An Exploration of How Generative AI Affects Workflow and Collaboration in a Software Engineering Course

Marie Salomon
University of British Columbia
Vancouver, Canada
mariesal@cs.ubc.ca

Kyle D. Chin
University of British Columbia
Vancouver, Canada
kdchin@cs.ubc.ca

Reid Holmes
University of British Columbia
Vancouver, Canada
rtholmes@cs.ubc.ca

Thomas Fritz
University of Zurich
Zurich, Switzerland
fritz@ifi.uzh.ch

Gail C. Murphy
University of British Columbia
Vancouver, Canada
murphy@cs.ubc.ca

Abstract

How does Generative AI (GenAI) impact how students work and collaborate in a software engineering course? To explore this question, we conducted an exploratory study in a project-based course where students developed three versions of a system across agile sprints, with unrestricted access to GenAI tools. From survey responses of 349 students, we found that the technology was used extensively with 84% of students reporting use and 90% of them finding the technology useful. Through semi-structured interviews with 24 of the students, we delved deeper, learning that students used GenAI pervasively, not only to generate code but also to validate work retrospectively, such as checking alignment with requirements and design after implementation had begun. Students often turned to GenAI as their first point of contact, even before consulting teammates, which reduced direct interpersonal collaboration. These results suggest the need for new pedagogical strategies that address not just individual tool use, but also design reasoning and collaborative practices in GenAI-augmented teams.

CCS Concepts: • **Software and its engineering → Collaboration in software development**.

Keywords: empirical study, computer science education, teamwork, survey, interviews

ACM Reference Format:
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 (SPLASH-E '25), October 12–18, 2025, Singapore, Singapore.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3758317.3759680

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.
SPLASH-E '25, Singapore, Singapore
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2142-7/25/10
https://doi.org/10.1145/3758317.3759680

1 Introduction

Recently, the rise of generative AI (GenAI) has fueled speculation and interest in how the technology will change software engineering and the education of software engineers. For instance, the popular press has speculated that human software engineers might be replaced by GenAI agents (e.g., [16, 22]), and educators have raised questions about how students will learn, e.g., different aspects of software engineering [1, 12, 14]. Professional developers reject the hypothesis that they will be replaced, instead suggesting how GenAI can make them more efficient during their workday (e.g., [13, 20–22, 35]). Educators, similarly, have shifted from questioning whether to engage with GenAI to considering how best to guide students in using it productively and responsibly [12, 15]. As such, education must now absorb this shift and better understand how students are encountering and incorporating these tools into their learning experiences. Interestingly, the perspectives of both professional developers and educators are similar, focusing on how GenAI can help each *individual* improve the activities they perform rather than on how it might support multiple people collaborating to perform the multiple activities required to develop multiple versions of a system.

A software system is *engineered* when its development process involves multiple people developing multiple versions of the system [31]. While simple, this description by Randell captures much of what is challenging when engineering software. For example, the multiple people involved must communicate, coordinate, and integrate independent decisions into a consistent whole, which then constantly evolves. Educators have long sought how best to simulate industrial environments in the educational environment to

enable students to experience and learn these software engineering concepts and skills (e.g., [10, 29, 36]).

To date, studies about how GenAI impacts software engineering and software engineering education largely align with the views that GenAI can assist individual developers (e.g., [3, 4, 24, 26, 40]). Although a few studies have considered how GenAI can be used in team, largely within educational environments (e.g., [39, 41]), the focus has still been on individual aspects of development, such as the use of GenAI for architectural decisions.

As a first step towards filling the gap of how GenAI impacts software engineers building multiple versions of a software system collaboratively, we set out to explore how GenAI was affecting our own students. Specifically, we examined its role in a project-based software engineering course, where students work collaboratively to iteratively design, build, and evolve software.

We focus on two research questions:

RQ1: How does GenAI impact the workflows of students engineering software? We use the term workflow to refer to the order and flow of phases in which software developers engage.

RQ2: How does GenAI impact how students collaborate to engineer a system?

To explore these research questions, we conducted an exploratory study targeting third-year computer science students who, as part of a course, were working in pairs to develop a software project over three iterations. Each of these iterations corresponds to a different version of the system, as requirements evolved as the software development proceeded. The students were permitted unrestricted use of GenAI tools as they developed the versions, allowing them the freedom to experiment and explore the full potential of GenAI tools across different aspects of software development. From a survey we distributed, which received 349 responses (a 96% response rate), we found that the students used GenAI pervasively with 84% reporting the use of the technology.

Seventy-five (21%) students indicated, through the survey, a willingness to participate in an interview about their experiences with GenAI. We randomly selected 25 of them for semi-structured interviews, each lasting 25–35 minutes. We analyzed the interview transcripts using thematic analysis, identifying five themes related to the research questions. Three themes were developed related to **RQ1**. GenAI meaningfully altered the order in which students performed development phases: students often jumped more quickly into implementation, leveraging GenAI to generate starting points or templates, which reduced the time spent on requirements analysis and design. They also used GenAI to retrospectively validate work—checking alignment with requirements and design after implementation had begun. Finally, students described relying on GenAI for instant code review, receiving feedback on correctness, style, and adherence to specifications without needing input from teammates.

Two themes developed related to **RQ2**. GenAI fundamentally changed how teammates collaborated with each other and with their tools: students often turned to GenAI as their first point of contact, even before consulting their teammates, which reduced direct interpersonal collaboration. While GenAI sometimes facilitated team discussions by providing suggestions or solutions that were further deliberated upon, it also created a reliance on technology as a standalone resource. Finally, students were divided on whether GenAI should be treated as another "teammate"—akin to a junior developer—or simply as a sophisticated tool. These results refine our understanding of how students interact with GenAI in educational settings, particularly within a project-based, upper-division software engineering course that emphasizes real-world development practices.

This paper makes the following key contributions:

- *Empirical Insights:* This work presents themes about how GenAI impacts workflows and collaboration in teams of software engineering students, highlighting shifts in the order and flow of development activities and the evolving role of GenAI as a teammate or tool.
- *Educational Implications:* This study provides pedagogical guidance for integrating GenAI into software engineering courses. It identifies how GenAI shifts student engagement away from traditional planning phases and reduces peer collaboration, prompting the need for revised instructional strategies.
- *Open Dataset:* It provides a dataset of survey results, interview transcripts, and thematic analysis results to support replication and further analysis by other researchers.[1]

We begin with a description of related studies (Section 2). We then describe the target students and the context in which they were working, as well as methods used for data collection and analysis (Section 3). A pre-condition to explore our two research questions is that students are using GenAI in their workflow with others. To satisfy this pre-condition, we administered a survey to all students in the course. We report on the results of GenAI use in the course in (Section 4). We then present the results of our study (Section 5) and discuss pedagogical implications for educators (Section 6). We conclude the paper by presenting threats to the validity of the results (Section 6.2) and a summary of findings (Section 7).

2 Related Work

The promise and rising popularity of GenAI has led to studies of the impact of the technology on various activities, both for individuals working on an activity alone, where the bulk of

[1]This paper has supplementary material available on Zenodo (https://doi.org/10.5281/zenodo.16782575).

studies have been conducted, and, less commonly, for teams working collaboratively.

2.1 Use by Individuals

A number of studies have considered how GenAI impacts particular activities or tasks undertaken by individuals in software development.

Programming. Many of the studies have considered the use of GenAI when programming. For example, Amoozadeh et al. showed that GenAI supports learning new concepts in a study conducted with students through a survey [3]. Liang et al. and Nam et al. both demonstrated that GenAI enables faster completion and understanding of programming tasks [24, 26]. Vaithilingam et al. report on how GenAI reduces the time spent searching for code online [40] and Barke et al. found that generated results provide a good starting point [4]. Our findings reinforce these results, as we also observe that students use GenAI to quickly explore possible code implementations and accelerate task completion.

Not every report about the use of the technology for programming has been positive. Choudhuri et al. found increased frustration levels among participants [8]. With respect to using GenAI to generate code, Denny et al. report that developers have difficulties in understanding generated code [11] and Liang et al. report that participants do not use solutions from GenAI when certain functional or non-functional requirements are not met [24]. Nguyen et al. conducted a large-scale lab study with beginning programmers, showing that even in well-scaffolded tasks, students struggled to prompt and interpret LLM outputs [27]. In contrast, our study examines more advanced students in an upper-division, project-based software engineering course, where tasks are open-ended and collaborative. While Nguyen et al. emphasize prompt literacy for novices, we highlight how GenAI shifts attention away from design reasoning and teamwork, raising a different set of pedagogical implications in more complex, real-world learning environments.

Other Software Development Activities. Other studies have considered the use of GenAI for other software development activities. Khojah et al. analyzed the interactions of 24 professional software developers using ChatGPT[2] over one week and conducted exit surveys [21]. They found that developers often use ChatGPT for guidance and learning about tasks in abstract terms, rather than for generating ready-to-use software artifacts. Rajbhoj et al. explored how a systematic prompting approach might help various activities, such as generating requirements, designs, source code, and tests [32]. While potential was found, the authors also reported on manual review and edits that were necessary for completeness and accuracy. Champa et al. performed a quantitative analysis of 12 activities in which software developers use ChatGPT, showing for which tasks ChatGPT was most and least suited to assist [7]. Compared to these earlier studies, the study we report on goes beyond the individual and beyond particular activities, seeking to understand how GenAI impacts multiple students working together and its impact on the overall workflow.

[2] https://chat.openai.com/

2.2 Use by a Team

There are fewer studies about how GenAI impacts teams engineering software collaboratively.

Similar to our study, others have considered the impact of GenAI in course projects. Tanay et al. conducted a semester-long study of student teams building a software engineering project, focusing on how the students integrated GenAI into their coursework and on how the students' perceptions of learning were influenced [39]. Waseem et al. conducted an exploratory study to investigate the usage and effectiveness of GenAI in supporting seven 1st and 2nd-year undergrad students in different phases of software development workflow while developing a three-month project [41]. The authors report that GenAI enhanced the clarity and efficiency of requirements gathering, improved architectural decisions, and streamlined coding processes. These studies largely emphasized individual use patterns and learning gains, and in comparison to the study reported in this paper, did not closely examine how GenAI reshapes team workflows or collaborative practices.

Rasnayaka et al. comes closer to our context by studying upper-division students working in teams to build a compiler-like system with a custom language [33]. The study analyzed AI-generated code annotations, prompts, and survey-based perceptions. They found that LLMs were particularly useful in the early phases of the project for generating basic code structures, improving code quality, and debugging through prompt usage. In contrast, our study focuses on how GenAI reshapes collaborative workflows, including how students communicate within the context of a more realistic, end-to-end software engineering project. Our work uniquely investigates how students resequence core phases, such as requirements gathering, design, and implementation, when using GenAI, and how team interactions are altered as GenAI increasingly becomes the first point of contact, often before teammates, thereby changing how and when collaboration occurs.

3 Methods

We used a semi-structured interview method to explore our research questions. Interviewed students were randomly selected from those willing to participate based on a survey available to all students in the course. The study was approved by the ethics board of our institution, and informed consent was obtained from all students. We describe the course project, the survey, the interviews, and our analysis approach.

3.1 The Course Project

We conducted the study with third-year undergraduate students who were enrolled in a project-based software engineering course. The project consisted of building a TypeScript-based backend implementation of a data store and a custom domain-specific language for querying university courses and teaching spaces. The project was divided into three phases. In the first phase, students implemented basic data processing and a domain-specific language for basic queries. In the next phase, they built on this functionality to support complex data processing and queries. This phase typically requires students to practice design patterns and refactoring to adapt their first phase functionality to the new requirements. In the final phase, students build a frontend interface and backend server that uses the functionality they built in the prior two phases. Students had access to an automated grader that gave high-level feedback on their solution. Because this feedback is high-level, students typically write many tests which is consistent with an industrial software engineering context. Students worked on the project in teams of two students over 12 weeks to complete these three phases. Each team met with a teaching assistant once per week to check progress in a 15-minute, scrum-like meeting, but were not otherwise given explicit, low-level advice on how to approach the project or collaboration. Students also had acceess to office hours and an online forum to ask questions and receive help. Most project implementations consisted of over 1,600 lines of source and test code. A total of 364 students participated in the course during the study term and approximately 30% of them were female-identifying students.

Figure 1a provides an overview of the typical process used by a pair of students to work on a sprint. In the `Requirements` phase, students are provided with a description of stakeholder expectations from which they derive the requirements for the system. The students must then generate and consider multiple `Design` choices. As the amount of `Implementation` required is substantial, teammates generally divide the work to build the software. Teams are able to receive feedback on their solution by running their code against a regression test server to ensure basic functionality. As Figure 1b depicts, teammates typically interact with each other extensively through the sprints and consult outside documentation and websites as needed.

The course permitted unrestricted use of GenAI tools, allowing students to utilize these resources freely in terms of both functionality and input content, contrasting with the more regulated environments seen in many organizations. Students were required to annotate any AI-generated or influenced code to maintain transparency and academic integrity.

3.2 Survey

To explore our research questions, we needed to identify students in the course who were using GenAI in their workflow. We administered a survey, which was open to all students of the course, to identify potential students. As this survey included questions about how GenAI was being used, we were able to gain data about the use of GenAI beyond the students in the interview study.

In addition to demographic questions, the survey asked about students' use of code completion tools (e.g., GitHub Copilot[3]) and chatbots (e.g., ChatGPT[4]). Students who did not use GenAI were asked to share their reasons and concerns. The survey was piloted with four people and was updated prior to sending it out to the students to ensure clarity and consistency. The survey and results are included in the dataset for this paper.

The survey was advertised widely through the course discussion board and during lecture time, allowing students to complete it anytime before the end of the course. Students were incentivized with a 1% addition to their final grade simply for opening the survey; they were not required to complete it, nor were they obligated to use any GenAI tools to receive this incentive. As part of the survey, students were asked if they could be contacted later to participate in an interview.

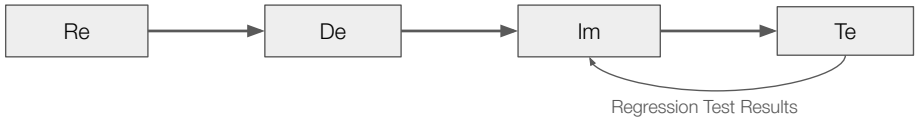
We recorded 349 completed responses to the survey, resulting in a response rate of 95.8%, from which we removed 2.3% of the responses due to a duplicate response and seven responses that were falsely included due to an initial bug in the survey workflow that was resolved after detection. We present the data about the use of GenAI from 341 responses (93.6%) in Section 4 to provide context for the responses from the interviews.

3.3 Interview

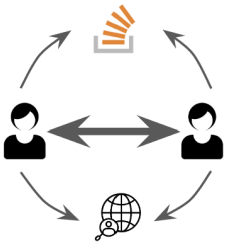
We explored our research questions using 19 main questions in semi-structured interviews. The interview included two scenarios presented to the students; however, the resulting data was inconclusive and is therefore not included in this paper. The dataset for this paper can be consulted for more detail about the full interview questions.

Only students who indicated usage of GenAI and a willingness to be interviewed in the survey were eligible for the interviews. From a pool of 79 willing students, we randomly sampled a subset of 25 to reduce bias and increase the representativeness of the sample, and contacted each of them via email to book an interview timeslot. We scheduled the interviews between the end of the project and the exam period for ethical and validity reasons. One scheduled interview student dropped out, leaving 24 students who completed

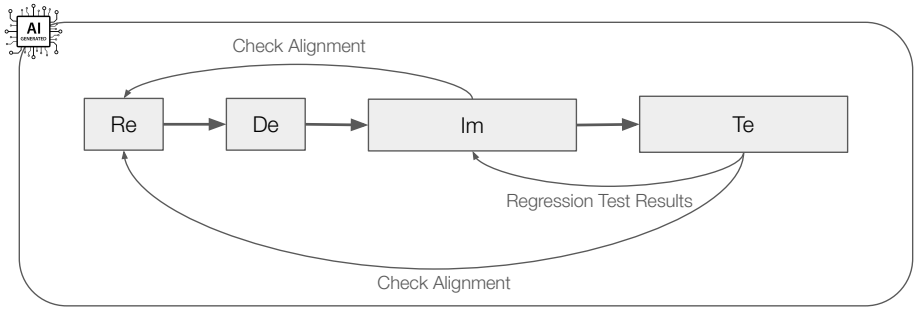
[3] https://github.com/features/copilot
[4] https://chat.openai.com/



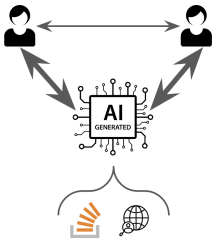
(a) Workflow **without** GenAI.



(b) Collaboration Pattern **without** GenAI.



(c) Workflow **with** GenAI.



(d) Collaboration Pattern **with** GenAI.

Figure 1. Impact of GenAI on Workflows and Collaboration Patterns (Re: Requirements, De: Design, Im: Implementation, Te: Testing).

interviews. Of the 24 students, 16 (64%) reported having professional, paid programming experience. All students stated that they have worked in a team to engineer software before. The reported team sizes in which students have previously worked varied from two to twenty people.

Each interview was designed to be around 25 minutes, and each student received a $25 gift card as compensation. The interviews were almost equally divided among two researchers, who are both authors of this paper. All interviews were performed through a video-conferencing application, audio-recorded, and transcribed automatically to ensure accurate capture of student responses. The automated transcriptions were reviewed by the authors of the paper to ensure accuracy.

3.4 Analysis of Qualitative Data

We conducted a thematic analysis with an emphasis on coding reliability on the qualitative interview data, an approach within the broader thematic analysis family as outlined by Braun & Clarke [6, 17, 28]. To guide the quality and transparency of our process, we drew on Braun & Clark's 15-point checklist for good thematic analysis [5]. Data analysis was performed using QSR International's NVivo 14 software[5] through an inductive approach, focusing on open coding without pre-existing theories influencing the analysis.

Coding was a collaborative effort between two coders (both authors of this paper), supported by a critical friend (also an author) who reviewed all coding to challenge assumptions and ensure depth in data interpretation [37]. The critical friend is an individual who provides an opportunity for a reflexive dialogue between researchers for critical reflection on coding practices, assumptions, and interpretations, ensuring consistency across the analysis process [9, 37, 38]. This coding strategy follows guidance for subjective approaches [5]. The coders independently conducted semantic coding on the interview data, focusing on the explicit content of participants' responses and surface-level meanings.

To build consistency in coding, a joint coding session was held early in the process and at the end, when the coders reviewed the mapping of each code to every statement in the interviews, underscoring the importance of thorough and iterative coding to ensure rigor in the research process. This collaborative approach was instrumental in refining the coding schema, enabling reliability of coding, and ensuring consistency across the dataset. If a new code emerged from an interview that was relevant to the research questions, previously coded interviews were revisited to check for the presence of this new code, ensuring comprehensive coverage.

[5] https://lumivero.com/products/nvivo/

This iterative coding process was critical in developing a robust thematic structure.

Themes were subsequently developed and named, with the process of writing and refining these themes being iterative and integrated throughout to ensure a thorough and comprehensive analysis. The final themes were developed collaboratively through multiple rounds of discussion and reflection on the coded data. The results reported in this paper are not comprehensive of all the themes identified bottom-up from the data. The decision to exclude sub-codes on certain topics, such as Usage of AI, Tool Improvements, Trust, and Concerns, was made after comparing the preliminary results with existing literature and engaging in a critical discussion following the analysis of the first ten interviews. The coders found that no new codes were developed after 15 interviews were coded, suggesting saturation. To ensure saturation was indeed reached, we continued until we had interviewed and coded the transcripts of 24 students. We considered this number to represent a sufficiently broad subset of the student pool. This methodology facilitated a thorough exploration of student perspectives, enhancing our understanding of the topics studied. Feedback from the critical friend was instrumental in maintaining analytical rigor, providing an external perspective to challenge and validate our findings.

3.5 The Study Population

The students who participated in the survey and interviews share backgrounds consistent with those of early-career professionals, with 64% of the interviewed students reporting professional programming experience on teams ranging from two to twenty members. Given that GenAI is a relatively new field, even experienced professional software developers have had limited exposure to these tools and are gradually adopting GenAI technologies while facing challenges themselves, for instance the lack of formal GenAI training [23, 30, 34]. In this respect, our students exhibit similarities to novice developers, who are also navigating a learning curve of integrating these emerging tools into their workflows.

The students in our study faced no restrictions regarding their use of GenAI, allowing them to freely explore and experiment with various tools. This stands in contrast to professional software developers, who are often constrained by organizational policies dictating tool selection, permissible inputs, and specific usage protocols for security, privacy, or intellectual property reasons [2, 19, 42]. The ability to apply these emerging technologies without these restrictions granted our students unique freedom, enabling them to experiment with GenAI across different aspects of programming, problem-solving, and collaboration, thereby fostering a deeper understanding of the tools' strengths and weaknesses and how they could integrate them into their work.

4 Use of Generative AI Tools in the Project

To explore how GenAI impacts students' workflows and collaborations in a software engineering course project, it is essential that students choose to use GenAI tools. Our survey results provide insights into the use of GenAI tools by students, helping determine whether the pre-condition of use of these tools was met.

From the survey, we learned that students did use GenAI tools extensively: 84% of the students reported using GenAI for some part of the development of the project. The students described using nine different GenAI tools, with the two most commonly cited being ChatGPT and GitHub Copilot. The primary reasons for using GenAI included troubleshooting code (81% of the 84%), asking how to use a library (65%), writing new code (64%), and optimizing existing code (57%). Students also described using GenAI for determining which library to use (40%) and creating data models (23%). Among the 16% of students who did not use GenAI, 35 (66% of the 16%) expressed concern that a wrong solution provided by the tool could negatively influence their project. Additionally, 32 students (60% of the 16%) were worried about becoming too reliant on the tools.

As part of the survey, we asked how students interacted with the GenAI tools to delve lightly into how the students collaborate with the tools. Only a small number of students (4%) found satisfactory answers immediately from GenAI without needing to refine their questions. More commonly, some collaboration was reported with the GenAI tools: 20% of students mentioned that they usually refine once before receiving a satisfying answer, and 70% said they typically needed to refine their questions multiple times to get a satisfying answer. Only 1% of the students chose not to refine their questions and ended up without a satisfactory answer, and 5% reported that even after multiple refinements, they still did not receive a satisfactory answer. Overall, 90% of the students indicated that they received a satisfactory answer with at least one round of question refinement. In other words, students were willing to engage and collaborate with the GenAI tools.

The survey results highlight that GenAI is perceived as valuable by the majority of the students in the course, with 94% who use GenAI finding them at least moderately useful. Despite facing difficulties with suggested solutions, most students are likely to continue using the technology. This indicates a strong overall acceptance and willingness to integrate GenAI into their workflows, underscoring the importance of investigating their impact on workflow and collaboration.

5 Results

We draw on our thematic analysis of interview data to explore the two research questions. Table 1 summarizes the themes that we developed that provide insight into each

research question. The table includes the number of participants from whom quotes support each theme. Representation of each theme ranged from 18 (75%) to 24 (100%) of the 24 participants, demonstrating that each theme was reflected across a substantial portion of the dataset. We discuss the themes supporting each research question.

5.1 RQ1: Software Engineering Workflows

We developed three themes related to how the students carried out their project tasks (**RQ1**).

Theme 1: Activity Changes. GenAI influenced what activities students performed in each phase of the development process by shifting or augmenting the kinds of work traditionally expected. For example, during the REQUIREMENTS phase, students described using GenAI to parse and summarize long and confusing specifications, helping them distill the most relevant information and streamline their understanding. Rather than deeply engaging with the specification documents themselves, students offloaded some of the comprehension work to GenAI, which allowed them to generate working summaries or jumpstart plans more quickly. These behaviors suggest a shift in how students engage with complex planning tasks, favoring speed and accessibility over deep understanding.

We identified a novel use of GenAI in this context. During the IMPLEMENTATION, students described using GenAI for "instant code review", changing how students debugged and improved their code. Instead of discussing their code with peers or waiting for TA feedback, students pasted code snippets into GenAI to get rapid feedback on structure, style, and correctness.

> *That one is more like to me it's like an instant code review. [...] I just give it the code. It gives me feedback and [I] go along with my day. (P3)*

Table 1. Themes and Support per Research Question.

RQ	Themes	# of Participants
RQ1 (Workflow)	1. Activity Changes	18
	2. Workflow Changes	18
	3. Flow Improvements	19
RQ2 (Team)	4. Collaboration Changes	18
	5. GenAI's Role in Collaboration	24

Theme 2: Workflow Changes. GenAI altered how much work students did within each phase and how they transitioned between the phases. Figure 1c depicts the overall changes in the students' workflow with GenAI.

In contrast to earlier offerings of the course (Figure 1a), students launched more directly into IMPLEMENTATION and spent less time in REQUIREMENTS and DESIGN. This change in approach was possible because GenAI gave students *"a good starting point"* (P9), allowing them to dive into IMPLEMENTATION more quickly. The diving in faster to IMPLEMENTATION is depicted in the shorter duration boxes for REQUIREMENTs and DESIGN in Figure 1c. Students even asked GenAI for *"a blueprint"* (P17) to get a starting point and jump-start the implementation.

> *If I have like ChatGPT as a tool, I can kind of skip the planning phase a bit more and just kind of go into doing stuff, cause I can kind of ask it. [...] How should I try and do this like, in what order should I do this? And then you can kind of just start go executing stuff rather than thinking, okay, I have to do. [...] (P24)*

A result of jumping in faster into IMPLEMENTATION is more effort spent by the students in activities related to IMPLEMENTATION, depicted by the longer box in Figure 1c. While this shift was not measured quantitatively through commit logs, students consistently described this transition in interviews and reflections.

Students also described changes in how they moved between phases, in particular, returning to REQUIREMENTS from the IMPLEMENTATION and TESTING phases. These 'returns', indicated by the additional arcs in Figure 1c, were due to asking GenAI to check alignment with specifications. For example, student P6 stated that when they are confused about what to implement, they refer back to the REQUIREMENTS with the help of GenAI, asking *"how the logic of the function could go"* according to the specs. As another example, student P17 referred back to the REQUIREMENTS when unsure how to implement or connect different parts of the code. These 'returns' to check alignment are faster for developers than fully understanding the requirements and design all up front. These findings suggest that students increasingly rely on GenAI as an intermediary to validate alignment between their code and the requirements, something they might otherwise seek from a teaching assistant or a more senior team member.

> *One thing I can do is like [...] put my test file on ChatGPT, and put like the main functions it is testing [...] and asking it like, Hey, is there any edge case that I'm missing? And a lot of times it might tell me that, hey? Either your test suite is pretty solid like, it's pretty proficient already, or it might just give you like, hey, this could be a potential thing that you haven't tested, and then I would go back and create a test for that. (P21)*

Table 2. Impacts of GenAI on Activities within the Development Cycle based on the Themes *Activity Changes* and *Workflow Changes*.

Phase	Impact and Evidence
Requirements	Students consulted GenAI for logic suggestions to clarify specifications, for advice on how to turn requirements into design, and for advice on how to start the project. Some students mentioned using GenAI to parse and summarize long and confusing specifications, providing a brief and generalized idea of the content, which streamlined their understanding and made the initial phase of their work more efficient.
Design	Students asked GenAI for high-level architectural suggestions, design advice for endpoints and the database, and how to approach the project. Instead of going to Google, watching tutorials, or reading documentation, they relied on GenAI to provide them with the necessary information.
Implementation	In addition to generating and refactoring code with GenAI, students described using GenAI as an instant code review step, providing them with feedback before committing the code and moving on to TESTING. Students asked GenAI how to implement a certain function or use a library instead of looking up the documentation themselves. They also requested GenAI to write documentation for their code and sought advice on how to integrate their code into the current codebase. Other impacted activities included debugging, finding errors in the codebase, and having error messages explained. Students consulted GenAI for these purposes. Additionally, students asked GenAI to explain code to them when they were unfamiliar or uncertain about its functionality.
Testing	Students described using GenAI to write, check, and identify missing cases to test software of interest. They also consulted GenAI to narrow down problems and find errors.

Theme 3: Flow Improvements. GenAI also helped students perform work more efficiently. Our students described three major ways in which the flow of work was improved.

First, GenAI helped students accelerate their work by providing a starting point. Students commonly used it to generate templates or initial code snippets they could then adapt to their needs.

> *I would say at the beginning of software development, when you kind of don't really have an idea of what you're doing, it's helpful to get started so you can ask it for like a beginning point, or something like that. (P24)*

Second, students described how GenAI reduced the effort needed to both locate and produce documentation. Rather than searching external sites like Stack Overflow, they turned to GenAI to explain or teach them how to use specific tools or libraries.

> *Well, most times I feel like, I don't even search up the documentation anymore. I just ask it to tell me like how to use it or like oh, like, teach me how to use this, or like something like that. (P18)*

Third, GenAI helped positively impact a student's flow by reducing mental load and helping to unblock the student.

> *Yeah, definitely does, because the like generating with AI is much faster and there's less mental load. (P14)*

These patterns reinforce prior findings in the literature. Liang et al. [25] similarly observed the use of GenAI tools for bootstrapping code, while Vaithilingam et al. [40] reported reductions in documentation search and writing effort. Our results build on these insights by situating them in the context of collaborative, multi-phase student projects. We discuss the impact of these flow improvements in Section 6.

5.2 RQ2: Team Interaction

Our second research question explores the impact of GenAI on how developers collaborate. From our qualitative analysis of the interview data, we developed two themes related to collaboration. For clarity of all themes identified, we continue the numbering of themes related to the first research question.

Theme 4: Collaboration changes. The use of GenAI altered how students communicated with each other. Figure 1d depicts the change from most interactions being between the human teammates to more communication happening between an individual team member and the GenAI tool. In addition, instead of directly interacting with outside documents, the human developers interacted with the information from documents through GenAI. We describe each of these two impacts in turn.

First, one significant shift was that GenAI often became the first point of contact, even before a teammate, particularly when teammates were unavailable. This use of GenAI did not just *reduced* the direct communication between teammates, hence the arrow is lighter between teammates in Figure 1d compared to Figure 1b, it also reduced students' time spent blocked, waiting on a partner for feedback, explanation, or

help. With GenAI, students could keep working independently and asynchronously, maintaining momentum even without synchronous team interaction.

> *Because I can ask copilot before I go to ask other developers in the team or something like that. (P9)*

Second, students used GenAI to help foster and seed discussions with their teammates, altering communication between teammates. Some students asked their teammate for help how to solve a certain problem and they referred them to GenAI to help them out *"I asked ChatGPT to do it"* (P18) and others informed their teammate that GenAI gave them a potential solution to an issue they have discussed before and they discussed that *"You know that problem we had with this I asked ChatGPT for like a potential solution, it gave me this back."* (P24)

> *[…] it's like them coming […] to me with a with a bug, and then I try to search and try ChatGPT and then I kind of like talk to ChatGPT about it. And then we discuss. Well, usually it's just me saying like this is what ChatGPT said and then they're like discuss on whether or not that's usable for our code and stuff. (P19)*

While these results are consistent with findings from Haque et al., our work extends their study by uncovering specific communication patterns that arise in collaborative, multi-phase project settings involving GenAI [18].

Theme 5: GenAI's Role in Collaboration. In addition to shaping communication, GenAI also influenced how teams divided their work and responsibilities based on the different views the students had about how GenAI fit in their teams, ranging from GenAI being just a tool to GenAI being a full team member. To the best of our knowledge, this is one of the first studies that considers how GenAI's role is perceived by students in a software engineering project and how students compare its qualities to human team members.

More than half of the students considered GenAI as a team member, likening GenAI to a junior developer, useful for providing general tips, helping when stuck, and learning new topics.

> *I think I like the analogy where you kind of treat ChatGPT or your AI tool as like a junior dev that you have on your own team. Even though it might not be the best at coding, you could still ask it as if it were if it like researched in a topic before, and just ask for maybe some general tips on what they found, and apply it to my own use case. (P1)*

Others, who viewed GenAI towards the teammate part of the spectrum, considered GenAI as an assistant useful for specific tasks, but who lacked the project-specific knowledge of a human team member.

> *It's definitely like an assistant. No, it can't be a teammate, because it doesn't have the knowledge that my teammate has about the project in terms of assistance it I think of it as someone that I can hand over a certain part of my code to, and it can evaluate that code and give me back suggestions to make the code more readable or to make it more precise and accurate. So it's something that I delegate what to, but not someone that I can work with. (P5)*

The other half of students considered GenAI as a tool, emphasizing its convenience, but cautioning against overreliance. Some likened it to advanced code completion or a faster Google search, useful for providing information and instant code reviews without taking on the responsibilities of a human team member.

> *It's more like a little Google search engine like someone who goes to search Google for you. So instead of you know, you looking through all the results, they look through all the results and summarize it for you. It's kind of like a sped up Google, I guess. (P7)*

Many students did note that GenAI was fundamentally different than a human team member because the technology does not engage in a two-way dialog and lacks the collaborative and intentional aspects of human teamwork.

> *I don't think it offers the full capacity of a teammate because like, if it's a teammate, then you make decisions together in terms of like, not just the code and making things run, but also like the intentions of the features that you're developing, for example and like you know owning our code base. (P2)*

6 Discussion

6.1 Pedagogical Implications

Each of the five themes identified in Section 5 have implications for how we train software engineers in an environment where AI-based tools play an increasingly integrated role. These implications are structured in terms of the traditional software engineering project phases depicted in Figure 1 and Table 2 and in terms of their impact on teamwork in the course project.

Requirements. Students frequently engaged with GenAI to synthesize complex documents into smaller pieces of work they could more directly work on (Theme 1). They would then use these smaller requirements to drive their implementation efforts. When authoring problem specifications, instructors should therefore be mindful that students may not be directly reading them, and prompt students to derive a deep understanding of requirements regardless. This places a greater importance on teaching testing as an integral part of the development workflow (e.g., via Test Driven Development (TDD)). Encouraging students to write tests that match the specification may induce them to engage directly with the specification and build better alignment with stakeholder intent. Future research could specifically investigate how students are using GenAI as part of TDD workflows and how effective GenAI is in this role.

Design. Since GenAI tools can generate source code directly from a given textual requirement, students altered their workflow in a way that largely bypassed the design process (Theme 2) (i.e., by asking GenAI to write the code for them). However, the role of design is more important than ever to ensure that any generated source code integrates well with the current project, and to ensure it fulfills the quality attributes (e.g., evolvability, testability) that are important to the long-term success of a project. In other words, generating *code* is different than generating *software*. Therefore it is ever-important to craft assignments where students must integrate code into existing software and are forced to evolve their own software. This would encourage students to better understand the trade-offs of their high-level design, whether it was initially generated by GenAI or themselves.

Implementation. Unsurprisingly, implementation activites were heavily impacted by GenAI. By allowing instant code review (Theme 1), supporting quicker prototyping (Theme 2), and allowing a direct feedback loop betweeen requirements and implementation (Theme 3), students spent a higher proportion of their time directly working with code. Since GenAI tools can generate code so quickly, students need to become more adept at understanding and judging the quality of code they did not write themselves. Pedagogically, a key concern is that students may over-rely on GenAI during implementation and, as a result, may under-engage with other critical aspects of the software development practices, such as requirements analysis, high-level design, or system architecture that are known to be important for long-term system success. This is problematic because it could encourage solutions that work well now but might not be amenable to future change which is important for all long-lived systems. As part of future work, we plan to analyse students' code repositories to explore patterns of implementation, code quality, and progress toward features. This will allow for a more quantitative investigation of how GenAI impacted the final solution and students' work patterns during the project, including potential comparisons with previous years.

Testing. Students understood the increased importance of testing when using GenAI and demonstrated this by both interrogating their test cases in terms of the requirements and generating test cases directly from requirements (Theme 2). For education, this suggests that testing activities should be more actively framed as a way both to validate the currently generated code, but also to ensure that newly-generated code does not cause unexpected regressions. While testing is good at checking alignment between requirements and source code, it is not able to validate alignment with an intended design; covering aspects of validation that are not entirely based on unit testing (e.g., code review or design walkthroughs) can help clarify this shortcoming.

Teamwork. Interestingly, GenAI seems to have *improved* how students collaborate with their teammates. Specifically, they spent less time blocked waiting for their partners (Theme 4), and were able to seed their discussions with GenAI provided suggestions instead of a blank sheet (Theme 5). In our project, integrating GenAI as part of the development team was a net positive that improved how the team worked. While GenAI did reduce communication between teammates, it also helped focus their teammate interactions on more important questions they actually felt they most needed help with. Furthermore, GenAI served as a useful (albeit incomplete) replacement for students' information needs—needs like checking requirements and performing code review that they would typically receive from a teaching assistant (or senior developer in industry). This suggests that it would be valuable to further investigate the quality of help that GenAI provides in this vein and how the role of course staff should evolve in response. The reduced peer interaction also raises questions about how students develop essential workplace soft skills such as negotiating ideas, giving feedback, resolving disagreements, and willingness to seek help—areas that future work may need to examine more closely.

6.2 Threats to Validity

We explored our research questions through an interview study and considered threats to the qualitative data and our interpretation of it.

Transferability of Results. We interviewed 24 students in a software engineering course. By selecting interview students randomly from a large set (79 potential students) that agreed to be interviewed, we reduced the bias of selection, increasing the likelihood of a representative sample of students in the class. All students in this study worked in pairs throughout the course, consistently collaborating with the same teammate on a shared software project. While this setting allowed us to closely examine GenAI's impact on small-team collaboration, the limited team size poses a threat to the transferability of our findings. In particular, the dynamics observed in two-person teams may not fully represent those in larger, more hierarchical teams typical of industry settings. Similarly, it remains unclear how well these findings generalize to learners with very different levels of experience, e.g., complete novice students or seasoned professionals might engage with GenAI in ways not captured in our study. We view our results as primarily providing hypotheses to explore in future studies, particularly with GenAI's role in larger team environments or industrial settings. Students were given unrestricted access to GenAI tools, meaning there were no constraints on which tools they could use, how frequently, or for what tasks. While this offered students considerable freedom, it introduces variability in tool use that may affect the transferability of our findings.

Differences in access to paid tools may have shaped how students engaged with GenAI. While this was not the focus of this study, future work should look into the potential impact of these differences.

Credibility. A potential threat to our study is the consistency of the interview process given two interviewers (both authors of this paper). To mitigate this risk, the two interviewers conducted joint sessions where they reviewed and discussed all the interview questions together to ensure a shared understanding of each question's intent and appropriate follow-up prompts. Additionally, a standardized checklist, which included key points and guidelines was developed that both interviewers used during the interviews. The reported results could also be impacted by the coding process. We used two methods to reduce this risk. First, we undertook an iterative process in which two researchers (both authors of this paper) independently coded the data, followed by comprehensive joint reviews. This approach allowed us to cross-check and validate our individual interpretations, reducing potential bias and enhancing the reliability of our findings. Despite these measures, the subjective nature of qualitative data interpretation remains a consideration. To further mitigate this risk, we incorporated the feedback of a critical friend (author of this paper), whose role was to rigorously challenge the assumptions and interpretations made, thereby providing an external perspective that enriched the depth and accuracy of our analysis.

7 Summary

GenAI is changing how students collaborate and build software together in an educational setting. Through an exploratory study involving 24 students enrolled in a project-based software engineering course, we examined how GenAI in a team-based setting influenced the development of three software versions across multiple sprints. While prior work has largely focused on individual use of GenAI for tasks like code generation or architectural guidance (e.g., [32, 40]), our findings show that GenAI also alters workflow structure and team interaction patterns. For example, students often moved into implementation earlier, using GenAI to bootstrap their work and retrospectively validate alignment with requirements and design. These findings extend those of prior literature (e.g., [32, 33, 39, 41]) by revealing how these workflow shifts create new feedback loops, such as retrospective requirement checks and "instant code reviews", both of which substitute for traditional human feedback mechanisms. With respect to team interactions, we found that students held differing views on whether GenAI functioned more as a tool or as a teammate. These perceptions influenced collaboration dynamics, with GenAI sometimes becoming the first point of contact over human teammates. These findings demonstrate how GenAI reshapes traditional workflows, creates new feedback loops, and alters team interactions in collaborative software development. This work offers pedagogical implications for educators designing and teaching project-based courses in an era of pervasive GenAI use. It highlights the need to support student development not just in technical skills, but also in prompt literacy, design reasoning, and collaborative awareness in GenAI-augmented teams.

Acknowledgments

We thank the individuals who participated in this research, those who provided us with insightful comments, and the reviewers for their constructive feedback. This work was funded, in part, by the Natural Science and Engineering Research Council of Canada (NSERC) [RGPIN-2022-03139].

References

[1] Duha Ali, Yasin Fatemi, Elahe Boskabadi, Mohsen Nikfar, Jude Ugwuoke, and Haneen Ali. 2024. ChatGPT in Teaching and Learning: A Systematic Review. *Education Sciences* (2024). https://api.semanticscholar.org/CorpusID:270536748

[2] Amazon Web Services. 2023. AWS Responsible AI Policy. https://aws.amazon.com/ai/responsible-ai/policy/ Last updated: September 28, 2023.

[3] Matin Amoozadeh, David Daniels, Daye Nam, Aayush Kumar, Stella Chen, Michael Hilton, Sruti Srinivasa Ragavan, and Mohammad Amin Alipour. 2024. Trust in Generative AI among Students: An exploratory study. In *Proceedings of the Technical Symposium on Computer Science Education (SIGCSE)*. 67–73. https://doi.org/10.1145/3626252.3630842

[4] Shraddha Barke, Michael B. James, and Nadia Polikarpova. 2023. Grounded Copilot: How Programmers Interact with Code-Generating Models. *Proc. ACM Program. Lang.* 7, OOPSLA1, Article 78 (apr 2023), 27 pages. https://doi.org/10.1145/3586030

[5] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative Research in Psychology* 3 (01 2006), 77–101. https://doi.org/10.1191/1478088706qp063oa

[6] Virginia Braun and Victoria Clarke. 2023. Toward good practice in thematic analysis: Avoiding common problems and be(com)ing a knowing researcher. *International Journal of Transgender Health* 24, 1 (2023), 1–6. https://doi.org/10.1080/26895269.2022.2129597 arXiv:https://doi.org/10.1080/26895269.2022.2129597

[7] Arifa I. Champa, Md Fazle Rabbi, Costain Nachuma, and Minhaz F. Zibran. 2024. ChatGPT in Action: Analyzing Its Use in Software Development. In *Proceedings of the International Conference on Mining Software Repositories (MSR)*. 182–186.

[8] Rudrajit Choudhuri, Dylan Liu, Igor Steinmacher, Marco Gerosa, and Anita Sarma. 2023. How Far Are We? The Triumphs and Trials of Generative AI in Learning Software Engineering. arXiv:2312.11719 [cs.SE] https://arxiv.org/abs/2312.11719

[9] Arthur Costa and Bena Kallick. 1993. Through the Lens of a Critical Friend. *Educational Leadership* 51 (01 1993).

[10] Marian Daun, Andrea Salmon, Thorsten Weyer, Klaus Pohl, and Bastian Tenbergen. 2016. Project-Based Learning with Examples from Industry in University Courses: An Experience Report from an Undergraduate Requirements Engineering Course. In *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET)*. 184–193. https://doi.org/10.1109/CSEET.2016.15

[11] Paul Denny, Juho Leinonen, James Prather, Andrew Luxton-Reilly, Thezyrie Amarouche, Brett A. Becker, and Brent N. Reeves. 2023. Promptly: Using Prompt Problems to Teach Learners How to Effectively Utilize AI Code Generators. arXiv:2307.16364 [cs.HC]

https://arxiv.org/abs/2307.16364

[12] Umama Dewan, Ashish Hingle, Nora McDonald, and Aditya Johri. 2025. Engineering Educators' Perspectives on the Impact of Generative AI in Higher Education. arXiv:2502.00569 [cs.CY] https://arxiv.org/abs/2502.00569

[13] Christof Ebert and Panos Louridas. 2023. Generative AI for Software Practitioners. *IEEE Software* 40, 4 (2023), 30–38. https://doi.org/10.1109/MS.2023.3265877

[14] Mohammadreza Farrokhnia, Seyyed Kazem Banihashem, Omid Noroozi, and Arjen Evert Jan Wals. 2023. A SWOT analysis of ChatGPT: Implications for educational practice and research. *Innovations in Education and Teaching International* 61 (2023), 460 – 474. https://api.semanticscholar.org/CorpusID:257832542

[15] Vahid Garousi, Zafar Jafarov, Aytan Movsumova, Atif Namazov, and Huseyn Mirzayev. 2025. Encouraging Students' Responsible Use of GenAI in Software Engineering Education: A Causal Model and Two Institutional Applications. arXiv:2506.00682 [cs.SE] https://arxiv.org/abs/2506.00682

[16] Marco Gerosa, Bianca Trinkenreich, Igor Steinmacher, and Anita Sarma. 2024. Can AI serve as a substitute for human subjects in software engineering research? *Automated Software Engineering* 31, 1 (Jan 2024), 12 pages. https://doi.org/10.1007/s10515-023-00409-6

[17] Greg Guest, Kathleen M. MacQueen, and Emily E. Namey. 2012. *Applied Thematic Analysis.* SAGE Publications, Inc. https://doi.org/10.4135/9781483384436

[18] Ebtesam Haque, Chris Brown, Thomas LaToza, and Brittany Johnson. 2024. Information Seeking Using AI Assistants. https://doi.org/10.48550/arXiv.2408.04032

[19] Steffen Herbold, Brian Valerius, Anamaria Mojica-Hanke, Isabella Lex, and Joel Mittel. 5555. Legal Aspects for Software Developers Interested in Generative AI Applications . *IEEE Software* 01 (Oct. 5555), 1–7. https://doi.org/10.1109/MS.2024.3476677

[20] Saki Imai. 2022. Is GitHub Copilot a substitute for human pair-programming? An empirical study. In *Proceedings of the International Conference on Software Engineering: Companion Proceedings*. 319–321. https://doi.org/10.1145/3510454.3522684

[21] Ranim Khojah, Mazen Mohamad, Philipp Leitner, and Francisco Gomes de Oliveira Neto. 2024. Beyond Code Generation: An Observational Study of ChatGPT Usage in Software Engineering Practice. *Proc. ACM Softw. Eng.* 1, FSE, Article 81 (Jul 2024), 22 pages. https://doi.org/10.1145/3660788

[22] Mohammad Amin Kuhail, Sujith Mathew, Ashraf Khalil, Jose Berengueres, and Syed Jawad Shah. 2024. "Will I be replaced?" Assessing ChatGPT's effect on software development and programmer perceptions of AI tools. *Science of Computer Programming* 235 (05 2024). https://doi.org/10.1016/j.scico.2024.103111

[23] Ze Shi Li, Nowshin Nawar Arony, Ahmed Musa Awon, Daniela Damian, and Bowen Xu. 2024. AI Tool Use and Adoption in Software Development by Individuals and Organizations: A Grounded Theory Study. arXiv:2406.17325 [cs.SE] https://arxiv.org/abs/2406.17325

[24] Jenny T. Liang, Chenyang Yang, and Brad A. Myers. 2024. A Large-Scale Survey on the Usability of AI Programming Assistants: Successes and Challenges. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering* (Lisbon, Portugal) *(ICSE '24)*. Association for Computing Machinery, New York, NY, USA, Article 52, 13 pages. https://doi.org/10.1145/3597503.3608128

[25] Jenny T. Liang, Chenyang Yang, and Brad A. Myers. 2024. A Large-Scale Survey on the Usability of AI Programming Assistants: Successes and Challenges. In *Proceedings of the International Conference on Software Engineering (ICSE)*. Article 52, 13 pages. https://doi.org/10.1145/3597503.3608128

[26] Daye Nam, Andrew Macvean, Vincent Hellendoorn, Bogdan Vasilescu, and Brad Myers. 2024. Using an LLM to Help With Code Understanding. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering* (Lisbon, Portugal) *(ICSE '24)*. Association for Computing Machinery, New York, NY, USA, Article 97, 13 pages. https://doi.org/10.1145/3597503.3639187

[27] Sydney Nguyen, Hannah McLean Babe, Yangtian Zi, Arjun Guha, Carolyn Jane Anderson, and Molly Q Feldman. 2024. How Beginning Programmers and Code LLMs (Mis)read Each Other. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 651, 26 pages. https://doi.org/10.1145/3613904.3642706

[28] Lorelli S. Nowell, Jill M. Norris, Deborah E. White, and Nancy J. Moules. 2017. Thematic Analysis: Striving to Meet the Trustworthiness Criteria. *International Journal of Qualitative Methods* 16, 1 (2017), 1609406917733847. https://doi.org/10.1177/1609406917733847 arXiv:https://doi.org/10.1177/1609406917733847

[29] E. Oh and A. van der Hoek. 2002. Towards game-based simulation as a method of teaching software engineering. In *32nd Annual Frontiers in Education*, Vol. 3. S2G–. https://doi.org/10.1109/FIE.2002.1158674

[30] Ruchika Pandey, Prabhat Singh, Raymond Wei, and Shaila Shankar. 2024. Transforming Software Development: Evaluating the Efficiency and Challenges of GitHub Copilot in Real-World Projects. arXiv:2406.17910 [cs.SE] https://arxiv.org/abs/2406.17910

[31] David Lorge Parnas. 2011. *Software engineering: Multi-person development of multi-version programs.* Springer Berlin Heidelberg, 413–427. https://doi.org/10.1007/978-3-642-24541-1_31

[32] Asha Rajbhoj, Akanksha Somase, Piyush Kulkarni, and Vinay Kulkarni. 2024. Accelerating Software Development Using Generative AI: ChatGPT Case Study. In *Proceedings of the Innovations in Software Engineering Conference (ISEC)*. Article 5, 11 pages. https://doi.org/10.1145/3641399.3641403

[33] Sanka Rasnayaka, Guanlin Wang, Ridwan Shariffdeen, and Ganesh Neelakanta Iyer. 2024. An Empirical Study on Usage and Perceptions of LLMs in a Software Engineering Project. In *Proceedings of the 1st International Workshop on Large Language Models for Code* (Lisbon, Portugal) *(LLM4Code '24)*. Association for Computing Machinery, New York, NY, USA, 111–118. https://doi.org/10.1145/3643795.3648379

[34] Daniel Russo. 2024. Navigating the Complexity of Generative AI Adoption in Software Engineering. *ACM Trans. Softw. Eng. Methodol.* 33, 5, Article 135 (June 2024), 50 pages. https://doi.org/10.1145/3652154

[35] Jaakko Sauvola, Sasu Tarkoma, Mika Klemettinen, Jukka Riekki, and David Doermann. 2024. Future of software development with generative AI. *Automated Software Engineering* 31, 1 (2024), 26. https://doi.org/10.1007/s10515-024-00426-z

[36] David C. Shepherd, Felipe Fronchetti, Yu Liu, Daqing Hou, Jan DeWaters, and Mary Margaret Small. 2022. Project-Sized Scaffolding for Software Engineering Courses. In *2022 IEEE/ACM First International Workshop on Designing and Running Project-Based Courses in Software Engineering Education (DREE)*. 27–31. https://doi.org/10.1145/3524487.3527362

[37] Brett Smith and Kerry McGannon. 2017. Developing Rigor in Qualitative Research: Problems and Opportunities within Sport and Exercise Psychology. *International Review of Sport and Exercise Psychology* 11 (05 2017). https://doi.org/10.1080/1750984X.2017.1317357

[38] Andrew C. Sparkes and Brett Smith. 2013. *Qualitative Research Methods in Sport, Exercise and Health: From Process to Product* (1st ed.). Routledge. https://doi.org/10.4324/9780203852187

[39] Ben Arie Tanay, Lexy Arinze, Siddhant S. Joshi, Kirsten A. Davis, and James C. Davis. 2024. An Exploratory Study on Upper-Level Computing Students' Use of Large Language Models as Tools in a Semester-Long Project. arXiv:2403.18679 [cs.SE] https://arxiv.org/abs/2403.18679 Accepted to the 2024 General Conference of the American Society for Engineering Education (ASEE).

[40] Priyan Vaithilingam, Tianyi Zhang, and Elena L. Glassman. 2022. Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI EA '22)*. Association for Computing Machinery, New York, NY, USA, Article 332, 7 pages. https://doi.org/10.1145/3491101.3519665

[41] Muhammad Waseem, Teerath Das, Aakash Ahmad, Peng Liang, Mahdi Fahmideh, and Tommi Mikkonen. 2024. ChatGPT as a Software Development Bot: A Project-Based Study. In *Proceedings of the International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*, Hermann Kaindl, Mike Mannion, and Leszek A. Maciaszek (Eds.). 406–413. https://doi.org/10.5220/0012631600003687

[42] Yi Zeng, Kevin Klyman, Andy Zhou, Yu Yang, Minzhou Pan, Ruoxi Jia, Dawn Song, Percy Liang, and Bo Li. 2024. AI Risk Categorization Decoded (AIR 2024): From Government Regulations to Corporate Policies. https://doi.org/10.48550/arXiv.2406.17864

Received 2025-06-30; accepted 2025-08-01