as a small-scale experiment, enabling a direct comparison between AI-assisted and manual approaches. In the debrief, we discussed the choices involved in operationalizing (e.g., productivity and code quality) and how different instruments might capture such constructs.

Session 3: Debugging. The third hands-on session focused on debugging. Students received one larger project with a test suite and had 60 minutes to identify and fix seeded bugs. In contrast to the previous sessions, all students were permitted to use AI assistants throughout the task. The session concluded with a discussion on how such a setup could be studied empirically, highlighting threats to validity (e.g., defect difficulty, prior experience) and possible strategies for data collection and analysis.

2.5 Additional Instructive Sessions

We offered four instructional sessions: (1) introduction to GitHub Copilot, covering student access, integration in Visual Studio Code, and available functionality; (2) literature search strategies for finding and accessing related work; (3) principles of empirical study design, with a focus on controlled experiments, were introduced to frame the hands-on sessions and guide students' own study planning according to best practices; and (4) scientific presentation practices, including dos and don'ts for effective scientific talks. These sessions were designed as concise complements to existing, more comprehensive university courses (e.g., on empirical study design), ensuring that all students, regardless of prior knowledge, were sufficiently prepared to complete the seminar.

2.6 Empirical Studies

Each student designed a small-scale empirical study lasting 25–30 minutes. Grading requirements at our university necessitated the decision that each student develop their own study design independently rather than in a team. Students first submitted a draft study design, then conducted a pilot with a peer while reciprocally participating as a subject. They had two weeks to revise their design and submitted a final version accompanied by a brief rationale that documented changes and lessons learned from the first pilot. A second pilot followed, allowing further refinement before a hypothetical final study conduct (not part of the seminar).

To provide focus and comparability, we offered five topic areas. Students indicated preferences, and we assigned topics accordingly. The areas reflected different dimensions of AI coding assistants at the time: (1) *code explanation and comprehension*, (2) *prompt engineering and interaction with coding assistants*, (3) *correctness and quality of AI-generated code*, (4) *the educational impact of coding assistants*, and (5) *AI in practice: professional use and workflow integration*. For each topic, we provided a reading list to help students ground their research questions in existing literature.

2.7 Final Presentations

Students prepared a 12-minute presentation in which they introduced their study design, explained how they refined it across the two pilot iterations, and shared preliminary insights from exploratory analyses of the pilot data. Importantly, they were expected to critically reflect on both the potential and the limitations of their data—acknowledging, for example, that results based on two participants cannot substantiate strong quantitative claims but can nevertheless reveal design flaws or suggest promising directions. Furthermore, we asked students to reflect on their experience of working with an empirical approach.

Each talk was followed by a substantive discussion with peers and instructors. These discussions clarified design details and probed the reasoning behind methodological choices, the adequacy of measurement instruments, and the interpretation of pilot findings.

2.8 Seminar Paper

Finally, students submitted a 15-page, single-column seminar paper describing their study. The paper followed a standard research-paper structure (i.e., introduction, background, related work, study design, and conclusion) and additionally documented design learnings, changes after the first pilot, and initial results from pilot participants, with brief implications for a future full study. Beyond being a required deliverable, the seminar paper aimed to give students hands-on practice in scientific writing and proper citation, complementing the oral presentation by allowing them to exercise precise and accurate scientific communication. For institutions with shorter seminar durations, this written report could be treated as an optional component, while still achieving the core learning objectives of the seminar, particularly the focus on empirical thinking.

3 Teacher's Perspective: Classroom Impressions

In this section, we report our impressions on teaching this seminar, in particular, how students interacted with the material and how they conducted their studies.

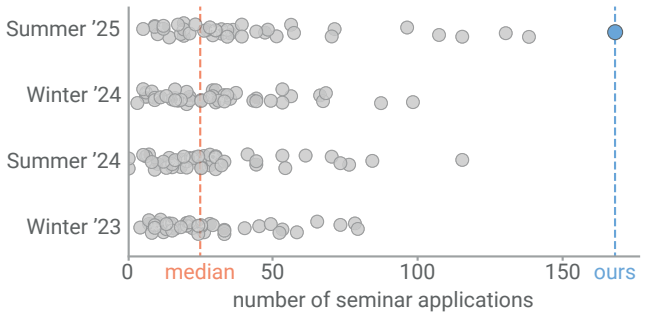


Figure 2: Number of student applications for each seminar offered over the past four semesters, highlighting our seminar with 168 applications.

3.1 Strong Interest in the Course

Since the computer science faculty at Saarland University employs a centrally organized seminar system, it is very transparent what seminars are offered and to gauge student interest in them. We offered 18 spots in our seminar, but demand far exceeded our capacity, with a total of 168 student applications.

To put this into perspective, we compare this number with data for the last four semesters at our faculty. Across the last four semesters, around 45 seminars were offered each semester for a total of 176 seminars. Each received a decent number of applications (median: 25, mean: 33). Our seminar far exceeded the average level of interest, with 168 applications, the highest number of recorded applications across these four semesters. This underlines how the excitement surrounding AI coding assistants is intriguing to students. We visualize all application numbers in Figure 2.

Our description of the seminar clearly communicated that the course will not just cover AI coding assistants, but to approach them with an empirical mindset. We have offered seminars covering empirical methods in the past; however, with less measurable interest. The seminar applications, which include students' motivation for participation, further highlight that students focused on the hype topic, in particular the recency of the AI assistant topic and its persuasive effects. For example, two students included the following phrases in their motivation:

- *"This seminar is very important to me as it will help me to stay up to date with the latest advancements in my research area, receive guidance from lecturers, and encounter other students' opinions."*
- *"I see that AI is becoming a more and more important topic in software engineering, and I find it interesting how its integration influences my personal coding experience and whether it provides an advantage in practice."*

Overall, we not only met the expected perceived hype around the topic for our students, but it far exceeded our expectations.

> The seminar's popularity was seemingly driven by the timely and relevant topic of AI coding assistants, demonstrating that students are interested in exploring hype topics in an academic setting, even when they involve complex and abstract concepts, such as empirical methods.

3.2 Reflection during Hands-On Sessions

At the beginning of the seminar, we conducted three hands-on sessions (cf. Section 2.4). For each of the hands-on sessions, we aimed for students not only to obtain practical experience with AI coding assistants in a controlled environment, but also to actively reflect on their experience. To encourage students to reflect on several important aspects of AI coding assistants (e.g., helpfulness, productivity, accuracy)—including tasks where the AI coding assistants were unavailable—we designed a small questionnaire with a mix of quantitative and qualitative questions. After the first session, we asked all students to fill out this questionnaire. While the students were working on the second task, one instructor analyzed the questionnaire data. We then summarized the results as a basis for the group discussion at the end of the hands-on session. We provide a subset of the quantitative data in Table 1.

Overall, the results of the questionnaire for the first session (code generation) were mixed across the two tasks. Notably, for the first task, the group without AI assistance completed more subtasks than the ones with AI ($Q_{R\text{-}1}$). Nevertheless, all students across both groups reported that they would be more productive with AI coding assistants ($Q_{R\text{-}2}$). While students also relied on GitHub Copilot for code completions ($Q_{R\text{-}3}$) and perceived it as fairly reliable ($Q_{R\text{-}4}$), they also reported that it often made unexpected (38–50%, $Q_{R\text{-}5}$) or incorrect (60–63%, $Q_{R\text{-}6}$) code completion suggestions. These contradicting experiences, which also appear to be quite task-specific, were part of the in-class discussions. The open-text questions about the students' experience similarly covered negative and positive experiences, such as:

- *"I tried commenting while Copilot was editing... bad idea, the edit didn't make any sense."*
- *"Copilot did basically everything [...] it was completely helpful"*

For the second task in the first session, the group using AI coding assistants achieved a notably higher completion rate, suggesting higher productivity. Otherwise, results were similar regarding perceived usefulness of the assistants and their incorrect completions.

We provided students with a test suite as an indicator of task completion but did not disclose additional secret tests covering edge cases. We intuitively expected that students coding without AI assistants would develop a deeper understanding and thus miss fewer edge cases than those relying on generated code. However, because students progressed at different rates within the limited time, the hidden test results were not comparable between groups.

For the second and third sessions, students could still use the same questionnaire for personal reflection, but we did not demand it in a structured manner on a classroom level. Instead, we employed a live group discussion covering the same topics without presenting specific data, since students were already familiar with the aspects that we considered worthy of a discussion.

During the hands-on sessions, we encouraged students to reflect on their experiences with AI coding assistants via a questionnaire, which provided insights into their perceptions and experiences. It also facilitated a group discussion, in which students could share their thoughts and observations about using these tools in our scenario.

3.3 Conducting Empirical Studies

After the hands-on sessions, it was the students' turn to design their own empirical study on one small aspect of AI coding assistants.

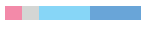
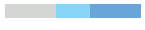
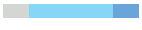
3.3.1 Students' Research Goals and Methods. We assigned the students to five different research areas (cf. Section 2.6). This way, we enforced variety in the students' topics. However, while we did not restrict students regarding their choice of research method, the chosen methodology was unexpectedly very uniform across our students. Except for one student, all students opted for some kind of (controlled) experiment. This was not our intention, as we did not restrict the methods and let students freely choose. We suspect that several factors may drive this trend and make other methods less feasible in our format. Specifically, our hands-on sessions were already in a style of controlled experiment. Furthermore, our student pilot study setup assigned each student one participant for half an hour. We paired the students together, which means they could not search for participants with a specific profile themselves. For future seminars, in which different methods would be essential, a different pilot study setup should be considered.

In combination with a controlled experiment, many students used a post-questionnaire to collect self-reported perspectives alongside behavioral measures. Again, this approach may have been strongly inspired by our hands-on sessions, in which we also combined a practical task with reflection driven by a questionnaire, and explored differences in behavior and subjective viewpoints. Furthermore, many students employed comparative setups (e.g., prompting style A vs. prompting style B, or with and without AI coding assistants), again similar to our hands-on sessions.

While the methods were fairly uniform, the conceptual goals differed between students and fully covered our intended variety of research topics. The students explored different research topics within the assigned areas, from very technical topics (e.g., "Which prompting style produces more correct code?" or "To what extent does GitHub Copilot generate secure and correct API implementations using FastAPI and PostgreSQL?") to their practical effects (e.g., "How reliable is code generated by GitHub Copilot perceived by developers?"), and to their role in a learning environment (e.g., "Do AI-coding tools help novice programmers learn a new language?"). From our perspective, this variety is quite valuable, as students can learn not just from their own study but also benefit from the insights of their fellow students. Retrospectively, we have to note that the seminar structure does make some topics less feasible. For example, generally interesting research topics, such as using AI coding assistants in a collaborative setting or the effects of AI coding assistants over an extended period of time, were not feasible. If this is of interest, it might be worth considering refinements for future iterations of this seminar (cf. Section 5.5).

Evolution of Study Designs. Students first created an initial study design, conducted a pilot, and then refined their design before a second pilot. We reviewed the students' submissions and analyzed the changes between their initial experiment design and the refined version. While we expressed that students only needed to refine the study design, many also improved their writing clarity and presentation quality. In retrospect, this made it difficult to separate presentation gains from methodological improvements. To more

Table 1: Overview of a small subset of gathered quantitative reflection data during the first hands-on session on code generation. The session included two tasks. For both tasks, we split the students into a without-AI and with-AI group.

Question	Task 1		Task 2	
	Without AI	With AI	Without AI	With AI
$Q_{R\text{-}1}$: Task Completion Rate	71%	50%	13%	60%
$Q_{R\text{-}2}$: Do you think being allowed to use GitHub Copilot (would have) made this task easier?	100% Yes	100% Yes	100% Yes	100% Yes
$Q_{R\text{-}3}$: Did you rely on GitHub Copilot for code completion? (strongly disagree ■■■■■ strongly agree)	–		–	
$Q_{R\text{-}4}$: How often did GitHub Copilot provide accurate code completions? (never ■■■■■ almost always)	–		–	
$Q_{R\text{-}5}$: Were there any instances where GitHub Copilot provided unexpected code completions?	–	38% Yes	–	50% Yes
$Q_{R\text{-}6}$: Were there any instances where GitHub Copilot provided incorrect code completions?	–	63% Yes	–	60% Yes

clearly capture this learning step, future iterations should require an explicit section detailing (1) what changed and (2) why.

Across all students, we observed a wide range of revisions. While a couple of students only applied minor tweaks, many substantially improved their studies—e.g., reducing task complexity, or introducing randomization. Notably, a few students overhauled their designs, changing the population (from targeting experienced programmers to novice programmers), experimental setup (different programming language), or even the research question. Overall, we were impressed with students quickly being able to refine their study designs after the first pilot, despite little prior experience. This aligns with prior findings that small projects [11] and practical hands-on work are often relevant for teaching empirical software engineering [25]. In our case, all students demonstrated that they can pick up one key aspect of empirical research—adapting and improving a methodology in response to unexpected challenges or practical limitations—deepening their understanding of the complexities involved in designing and conducting empirical studies. Our live observations of students conducting pilot studies confirmed our analysis of the study design documents.

Reflection of Learnings from Conducting Studies. In their final presentations, students shared both topic-related insights (e.g., "LLM explanations give answers, not insights") and reflections on their empirical work. Many discussed strengths and weaknesses of their designs (e.g., "Within-subject counterbalancing minimized individual and learning biases"), challenges (e.g., "measuring *understanding* was hard"), and ideas for improvement (e.g., "prepare a scripted set of interview questions in advance"). Some also emphasized the importance of considering confounding factors early on. These are all valuable considerations even for experienced researchers.

> Our seminar allowed students to design and conduct their own small-scale empirical studies. While the setup may have limited the range of methods, the two-pilot design proved valuable by supporting refinement and improving study quality. Many students also reported gaining key insights into the challenges of empirical research.

4 Student Perspective: Linking Hype to Empirical Thinking

To complement our subjective impressions, we aimed to understand how students perceived the seminar and how engaging with a currently popular topic influenced their learning experience. Thus, we invited all students to participate in a voluntary online survey following the final presentations. Note that while our primary goal is to provide an experience report, we complement it with small-scale empirical evaluations to strengthen the insights gained.

4.1 Methodology

To ensure scientific rigor, we follow Kasunic's seven-step guideline for survey design [16], complemented by Linåker et al.'s more recent annotations [22]. We collect both qualitative data, which provide detailed insights into individual participants' experiences [29], as well as quantitative data, which—despite the small sample size—are analyzed exploratively to offer a more general indication of the seminar's effectiveness.

Sampling Strategy. After the final presentations (cf. Section 2.7), we shared the survey link with all seminar participants and invited them to take part without offering incentives. They could either use the remaining time of the seminar session to complete the survey on site, fill it out at home, or choose not to participate at all. The survey was anonymous, and participation was entirely voluntary. In line with standard practice at Saarland University, this anonymous course evaluation did not require institutional review board approval.

Questionnaire Design. The questionnaire began with a consent form that informed participants about the purpose of the survey and the data processing procedures. The questionnaire consisted of four main sections spread across four pages. The first section focused on students' perceptions of AI coding assistants, prompting them to reflect on their views before and after the seminar. The second section assessed the achievement of empirical thinking goals by asking students whether and how they adopted an empirical mindset throughout the course. The third section explored the link between working on a currently popular topic and students' learning outcomes, for example, by asking how engaging with a trending technology such as AI coding assistants influenced their interest in designing and conducting their own study. The final section gathered demographic information, including participants' study program, current semester, prior experience with AI coding assistants, and prior experience with conducting empirical research.

Pilot Test and Data Collection. It is generally recommended to pilot surveys [16]. However, since our intended sample coincides entirely with the population of students who can meaningfully comment on the seminar, we opted against a traditional pilot. Instead,

Table 2: Overview of our student demographics and experience with AI coding assistants and empirical research.

Measure		No./Mean ± SD
Number of students completing the seminar		13
Number of students who completed the survey		10
Undergraduate students		4
→ Current semester		5.25 ± 0.83
Graduate students		6
→ Current semester		1.66 ± 0.75
Experience with AI coding assistants	None	2
	Tried once or twice	1
	Used occasionally	3
	Used regularly	4
Experience with Empirical Research	None	4
	Minimal	4
	Some	2
	Solid	0

we asked two doctoral students familiar with the seminar to review the questionnaire and included their feedback in the final version. The survey was online for 14 days. In total, 11 of 13 students started the survey, of which 10 students completed the survey.

Demographic Data. The survey included typical demographic and experience questions, which we present in Table 2.

Quantitative Data Analysis. To analyze the quantitative data from the Likert-scale [21] questions of the survey, we mapped the response data to the respective categories (e.g., "strongly agree", "disagree"). We then computed descriptive statistics in terms of mean and standard deviation (SD). Note that we are aware of an open debate on whether Likert scales represent ordinal or continuous intervals [32] and that we opted to consider Likert items as discrete values on a continuous scale. We provide the results, including visualizations, in Table 3.

Qualitative Data Analysis. To analyze the qualitative data from the open-ended survey questions, we conducted a lightweight coding and thematic analysis, roughly following the recommendations and steps of Cruzes and Dyba [9] and Defranco and Laplante [10]. Specifically, we conducted the analysis in the following steps:

(1) **Prepare data**: We extracted all answers for each question.
(2) **Code data**: One author initially coded each available answer. This was done with an open, inductive coding approach due to the novelty of the questions and the small dataset.
(3) **Create themes from codes**: The same author then found common themes without a hierarchy across codes.
(4) **Review coding and themes**: Another author reviewed and checked the coding, themes, and interpretation. They also assisted in discussing difficult cases.
(5) **Assess overall quality**: A third author conducted a final quality assessment based on Treude's quality criteria [39].

We discuss the results and our interpretation in the next section. For full transparency, we provide all answers with assigned codes and themes in our replication package.

4.2 Results

We structured the content-related part of the survey into three sections, each covering a different topic. We discuss each section in the following.

4.2.1 Perception of AI Coding Assistants. Our first section of the survey investigated the students' diverse prior knowledge ($Q_{AI\text{-}L1}$) and perception ($Q_{AI\text{-}L2}$) of AI coding assistants, and how it changed throughout this seminar (cf. Table 3). Initially, students were hesitant about the accuracy and reliability of AI coding assistants ($Q_{AI\text{-}L2}$), which shifted slightly more positive throughout the seminar ($Q_{AI\text{-}L3}$). The majority of our students reported that our seminar has helped them to understand the capabilities and limitations of AI coding assistants ($Q_{AI\text{-}L4}$, $Q_{AI\text{-}L5}$) and that they are more likely to critically evaluate them ($Q_{AI\text{-}L6}$).

Analyzing the open-ended questions supported this viewpoint, which we show in Table 4(a). Throughout the seminar, the majority of students report that they have developed a more comprehensive understanding of AI coding assistants ($Q_{AI\text{-}O7}$). In particular, they are more aware of their risks and limitations (S2, S4, S6, S5, S7, S9, S11), but also their selective usefulness when used appropriately (S2, S4, S7, S7, S10, S11). Two students (S4, S11) highlighted that AI coding assistants can be problematic for learning students. One student (S1) also pointed out that they are now more aware that there is still much to explore in this research field. For example, S9 states "I started to see AI more with respect to its limitations but maybe I was getting too reliant on it and this seminar was a wakeup call", while S11 stated "I learned to what extent AI coding assistants can be used for learning but also where the limitations are, so where students need to make progress by themselves."

For the majority of our students (S1, S2, S4, S6, S7, S11), the hands-on sessions were perceived as instructive ($Q_{AI\text{-}O8}$). They report that across all three hands-on sessions (code generation, code explanation, debugging) some or multiple sessions provided specific learnings. They also report that they discovered new features and explored use cases they had not considered before. For example, S6 states: "I know that Copilot can directly edit our entire code file, and we can chat with the assistant. Especially helpful when asking it to explain the function by providing example input and output". Some students (S4, S7, S11) experienced firsthand the limitations of AI coding assistants, such as incorrect suggestions. Only a couple of students (S9, S10) stated that the sessions had no substantial impact on how they perceived AI coding assistants.

Through their own studies, many students (S1, S2, S7, S9, S10) reported an increased awareness of AI coding assistants' strengths and limitations ($Q_{AI\text{-}O9}$). In particular, they discussed their fast development, discovered new use cases, but also noted potential issues. Some students (S4, S10) emphasized the impact of the human factor, in particular that other students use AI coding assistants differently, and this strongly influences their success and perception. Two students (S2, S7) discussed that prompt engineering is a key factor to successfully use AI coding assistants. Two different students (S6, S11) reported no substantial change in perception toward AI coding assistants from conducting their own studies.

4.2.2 Empirical Thinking. While the first section of our survey confirmed the learning effects regarding AI coding assistants, one

Table 3: Overview of gathered quantitative reflection data of the post-seminar survey. All questions are on a 1→5 Likert-scale [21] from ■ "strongly disagree" to ■ "strongly agree". We excluded "I do not know" answers.

	Question	Mean ± SD	Response Distribution
(a) AI coding assistants	$Q_{AI\text{-}L1}$: Before the seminar, I had a clear idea of what AI coding assistants can and cannot do	3.50 ± 1.27	10% 10% 20% 40% 20%
	$Q_{AI\text{-}L2}$: Before the seminar, I perceived AI coding assistants as accurate and reliable tools	3.20 ± 0.92	30% 20% 50%
	$Q_{AI\text{-}L3}$: After the seminar, I perceived AI coding assistants as accurate and reliable tools	3.50 ± 0.97	20% 20% 50% 10%
	$Q_{AI\text{-}L4}$: During this seminar, my understanding of the capabilities and limitations of AI coding assistants changed	4.00 ± 1.05	10% 20% 30% 40%
	$Q_{AI\text{-}L5}$: After this seminar, I better understand both the strengths and limitations of AI coding assistants	4.50 ± 0.71	10% 30% 60%
	$Q_{AI\text{-}L6}$: After the seminar, I am more likely to critically evaluate the output of AI coding assistants	4.20 ± 0.63	10% 60% 30%
(b) Empirical Mindset	$Q_{Emp\text{-}L10}$: Helped to distinguish between anecdotal and empirical evidence	4.00 ± 0.71	22% 55% 22%
	$Q_{Emp\text{-}L11}$: More likely to seek out comprehensive data and robust evidence	4.20 ± 0.63	10% 60% 30%
	$Q_{Emp\text{-}L12}$: More confident in assessing claims about the capabilities of AI coding assistants	4.20 ± 0.92	10% 50% 40%
	$Q_{Emp\text{-}L13}$: More confident applying an empirical mindset to other emerging technologies	4.20 ± 0.63	10% 60% 30%
	$Q_{Emp\text{-}L14}$: My own empirical study helped me better understand how empirical research works	4.70 ± 0.48	30% 70%
	$Q_{Emp\text{-}L15}$: My own study made me feel invested in the results and motivated to share my findings	4.50 ± 0.53	50% 50%
	$Q_{Emp\text{-}L16}$: Individual supervision played an important role in guiding my empirical work	4.00 ± 0.94	10% 10% 50% 30%
	$Q_{Emp\text{-}L17}$: Questions from my peers helped me reflect on methodological issues with my study design	4.10 ± 0.74	20% 50% 30%
	$Q_{Emp\text{-}L18}$: Discussing my study improved understanding more than theory alone	4.20 ± 0.79	20% 40% 40%
(c) Linking Hype	$Q_{Hype\text{-}L21}$: Conducting a study on a hyped topic motivates me more than working on a conventional topic	3.80 ± 0.79	40% 40% 20%
	$Q_{Hype\text{-}L22}$: Working on AI coding assistants helped me better understand the value of empirical investigation	4.10 ± 0.74	20% 50% 30%
	$Q_{Hype\text{-}L23}$: A predefined topic area with provided literature eased developing a feasible study design	4.30 ± 0.67	10% 50% 40%
	$Q_{Hype\text{-}L24}$: Conducting own research deepened my critical understanding more than traditional lectures	4.30 ± 0.82	20% 30% 50%

fundamental goal was to teach empirical thinking. We devised 9 questions targeting various aspects of how participating in our seminar affected students' empirical thinking in general and specifically regarding AI coding assistants. The overwhelming majority consistently agreed or strongly agreed with the positive effects of our seminar ($Q_{Emp\text{-}L7}$–$Q_{Emp\text{-}L15}$, cf. Table 3(b)).

For our open-ended questions (cf. Table 4(b)), students encountered various difficulties when conducting their empirical pilot studies ($Q_{Emp\text{-}O19}$). Half of the responses (S2, S5, S6, S7) covered typical challenges of empirical research, such as defining research questions, selecting measures, and trade-offs in experiment design and validity considerations. Another set of challenges was reported due to the design and conduct of our seminar (S5, S6, S9, S10), which limited the students' research to two short pilot studies (S9: "There were only two participants"), but also due to issues in providing sufficient information. One student (S11) reported technical difficulties but was able to overcome them.

Within the scope of our seminar, students reported a lot of refinements in their experiments ($Q_{Emp\text{-}O20}$). For the majority (S1, S2, S4, S5, S7), this meant a reduction in scope by simplifying the task or reducing the number of tasks. This was partially driven by the artificial time limit we had set to stay within the seminar time slot. Students also recognized general challenges of designing empirical studies, either due to biases or due to the difficulty of the human factor in software engineering. Finally, a few students (S4, S6, S11) noted that they were unsure what to expect, and conducting the pilot studies was important for refinement.

4.2.3 Linking Hype and Empirical Thinking. In the final section of the questionnaire, we explored how a hype topic could be used to engage students in empirical work. Here, we received a few neutral responses, but still a majority of students agreed or strongly agreed that AI coding assistants were able to motivate them for empirical work ($Q_{Hype\text{-}L21}$, $Q_{Hype\text{-}L22}$). Additionally, they agreed that conducting their own study is more effective than theoretical lectures ($Q_{Hype\text{-}L24}$). We initially were concerned that students would struggle with finding relevant literature for a novel research topic. Thus, we pre-defined five research areas and, for each, provided them with a list of literature as a starting point. Almost all students confirmed this was helpful for developing their studies ($Q_{Hype\text{-}L23}$).

The answers to the open-text questions (cf. Table 4(c)) complemented and reinforced the trends found in the Likert-scale responses. Many students (S4, S6, S7, S8, S9, S11) reported that working on the hype topic of AI coding assistants increased their interest in empirical research, especially due to its novelty and personal relevance. In particular, students mentioned: "It made it way more interesting!" (S4) and "I'm not really interested in research side but conducting this study made me curious" (S9). A few students (S2, S5, S7, S10) explicitly stated that conducting a pilot study helped them to better understand empirical research (S1: "It taught me basically how to conduct studies"). Some students (S2, S5, S6, S9,

Table 4: Themes from answers to open-ended questions in three topic areas of our post-seminar survey. Frequency denotes the count of participants contributing to a given theme.

(a) Themes on AI Coding Assistants	Frequency
$Q_{AI\text{-}O7}$: How has your perception of AI coding assistants changed throughout the seminar?	
Limitation and Risk Awareness	6/8
Usefulness When Used Carefully	6/8
Realizing Drawbacks for Student Learning	2/8
Awareness of Research Gaps	1/8
$Q_{AI\text{-}O8}$: Were there particular hands-on session(s) that changed how you see AI coding assistants?	
Confirming/Positive Response	6/8
Discovering Features and Use Cases	6/8
Experience Limitations	3/8
Negative Response	2/8
$Q_{AI\text{-}O9}$: Was there a particular part of your own study that changed how you see AI coding assistants?	
Change in Perception (e.g., Quality)	5/8
Human Factor in Tool Effectiveness	2/8
Influence of Prompt Engineering	2/8
No Change	2/8
(b) Themes on Empirical Thinking	**Frequency**
$Q_{Emp\text{-}O19}$: What challenges did you encounter when trying to apply an empirical approach to studying AI coding assistants?	
General Empirical Difficulties	4/8
Seminar-Specific Difficulties	4/8
Technical Issues	1/8
$Q_{Emp\text{-}O20}$: How did conducting a first pilot study influence your expectations or plans for the final study design?	
Challenges in Running Experiments	7/8
Experiment Redesign/Reduction	5/8
Human Variability in Software Studies	4/8
Reducing Uncertainty	3/8
(c) Themes on Linking Hype	**Frequency**
$Q_{Hype\text{-}O25}$: How did working on a current topic of AI coding assistants affect your interest in designing and conducting your own study?	
Hype Increased Interest	5/8
Personal Relevance and Learning	2/8
Empirical Learning/Research	2/8

S11) also emphasized that the research results were meaningful to them personally. Interestingly, one student (S10) stated that their interest in empirical methods was genuine and not because of AI.

4.2.4 Final Notes. In the final section, a few students (S2, S9, S10, S11) offered closing remarks, all highlighting positive aspects of the seminar, such as an overall enjoyable experience (S9: "I had fun"), the seminar's structure, and the opportunity to receive feedback. Two students (S9, S10) also noted a negative aspect, namely unclear milestone expectations.

In a nutshell, survey results, with Likert-scale responses from 10 students and open-ended feedback from 8 (of 13 total), indicate the seminar was perceived very positively. Students gained a deeper understanding of AI coding assistants and further developed their empirical thinking. Additionally, our data suggest that leveraging hype topics can effectively engage students in empirical research, especially when exploring a relevant real-world application through their pilot study.

5 Discussion and Lessons Learned

In this section, we reflect on our experiences teaching the seminar and highlight lessons to inform actionable advice for others.

5.1 Hype Can Be a Pedagogical Catalyst

Our seminar on AI coding assistants suggests that hype can be a powerful entry point to core methodological training. A record number of applicants and positive student feedback indicated that hype-driven topics can draw students into empirical thinking and research skills. One benefit is that limited prior research on hype topics makes research feel more relevant: students reported being motivated by working in a novel area and by having to think more carefully about study design—which made the work more exciting. Framing the seminar this way helped create a learning environment where students were motivated and invested in the material.

However, instructors must ensure that hype does not overshadow careful methodological rigor. Because fads are temporary but empirical skills endure, we recommend using hype-driven topics primarily as an entry point, explicitly anchored to non-negotiable methodological learning goals.

5.2 Managing Expectations Beyond the Hype

Building on the motivational benefits of hype, we also observed its risks. From 168 applications, we admitted 18 students, of whom 13 completed their empirical studies. Some dropout is expected, for example, due to university changes or personal circumstances, but others left without clear reasons. This raises an important question: Was there a mismatch between the expectations of these students (possibly focused on AI coding assistants) and the actual content of the seminar (with focus on empirical methods)? Hype may have attracted many applicants, but it may also have drawn in students mainly interested in "playing with" AI coding assistants rather than engaging in empirical research.

A key lesson learned is that hype amplifies the cost of misaligned expectations. While hype can make complex topics accessible, if not framed carefully, it can foster unrealistic expectations and student disengagement. Although our seminar description clearly stated its goals (i.e., empirical thinking, research design, and pilot studies), some students may have focused more on the topic than on this framing. This emphasizes the importance of transparent communication about workload, deliverables, and intended learning outcomes, in line with prior teaching experiences [42]. In practice, our experience suggests foregrounding methods, workload, and deliverables more prominently than the topic itself in course descriptions and early sessions.

5.3 Ownership of Inquiry Fosters Insights

When students design and carry out their own studies, they naturally want to share their findings—deepening their critical engagement with the topic [40]. In our seminar, this sense of ownership was amplified through the iterative process of running two pilot studies, reflecting, and refining. Students could clearly see how their design evolved through their own decisions, which made them more invested in the outcome. Felderer and Kuhrmann found a similar effect when their students conducted small studies [11].

Presenting study designs to peers further created accountability and learning opportunities [42]. Group discussions about threats to validity were more productive when grounded in actual student projects rather than hypothetical examples. This allowed students to learn not only from their own project but also from those encountered by their peers. Overall, our experience suggests that student-driven inquiry is highly beneficial for developing empirical thinking, echoing prior work [25], but instructors should plan sufficient feedback capacity and structured checkpoints to support students throughout the process.

5.4 Balance of Dual-Purpose Seminars

While our primary learning goals focused on empirical thinking, the seminar also gave students extensive exposure to AI coding assistants. This showed that technical and research skills do not need to be taught in isolation—integrating them can make each side more meaningful. By combining hands-on exercises with empirical study design, we created a dual-purpose learning pathway: students practiced with AI coding assistants in realistic tasks while also reflecting on these experiences through the lens of research.

This approach was effective for students motivated by technology but less inclined toward pure methodological training. Integrating technical and research perspectives made both more meaningful and engaging. This shows that even within a limited time frame (compared to, e.g., completing a graduate program), students can meaningfully experience a well-scoped research process. The pilot studies provided a "scaled-down but authentic" research experience, mimicking the workflow of a full research project. While these pilot studies cannot replace full-scale empirical projects, they are an efficient and effective compromise for a seminar.

Teaching this seminar corroborated that frequent, short-cycle feedback is essential to ensure students develop meaningful and methodologically sound study designs [42]. This was particularly crucial for those without prior empirical training. Without structured feedback and multiple interim deadlines, students risk drifting into pseudo-scientific territory.

5.5 Limitations of Course Format

The seminar offered valuable hands-on experience in empirical study design, but it came with some limitations. Students mainly conducted two short pilot experiments, so longer-term or qualitative studies (e.g., field studies, interviews with practitioners) were not feasible. Adapting the structure would be needed for different methods or extended investigations. Moreover, given the participants' limited experience in empirical studies, students may not execute everything flawlessly unless provided with sufficient guidance. In this regard, the high staff-to-student ratio helped provide individualized guidance but may be difficult to replicate in contexts with fewer instructors. In such a case, supervision could shift toward in-class group feedback while still supporting student learning.

Finally, while designed for software engineering, similar approaches could work in other disciplines where trending topics or emerging technologies spark curiosity and can motivate students to explore questions critically. In these fields, instructors could leverage current debates, innovations, or emerging tools to create a comparable learning environment. By aligning hands-on engagement with structured reflection, students in any domain can be guided to critically evaluate claims, gather evidence, and develop robust, evidence-based insights. Overall, our experience suggests that this format works best for short, tightly scoped empirical studies and may require careful adaptation when transferred to contexts with fewer instructors or different methodological goals.

6 Conclusion

Hype cycles are a recurring reality in software engineering, shaping industry practices, research agendas, and also what students expect to learn. Although hype in software engineering is rarely met with empirical scrutiny, its bold claims can serve as a powerful teaching opportunity to engage students in formulating and testing hypotheses about what these technologies can and cannot deliver. In this experience report, we reflected on our seminar on AI coding assistants, which embraced this dynamic: We used the hype to spark interest and directed it toward deeper engagement with empirical research. Through hands-on exercises, pilot studies, and iterative reflection, students learned to design, evaluate, and communicate empirical work, while instructors provided close guidance and structured feedback.

From our experience, we derived several lessons learned: hype can be a powerful pedagogical catalyst, but it can also raise false expectations if students primarily seek technical training rather than research practice; ownership of inquiry fosters ownership of insight; and dual-purpose formats that balance technical skills with empirical reflection can be effective.

We hope this report encourages others to treat the next hype not just as a teaching challenge, but as an opportunity to equip students with empirical thinking skills—skills that will remain essential, no matter what the next trend may be.

Acknowledgments

We thank all participating students for their curiosity, engagement, and valuable contributions to both the seminar and our accompanying research. This research is supported by ERC Advanced Grant "Brains On Code" (101052182) and DFG through the Collaborative Research Center TRR 248, project ID 389792660 (https://perspicuous-computing.science).

References

[1] Vard Antinyan and Mikael Grah. 2026. Fashion-Driven Software Engineering: Industry Experiences of Form Over Substance. *IEEE Software* 43, 1 (2026), 112–118.

[2] Brett Becker, Michelle Craig, Paul Denny, Natalie Kiesler, and James Prather. 2025. *Dagstuhl Seminar 25311: Generative AI in Programming Education.* https://www.dagstuhl.de/25311

[3] Brett Becker, Paul Denny, James Finnie-Ansley, Andrew Luxton-Reilly, James Prather, and Eddie Antonio Santos. 2023. Programming Is Hard - Or at Least It

Used to Be: Educational Opportunities and Challenges of AI Code Generation. In *Proc. ACM Technical Symposium on Computer Science Education* (Toronto, Canada) *(SIGCSE 2023)*. ACM, 500–506.

[4] Santiago Berrezueta-Guzman, Stephan Krusche, and Stefan Wagner. 2025. From Coders to Critics: Empowering Students through Peer Assessment in the Age of AI Copilots. In *Int'l Symposium on Educational Technology (ISET)*. IEEE, 66–71.

[5] John Biggs and Catherine Tang. 2011. *Teaching for Quality Learning at University* (4 ed.). Open University Press / McGraw-Hill.

[6] Manfred Broy and Bran Selić. 2025. The Effects of Hype in the Software Domain: Causes, Consequences, and Mitigations. *IEEE Software* 42, 2 (2025), 98–102.

[7] Christopher Bull and Ahmed Kharrufa. 2024. Generative Artificial Intelligence Assistants in Software Development Education: A Vision for Integrating Generative Artificial Intelligence Into Educational Practice, Not Instinctively Defending Against It. *IEEE Software* 41, 2 (March 2024), 52–59.

[8] 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 *Int'l Conf. on Software Engineering Education and Training (CSEE&T)*. IEEE, 1–12.

[9] Daniela Cruzes and Tore Dyba. 2011. Recommended Steps for Thematic Synthesis in Software Engineering. In *Int'l Symposium Empirical Software Engineering and Measurement*. IEEE, 275–284.

[10] Joanna DeFranco and Phillip Laplante. 2017. A Content Analysis Process for Qualitative Software Engineering Research. *Innovations in Systems and Software Engineering* 13, 2 (2017), 129–141.

[11] Michael Felderer and Marco Kuhrmann. 2019. Using Mini-Projects to Teach Empirical Software Engineering. In *SEUH*. 75–86.

[12] Jonas Fritzsch, Marvin Wyrich, Justus Bogner, and Stefan Wagner. 2021. Résumé-Driven Development: A Definition and Empirical Characterization. In *43rd IEEE/ACM International Conference on Software Engineering: Software Engineering in Society, ICSE (SEIS) 2021, Madrid, Spain, May 25-28, 2021*. IEEE, 19–28.

[13] Jonas Fritzsch, Marvin Wyrich, Justus Bogner, and Stefan Wagner. 2023. Resist the Hype!: Practical Recommendations to Cope With Résumé-Driven Development. *IEEE Softw.* 40, 5 (2023), 41–49.

[14] Matthew Galej. 2025. The Risks of AI Coding Assistants: Novice Programmers Beware. *XRDS* 31, 4 (Aug. 2025), 10–11.

[15] GitHub, Inc. 2025. *GitHub Copilot • Your AI pair programmer*. https://web.archive.org/web/20250728061412/https://github.com/features/copilot

[16] Mark Kasunic. 2005. *Designing an Effective Survey*. Technical Report September. Carnegie Mellon University, Software Engineering Institute. 143 pages.

[17] Majeed Kazemitabaar, Runlong Ye, Xiaoning Wang, Austin Zachary Henley, Paul Denny, Michelle Craig, and Tovi Grossman. 2024. Codeaid: Evaluating a Classroom Deployment of an LLM-Based Programming Assistant that Balances Student and Educator Needs. In *Proc. CHI Conference on Human Factors in Computing Systems*. ACM, 1–20.

[18] Vassilka Kirova, Cyril Ku, Joseph Laracy, and Thomas Marlowe. 2024. Software Engineering Education Must Adapt and Evolve for an LLM Environment. In *Proc. Technical Symposium on Computer Science Education*. 666–672.

[19] Sam Lau and Philip Guo. 2023. From "Ban It Till We Understand It" to "Resistance Is Futile": How University Programming Instructors Plan to Adapt as More Students Use AI Code Generation and Explanation Tools such as ChatGPT and GitHub Copilot. In *Proc. Conf. on International Computing Education Research*. 106–121.

[20] Juho Leinonen, Paul Denny, Stephen MacNeil, Sami Sarsa, Seth Bernstein, Joanne Kim, Andrew Tran, and Arto Hellas. 2023. Comparing Code Explanations Created by Students and Large Language Models. In *Proc. Conf. Innovation and Technology in Computer Science Education (ITiCSE)*. ACM, 124–130.

[21] Rensis Likert. 1932. A Technique for the Measurement of Attitudes. *Archives of Psychology* 22, 140 (1932), 1–55.

[22] Johan Linåker, Sardar Muhammad Sulaman, Rafael Maiani de Mello, and Martin Höst. 2015. *Guidelines for Conducting Surveys in Software Engineering*. Technical Report. Lund University.

[23] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation. *Advances in Neural Information Processing Systems* 36 (2023), 21558–21572.

[24] Zhijie Liu, Yutian Tang, Xiapu Luo, Yuming Zhou, and Liang Feng Zhang. 2024. No Need to Lift a Finger Anymore? Assessing the Quality of Code Generation by ChatGPT. *IEEE Transactions on Software Engineering* 50, 6 (2024), 1548–1584.

[25] Carlos Danilo Luz, Edson OliveiraJr, and Igor Steinmacher. 2022. How Experimentation in Software Engineering Has Been Taught? Survey and Research Agenda. In *Proc. Int'l Conf. on Evaluation and Assessment in Software Engineering*. 299–304.

[26] Joyce Mahon, Brian Mac Namee, and Brett Becker. 2024. Guidelines for the Evolving Role of Generative AI in Introductory Programming Based on Emerging Practice. In *Proc. Innovation and Technology in Computer Science Education* (Milan, Italy) *(ITiCSE 2024)*. ACM, 10–16.

[27] Lauren Margulieux, James Prather, Brent Reeves, Brett 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 *Proc. Innovation and Technology in Computer Science Education*. 276–282.

[28] Maria Alcimar Costa Meireles, Sabrina Rocha, Jose Carlos Maldonado, and Tayana Conte. 2024. An Experience Report on the Use of Active Learning in Empirical Software Engineering Education: Understanding the Pros and Cons From the Student's Perspective. In *Proc. Int'l Conf. on Software Engineering: Software Engineering Education and Training*. 380–390.

[29] Jorge Melegati, Kieran Conboy, and Daniel Graziotin. 2024. Qualitative Surveys in Software Engineering Research: Definition, Critical Review, and Guidelines. *IEEE Transactions on Software Engineering* (2024).

[30] Daniel Mendez, Paris Avgeriou, Marcos Kalinowski, and Nauman bin Ali. 2025. *Handbook on Teaching Empirical Software Engineering*. Springer.

[31] Jefferson Seide Molléri and Kai Petersen. 2024. *Teaching Research Design in Software Engineering*. Springer Nature Switzerland, 71–100.

[32] Jacqueline Murray. 2013. Likert Data: What to Use, Parametric or Non-Parametric? *International Journal of Business and Social Science* 4, 11 (2013), 258–264.

[33] Daye Nam, Andrew Macvean, Vincent Hellendoorn, Bogdan Vasilescu, and Brad Myers. 2024. Using an LLM to Help With Code Understanding. In *Proc. Int'l Conference on Software Engineering*. 1–13.

[34] Farman Ali Pirzado, Awais Ahmed, Román Mendoza-Urdiales, and Hugo Terashima-Marin. 2024. Navigating the Pitfalls: Analyzing the Behavior of LLMs as a Coding Assistant for Computer Science Students-A Systematic Review of the Literature. *IEEE Access* (2024).

[35] James Prather, Paul Denny, Juho Leinonen, Brett Becker, Ibrahim Albluwi, Michelle Craig, Hieke Keuning, Natalie Kiesler, Tobias Kohn, Andrew Luxton-Reilly, Stephen MacNeil, Andrew Petersen, Raymond Pettit, Brent Reeves, and Jaromir Savelka. 2023. The Robots Are Here: Navigating the Generative AI Revolution in Computing Education. In *Proc. Working Group Reports on Innovation and Technology in Computer Science Education* (Turku, Finland) *(ITiCSE-WGR '23)*. ACM, 108–159.

[36] James Prather, Brent Reeves, Juho Leinonen, Stephen MacNeil, Arisoa Randrianasolo, Brett Becker, Bailey Kimmel, Jared Wright, and Ben Briggs. 2024. The Widening Gap: The Benefits and Harms of Generative AI for Novice Programmers. In *Proc. Conf. on International Computing Education Research* (Melbourne, VIC, Australia) *(ICER '24)*. ACM, 469–486.

[37] Mohammed Latif Siddiq, Lindsay Roney, Jiahao Zhang, and Joanna Cecilia Da Silva Santos. 2024. Quality Assessment of ChatGPT Generated Code and Their Use by Developers. In *Proc. Int'l Conf, Mining Software Repositories*. 152–156.

[38] Lucas Siqueira Rodrigues, Antonio Rueda-Toicen, and Thomas Kosch. 2025. Redesigning Large Language Model Coding Assistants for Software Engineering Education. Mensch und Computer 2025 - Workshopband.

[39] Christoph Treude. 2024. Qualitative Data Analysis in Software Engineering: Techniques and Teaching Insights. In *Handbook on Teaching Empirical Software Engineering*. Springer, 155–176.

[40] Sira Vegas and Natalia Juristo. 2024. *A Course on Experimentation in Software Engineering: Focusing on Doing*. Springer Nature Switzerland, 301–324.

[41] Michael Vierhauser, Iris Groher, Tobias Antensteiner, and Clemens Sauerwein. 2024. Towards Integrating Emerging AI Applications in SE Education. In *2024 36th International Conference on Software Engineering Education and Training (CSEE&T)*. 1–5.

[42] Jéssyka Vilela and Carla Silva. 2024. Empowering Undergraduates in Empirical Research Methods: an Experience Report. In *Simpósio Brasileiro de Engenharia de Software (SBES)*. SBC, 543–553.

[43] Greg Wilson. 2025. *No Satisfaction*. https://third-bit.com/2025/07/18/no-satisfaction/

[44] Ramazan Yilmaz and Fatma Gizem Karaoglan Yilmaz. 2023. The Effect of Generative Artificial Intelligence (AI)-Based Tool Use on Students' Computational Thinking Skills, Programming Self-Efficacy and Motivation. *Computers and Education: Artificial Intelligence* 4 (2023), 100147.