Software Engineering Education in the Age of AI: Challenges, Opportunities, and Pedagogical Shifts

Remzi Cetin

University of Innsbruck, Innrain 52, 6020 Innsbruck, Austria
remzi.cetin@student.uibk.ac.at

Abstract. Developments in artificial intelligence, especially generative AI and large language models, are changing how students learn and how educators teach software engineering at university level. While students already use tools like *ChatGPT* and AI coding assistants for code generation, debugging, and explanations of concepts, there is still limited overall understanding of how these tools are integrated into software engineering education and what educational consequences follow. In this thesis, a systematic literature review is conducted following the guidelines proposed by *Kitchenham*. It explores how AI is used in software engineering education at university-level, how educators adapt their teaching in response, and what impacts have been reported. Peer-reviewed conference and journal papers published between 2021 and 2025 were retrieved from major digital libraries and systematically screened, assessed for quality, and synthesized with respect to (i) AI tools and application areas, (ii) pedagogical responses by educators, and (iii) reported impacts on teaching and learning. The findings show that the literature is dominated by general-purpose LLM tools and that most use cases focus on introductory programming contexts. Across studies, educators increasingly move from prohibition toward guided integration of AI.

Keywords: software engineering education · generative AI · large language models · ChatGPT · systematic literature review.

1 Introduction

Nowadays, how software is specified, designed, implemented, and maintained is changed by the developments in artificial intelligence (AI), and also the rise of generative models such as large language models (LLMs) [16, 40, 64]. Tools such as *ChatGPT* [80], *GitHub Copilot* [31], or *Gemini* [32] are more and more integrated into development workflows. From automated code completion and debugging support to test generation and assisting in documentation [52]. Not only professional software engineering practices are changed by this fast technological shift, but also how future software engineers are educated [89].

In higher education, many students now use AI tools that can generate code, explain programming concepts, or provide feedback on assignments [9, 70, 96].

For many of them, these tools have become a natural part of their study routine [96]. Students use them even before they open lecture slides or ask their professors questions [82, 110]. This creates both opportunities and concerns. On the one hand, AI can personalize learning, lower entry barriers, and make concepts easier to understand [19, 35, 48]. On the other hand, there is a risk that students may over-rely on AI tools and their generated solutions and engage less deeply with core concepts. Therefore, there is the risk that students may develop a superficial understanding of software engineering principles [65, 76, 103].

These developments lead to the situation that teachers or educators ask themselves important questions. How should courses be designed when students can generate code or explanations on demand [42, 87, 89]? How can AI be used by educators in order to support learning while still helping to improve critical thinking and problem solving [19, 40]? What kinds of guidance, rules, or scaffolding are necessary to prevent over-reliance, academic integrity issues, and shallow learning [43]? And more broadly, on which areas of software engineering education does AI currently have the most effect, and whether there is any evidence about its impact on learning outcomes, motivation, and student behavior [5, 98, 109]?

AI tools have become widely available, but there is still little clear understanding of how AI is actually being used in university-level software engineering education and what consequences this has in practice. To address this gap, this thesis conducts a systematic literature review on *software engineering education in the age of AI*.

The work is guided by the following research questions:

- **RQ**$_1$: In what areas of software engineering education is AI currently being used, and which AI tools or techniques are reported in the literature?
- **RQ**$_2$: How are software engineering educators adjusting their teaching methods in response to AI integration?
- **RQ**$_3$: What impact does AI integration have on software engineering education?

By finding answers to these questions, the goal of the thesis is to provide an overview of the current landscape of how AI is used in software engineering education (RQ$_1$), identify pedagogical patterns and strategies on how to integrate AI into teaching (RQ$_2$), and report benefits, challenges, and risks that are associated with AI from the perspective of the students and teachers (RQ$_3$).

2 Background and Related Work

Software engineering education has always been influenced by tools. These include integrated developments environments, version control systems, automated build pipelines, unit-testing frameworks, continuous integration platforms, or

static analysis tools. These tools have an influence on how students work, what kind of tasks instructors assign, and how the students' learning is assessed. In this sense, AI can be basically seen as the next important tool shift [42, 89]. But AI might be more concerning because it can generate content that looks like a student's own work. In contrast to traditional help, which is used for programming for instance, systems that are LLM-based are able to generate complete solutions, suggest alternative designs, explain code in natural language, or create tests and documentation. Students are able to interact with these systems without even needing a deep technical setup [47, 70].
This shift introduces new challenges for teaching and assessment of course. When students can get solutions that are correct and working or detailed explanations right away, classic assignments may no longer measure the intended learning objectives [81]. At the same time AI tools can have a positive impact on education. They may provide rapid feedback, reduce frustration, help students overcome entry barriers, and support self-paced learning. That is why the current situation can be described by the tension between two perspectives. One perspective sees AI as an opportunity to improve learning through personalization, scaffolding, and support [28, 72, 90]. The other highlights risks. These risks include over-reliance, reduced conceptual understanding, biased or incorrect outputs, and academic integrity concerns [68, 97]. Universities and educators are therefore required to define policies and teaching strategies that brings balance to these aspects [93, 96].

A recent systematic literature review [83] shows that LLMs are widely used in programming education to scale feedback and evaluation. It points out that tools that are LLM-based can generate immediate formative feedback, support debugging and repair programs, and enable more personalized guidance. This can reduce the overall workload and improve iteration speed (see Section 4.5). However, it was also stated that these benefits depend heavily on how carefully the prompt was designed and how instructions were given, because LLM outputs can be incorrect or misleading and may introduce bias.
Another review research also shows the limitations of LLMs when they are used as coding assistants [84]. While models often produce fluent explanations and syntactically plausible code, they often fail on more complex tasks (see Section 4.6). The paper by Gaitantzi and Kazanidis [30] highlights that research is concentrated in programming, while areas like database education remain understudied. Here, AI can provide on-demand tutoring and feedback but struggles again with correctness in complex cases. Another related work [11] strongly connects LLM usage to academic integrity and assessment redesign. It is reported that educators try to develop different pedagogical strategies (see Section 4.4) with the goal of making student understanding more visible.

Many of these AI tools can currently be accessed by everyone and research on AI in software engineering education is still developing. Studies differ in their scope and methods they use, which ranges from small course interventions and reports

of experiences [5, 50] to survey and evaluations of teaching approaches that are AI-supported [29, 64, 75]. This motivates the need for a structured synthesis of current evidence, which is provided through the systematic literature review presented in this thesis.

3 Methodology

The thesis follows the guidelines for a systematic literature review process proposed by Kitchenham [49]. It is based on guidelines for conducting SLRs in software engineering. The review uses the core phases: *planning*, *conducting*, and *reporting*.

3.1 Review Process

In the *planning* phase, the aim of the review and three research questions were defined. A protocol was created which specified the search scope (digital libraries), the search strings, and the criteria for including and excluding studies. In the *conducting* phase, the protocol was implemented. Studies were identified, duplicates were removed, screening was conducted, and the inclusion/exclusion criteria were applied. After that, each study was assessed using a point-based quality checklist, data was extracted, and synthesized.

Table 1. Overview of selected papers from the exploratory set

Paper Title	**Source / Venue**
Will Your Next Pair Programming Partner Be Human? An Empirical Evaluation of Generative AI as a Collaborative Teammate in a Semester-Long Classroom Setting [59]	L@S (2025)
What an AI-Embracing Software Engineering Curriculum Should Look Like: An Empirical Study [89]	IEEE Software
Using Generative AI to Create User Stories in the Software Engineering Classroom [1]	Conference on Software Engineering Education and Training (CSEE&T, 2024)
Enhancing Python learning with PyTutor: Efficacy of a ChatGPT-based intelligent tutoring system in programming education [112]	Computers and Education: Artificial Intelligence
Teaching CS50 with AI: Leveraging Generative Artificial Intelligence in Computer Science Education [58]	SIGCSE Technical Symposium on Computer Science Education (2024)
Unlocking Potential with Generative AI Instruction: Investigating Mid-level Software Development Student Perceptions, Behavior, and Adoption [33]	SIGCSE Technical Symposium on Computer Science Education (2025)
Students' Perception of ChatGPT in Software Engineering: Lessons Learned from Five Courses [52]	Conference on Software Engineering Education and Training (CSEE&T, 2025)
The effect of generative artificial intelligence (AI)-based tool use on students' computational thinking skills, programming self-efficacy and motivation [113]	Computers and Education: Artificial Intelligence
Can Generative Artificial Intelligence Outperform Self-Instructional Learning in Computer Programming? [66]	Applied Sciences
Applying Large Language Models to Enhance the Assessment of Java Programming Assignments [34]	ACM International Conference on the Foundations of Software Engineering – Companion (2025)

3.2 Search Strategy

The search strategy began with an informal exploratory search. Using *Web of Science*, initial queries like *AI software engineering education* were executed. From this phase, approximately 50-60 papers were collected. They were used to understand broadly of how AI is currently discussed in software engineering education, to identify common areas where AI is integrated (e.g. AI tutoring and support, student prompting and behavior) and to detect frequently used keywords, such as *"generative AI"*, *"large language models"* and *"ChatGPT"*, as well as education-related keywords such as *"programming education"*.

From the exploratory set, a known set of ten relevant studies were manually selected (see Table 1). These papers covered different AI tools and areas in which they were used. Across all databases and digital libraries used - *Web of Science* [21], *IEEE Xplore* [38], *ACM Digital Library* [10], and *ScienceDirect* [27] - the queries achieved an overall recall of at least 80% on the benchmark set. These sources were selected because they cover the major venues in software engineering, computing education and educational technology. Each of the search queries (see Table 2) were executed in all the four sources. When necessary, search syntax was adapted to the respective platform (e.g. using *TS=* for *Web of Science*).

Table 2. Search queries

Research Question	Search Query
RQ_1	("software engineering education" OR "computer science education") AND ("artificial intelligence" OR "machine learning" OR "generative AI" OR "large language models" OR ChatGPT) AND (tool OR system OR technique OR application OR "AI tool" OR "AI technique") AND ("university" OR "higher education" OR "tertiary education" OR "postsecondary education" OR "college" OR "academic" OR "undergraduate" OR "graduate")
RQ_2	("software engineering education" OR "computer science education") AND ("artificial intelligence" OR "machine learning" OR "generative AI" OR "large language models" OR ChatGPT) AND (teaching OR pedagogy OR "teaching methods" OR curriculum OR "educational practices" OR instruction) AND ("university" OR "higher education" OR "tertiary education" OR "postsecondary education" OR "college" OR "academic" OR "undergraduate" OR "graduate")
RQ_3	("software engineering education" OR "computer science education") AND ("artificial intelligence" OR "machine learning" OR "generative AI" OR "large language models" OR ChatGPT) AND (impact OR effect OR evaluation OR benefit OR challenge OR outcome OR "student performance") AND ("university" OR "higher education" OR "tertiary education" OR "postsecondary education" OR "college" OR "academic" OR "undergraduate" OR "graduate")
Broad Query	("software engineering education" OR "computer science education" OR "computing education" OR "programming education") AND ("artificial intelligence" OR AI OR "machine learning" OR "generative AI" OR "large language models" OR LLM OR ChatGPT) AND ("university" OR "higher education" OR "tertiary education" OR "postsecondary education" OR "college" OR "academic" OR "undergraduate" OR "graduate")

3.3 Inclusion and Exclusion Criteria

To ensure that the selected studies are relevant, a set of inclusion and exclusion criteria was defined. These criteria were applied during both title/abstract and full-text screening.

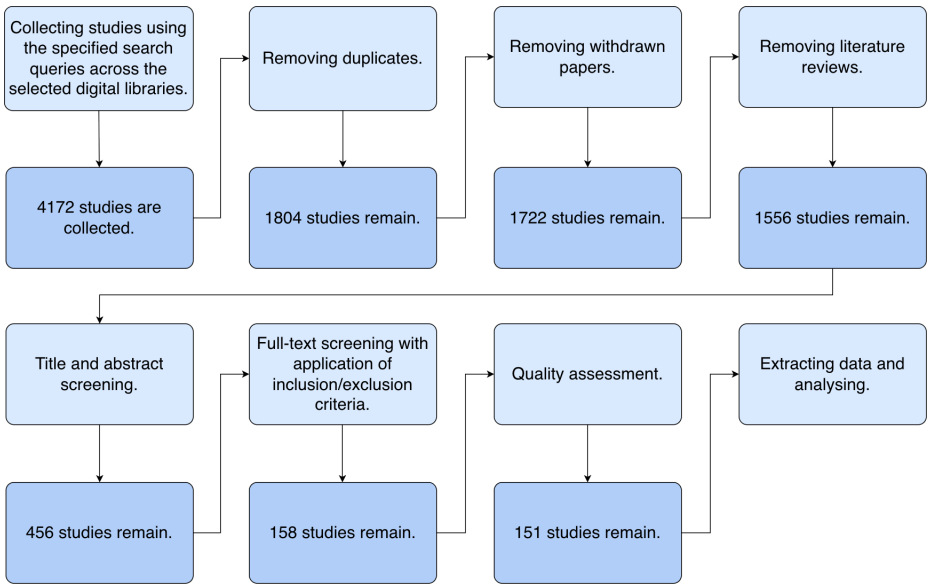

Fig. 1. Study selection process

Inclusion Criteria:
A paper was included in the review if all the following criteria were met:

1. The paper is written in English.
2. The paper was published between 2021 and 2025.
3. The paper is a peer-reviewed conference or journal publication.
4. The full text of the paper is accessible.
5. The paper must be relevant to the topic of AI in software engineering education in a university context, satisfying at least one of the defined research questions (RQ_1 - RQ_3).

Exclusion Criteria:
A paper was excluded if any of the following applied:

1. The paper focuses on AI in general education or other disciplines, without a clear link to software engineering or closely related computing education.
2. The paper focuses on AI for software engineering practice without an explicit educational context.
3. The paper itself is a literature review.

3.4 Quality Assessment

A quality assessment was conducted so that the trustworthiness of the included studies could have been evaluated. The assessment consisted of two parts: general quality questions and research-question specific questions (see Table 3).

Table 3. Quality Assessment Questions

Category	Question
General	1. Does the study clearly state its research objective or purpose? 2. Does the study provide sufficient background or motivation for the research? 3. Is the context of the study described in sufficient detail? 4. Is the research design appropriate for addressing the stated objectives? 5. Are the data collection and analysis procedures clearly explained? 6. Are the findings supported by sufficient data, examples, or empirical evidence? 7. Are the results clearly presented and logically connected to the research questions or objectives? 8. Does the study discuss limitations, assumptions, or potential biases? 9. Does the study acknowledge threats to validity or constraints in generalizing results?
RQ$_1$	10. Does the study specify the AI tools, methods, or technologies used in software engineering education? 11. Does the study describe how these tools are integrated into the learning or teaching process?
RQ$_2$	12. Does the study describe any pedagogical or curricular adjustments due to AI? 13. Does the study include examples of AI-supported learning or teaching activities?
RQ$_3$	14. Does the study report measurable or observed impacts on students' learning outcomes, engagement, or performance? 15. Does the study discuss how educators and students perceive the use of AI in software engineering education?

For each question, a study received 1 point if the criterion was fully met, 0.5 points if it was partly met, and 0 points if it was not met. Research-question-specific questions were only answered when the study also addressed them. For each of the relevant RQ, the points of its questions were summed and normalized to a maximum of 1:

$$\mathrm{RQ}_k^{\mathrm{norm}} = \frac{\text{score for } \mathrm{RQ}_k}{2}$$

The normalization of the RQ-specific scores was integrated so that the comparability across research questions can be ensured, independent of the number of

quality criteria questions associated with each RQ. Although all research questions in this review are associated with the same number of quality criteria, normalization was still applied to define a general and extensible scoring model. This makes it possible that quality assessment remains comparable in cases where research questions are associated with differing numbers of criteria.
If a study contributed to more than one research question, the normalized scores were averaged:

$$\text{RQ-score} = \frac{\sum_{k=1}^{n} \text{RQ}_k^{\text{norm}}}{n}$$

This prevents studies who address multiple research questions from being favored due to quantity. The final quality score of a study is defined as: Final QA score = General score + RQ-score. Studies achieving a final score below 7 were not included in the synthesis process, except when they reached an *RQ-score* of 1. From the 158 studies (see Fig. 1), 151 remain. Thus, all the outcomes from Section 4 are based on the review of these 151 studies.

3.5 Data Extraction

For each primary study, relevant information was collected in a data-extraction spreadsheet. The aim was to capture both general study characteristics and details directly related to the three research questions:

- **Identification data**: title, author(s), year of publication, source/venue
- **Study characteristics**: publication type, study type/methodology, participants, education level or context
- **AI usage ($\mathbf{RQ_1}$)**: AI tool used, its purpose and function in the course, and the area of software engineering education it was applied
- **Pedagogical changes ($\mathbf{RQ_2}$)**: the pedagogical approach or teaching method, any adjustments by educators when integrating AI
- **Impacts and perceptions ($\mathbf{RQ_3}$)**: reported benefits or opportunities, reported challenges or risks, as well as the impact on learning outcomes

4 Results

This section reports the results of the systematic literature review based on the analysis of 151 primary studies. The findings are organized according to the three research questions. They provide a structured overview of the AI tools used in software engineering education (RQ_1), pedagogical approaches applied by educators in response to AI integration (RQ_2), and reported benefits, risks, and impacts on learning outcomes (RQ_3).

4.1 AI Tools Used - RQ1

The 151 reviewed studies were analyzed with respect to the specific AI tools which have been used. It is important to note that a single study may reference more than one AI tool, and therefore the frequencies and percentages do not sum up to 100%.

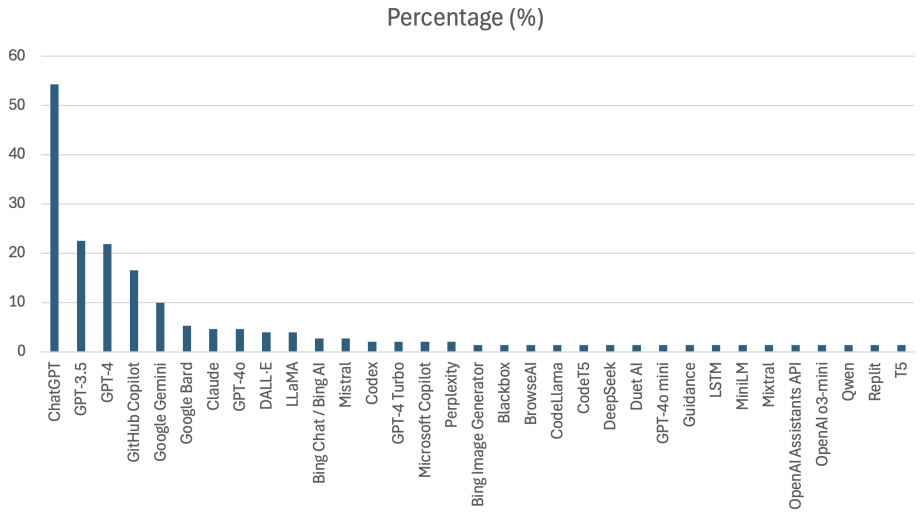

Fig. 2. AI Tools Usage

As can be clearly seen in Fig. 2, among these tools, *ChatGPT* and related *GPT*-based models dominate the research landscape. This shows that they are seen as useful in many ways and easy-to-use tools that support many educational tasks like code explanation, debugging assistance, and general learning support [36, 55]. AI-powered coding assistants like *GitHub Copilot* represent another important category of tools used in studies [4, 77].
Google Gemini from Google's ecosystem is also present. In the paper by E. Smolić et al. [25], it was analyzed how it can grade students' *C* programming exam solutions compared to human teaching assistants. Cubillos et al. [18] examined whether it can help learning data structures in *C* compared to a traditional method with an explanatory *YouTube* video.

Overall, the findings show that software engineering education research in the age of AI is currently heavily centered on general-purpose large language models. While a growing number of studies experiment with open-source LLMs (e.g. *LLaMA* [95]) and multi-model architectures, their usage remains comparatively limited. So, the most evaluations still rely on a small set of widely accessible, commercial AI tools such as *ChatGPT* or *GitHub Copilot*.

Table 4. Overview of infrequently reported AI tools and models

Category	**Tools / Models**
Coding assistants and IDE-integrated tools	Amazon CodeWhisperer [41], Codeium [78], Cody [63], Cursor [111], JetBrains AI Assistant [13], TabNine [63], AskCodi [63]
Educational tutoring systems and course-specific assistants	AIELA [104], ALEKS [48], AlgoBo [45], Artemis [14], AutoTA [22], CS50 Duck [58], Tutor Kai [91], GreAIter [34], IPSSC [43], Iris Tutor [14], KOGI [51], Knewton [48], PuppyCodeReview [102], PyChatAI [7], PyTutor [112], SSD-Tutor [73], TeachYou [45], VisualCodeMOOC [56], UnrealMentorGPT [114], StepGrade [61], SPARK [60]
LLM families and foundation models	GPT-2 [100], Gemma [22], PaLM [53], Phi-3 [22], TinyBERT [90], RoBERTa [17], BART [24]
AI platforms, APIs, orchestration, and infrastructure	Azure OpenAI Studio [108], Vertex AI [65], Vision Studio [108], OpenRouter [102], Hugging Face [63], Replicate [63], LangChain [22], Langdock [23], Faiss [101], ScraperAPI [63]
Search/answer engines	Phind [88], SQLAI [88], You.com [9]
Generative media (image/audio)	MidJourney [111], Stable Diffusion [50], Whisper [105]

Table 4 summarizes AI tools and models that were mentioned only once across the analyzed studies and were therefore grouped by functional category. A notable group is about *coding assistants and IDE-integrated tools*. Another important group includes *educational tutoring systems and course-specific assistants*. These tools are usually developed for courses or pedagogical experiments, and they focus on specific learning goals (see Section 4.4).

4.2 Purpose and Function - RQ1

The most frequently identified use of AI tools is about explaining concepts, code, or error messages (see Fig. 3). Students use them to understand concepts, interpret the feedback generated by the AI tool, and get contextual explanations [99]. They see these tools as a smart friend or a study buddy [70]. In the study by Mutanga et al. [68], it was reported that most of the students asked "tell me what X is" questions and students treated the LLM like a smart textbook.

Another prominent use case is debugging and bug fixing. Thus, AI tools help students to find syntax errors, logical flaws, and runtime issues in their code. Students paste error messages, receive the proposed fixes, and iterate [99].
Code generation and completion represents another functionality of many AI tools. They are used to generate code fragments or complete partial solutions.

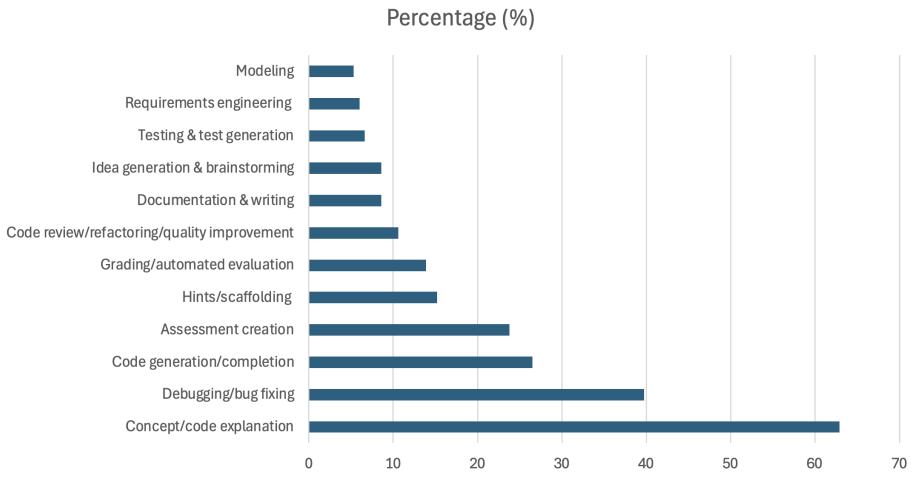

Fig. 3. Purpose and Function of the AI tools

While this capability can significantly increase productivity and lower entry barriers for beginners, there is a risk of over-reliance [23, 92].
AI tools are also used for assessment creation, including the generation of programming exercises, quizzes, and exam questions to reduce the workload of educators. However, it was often highlighted that AI-generated questions still need human review, because mistakes and misalignment with the course and learning objectives can occur [107]. Educators also may use AI for grading. They assess student submission, assign scores, and provide feedback. This is beneficial because it can make grading faster, more consistent, more thorough, as long as humans stay in the loop to review [34]. AI tools can, for example, handle the repetitive scoring, while educators handle oversight, edge cases, and final decisions [20].
Another important use of AI tools is for hints and scaffolding without providing full solutions. Systems integrate AI carefully to offer partial guidance, leading questions, or suggestions (see Section 4.4).
The categories *Requirements Engineering* and *Modeling*, particularly UML diagrams, show that AI tools are beginning to support higher-level software engineering activities. For example, the paper by A. M. Abdelfattah et al. [3] describes how *ChatGPT* is used as an assistant to learn requirements engineering. The authors suggest that students actively interact with *ChatGPT* to define requirements, write user stories, create *UML* diagrams (use case-, class-, sequence-diagrams), and improve them through feedback in an iterative way.

4.3 Area of Software Engineering Education - RQ1

It can be clearly seen that introductory programming and coding is the most dominant area where AI is used (see Fig. 4). AI is able to help students to

solve assignments or to learn the fundamental concepts. Students may want to learn different languages, at different speed, with different resources. LLMs allow explanations tailored to the student's level and learning at one's own pace [15].

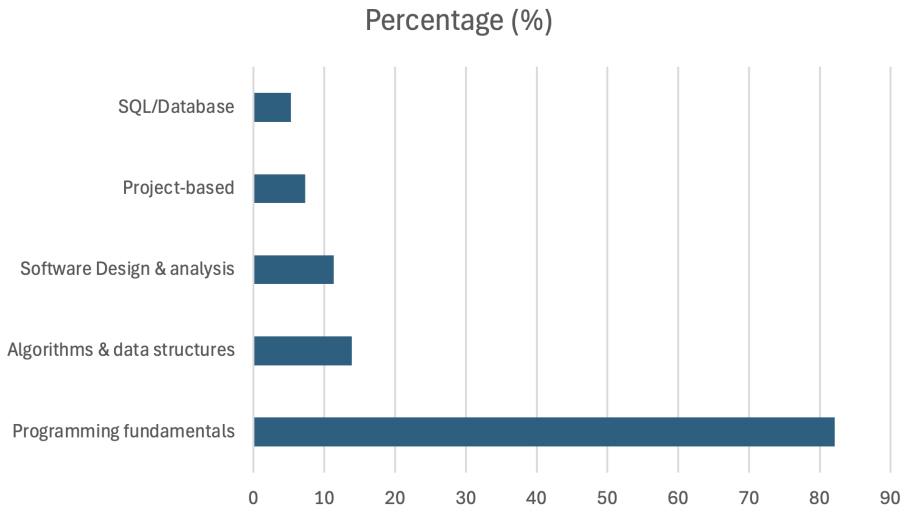

Fig. 4. Areas of SE Education

Of course, it occurs much less frequently, but the usage of AI in *algorithms and data structures* still appears several times. AI is used here, for example, to handle exam-style, free response questions [57] or it is used to explicitly learn data structures [18].
A smaller category is project-based software engineering education. Analyses are conducted on how LLMs like *ChatGPT* and *GitHub Copilot* are actually used by software engineering students while they are building a big semester project [4, 16] or on how to integrate an AI-assistant that support students during a software engineering capstone project [74].
The area *software design and analysis* are about activities related to requirements engineering, system analysis, and modeling, including the use of user stories, architectural descriptions, and UML diagrams. A smaller group of studies focuses on SQL and database education including relational schema design, and query formulation [26].

4.4 Pedagogical approaches - RQ2

Recurring teaching approaches either guide how students use AI [6, 33, 72], integrate AI as tutor [43, 58], teach students how to use AI critically [23], or redesign assessment or curricula in order to integrate AI into software engineering edu-

cation in a proper way [2, 113]. Across these approaches, a clear trend shows up. Instead of trying to ban AI, educators are focusing on guided integration.

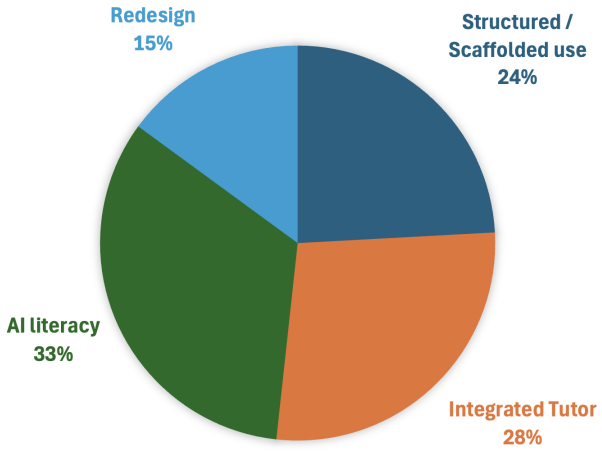

Fig. 5. Distribution of pedagogical approaches

Structured or scaffolded use of AI: A common approach is to limit and guide how students use AI instead of giving them the opportunity to adopt the "*ask ChatGPT for the solution*" behavior (see Fig. 5). AI is used at specific stages of the learning process, like explaining concepts, debugging, or reviewing code. The goal is basically to shift attention from simply getting the solution to understand how and why decision are made. Typical implementations include phased workflows (e.g., learn → use AI → reflect), structured prompts, and small iterations instead of one-shot generation of the solution [6]. The limitation of AI is something that is intentional: helpful enough to reduce frustration, but not so powerful that students outsource the entire task [19]. So, the main idea is that a structured approach prevents this copy-paste behavior.
The study by Nathaniel et al. [72] verifies whether generative AI can be integrated into introductory programming education without undermining higher-order thinking skills and understanding of programming logic. It was a seven-week experiment with 25 computer science students who are new to C++. The authors applied the *GenAI-Ped* framework, which combines scaffolding, self-regulated learning, *Bloom's taxonomy*, and prompt engineering training to guide how students use tools like *ChatGPT*. Pretest-posttest results showed improvements in problem-solving, critical thinking, and core programming logic.

A similar paper by Abouelenein et al. [6] also addresses the problem that students are using *ChatGPT* more and more in an unstructured way. This leads to copy-paste coding, passive learning, and weak conceptual understanding. The authors

argue that *ChatGPT* itself is not the problem. The problem is how it is used in education. Thus, their core question is: *Can a structured pedagogical model turn ChatGPT into a learning tool that actually improves programming skills?*
They introduce a model called *R5E*. Students are forced by this model to think before asking *ChatGPT* for answers. It consists of these phases:

1. **Recognition:** The students identify and familiarize themselves with the problem and the relevant concepts.
2. **Exploration:** They break down the problem and design a strategy to solve it. *ChatGPT* is only used to collect conceptual information. At the end of this phase, the constructed strategy is transformed into flowcharts.
3. **Engagement:** Students share and discuss their proposed solutions with other students and the instructor.
4. **Execution:** The agreed solution is implemented by writing the program code (in *C++*) themselves. *ChatGPT* may help with syntax or for explanations.
5. **Evaluation:** Students test the program and identify errors. *ChatGPT* is used to explain errors and debugging strategies.
6. **Editing:** The identified errors are corrected and the program is refined.

An experiment with 70 undergraduate IT students with basic prior programming knowledge was conducted. They were divided into two groups. The first group used *ChatGPT* with the *R5E* structure and the second group used it freely and in an unstructured way. Both groups learned *C++*, had the same lectures, and solved the same programming tasks during the lab sessions. The study took 12 weeks. Two dimensions of programming skills were measured: *cognitive understanding* and *practical performance*. The *cognitive understanding* part consisted of a multiple-choice test which checked the theoretical understanding of *C++* concepts like variables, control flow, algorithms, and syntax. The *practical performance* measured students' actual ability to program by observing them while they solve coding tasks.
The *R5E* group improved much more on both conceptual knowledge and hands-on performance. The findings show that *ChatGPT* can help in learning programming if it is used properly.

AI as an integrated tutor or teaching assistant: A second approach integrates AI directly into the learning environment. The integrated AI can then access task description and student's code [14]. And since the AI is part of the course system, its feedback can be aligned with lectures and course materials [58, 114]. These systems are designed to provide bounded support, such as hints, explanations, or small fixes [35, 58]. They avoid full solutions. The AI is meant to behave more like a tutor or coach than a solution generator.

The paper by Liu et al. [58] describes how the instructors of *CS50* at Harvard University integrated generative AI into an introductory computer science course rather than banning it. Instead of seeing AI as a threat, they say that AI can be

reshaped into a teaching assistant which supports learning while still protecting educational values. Their goal is to approximately have a 1:1 teacher-student ratio using AI. The paper introduces a platform which is called *CS50.ai*. One important component of this platform gives students the ability to highlight code in their editor and get natural language explanations of what that code does. Another component shows suggestions on how to format the written code and explains why those changes improve the readability.
The most remarkable tool is the *CS50 Duck* (see Fig. 6). This is an assistant inspired by *rubber-duck debugging*. The tool is prevented from producing full solutions or writing assignment code for students. Instead, it only gives hints and clarifies errors.

Fig. 6. CS50 Duck [58]

Another notable example is the paper by Jacobs and Jaschke [91]. Their goal is that large language models are used to generate feedback that is pedagogically meaningful for programming exercises when they are explicitly integrated. They argue that while generative AI tools like GPT-4 are strong in programming-related tasks, their application in educational settings is problematic because they tend to provide complete solutions, rely on generic explanations, and provide information that is not consistent with the course material.

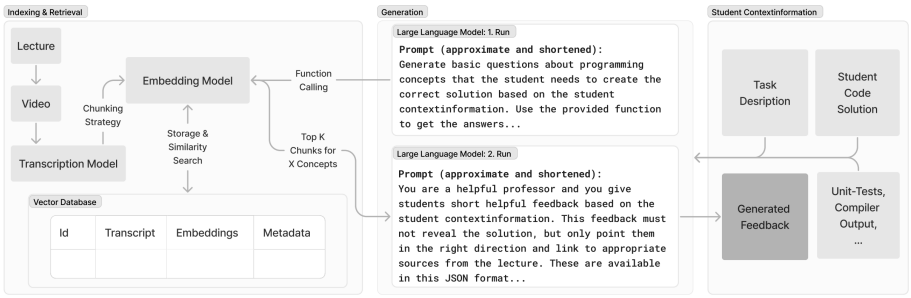

Fig. 7. Tutor Kai - System design [91]

Therefore, the authors introduce *Tutor Kai* (see Fig. 8). This is a platform which allows students to submit their code solutions and get feedback based on *Retrieval-Augmented Generation (RAG)*. Here, GPT-4 is integrated with knowledge that is externally retrieved. The lecture content consists of recorded programming lectures. They were transcribed using *Whisper* [79]. These transcripts include timestamps, which allows specific textual explanations to be linked back to precise moments in the lecture videos. The transcribed text is split into chunks of 512 characters, embedded, and stored in a vector database together with metadata such as the corresponding video file and timestamp.

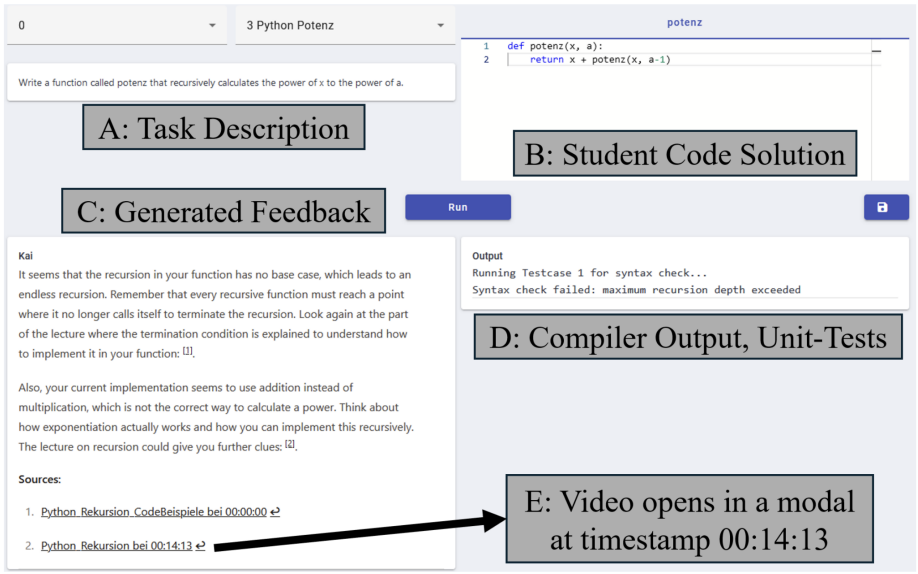

Fig. 8. Tutor Kai - User Interface [91]

The feedback generation process is implemented as a two-step prompt chain (see Fig. 7). In the first step, GPT-4 analyzes the student's submission and also the task description, the code, compiler output, and unit test results. Based on this, the model formulates one or two concept-focused questions. These are then used as queries to retrieve relevant lecture transcript segments from the vector database. In the second step, GPT-4 generates the final feedback. This is done by combining the retrieved lecture passages with the student's context information.

Critical AI use and AI literacy: A very important pedagogical approach is to teach AI literacy explicitly [33]. This also includes teaching how to write effective prompts. Students are encouraged to write prompts so that their thinking becomes visible [68, 85]. They should refine their prompts based on feedback, clarify misunderstandings, and move step by step from broad questions to targeted ones.
AI can be wrong and students must verify the results through testing, careful reading, and reasoning. Ethical and responsible use is also discussed. This includes that students should be transparent about AI-assisted work, avoid overreliance, and understand limitations such as hallucinations. They should also reflect on when AI is appropriate [13, 33].

The paper by Gorson Benario et al. [33] examines what effects it has if instructions on how to use generative AI tools into a software development course are integrated. Throughout the semester, students were taught the fundamentals of AI, its role, and its practical applications in software development. The instruction also covered how LLMs work, techniques for prompt engineering, and common limitations. The students were encouraged to use generative AI only as a supportive learning aid.

It was reported that the students were already using generative AI but mostly without guidance. After the students got the instructions, they mostly used generative AI for learning-related activities like explaining concepts or understanding the code. They compared pre- and post-scores on how often students conduct specific actions when they get a coding suggestion from generative AI. Significant increases include: read the code more often, test the response more often, and understand the code provided by the tool more often. Students felt that AI helped them learn programming and problem solving. They also generally believed that they understood their own code when using generative AI tools.

Redesign: Emphasis is placed on redesigning assessments and course structures to ensure that learning outcomes still remain meaningful [2, 42, 89]. Common strategies include making assignments more resistant to simple AI solution [2]. This can include adding constraints, multi-part tasks, partial templates, hidden tests, or additional artifacts such as diagrams [12, 81]. Exam formats should also

be adjusted, with more in-person, oral, or live assessments [13, 87]. There must be a stronger focus on higher-level skills such as requirements, modeling, architecture, or design [52].

Berrezueta-Guzman and Krusche [39] want to find out the impact of large language models, particularly *ChatGPT* (GPT-3.5 and GPT-4), on learning outcomes in introductory programming education. They observe the fact that LLMs are easy to access and that they have the ability to generate correct code solutions may undermine the pedagogical value of programming exercises. Because programming skills are developed through practice and active problem-solving, relying too much on AI-generated solutions enable students to pass courses without even developing them.
The authors conduct an experimental analysis using real programming exercises. A total of 22 homework exercises were selected. Each of them was solved using *ChatGPT* and the required time, the number of passed tests, and the achieved grade was documented. These results were then compared with the average grades and effort invested by students.
A key finding of the study is that *ChatGPT* achieved an average grade of 55%, which exceeds the minimum passing threshold of the course (50%). So, a student could pass the course by using *ChatGPT* with almost no understanding. *ChatGPT* was effective at solving exercises that were formally specified, self-contained, and did not require interpretation.
Based on these findings, the paper proposes a set of design recommendations so that the effectiveness of *ChatGPT* can be reduced:

- Reduce the problem statement text
- Use TODOs spread across multiple files
- Provide partial implementations
- Use images for specifications
- Hidden test cases
- Dynamic test cases
- Evaluate runtime/memory
- Assess code quality
- Require documentation
- Customize task/requirements
- Plagiarism checks

4.5 Reported Benefits - RQ3

Increased time savings and productivity: AI tools are perceived to speed up work and reduce the time that is needed to complete specific tasks [16, 97]. AI is seen as fast and convenient in comparison to the traditional methods or workflows such as manual web searching or reading documentation [82, 98].

Improved understanding and learning support: Another major benefit is the learning support. They help students understand difficult material and stay

engaged. AI tools are able to explain unfamiliar concepts often in a clearer language than in textbooks or official documentation [37, 69, 108].

High perceived usefulness, ease of use, and convenience: Students see AI tools as useful, easy to use, and convenient to include them into their workflow [86, 90]. This perception is often connected with needing less effort when working on assignments or tasks. Thus, AI is described as smoothing the learning process.

Motivation, confidence, and emotional support: Students report that have more motivation, more interest in learning, and higher confidence when AI tools are available. Moreover, they mention that their self-efficacy also improves. They have the feeling that they are able to solve (programming) tasks, along with reduced anxiety [8, 48, 106]. These benefits are meaningful for weaker or less confident students [108]. Students describe experiencing less "imposter syndrome". An important reason for this is that AI is non-judgmental. Students may feel more comfortable asking "basic" questions or showing their misunderstandings because they do not fear to get negative reactions [15].

Personalized tutoring and 24/7 availability: Students often perceive AI tools as a form of personalized tutor [55]. AI can adapt explanations and suggestions to the student's current level, pace, and context. This perceived personalization is also closely tied to the 24/7 access. So, personalization and continuous availability make more individualized learning support possible.

Reduced effort and workload: There are reduced effort on both sides. For students, AI reduces the workload by providing faster guidance. For educators, AI is reported as helpful for creating exercises or assisting with grading-related activities, typically with human oversight [25, 51, 67]. Overall, AI gives the opportunity to reduce routine workload.

4.6 Reported Risks and Challenges - RQ3

Output quality and limitations: A dominant concern is that AI-generated outputs can be incorrect. They can include hallucinations, wrong explanations, or faulty code [88, 99]. The outputs are sometimes too complex, too simplistic or too long [37, 62, 71]. Closely related are reports about the limitations of AI tools. They struggle when the context is larger or when the requirements are complex. All of this shows that verification of AI outputs is essential.

Overreliance and shallow learning Another major theme is overreliance. Students may rely too much on AI to solve problems for them [23, 46, 110]. This can reduce critical thinking, problem-solving skills, and deep understanding of concepts. Students get to a working solution faster but may learn less. So, there is a risk of a shift from *learning to solve problems* towards *learning to ask the tool*. The result of that can be shallow learning. Students may submit the right

solutions but struggle to explain them.

Prompting difficulty: Using AI requires writing good prompts, but many students have difficulties with that. Prompts are often too vague or missing important details, which can lead to misleading results. Authors emphasize that AI tools should be integrated into software engineering education, but with explicit teaching on prompt engineering [9, 52].

Academic integrity and ethical concerns: AI makes it easy for students to generate code and text that looks original, which makes plagiarism harder to detect. The output may not show the student's own work or understanding. Studies also mention ethical and practical issues that must be managed. They stress the need for clear rules about when and how AI is allowed. Without clear policies, students may be unsure what is acceptable, and instructors may struggle to make sure that fairness is achieved [5, 44, 94].

Assessment validity and reliability: AI tools challenge the validity of traditional assessments. They may weak indicators of student learning. So, there is a difficulty determining whether a submission reflects a student's understanding or simply the tool's output. This undermines assessment reliability [42, 97]. As a consequence, studies suggest that courses must rethink what they measure and how they measure it.

Reduced social interaction: The reliance on AI tools can reduce student-student and student-educator interaction [54]. Students may ask fewer questions in labs, collaborate less, and engage less in peer learning. This matters because skills such as communication, coordination, and shared problem-solving are important learning outcomes.

4.7 Impact on Learning Outcomes - RQ3

As shown in Fig. 9, all reported positive learning outcomes (25), associated with the use of AI for students, occurred along with a clearly defined pedagogical approach, as described in Section 4.4. In this context, *positive learning outcomes* are about improvements such as higher examination or assignment scores, as well as student's self-reported perceptions of better understanding of the covered concepts.

Despite the overall trend of positive learning outcomes shown in Fig. 9, two studies that integrated explicit pedagogical approaches also reported negative effects alongside the positive results (see Fig. 10). One such study [43] introduced an *Intelligent Programming Scaffolding System using ChatGPT* in a *Data Structured and Algorithms* course. It was designed in order to support students' computational thinking. The system consisted of three main components: *Solution Assessment*, in which students submitted proposed solution approaches that were critiqued by *ChatGPT*, *Code Assessment*, where students submitted source code so that they can get a AI-based review, debugging, and suggestions

for improvement, and *Free Interaction*, which allowed the students to interact with the AI in an unrestricted way for issues not covered in the previous parts of the system.

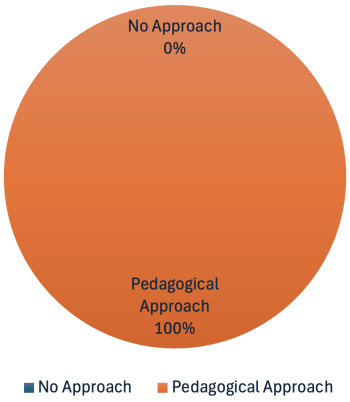

Fig. 9. Distribution of positive learning outcomes with and without pedagogical approaches

Using a combination of computational thinking questionnaire, system usage logs, and students' interviews, the authors found that students using the system show improvements in overall computational thinking, particularly in creativity, algorithmic thinking, and critical thinking.

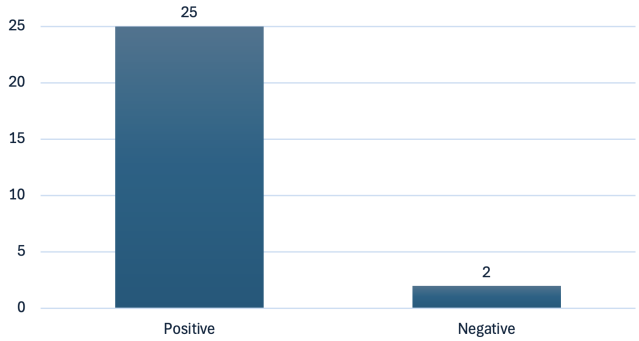

Fig. 10. Positive and Negative Learning Outcomes in Pedagogical Approaches

However, student's self-reported problem-solving ability decreased. The authors interpret this decline to an increased dependence on *ChatGPT*, as many students

tended to rely heavily on the *Free Interaction* component.

A similar pattern of mixed outcomes can be found in the study described in Section 4.4. The study proposes the *GenAI-Ped* framework. Over a seven-week period, students learned C++ under clearly defined rules for how they are permitted to use AI. These include structured planning, careful prompt construction, and the use of generative AI primarily as an explainer and debugging assistant. The pretest-posttest results show improvements in problem-solving skills, critical thinking, and core programming logic. But enhancements in more advanced skills, such as code optimization and the evaluation of alternative solutions, were weak or not significant.
These findings show that while pedagogically guided AI use can improve foundational understanding and reasoning skills, certain design choices may limit or even undermine some abilities. Thus, they may require further refinement and fine-tuning.

5 Discussion

Across the studies, a clear picture shows up: software engineering education is adapting more and more to the availability of AI. There is a shift from the question of whether AI will be used at all to how it can be integrated without undermining the learning goals.
RQ_1 asked: *In what areas of software engineering education is AI currently being used, and which AI tools or techniques are reported in the literature?* The findings show that AI is already widely used across multiple areas of software engineering education. There is a strong concentration on introductory programming and programming fundamentals. Often, general-purpose large language models were used, particularly *ChatGPT* and related *GPT*-models, as well as coding assistants such as *GitHub Copilot*. There is a small but emerging use in project-based courses, design/modeling, and databases. The most common functions include concept/code explanation, debugging support, or code generation/completion.
RQ_2 asked: *How are software engineering educators adjusting their teaching methods in response to AI integration?* A clear pedagogical shift can be seen: the discussion is moving away from whether AI will be used at all toward how it can be integrated without undermining the learning goals. Also, because a lot of students use AI tools and will use it anyway even if they are prohibited to use it, a strict "ban" approach is replaced by guided and structured integration. The approaches can be grouped into:

1. **Structured/scaffolded AI use**, including workflows with phases and constraints in order to prevent "one-shot" solution generation
2. **AI as an integrated tutor/assistant** embedded into course environments with bounded behavior
3. Explicit **AI literacy and critical use**. This includes verification of responses, prompt quality, and awareness of limitations.

4. **Redesign of assessments and assignments** in order to make the thinking process of students visible and to keep assessment valid when AI is available.

These responses emphasize process, reasoning, and reflection instead of only final artifacts.
RQ_3 asked: *What impact does AI integration have on software engineering education?* The reported impact is not positive or negative in a uniform way. It depends on how AI is used and how learning activities are designed. Benefits include faster access to help, improved understanding through explanations, reduced frustration, and perceived usefulness and convenience. At the same time, major risks and challenges repeatedly mentioned are output quality issues, including hallucinations or faulty code, students' difficulty in prompting effectively, overreliance and shallow learning, reduced interaction with other students and instructors, and challenges to academic integrity and assessment validity.
An important pattern across the studies is that positive learning impacts are associated with bounded or structured use. On the other hand, unstructured use is more frequently connected to weaker understanding of concepts and reduced deep learning. This shows that "integration with constraints" is essential. AI can support learning, but only when full outsourcing is prevented and students are required to demonstrate their understanding.

6 Threats to Validity

One existing threat is that the search strategy may not have retrieved all relevant studies. Major digital libraries were used, and the search strings included a wide set of synonyms for both AI and education contexts. Despite that, relevant papers may still be missed. To reduce this, the search strings were validated against a known set of papers collected from an exploratory search ($\geq$ 80% recall). Also, results may be time-sensitive. AI tools such as LLMs evolve very quickly. This can affect both educational use cases and observed impacts. That is why conclusions drawn from studies in 2021-2025 might not fully generalize to future generations of these tools. Furthermore, the available literature may be influenced by publication bias: positive or "successful" integrations. That is why, even though only limited negative results were found from the application of pedagogical approaches, these negative aspects were extracted as shown in Section 4.7. Also, overall negative aspects of AI usage were discussed in Section 4.6.

7 Conclusion

Overall, the reviewed literature shows that AI is becoming a normal component and a critical part of software engineering education. It is currently dominated by generative AI and LLM-based tools, especially *ChatGPT* and GPT-based systems, with additional attention to coding assistants. These tools are mainly used for code explanation, debugging support, and code generation, most often

in introductory programming contexts.
Reported benefits include getting on-demand support and increase in efficiency, while major risks include hallucinations, overreliance, and threats to assessment validity and academic integrity. Therefore, many studies describe a shift to guided integration. An in all, the evidence highlights that outcomes depend strongly on instructional design: structured and bounded use is more likely to support learning than unrestricted use. Based on this, important outcomes show up: guided integration does promise more than banning, assessment redesign is not avoidable, AI literacy should be taught explicitly, and human oversight is still essential, especially if AI is used for grading support or tutoring.

References

1. A. Brockenbrough, D. Salinas: Using Generative AI to Create User Stories in the Software Engineering Classroom. In: 2024 36th International Conference on Software Engineering Education and Training (CSEE&T). pp. 1–5 (Aug 2024). https://doi.org/10.1109/CSEET62301.2024.10662994, journal Abbreviation: 2024 36th International Conference on Software Engineering Education and Training (CSEE&T)
2. A. L. Felipe, U. S. Khwakhali, T. N. Nguyen: A Framework for Assessment Design in the Era of Generative AI: Case Study of Take-Home Assignment in Software-related Courses. In: 2025 10th International STEM Education Conference (iSTEM-Ed). pp. 1–6 (Aug 2025). https://doi.org/10.1109/iSTEM-Ed65612.2025.11129352, journal Abbreviation: 2025 10th International STEM Education Conference (iSTEM-Ed)
3. A. M. Abdelfattah, N. A. Ali, M. A. Elaziz, H. H. Ammar: Roadmap for Software Engineering Education using ChatGPT. In: 2023 International Conference on Artificial Intelligence Science and Applications in Industry and Society (CAISAIS). pp. 1–6 (Sep 2023). https://doi.org/10.1109/CAISAIS59399.2023.10270477, journal Abbreviation: 2023 International Conference on Artificial Intelligence Science and Applications in Industry and Society (CAISAIS)
4. A. Naman, R. Shariffdeen, G. Wang, S. Rasnayaka, G. N. Iyer: Analysis of Student-LLM Interaction in a Software Engineering Project. In: 2025 IEEE/ACM International Workshop on Large Language Models for Code (LLM4Code). pp. 112–119 (May 2025). https://doi.org/10.1109/LLM4Code66737.2025.00019, journal Abbreviation: 2025 IEEE/ACM International Workshop on Large Language Models for Code (LLM4Code)
5. Abdulla, S., Ismail, S., Fawzy, Y., Elhaj, A.: Using ChatGPT in Teaching Computer Programming and Studying its Impact on Students Performance. ELECTRONIC JOURNAL OF E-LEARNING **22**(6), 66–81 (Dec 2024). https://doi.org/10.34190/ejel.22.6.3380
6. Abouelenein, Y.A.M., Ghazala, A.F.A., Mahdy, E.M.M., Khalaf, M.H.R.: The R5E pattern: can artificial intelligence enhance programming skills development? EDUCATION AND INFORMATION TECHNOLOGIES **30**(15), 22177–22205 (Oct 2025). https://doi.org/10.1007/s10639-025-13616-3
7. Alanazi, M., Li, A., Samra, H., Soh, B.: Examining the Influence of AI on Python Programming Education: An Empirical Study and Analysis of Student Acceptance Through TAM3. COMPUTERS **14**(10) (Sep 2025). https://doi.org/10.3390/computers14100411

8. Ali, F., Ahmed, A., Alipour, M.A., Terashima-Marin, H.: Adoption of AI-coding assistants in programming education: exploring trust and learning motivation through an extended technology acceptance model. JOURNAL OF COMPUTERS IN EDUCATION (Oct 2025). https://doi.org/10.1007/s40692-025-00375-w
9. Alpizar-Chacon, I., Keuning, H.: Student's Use of Generative AI as a Support Tool in an Advanced Web Development Course. In: Proceedings of the 30th ACM Conference on Innovation and Technology in Computer Science Education V. 1. pp. 312–318. ITiCSE 2025, Association for Computing Machinery, New York, NY, USA (2025). https://doi.org/10.1145/3724363.3729106, https://doi.org/10.1145/3724363.3729106, event-place: Nijmegen, Netherlands
10. Association for Computing Machinery: Acm digital library. https://dl.acm.org (2024), digital library for ACM journals, conferences, and proceedings
11. Azoulay, R., Hirst, T., Reches, S.: Large language models in computer science classrooms: Ethical challenges and strategic solutions. Applied Sciences (2076-3417) **15**(4) (2025)
12. B. P. Cipriano, P. Alves: LLMs Still Can't Avoid Instanceof: An Investigation Into GPT-3.5, GPT-4 and Bard's Capacity to Handle Object-Oriented Programming Assignments. In: 2024 IEEE/ACM 46th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET). pp. 162–169 (Apr 2024). https://doi.org/10.1145/3639474.3640052, journal Abbreviation: 2024 IEEE/ACM 46th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)
13. B. Zönnchen, V. Thurner, A. Böttcher: On the Impact of ChatGPT on Teaching and Studying Software Engineering. In: 2024 IEEE Global Engineering Education Conference (EDUCON). pp. 1–10 (May 2024). https://doi.org/10.1109/EDUCON60312.2024.10578680, journal Abbreviation: 2024 IEEE Global Engineering Education Conference (EDUCON)
14. Bassner, P., Frankford, E., Krusche, S.: Iris: An AI-Driven Virtual Tutor for Computer Science Education. In: Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1. pp. 394–400. ITiCSE 2024, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3649217.3653543, https://doi.org/10.1145/3649217.3653543, event-place: Milan, Italy
15. Boguslawski, S., Deer, R., Dawson, M.G.: Programming education and learner motivation in the age of generative AI: student and educator perspectives. INFORMATION AND LEARNING SCIENCES (Jul 2024). https://doi.org/10.1108/ILS-10-2023-0163
16. Borghoff, U.M., Minas, M., Schopp, J.: Generative AI in Student Software Development Projects: A User Study on Experiences and Self-Assessment. In: Proceedings of the 6th European Conference on Software Engineering Education. pp. 161–170. ECSEE '25, Association for Computing Machinery, New York, NY, USA (2025). https://doi.org/10.1145/3723010.3723012, https://doi.org/10.1145/3723010.3723012
17. Brito, F., Mekdad, Y., Ross, M., Finlayson, M.A., Uluagac, S.: Enhancing Cybersecurity Education with Artificial Intelligence Content. In: Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1. pp. 158–164. SIGCSETS 2025, Association for Computing Machinery, New York, NY, USA (2025). https://doi.org/10.1145/3641554.3701958, https://doi.org/10.1145/3641554.3701958, event-place: Pittsburgh, PA, USA

18. C. Cubillos, R. Mellado, D. Cabrera-Paniagua, E. Urra: Generative Artificial Intelligence in Computer Programming: Does It Enhance Learning, Motivation, and the Learning Environment? IEEE Access **13**, 40438–40455 (2025). https://doi.org/10.1109/ACCESS.2025.3532883
19. Chen, P.H., Huang, Y.M., Wu, T.T., Lee, H.Y.: Coding Error-Based Reflective Learning with ChatGPT: A Study on Higher-order Thinking Skills in Programming Education. In: Proceedings of the 6th International Conference on Modern Educational Technology. pp. 48–54. ICMET '24, Association for Computing Machinery, New York, NY, USA (2025). https://doi.org/10.1145/3729434.3729435, https://doi.org/10.1145/3729434.3729435
20. Cisneros-Gonzalez, J., Gordo-Herrera, N., Barcia-Santos, I., Sanchez-Soriano, J.: JorGPT: Instructor-Aided Grading of Programming Assignments with Large Language Models (LLMs). FUTURE INTERNET **17**(6) (Jun 2025). https://doi.org/10.3390/fi17060265
21. Clarivate: Web of science. https://www.webofscience.com (2024), citation indexing and abstracting database
22. Dahal, R., Murray, G., Chataut, R., Hefeida, M., Srivastava, A.K., Gyawali, P.K.: AutoTA: A Dynamic Intent-Based Virtual Teaching Assistant for Students Using Open Source LLMs. IEEE ACCESS **13**, 118122–118134 (2025). https://doi.org/10.1109/ACCESS.2025.3576329
23. Dawson, M.G., Deer, R., Boguslawski, S.: Cognitive dissonance in programming education: A qualitative exploration of the impact of generative AI on application-directed learning. COMPUTERS IN HUMAN BEHAVIOR REPORTS **19** (Aug 2025). https://doi.org/10.1016/j.chbr.2025.100724
24. Du, H., Jia, Q., Gehringer, E., Wang, X.: Harnessing large language models to auto-evaluate the student project reports. Computers and Education: Artificial Intelligence **7**, 100268 (Dec 2024). https://doi.org/10.1016/j.caeai.2024.100268, https://www.sciencedirect.com/science/article/pii/S2666920X24000717
25. E. Smolić, M. Pavelić, B. Boras, I. Mekterović, T. Jagušt: LLM Generative AI and Students' Exam Code Evaluation: Qualitative and Quantitative Analysis. In: 2024 47th MIPRO ICT and Electronics Convention (MIPRO). pp. 1261–1266 (May 2024). https://doi.org/10.1109/MIPRO60963.2024.10569820, journal Abbreviation: 2024 47th MIPRO ICT and Electronics Convention (MIPRO)
26. E. Tuparova, D. Tuparova, G. Tuparov: Exploring the Role of Large Language Models in Advancing Student Assessment in Database Courses. In: 2025 MIPRO 48th ICT and Electronics Convention. pp. 1448–1453 (Jun 2025). https://doi.org/10.1109/MIPRO65660.2025.11131984, journal Abbreviation: 2025 MIPRO 48th ICT and Electronics Convention
27. Elsevier: Sciencedirect. https://www.sciencedirect.com (2024), scientific literature platform for journals and books
28. Fan, G., Liu, D., Zhang, R., Pan, L.: The impact of AI-assisted pair programming on student motivation, programming anxiety, collaborative learning, and programming performance: a comparative study with traditional pair programming and individual approaches. INTERNATIONAL JOURNAL OF STEM EDUCATION **12**(1) (Mar 2025). https://doi.org/10.1186/s40594-025-00537-3
29. G. Cheng, W. Wong, L. Luo, M. Yu: Integrating a Scaffolding-Based, LLM-Driven Chatbot into Programming Education: A University Case Study. In: 2025 International Symposium on Educational Technology (ISET). pp. 1–5 (Jul 2025). https://doi.org/10.1109/ISET65607.2025.00047, journal Abbreviation: 2025 International Symposium on Educational Technology (ISET)

30. Gaitantzi, A., Kazanidis, I.: The role of artificial intelligence in computer science education: A systematic review with a focus on database instruction. Applied Sciences **15**(7), 3960 (2025)
31. GitHub: Github copilot. https://github.com/features/copilot (2021), aI-powered code completion tool
32. Google: Gemini. https://deepmind.google/technologies/gemini/ (2023), large multimodal language model
33. Gorson Benario, J., Marroquin, J., Chan, M.M., Holmes, E.D., Mejia, D.: Unlocking Potential with Generative AI Instruction: Investigating Mid-level Software Development Student Perceptions, Behavior, and Adoption. In: Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1. pp. 395–401. SIGCSETS 2025, Association for Computing Machinery, New York, NY, USA (2025). https://doi.org/10.1145/3641554.3701859, https://doi.org/10.1145/3641554.3701859, event-place: Pittsburgh, PA, USA
34. Grandel, S., Schmidt, D., Leach, K.: Applying Large Language Models to Enhance the Assessment of Java Programming Assignments. In: Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering. pp. 789–799. FSE Companion '25, Association for Computing Machinery, New York, NY, USA (2025). https://doi.org/10.1145/3696630.3727236, https://doi.org/10.1145/3696630.3727236, event-place: Clarion Hotel Trondheim, Trondheim, Norway
35. H. Nutalapati, S. Velmurugan, N. M. Tiglao: Coding Buddy: An Adaptive AI-Powered Platform for Personalized Learning. In: 2024 International Symposium on Networks, Computers and Communications (ISNCC). pp. 1–6 (Oct 2024). https://doi.org/10.1109/ISNCC62547.2024.10759044, journal Abbreviation: 2024 International Symposium on Networks, Computers and Communications (ISNCC)
36. Haindl, P., Weinberger, G.: Does ChatGPT Help Novice Programmers Write Better Code? Results From Static Code Analysis. IEEE ACCESS **12**, 114146–114156 (2024). https://doi.org/10.1109/ACCESS.2024.3445432
37. Haindl, P., Weinberger, G.: Students' Experiences of Using ChatGPT in an Undergraduate Programming Course. IEEE ACCESS **12**, 43519–43529 (2024). https://doi.org/10.1109/ACCESS.2024.3380909
38. IEEE: Ieee xplore digital library. https://ieeexplore.ieee.org (2024), digital library for IEEE journals, conferences, and standards
39. J. Berrezueta-Guzman, S. Krusche: Recommendations to Create Programming Exercises to Overcome ChatGPT. In: 2023 IEEE 35th International Conference on Software Engineering Education and Training (CSEE&T). pp. 147–151 (Aug 2023). https://doi.org/10.1109/CSEET58097.2023.00031, journal Abbreviation: 2023 IEEE 35th International Conference on Software Engineering Education and Training (CSEE&T)
40. J. Cámara, J. Troya, J. Montes-Torres, F. J. Jaime: Generative AI in the Software Modeling Classroom: An Experience Report With ChatGPT and Unified Modeling Language. IEEE Software **41**(6), 73–81 (Dec 2024). https://doi.org/10.1109/MS.2024.3385309
41. J. I. C. Maurat, M. P. Jamison: Correlation Between AI Tool Usage Frequency and Perceived Learning Effectiveness in Programming Education. In: 2025 IEEE 15th Symposium on Computer Applications & Industrial Electronics (ISCAIE). pp. 260–264 (May 2025). https://doi.org/10.1109/ISCAIE64985.2025.11080867, journal Abbreviation: 2025 IEEE 15th Symposium on Computer Applications & Industrial Electronics (ISCAIE)

42. J. L. Weber, B. M. Neda, K. C. Juarez, J. Wong–Ma, S. Gago–Masague, H. Ziv: Beyond the Hype: Perceptions and Realities of Using Large Language Models in Computer Science Education at an R1 University. In: 2024 IEEE Global Engineering Education Conference (EDUCON). pp. 01–08 (May 2024). https://doi.org/10.1109/EDUCON60312.2024.10578596, journal Abbreviation: 2024 IEEE Global Engineering Education Conference (EDUCON)
43. J. Liao, L. Zhong, L. Zhe, H. Xu, M. Liu, T. Xie: Scaffolding Computational Thinking With ChatGPT. IEEE Transactions on Learning Technologies **17**, 1628–1642 (2024). https://doi.org/10.1109/TLT.2024.3392896
44. J. Pereira, J. -M. López, X. Garmendia, M. Azanza: Leveraging Open Source LLMs for Software Engineering Education and Training. In: 2024 36th International Conference on Software Engineering Education and Training (CSEE&T). pp. 1–10 (Aug 2024). https://doi.org/10.1109/CSEET62301.2024.10663055, journal Abbreviation: 2024 36th International Conference on Software Engineering Education and Training (CSEE&T)
45. Jin, H., Lee, S., Shin, H., Kim, J.: Teach AI How to Code: Using Large Language Models as Teachable Agents for Programming Education. In: Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems. CHI '24, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3613904.3642349, https://doi.org/10.1145/3613904.3642349, event-place: Honolulu, HI, USA
46. Jost, G., Taneski, V., Karakatic, S.: The Impact of Large Language Models on Programming Education and Student Learning Outcomes. APPLIED SCIENCES-BASEL **14**(10) (May 2024). https://doi.org/10.3390/app14104115
47. K. Zheng, Y. Shen, Y. Tao: Automatic Unit Test Generation for Programming Assignments Using Large Language Models. In: 2025 IEEE/ACM 37th International Conference on Software Engineering Education and Training (CSEE&T). pp. 242–252 (May 2025). https://doi.org/10.1109/CSEET66350.2025.00031, journal Abbreviation: 2025 IEEE/ACM 37th International Conference on Software Engineering Education and Training (CSEE&T)
48. Katona, J., Gyonyoru, K.I.K.: Integrating AI-based adaptive learning into the flipped classroom model to enhance engagement and learning outcomes. Computers and Education: Artificial Intelligence **8**, 100392 (Jun 2025). https://doi.org/10.1016/j.caeai.2025.100392, https://www.sciencedirect.com/science/article/pii/S2666920X25000323
49. Keele, S., et al.: Guidelines for performing systematic literature reviews in software engineering. Tech. rep., Technical report, ver. 2.3 ebse technical report. ebse (2007)
50. Kozov, V., Ivanova, G., Atanasova, D.: Practical Application of AI and Large Language Models in Software Engineering Education. INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS **15**(1), 690–696 (Jan 2024)
51. Kuramitsu, K., Obara, Y., Sato, M., Obara, M.: KOGI: A Seamless Integration of ChatGPT into Jupyter Environments for Programming Education. In: Proceedings of the 2023 ACM SIGPLAN International Symposium on SPLASH-E. pp. 50–59. SPLASH-E 2023, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3622780.3623648, https://doi.org/10.1145/3622780.3623648, event-place: Cascais, Portugal
52. L. Baresi, A. De Lucia, A. Di Marco, M. Di Penta, D. Di Ruscio, L. Mariani, D. Micucci, F. Palomba, M. T. Rossi, F. Zampetti: Students' Per-

ception of ChatGPT in Software Engineering: Lessons Learned from Five Courses. In: 2025 IEEE/ACM 37th International Conference on Software Engineering Education and Training (CSEE&T). pp. 158–169 (May 2025). https://doi.org/10.1109/CSEET66350.2025.00023, journal Abbreviation: 2025 IEEE/ACM 37th International Conference on Software Engineering Education and Training (CSEE&T)

53. L. Frank, F. Herth, P. Stuwe, M. Klaiber, F. Gerschner, A. Theissler: Leveraging GenAI for an Intelligent Tutoring System for R: A Quantitative Evaluation of Large Language Models. In: 2024 IEEE Global Engineering Education Conference (EDUCON). pp. 1–9 (May 2024). https://doi.org/10.1109/EDUCON60312.2024.10578933, journal Abbreviation: 2024 IEEE Global Engineering Education Conference (EDUCON)
54. Lee, H.Y., Chen, P.H., Lin, C.J., Huang, Y.M., Wu, T.T.: Leveraging ChatGPT for personalized reflective learning in programming education: effects on self-efficacy, higher-order thinking, and project implementation skills. EDUCATION AND INFORMATION TECHNOLOGIES (Sep 2025). https://doi.org/10.1007/s10639-025-13733-z
55. Lepp, M., Kaimre, J.: Does generative AI help in learning programming: Students' perceptions, reported use and relation to performance. COMPUTERS IN HUMAN BEHAVIOR REPORTS **18** (May 2025). https://doi.org/10.1016/j.chbr.2025.100642
56. Li, M., Wang, D., Purwanto, E., Selig, T., Zhang, Q., Liang, H.N.: VisualCodeMOOC: A course platform for algorithms and data structures integrating a conversational agent for enhanced learning through dynamic visualizations. SoftwareX **30**, 102072 (May 2025). https://doi.org/10.1016/j.softx.2025.102072, https://www.sciencedirect.com/science/article/pii/S2352711025000391
57. Li, N., Broner, S., Kim, Y., Mizuo, K., Sauder, E., To, C., Wang, A., Gila, O., Shindler, M.: Investigating the Capabilities of Generative AI in Solving Data Structures, Algorithms, and Computability Problems. In: Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1. pp. 659–665. SIGCSETS 2025, Association for Computing Machinery, New York, NY, USA (2025). https://doi.org/10.1145/3641554.3701946, https://doi.org/10.1145/3641554.3701946, event-place: Pittsburgh, PA, USA
58. Liu, R., Zenke, C., Liu, C., Holmes, A., Thornton, P., Malan, D.J.: Teaching CS50 with AI: Leveraging Generative Artificial Intelligence in Computer Science Education. In: Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1. pp. 750–756. SIGCSE 2024, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3626252.3630938, https://doi.org/10.1145/3626252.3630938, event-place: Portland, OR, USA
59. Lyu, W., Wang, Y., Sun, Y., Zhang, Y.: Will Your Next Pair Programming Partner Be Human? An Empirical Evaluation of Generative AI as a Collaborative Teammate in a Semester-Long Classroom Setting. In: Proceedings of the Twelfth ACM Conference on Learning @ Scale. pp. 83–94. L@S '25, Association for Computing Machinery, New York, NY, USA (2025). https://doi.org/10.1145/3698205.3729544, https://doi.org/10.1145/3698205.3729544, event-place: Palermo, Italy
60. M. A, N. M, M. S. M, R. S, N. M: Smart Problem-Solving Using AI Driven Resource Kit. In: 2024 International Conference on Emerging Research in Computational Science (ICERCS). pp. 1–7 (Dec 2024). https://doi.org/10.1109/ICERCS63125.2024.10895204, journal Abbrevia-

tion: 2024 International Conference on Emerging Research in Computational Science (ICERCS)
61. M. Akyash, K. Z. Azar, H. Mardani Kamali: StepGrade: Grading Programming Assignments with Context-Aware LLMs. In: 2025 IEEE Integrated STEM Education Conference (ISEC). pp. 1–6 (Mar 2025). https://doi.org/10.1109/ISEC64801.2025.11147374, journal Abbreviation: 2025 IEEE Integrated STEM Education Conference (ISEC)
62. M. Horváth, L. Bubeňková, E. Pietriková: Exploring GPT-Generated Variations in C Programming Assignments. In: 2025 IEEE 23rd World Symposium on Applied Machine Intelligence and Informatics (SAMI). pp. 000331–000336 (Jan 2025). https://doi.org/10.1109/SAMI63904.2025.10883288, journal Abbreviation: 2025 IEEE 23rd World Symposium on Applied Machine Intelligence and Informatics (SAMI)
63. M. Yabaku, N. Pombo, S. Ouhbi: Exploring the Potential Use of Generative AI in Software Engineering Education. In: 2024 IEEE 18th International Conference on Application of Information and Communication Technologies (AICT). pp. 1–7 (Sep 2024). https://doi.org/10.1109/AICT61888.2024.10740416, journal Abbreviation: 2024 IEEE 18th International Conference on Application of Information and Communication Technologies (AICT)
64. M. Yabaku, S. Ouhbi: University Students' Perception and Expectations of Generative AI Tools for Software Engineering. In: 2024 36th International Conference on Software Engineering Education and Training (CSEE&T). pp. 1–5 (Aug 2024). https://doi.org/10.1109/CSEET62301.2024.10663035, journal Abbreviation: 2024 36th International Conference on Software Engineering Education and Training (CSEE&T)
65. Mejia, D., Holmes, E.D., Marroquin, J., Gorson Benario, J.: Bridging Academia and Industry: Leveraging Generative AI in a Software Engineering Course for Practical Industry Experiences. In: Proceedings of the 30th ACM Conference on Innovation and Technology in Computer Science Education V. 1. pp. 507–513. ITiCSE 2025, Association for Computing Machinery, New York, NY, USA (2025). https://doi.org/10.1145/3724363.3729036, https://doi.org/10.1145/3724363.3729036, event-place: Nijmegen, Netherlands
66. Mellado, R., Cubillos, C.: Can Generative Artificial Intelligence Outperform Self-Instructional Learning in Computer Programming?: Impact on Motivation and Knowledge Acquisition. APPLIED SCIENCES-BASEL **15**(11) (May 2025). https://doi.org/10.3390/app15115867
67. Mohamed, K., Yousef, M., Medhat, W., Mohamed, E.H., Khoriba, G., Arafa, T.: Hands-on analysis of using large language models for the auto evaluation of programming assignments. Information Systems **128**, 102473 (Feb 2025). https://doi.org/10.1016/j.is.2024.102473, https://www.sciencedirect.com/science/article/pii/S0306437924001315
68. Mutanga, M.B., Msane, J., Mndaweni, T.N., Hlongwane, B.B., Ngcobo, N.Z.: Exploring the Impact of LLM Prompting on Students' Learning. TRENDS IN HIGHER EDUCATION **4**(3) (Jun 2025). https://doi.org/10.3390/higheredu4030031
69. Mwaniki, S., Araka, E., Kituku, B., Maina, E.: Students' perceptions on the use of generative ai in enhancing teaching and learning computer science courses. In: 2025 IST-Africa Conference (IST-Africa). pp. 1–11. IEEE (2025)
70. N. Andersen-Kiel, P. P. Linos: Using ChatGPT in Undergraduate Computer Science and Software Engineering Courses: A Students' Perspective. In:

2024 IEEE Frontiers in Education Conference (FIE). pp. 1–9 (Oct 2024). https://doi.org/10.1109/FIE61694.2024.10892934, journal Abbreviation: 2024 IEEE Frontiers in Education Conference (FIE)
71. N. Kiesler, D. Lohr, H. Keuning: Exploring the Potential of Large Language Models to Generate Formative Programming Feedback. In: 2023 IEEE Frontiers in Education Conference (FIE). pp. 1–5 (Oct 2023). https://doi.org/10.1109/FIE58773.2023.10343457, journal Abbreviation: 2023 IEEE Frontiers in Education Conference (FIE)
72. Nathaniel, J., Oyelere, S.S., Suhonen, J., Tedre, M.: Investigating the impact of generative AI integration on the sustenance of higher-order thinking skills and understanding of programming logic. Computers and Education: Artificial Intelligence **9**, 100460 (Dec 2025). https://doi.org/10.1016/j.caeai.2025.100460, https://www.sciencedirect.com/science/article/pii/S2666920X25001006
73. Newar, D.K.P., Zhao, R., Siy, H., Soh, L.K., Song, M.: SSDTutor: A feedback-driven intelligent tutoring system for secure software development. Science of Computer Programming **227**, 102933 (Apr 2023). https://doi.org/10.1016/j.scico.2023.102933, https://www.sciencedirect.com/science/article/pii/S0167642323000151
74. Neyem, A., Gonzalez, L.A., Mendoza, M., Alcocer, J.P.S., Centellas, L., Paredes, C.: Toward an AI Knowledge Assistant for Context-Aware Learning Experiences in Software Capstone Project Development. IEEE TRANSACTIONS ON LEARNING TECHNOLOGIES **17**, 1639–1654 (2024). https://doi.org/10.1109/TLT.2024.3396735
75. O. L. Dos Santos, D. Cury: Challenging the Confirmation Bias: Using ChatGPT as a Virtual Peer for Peer Instruction in Computer Programming Education. In: 2023 IEEE Frontiers in Education Conference (FIE). pp. 1–7 (Oct 2023). https://doi.org/10.1109/FIE58773.2023.10343247, journal Abbreviation: 2023 IEEE Frontiers in Education Conference (FIE)
76. O. Manakina, C. -H. Lung: Designing Reusable LLM-Enhanced Assignments: A Quality-Oriented Framework for Software Engineering Education. In: 2025 IEEE 49th Annual Computers, Software, and Applications Conference (COMPSAC). pp. 2212–2217 (Jul 2025). https://doi.org/10.1109/COMPSAC65507.2025.00310, journal Abbreviation: 2025 IEEE 49th Annual Computers, Software, and Applications Conference (COMPSAC)
77. O. Timcenko: Case Study: Using Artificial Intelligence as a Tutor for a Programming Course. In: 2024 21st International Conference on Information Technology Based Higher Education and Training (ITHET). pp. 1–4 (Nov 2024). https://doi.org/10.1109/ITHET61869.2024.10837624, journal Abbreviation: 2024 21st International Conference on Information Technology Based Higher Education and Training (ITHET)
78. Omeh, C.B., Olelewe, C.J., Ohanu, I.B.: Impact of Artificial Intelligence Technology on Students' Computational and Reflective Thinking in a Computer Programming Course. COMPUTER APPLICATIONS IN ENGINEERING EDUCATION **33**(3) (May 2025). https://doi.org/10.1002/cae.70052
79. OpenAI: Whisper. https://openai.com/research/whisper (2022), automatic speech recognition model
80. OpenAI: Chatgpt. https://chat.openai.com (2023), large language model
81. Ouh, E.L., Tan, K.W., Lo, S.L., Gan, B.K.S.: Evaluating ChatGPT to Answer Multi-Modal Exercises in Computer Science Education. In: Proceedings of the 30th ACM Conference on Innovation and Technology in Computer Sci-

ence Education V. 1. pp. 58–64. ITiCSE 2025, Association for Computing Machinery, New York, NY, USA (2025). https://doi.org/10.1145/3724363.3729056, https://doi.org/10.1145/3724363.3729056, event-place: Nijmegen, Netherlands
82. Penney, J., Acharya, P., Hilbert, P., Parekh, P., Sarma, A., Steinmacher, I., Gerosa, M.: Understanding Programming Students' Help-Seeking Preferences in the Era of Generative AI. In: Proceedings of the ACM Global on Computing Education Conference 2025 Vol 1. pp. 15–21. CompEd 2025, Association for Computing Machinery, New York, NY, USA (2025). https://doi.org/10.1145/3736181.3747165, https://doi.org/10.1145/3736181.3747165, event-place: Gaborone, Botswana
83. Pereira, A.F., Mello, R.F.: A systematic literature review on large language models applications in computer programming teaching evaluation process. IEEE Access (2025)
84. Pirzado, F.A., Ahmed, A., Mendoza-Urdiales, R.A., Terashima-Marin, H.: Navigating the pitfalls: Analyzing the behavior of llms as a coding assistant for computer science students—a systematic review of the literature. IEEE Access **12**, 112605–112625 (2024)
85. Pădurean, V.A., Denny, P., Gotovos, A., Singla, A.: Prompt Programming: A Platform for Dialogue-based Computational Problem Solving with Generative AI Models. In: Proceedings of the 30th ACM Conference on Innovation and Technology in Computer Science Education V. 1. pp. 458–464. ITiCSE 2025, Association for Computing Machinery, New York, NY, USA (2025). https://doi.org/10.1145/3724363.3729094, https://doi.org/10.1145/3724363.3729094, event-place: Nijmegen, Netherlands
86. R. Mellado, C. Cubillos, G. Ahumada: Effectiveness of Generative Artificial Intelligence in learning programming to higher education students. In: 2024 IEEE International Conference on Automation/XXVI Congress of the Chilean Association of Automatic Control (ICA-ACCA). pp. 1–7 (Oct 2024). https://doi.org/10.1109/ICA-ACCA62622.2024.10766746, journal Abbreviation: 2024 IEEE International Conference on Automation/XXVI Congress of the Chilean Association of Automatic Control (ICA-ACCA)
87. Rahe, C., Maalej, W.: How Do Programming Students Use Generative AI? Proc. ACM Softw. Eng. **2**(FSE) (Jun 2025). https://doi.org/10.1145/3715762, https://doi.org/10.1145/3715762, place: New York, NY, USA Publisher: Association for Computing Machinery
88. Ramirez Osorio, V., Zavaleta Bernuy, A., Simion, B., Liut, M.: Understanding the Impact of Using Generative AI Tools in a Database Course. In: Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1. pp. 959–965. SIGCSETS 2025, Association for Computing Machinery, New York, NY, USA (2025). https://doi.org/10.1145/3641554.3701785, https://doi.org/10.1145/3641554.3701785, event-place: Pittsburgh, PA, USA
89. Randall, N., Waeckerle, D., Stein, N., Gosler, D., Bente, S.: What an AI-Embracing Software Engineering Curriculum Should Look Like: An Empirical Study. IEEE SOFTWARE **41**(2), 36–43 (Apr 2024). https://doi.org/10.1109/MS.2023.3344682
90. Roldan-Alvarez, D., Mesa, F.J.: Intelligent Deep-Learning Tutoring System to Assist Instructors in Programming Courses. IEEE TRANSACTIONS ON EDUCATION **67**(1), 153–161 (Feb 2024). https://doi.org/10.1109/TE.2023.3331055
91. S. Jacobs, S. Jaschke: Leveraging Lecture Content for Improved Feedback: Explorations with GPT-4 and Retrieval Augmented Generation. In: 2024 36th Interna-

tional Conference on Software Engineering Education and Training (CSEE&T). pp. 1–5 (Aug 2024). https://doi.org/10.1109/CSEET62301.2024.10663001, journal Abbreviation: 2024 36th International Conference on Software Engineering Education and Training (CSEE&T)
92. S. M. Park, M. Ho, M. P. -C. Lin, J. Ryoo: Evaluating the Impact of Assistive AI Tools on Learning Outcomes and Ethical Considerations in Programming Education. In: 2025 IEEE Global Engineering Education Conference (EDUCON). pp. 1–10 (Apr 2025). https://doi.org/10.1109/EDUCON62633.2025.11016517, journal Abbreviation: 2025 IEEE Global Engineering Education Conference (EDUCON)
93. S. Mackay, K. Eiselt, A. Decker: Two Sides of the Same Coin: Differing Approaches to Generative AI in Two Computer Science Classrooms. In: 2024 IEEE Frontiers in Education Conference (FIE). pp. 1–9 (Oct 2024). https://doi.org/10.1109/FIE61694.2024.10893388, journal Abbreviation: 2024 IEEE Frontiers in Education Conference (FIE)
94. S. Mwaniki, E. Araka, B. Kituku, E. Maina: Students' Perceptions on the Use of Generative AI in Enhancing Teaching and Learning Computer Science Courses. In: 2025 IST-Africa Conference (IST-Africa). pp. 1–11 (May 2025). https://doi.org/10.23919/IST-Africa67297.2025.11060553, journal Abbreviation: 2025 IST-Africa Conference (IST-Africa)
95. S. Vemula: Enriching Python Programming Education With Generative AI: Leveraging Large Language Models for Personalized Support and Interactive Learning. In: 2024 IEEE Frontiers in Education Conference (FIE). pp. 1–8 (Oct 2024). https://doi.org/10.1109/FIE61694.2024.10893561, journal Abbreviation: 2024 IEEE Frontiers in Education Conference (FIE)
96. Schefer-Wenzl, S., Vogl, C., Peiris, S., Miladinovic, I.: Exploring the Adoption of Generative AI Tools in Computer Science Education: A Student Survey. In: Proceedings of the 2024 16th International Conference on Education Technology and Computers. pp. 173–178. ICETC '24, Association for Computing Machinery, New York, NY, USA (2025). https://doi.org/10.1145/3702163.3702188, https://doi.org/10.1145/3702163.3702188
97. Stoyanova, D., Stoyanova-Petrova, S., Mileva, N.: Exploring Students' and Teachers' Perceptions about Using ChatGPT in Programming Education. INTERNATIONAL JOURNAL OF ENGINEERING PEDAGOGY **15**(2), 15–41 (2025). https://doi.org/10.3991/ijep.v15i2.50607
98. Sun, D., Boudouaia, A., Zhu, C., Li, Y.: Would ChatGPT-facilitated programming mode impact college students' programming behaviors, performances, and perceptions? An empirical study. INTERNATIONAL JOURNAL OF EDUCATIONAL TECHNOLOGY IN HIGHER EDUCATION **21**(1) (Feb 2024). https://doi.org/10.1186/s41239-024-00446-5
99. Sun, D., Boudouaia, A., Zhu, C., Li, Y.: Would chatgpt-facilitated programming mode impact college students' programming behaviors, performances, and perceptions? an empirical study. International Journal of Educational Technology in Higher Education **21**(1), 14 (2024)
100. T. C. Freitas, M. João Varanda Pereira, A. C. Neto, P. Rangel Henriques: Goliath, a Programming Exercises Generator Supported by AI. In: 2024 19th Conference on Computer Science and Intelligence Systems (FedCSIS). pp. 331–342 (Sep 2024). https://doi.org/10.15439/2024F8479, journal Abbreviation: 2024 19th Conference on Computer Science and Intelligence Systems (FedCSIS)
101. T. N. B. Duong, C. Y. Meng: Automatic Grading of Short Answers Using Large Language Models in Software Engineering Courses. In: 2024 IEEE

Global Engineering Education Conference (EDUCON). pp. 1–10 (May 2024). https://doi.org/10.1109/EDUCON60312.2024.10578839, journal Abbreviation: 2024 IEEE Global Engineering Education Conference (EDUCON)
102. Tseng, C.H., Lin, H.C.K., Huang, A.C.W., Lin, J.R.: Investigating the effects of PuppyCodeReview, an AI-based code review system, on students' cognitive load. SoftwareX **31**, 102283 (Sep 2025). https://doi.org/10.1016/j.softx.2025.102283, https://www.sciencedirect.com/science/article/pii/S235271102500250X
103. V. Bhatt, Z. Yu, Y. Hou, J. Jin: ChatGPT as a Programming Tutor: Student Perceptions, Effectiveness, and Challenges. In: 2025 IEEE Global Engineering Education Conference (EDUCON). pp. 1–10 (Apr 2025). https://doi.org/10.1109/EDUCON62633.2025.11016463, journal Abbreviation: 2025 IEEE Global Engineering Education Conference (EDUCON)
104. V. Ramasamy, E. Kulpinski, T. Beaupre, A. Antreassian, Y. Jeong, P. J. Clarke, A. Aiello, C. Ray: Enhancing CS Education with LAs Using AI-Empowered AIELA Program. In: 2024 IEEE Frontiers in Education Conference (FIE). pp. 1–9 (Oct 2024). https://doi.org/10.1109/FIE61694.2024.10893294, journal Abbreviation: 2024 IEEE Frontiers in Education Conference (FIE)
105. V. Ramasamy, S. Ramamoorthy, G. S. Walia, E. Kulpinski, A. Antreassian: Enhancing User Story Generation in Agile Software Development Through Open AI and Prompt Engineering. In: 2024 IEEE Frontiers in Education Conference (FIE). pp. 1–8 (Oct 2024). https://doi.org/10.1109/FIE61694.2024.10893343, journal Abbreviation: 2024 IEEE Frontiers in Education Conference (FIE)
106. Wang, H., Wang, C., Chen, Z., Liu, F., Bao, C., Xu, X.: Impact of AI-agent-supported collaborative learning on the learning outcomes of University programming courses. EDUCATION AND INFORMATION TECHNOLOGIES **30**(12), 17717–17749 (Aug 2025). https://doi.org/10.1007/s10639-025-13487-8
107. Y. Kumar, A. Manikandan, J. J. Li, P. Morreale: Optimizing Large Language Models for Auto-Generation of Programming Quizzes. In: 2024 IEEE Integrated STEM Education Conference (ISEC). pp. 1–5 (Mar 2024). https://doi.org/10.1109/ISEC61299.2024.10665141, journal Abbreviation: 2024 IEEE Integrated STEM Education Conference (ISEC)
108. Y. Kumar, A. Manikandan, J. J. Li, P. Morreale: Preliminary Results from Integrating Chatbots and Low-Code AI in Computer Science Coursework. In: 2024 IEEE Integrated STEM Education Conference (ISEC). pp. 1–4 (Mar 2024). https://doi.org/10.1109/ISEC61299.2024.10665039, journal Abbreviation: 2024 IEEE Integrated STEM Education Conference (ISEC)
109. Y. Li, J. Keung, X. Ma: Integrating Generative AI in Software Engineering Education: Practical Strategies. In: 2024 International Symposium on Educational Technology (ISET). pp. 49–53 (Aug 2024). https://doi.org/10.1109/ISET61814.2024.00019, journal Abbreviation: 2024 International Symposium on Educational Technology (ISET)
110. Y. Xue, H. Chen, G. R. Bai, R. Tairas, Y. Huang: Does ChatGPT Help With Introductory Programming? An Experiment of Students Using ChatGPT in CS1. In: 2024 IEEE/ACM 46th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET). pp. 331–341 (Apr 2024). https://doi.org/10.1145/3639474.3640076, journal Abbreviation: 2024 IEEE/ACM 46th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)
111. Y. Zhuang, Y. Ji, Z. Huang, Y. Shang, X. Li, X. Su: Exploring the Integration of Artificial Intelligence and Programming Education: A Practice-

Oriented Study on the Development of Computational Thinking Based on Human-AI Co-Creation. In: 2025 IEEE 5th International Conference on Software Engineering and Artificial Intelligence (SEAI). pp. 325–329 (Jun 2025). https://doi.org/10.1109/SEAI65851.2025.11108869, journal Abbreviation: 2025 IEEE 5th International Conference on Software Engineering and Artificial Intelligence (SEAI)

112. Yang, A.C., Lin, J.Y., Lin, C.Y., Ogata, H.: Enhancing python learning with PyTutor: Efficacy of a ChatGPT-Based intelligent tutoring system in programming education. Computers and Education: Artificial Intelligence **7**, 100309 (Dec 2024). https://doi.org/10.1016/j.caeai.2024.100309, https://www.sciencedirect.com/science/article/pii/S2666920X24001127
113. Yilmaz, R., Karaoglan Yilmaz, F.G.: 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**, 100147 (Jan 2023). https://doi.org/10.1016/j.caeai.2023.100147, https://www.sciencedirect.com/science/article/pii/S2666920X23000267
114. Zhu, H., Xiang, J., Yang, Z.: UnrealMentor GPT: A System for Teaching Programming Based on a Large Language Model. COMPUTER APPLICATIONS IN ENGINEERING EDUCATION **33**(3) (May 2025). https://doi.org/10.1002/cae.70023