Large Language Models in Computer Science Education: A Systematic Literature Review

Nishat Raihan
mraihan2@gmu.edu
George Mason University
Fairfax, VA, USA

Mohammed Latif Siddiq
msiddiq3@nd.edu
University of Notre Dame
Notre Dame, IN, USA

Joanna C. S. Santos
joannacss@nd.edu
University of Notre Dame
Notre Dame, IN, USA

Marcos Zampieri
mzampier@gmu.edu
George Mason University
Fairfax, VA, USA

Abstract

Large language models (LLMs) are becoming increasingly better at a wide range of Natural Language Processing tasks (NLP), such as text generation and understanding. Recently, these models have extended their capabilities to coding tasks, bridging the gap between natural languages (NL) and programming languages (PL). Foundational models such as the *Generative Pre-trained Transformer (GPT)* and *LLaMA* series have set strong baseline performances in various NL and PL tasks. Additionally, several models have been fine-tuned specifically for code generation, showing significant improvements in code-related applications. Both foundational and fine-tuned models are increasingly used in education, helping students write, debug, and understand code. We present a comprehensive systematic literature review to examine the impact of LLMs in computer science and computer engineering education. We analyze their effectiveness in enhancing the learning experience, supporting personalized education, and aiding educators in curriculum development. We address five research questions to uncover insights into how LLMs contribute to educational outcomes, identify challenges, and suggest directions for future research.

CCS Concepts

• **Applied computing** → **Education**; • **Computing methodologies** → *Natural language processing*; **Natural language processing**.

Keywords

Large Language Models, Code Generation, CS Education

ACM Reference Format:
Nishat Raihan, Mohammed Latif Siddiq, Joanna C. S. Santos, and Marcos Zampieri. 2025. Large Language Models in Computer Science Education: A Systematic Literature Review. In *Proceedings of The Technical Symposium on Computer Science Education (SIGCSE TS) (SIGCSE TS '25).* ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

SIGCSE TS '25, Feb 26–Mar 01, 2025, Pittsburgh, PA

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Recent advances in Generative AI and LLMs, exemplified by GitHub Copilot [1] and ChatGPT [2], highlight their promising ability to address complex problems with human-like expertise. These advancements have a significant impact on education, where students may either benefit from or misuse these tools, compromising the integrity and quality of education [3]. This issue is particularly important in introductory Computer Science (CS) courses, which are directly affected by the capabilities of LLMs [4].

The ability of LLMs to efficiently handle programming tasks allows them to successfully complete assignments typically given in beginner courses, making them highly attractive to students seeking effortless solutions. Researchers have been examining the role of LLMs in CS education, focusing on how these models perform with current datasets and past assignments [5]. Identifying the use of AI tools in student work is another area of interest [6]. However, current methods, including plagiarism detection software, often fail to deliver reliable performance when handling output from a recently introduced LLM [7].

Tools powered by LLMs offer interesting opportunities to improve CS education [8]. When used responsibly and in the right way, they can be helpful for learning by providing students with quick feedback on coding assignments and creating different code examples to make programming concepts clearer [9]. Furthermore, as Generative AI tools become more common in real-world jobs [10], it is important to teach students about these tools in CS classes. This ensures that students are well-prepared for careers where such tools are widely used.

With students already adopting these tools [11], we do not yet fully understand their impact on learning. Due to the many challenges and benefits these technologies bring, understanding the impact of LLMs in CS education is paramount to improving areas such as curriculum design and assessment. While recent surveys and literature reviews have been published on different topics related to LLMs, such as their use in programming exercise generation [12], implications to security and privacy [13], and software development [14], to the best of our knowledge, no comprehensive survey has been published on the impact of LLMs in CS education. A couple of surveys published on related educational topics are

the one by Vierhauser et al. (2024) [15] that focuses on software engineering education and the one by Cambaz et al. (2024) [16] that focuses on programming alone.

In this paper, we fill this important gap in the literature by presenting the first comprehensive systematic literature review (SLR) investigating the impact of LLMs on CS education. We address five carefully crafted research questions (RQs) to understand how these models influence educational outcomes, pinpoint challenges, and suggest future research directions. Following Kitchenham and Charters (2007) guidelines [17] for conducting SLR, we use a thorough search strategy across multiple databases, applying strict inclusion and exclusion criteria to ensure the studies selected are relevant and of high quality. Our results reveal the transformative impact of LLMs in CS educational practices. To enable reproducibility, our scripts and data are available in a GitHub repository. [1]

2 Methodology

To conduct this SLR, we follow the aforementioned guidelines [17] that suggest three main steps: *planning* the literature review, *conducting* the literature review, and *reporting* the results. During the planning phase, we set five clear RQs and create a detailed plan for our SLR. In the conducting phase, we search for relevant studies and select them based on specific criteria. Finally, in the reporting phase, we organize and present our findings in this paper.

2.1 Research Questions

The five RQs addressed in this SLR to understand the use and impact of LLMs in CS education are the following:

RQ1: ***What are the educational levels in which LLMs are used?***

In this RQ, we examine the education stages (*e.g.*, undergraduate level, graduate level, *etc.*), in which LLMs are integrated. This examination aims to identify the most effective stages for introducing and integrating these models, enhancing learning outcomes, and guiding future educational practices.

RQ2: ***What are the sub-disciplines of CS that are the focus of the studied papers?***

We investigate what are the specific sub-disciplines (*e.g.*, CS1, software testing, *etc.*) that were the target of the studied works.[2] This RQ aims to identify areas needing further research.

RQ3: ***What research methodologies are mostly used in the papers?***

We explore the research methodologies, such as experimental designs and data analysis techniques, employed in the selected papers. This RQ aims to understand the field's research practices and standards with respect to this ML-based technology.

RQ4: ***What are the most commonly used programming languages (PLs) in studies involving LLMs?***

In this RQ, we examine the programming languages that are the focus of the paper. Identifying the frequently targeted languages helps reveal trends and gaps in educational research and practice.

RQ5: ***Which large language models (LLMs) are employed in these studies?***

In this RQ, we study the LLMs most widely used in research papers. Cataloging the specific LLMs used provides insights into the diversity and rationale for model selection.

2.2 Search Method

To answer our RQs, we use the search query below to retrieve all ***primary*** works related to LLMs for CS education:

```
("software engineering" OR "programming" OR "software development" OR "computer science" OR "computer engineering") AND ("education" OR "teaching") AND ("LLM" OR "large language model")
```

We apply this query to the following library databases [3]: the ACM Digital Library, IEEE Xplore, Scopus, the ACL Anthology, ISI Web of Science, Springer Link, Science@Direct, and ArXiv. This search query results in a total of 1,735 papers.

2.3 Inclusion and Exclusion Criteria

We vet the papers to exclude those that do not meet our *inclusion* criteria or that meet our *exclusion* criteria. Our criteria, shown in Table 1, ensure the relevance and quality of the selected works.

Table 1: Inclusion and Exclusion criteria to Select Papers

Inclusion Criteria	Exclusion Criteria
I1 Full papers (*i.e.*, at least 4 full pages of text, excluding references). **I2** Written in English. **I3** Focus on or investigate the use of code LLMs to teach computing concepts. **I4** Written between January 2019 - June 2024.	**E1** Duplicated studies **E2** Not written in English. **E3** Abstracts, posters, or extended abstracts with less than 4 full pages of text. **E4**: Survey and Systematic Literature Reviews (SLRs).

We begin with a total of 1,735 primary studies. First, we removed duplicated studies and papers not published within the past 5 years, obtaining a total of 1,423 papers. Subsequently, we inspect each paper's *title*, *keywords*, *number of pages*, and *abstract* to determine their relevance based on our inclusion and exclusion criteria. This screening process reduces the number of papers to 187. Finally, we apply the same criteria to the full text of these papers, leaving us with **125** papers included in this literature review.

2.4 Data Extraction

As we reviewed the papers, we extracted the key information we were looking for to answer our RQs: the *educational level*, the *CS discipline*, and *programming languages* that were the focus of the paper as well as the *LLMs* and *research methodologies* that were employed. This data was extracted by two of the authors and peer-reviewed by the senior author.

3 Results

Upon carefully reviewing the 125 selected papers, we conducted a high-level analysis addressing the RQs presented in Section 2.1.

[1] https://anonymous.4open.science/status/llm-education-survey-EE2B

[2] As described in Section 2.2, we consider some sub-disciplines that are commonly within the scope of computer engineering along with CS sub-disciplines. The acronym used throughout the paper is CS.

[3] *portal.acm.org*. IEEE Xplore: *ieeexplore.ieee.org*. Scopus: *scopus.com*. ACL: *aclanthology.org*. Web of Science: *isiknowledge.com*. Springer: *link.springer.com*. Science@Direct: *sciencedirect.com*. ArXiv: *arxiv.org*.

3.1 RQ1: Educational Levels

As shown in Figure 1, **111** of the studied papers focus on *undergraduate-level* CS courses [1, 6, 9, 11, 18–124]. While **15** works explored advanced courses typically taught at the *graduate level* [21, 28, 30, 39, 52, 89, 92, 111, 114, 125–130], only **4** papers include *PhD-level* courses [21, 30, 92, 127], and just **2** papers addressed *K-12* education [131, 132]. There was one work [133] that examined how ChatGPT is used as a means for training employees in a *software engineering workplace* (professional context).

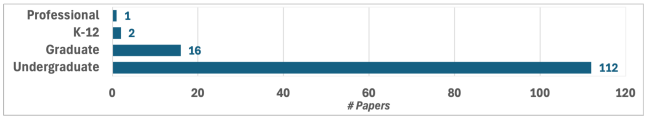

Figure 1: Educational Levels where LLMs are used.

This strong focus on undergraduate level is primarily due to the limited presence of CS courses at lower educational levels (K-12) and the current limitations of most LLMs in handling higher-level content [120, 128]. Although newer models like GPT-4 have shown promising results for some graduate courses [39, 90], many earlier studies [100, 128] did not have the chance to test these capabilities. Even in more recent works, GPT-4 is sometimes excluded due to its cost or other practical issues [99].

> **RQ1 Results:** Over **80%** of the papers focused on undergraduate-level CS education. Further studies are needed to assess the effectiveness and helpfulness of LLMs for graduate students and professionals undergoing CS training.

3.2 RQ2: CS Sub-disciplines

As shown in Table 2, over half of the 125 observed studies focus on *introduction to programming*, predominantly in Python and occasionally in Java. This emphasis is expected, given that CS students frequently use LLMs for code generation [30]. The emphasis on introductory programming arises from Python's widespread adoption as the first language in many CS curricula and Java's significant role in teaching object-oriented programming principles.

There were **19** works that focused on *introductory CS* concepts, featuring Q&As and multiple-choice questions related to basic computer science topics [48, 119, 124]. These studies not only assess LLM-generated code, solutions, and feedback but also explore the generation of tasks, assignments, and questions [9, 136]. More advanced courses, such as *Data Science*, present mixed findings regarding the effectiveness of LLMs; some studies claim that LLMs perform well [125], while others disagree [114, 130].

Due to space constraints, disciplines with less than 3 papers are aggregated as "Others" in Table 2. These other advanced topics were *Distributed Systems* [21, 69], *Operating Systems* [19, 69], *Computer Networks* [69], *Numerical Analysis* [69], *Interactive Systems* [69], *Real-Time Systems* [69], *Concurrent, Parallel and Distributed Computing* [126], *Software Testing* [128], *Information Technology* [50], *Computer Graphics* [43], *Human-computer Interaction* [46], *Databases* [91], *Automata Theory and Formal Languages* [83], *Bioinformatics* [90], *Software Security* [74], and *Data Visualization* [65].

Table 2: CS Disciplines Explored by the Studied Papers.

CS Discipline	Total	References
Introduction to Programming	65	[1, 11, 18–20, 22–27, 29, 31, 34, 36–38, 40, 41, 44, 45, 51, 53, 55, 57–64, 67, 68, 70, 72, 73, 77–80, 82, 84–86, 88, 93, 95, 100–108, 112, 116, 117, 121, 123, 132, 134, 135]
Introduction to CS	19	[1, 11, 37, 38, 48, 49, 58, 82, 86, 87, 102, 104, 105, 109, 113, 115, 118–120]
Data Science	9	[47, 69, 71, 76, 114, 120, 125, 129, 130]
Software Engineering	8	[6, 42, 52, 66, 69, 97, 99, 133]
Object-Oriented Programming	4	[9, 32, 33, 69]
Algorithms	4	[56, 71, 83, 110]
Web Development	3	[69, 81, 96]
Machine Learning	3	[39, 98, 120]
Computer architecture	3	[83, 124, 127]
CS Education in General	10	[28, 30, 35, 75, 89, 92, 94, 111, 122, 131]
Others	14	[19, 21, 43, 46, 50, 65, 69, 69, 69, 74, 83, 90, 91, 126, 128]

> **RQ2 Results: 67%** of works focus on *introduction to programming* and *introduction to CS*. There is limited focus on more advanced CS concepts, suggesting a need for further exploration on LLMs' ability in helping to teach advanced CS concepts.

3.3 RQ3: Research Methodologies

Table 3 summarizes the research methodologies followed by the studied papers. Our findings show that **38%** of papers use case studies and ethnography as their research method, which means these papers are focused on how LLMs can be used in different use cases of CS education. Moreover, 24% of the papers used the action research method, where the researchers introduced novel LLM-based tools and techniques and applied them in a CS educational context. We also found **24** works in which researchers did experiments with students from various levels as described in RQ1 (Section 3.1). In 14% of the works we analyzed, we found researchers were involved in data-centric analysis, and in 12% of cases, they used grounded theory for qualitative analysis. 12% of papers were involved in engineering research that invents and evaluates LLM-based artifacts for CS education. Researchers were also involved in interview studies and case studies. They also used multi-methodology and mixed-methods research to use LLMs in CS education. In 5% of works, they benchmarked LLMs for CS education tasks. The rest of the two works used Optimization Studies [9] and Repository Mining [121] as their research methodology.

While most studies conducted interviews with students, teachers as well as practitioners (humans), the work by Dengel *et al.* [35] conducted semi-structured interviews with LLMs. Their goal was to examine the applicability of qualitative research methods to interviews with LLMs. In their study, LLMs were asked questions related to the relevance of computer science in K-12 education.

Table 3: Research Methodologies used by the Papers.

Methodology	Total	References
Case Study and Ethnography	48	[9, 18, 25–27, 29, 31, 33, 37, 39, 41–43, 45, 46, 48, 49, 51, 54, 56, 57, 60, 62–64, 69, 70, 74, 76–78, 83, 90, 91, 100, 103, 112, 114–116, 120, 121, 125, 129, 131, 134, 135, 137]
Action Research	30	[20, 27, 29, 32, 34, 38, 41, 42, 44, 45, 50, 53, 56, 58, 61–63, 67, 74, 80, 81, 102, 106, 109, 119, 124, 130, 133, 135, 138]
Experiments with Human Participants	24	[18, 23, 24, 26, 35, 38, 47, 51, 63, 68, 69, 72, 74, 75, 85, 103, 106, 111, 115, 118, 123, 131, 132, 134]
Data Science	18	[1, 11, 24, 30, 33, 38, 47, 49, 55, 68, 70, 85, 89, 106, 110, 111, 125, 126]
Grounded Theory	15	[6, 19, 23, 28, 36, 37, 59, 66, 67, 72, 82, 84, 98, 108, 137]
Engineering Research (Design Science)	15	[1, 25, 31, 40, 71, 73, 76, 93, 96, 99, 101, 104, 113, 117, 128]
Qualitative Surveys	9	[21, 52, 55, 58, 65, 88, 118, 122, 123]
Longitudinal Studies	8	[6, 11, 59, 79, 82, 86, 98, 105]
Mixed Methods Research	7	[22, 43, 64, 82, 88, 91, 97]
Benchmarking	6	[95, 99, 104, 107, 128, 139]
Case Survey	4	[87, 92, 94, 127]
Optimization Studies	1	[9]
Repository Mining	1	[121]

RQ3 Results: Around **60%** works focus on case studies of using LLMs and action research for creating tools around LLMs for CS educational tasks. Researchers also did qualitative research by conducting surveys. Limited works explore data-centric analysis and benchmarking LLMs.

3.4 RQ4: Programming Languages

Figure 2 shows the top 10 programming languages that the reviewed papers focused on (*TypeScript* and *R* are tied for the 10th position). Among the programming languages, *Python* is the most studied, being mentioned in **55%** of the papers. This prominence is likely because many studies focus on introductory CS courses, where Python is often the first language taught. The second most used language is *Java* [120, 128], primarily taught as an object-oriented programming language. *C* [63] and *C++* [73] receive comparatively less attention. Fewer works focus on web languages like *JavaScript*, *HTML*, and *CSS* [50, 59, 65, 69, 81, 92, 92, 96, 120].

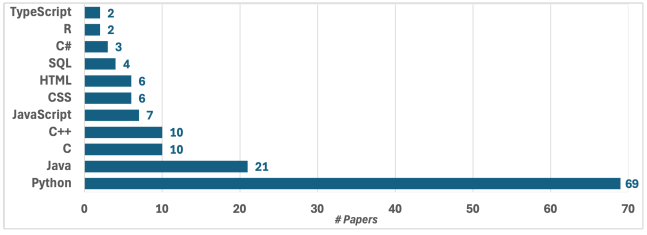

Figure 2: Top 10 Programming Languages.

Our findings also reveal that LLMs are employed not only for code generation but also for code-related Q&A tasks [82, 116, 119, 124] and MCQs [48, 49, 105, 109] across various CS courses. Moreover, some studies do not specify particular tasks but instead provide a broader overview, such as examining how LLMs are perceived by students [94] and teachers [111, 122], evaluating course feedback [75, 89, 126], and generating scaffolds [31].

RQ4 Results: Most papers focused on Python and Java code generation while neglecting other commonly used languages such as JavaScript, C & C++.

3.5 RQ5: Most commonly used LLMs

Most works used ChatGPT [140] without employing the API, primarily relying on the earlier GPT-3.5 model, with some using GitHub Copilot (OpenAI's Codex) [141] and GPT-4 [142] (see Figure 3). This trend is largely because many of the studies were conducted before the release of GPT-4 and the higher cost of GPT-4 compared to the mostly free GPT-3.5. Additionally, several works accessed GPT-3.5-Turbo through OpenAI's API. Besides OpenAI models, Microsoft's BingAI[4], as well as Google's BARD[5] and Gemini[6], have been used occasionally. Code-finetuned models like StarCoder [143] and CodeBERT [144] were also used a few times. Unlike OpenAI's models, StarCoder and CodeBERT are free to use.

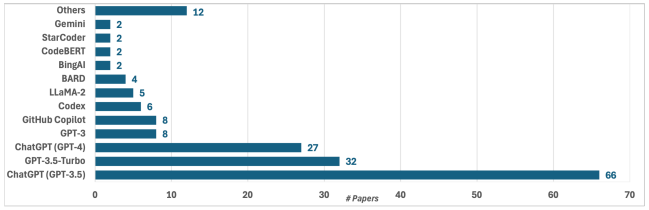

Figure 3: Most commonly used LLMs. *Others* include models that are used only once. A work may use more than one LLM.

There were 26 different models used only once across 12 papers. These include Anthropic's Claude[7] and earlier LLMs like Mistral [145], Falcon [146], MPT [147] etc.

RQ5 Results: Most papers used commercial models in their studies, with ChatGPT being the most used model due to its popularity among students. However, papers mostly used its older version (GPT-3.5) instead of GPT-4 due to its costs.

4 Discussion

Along with the five RQs answered in this SLR, in our comprehensive analysis, we also identified four important discussion points on LLMs in CS Education as follows:

– Students' and instructors' sentiment about using LLMs in CS Education: Students generally have positive experiences with LLMs and LLM-based tools [21, 93, 133–135]. Studies have shown that students consider examples generated by LLMs helpful [60]

[4] bing.com/chat
[5] bard.google.com
[6] gemini.google.com
[7] claude.ai

and perceive that LLMs could enhance their knowledge [6] by providing helpful feedback [123]. CS Students praised LLMs for providing explanations that were easy to understand [73] and thought that LLMs could be an additional agent with teaching assistants. However, studies have also shown that students expressed some frustration about crafting prompts that elicit the desired output [84]. Furthermore, some studies indicate that students found it hard to find relevant or accurate responses from LLMs [21]. From the perspective of the instructors, studies have indicated that CS instructors found a negative correlation between the usage of LLMs and students' grades [59]. They found LLMs could negatively affect students' ability to solve programming tasks independently [59]. They expressed concerns about proper learning, over-reliance on tools, and plagiarism when using LLMs in CS education [132].

– **Task completion with LLMs and LLM-based tools:** As described in the results of RQ2 and RQ4, LLMs and LLM-based tools were applied to solve assignments and programming problems in different PLs from different courses in CS. Regarding the successful completion of the tasks, we found mixed results regarding the use of LLMs. Studies have shown that LLMs can help students solve introductory programming problems, repair buggy code [68, 80, 117], and help write better code [62, 64, 93]. According to the findings, LLMs are generally better at writing code than solving question-answer [116]. They can also generate programming problems [101, 112], MCQs [109, 113], and detect AI-generated code despite having false positives [61]. LLMs can also provide feedback to the student to improve their code [75, 89]. However, LLMs can partially help with data science [114, 130] but hardly solve machine learning problems [39]. They also suffer from problems in other languages other than English. For example, LLM performed poorly on Chinese Python question-answering problems [121].

– **Adoption of LLMs in different use cases:** LLMs are heavily adopted by students [52, 82]; they often try ChatGPT to solve their problems but remain skeptical overall [82, 96]. The adoption of LLMs varied depending on the students' coding skills and prior experience [97]. It is also significantly influenced by their perception of future career norms [88]. There is potential for rapid iteration, creative ideation, and avoiding social pressures for using LLMs [54]. Students used it as a chatbot [63], integrated with the IDE [71], and as a substitute for the teaching assistant [6]. They usually read the generated code to solve a task and mostly understand them [115].

– **Expectation and future direction of using LLMs in CS Education:** Though LLMs can help solve assignments, provide feedback, and repair code, both students and instructors desire more than answers from LLMs [36]. Students and instructors agree that LLMs should be welcome in academia [66, 131] and that the integration of LLMs with teaching can lead to a better understanding [21]. Instructors ask to change the curriculum as they can solve most of the data structure problems [129] but are urged to handle LLMs carefully [28].

5 Conclusion

In this paper, we presented the first comprehensive SLR on LLMs in CS education. We identified and analyzed 1,735 related papers. After applying well-defined inclusion and exclusion criteria, we described 125 relevant papers in this SLR - the most related SLR to ours [16] has covered 21 papers focusing only on programming. Taking the 125 relevant papers into consideration, we answered five important RQs related to educational levels, sub-disciplines, methodologies, and PLs. We also presented a brief discussion on the adoption of LLMs, students' sentiments, and future directions.

Our findings indicate that most current research focuses on undergraduate education and introductory programming courses. They also indicate that most research applies case-based studies, while the most widely-used PL is Python. All in all, although students are usually positive about using LLMs, instructors are worried about learning effectiveness because of potential over-reliance on them. Our SLR also indicates that educators are gradually adopting LLMs in their courses but that most CS curricula still need to be changed to accommodate recent advances in AI. We hope that the insights gained from this comprehensive SLR will help inform and enhance future research and applications of LLMs in CS education, contributing to a deeper understanding of their role and effectiveness in various educational contexts.

References

[1] P. Denny, V. Kumar, and N. Giacaman. Conversing with copilot: Exploring prompt engineering for solving cs1 problems using natural language. In *SIGCSE*, 2023.
[2] OpenAI. Gpt-4 technical report. https://arxiv.org/abs/2303.08774, 2023.
[3] T. Phung, Victor-Alexandru, et al. Generative ai for programming education: Benchmarking chatgpt, gpt-4, and human tutors, 2023.
[4] K. Z. Zhou, Z. Kilhoffer, et al. "the teachers are confused as well": A multiple-stakeholder ethics discussion on large language models in computing education, 2024.
[5] Yann Hicke, Anmol Agarwal, Qianou Ma, and Paul Denny. Ai-ta: Towards an intelligent question-answer teaching assistant using open-source llms, 2023.
[6] B. Arie Tanay, L. Arinze, S. S. Joshi, K. A. Davis, and J. C. Davis. An exploratory study on upper-level computing students' use of large language models as tools in a semester-long project, 2024.
[7] J. Meyer, R. Urbanowicz, et al. Chatgpt and large language models in academia: opportunities and challenges. *BioData Mining*, 2023.
[8] Harsh Kumar, Ilya Musabirov, et al. Impact of guidance and interaction strategies for llm use on learner performance and perception, 2024.
[9] M. Pankiewicz and R. S. Baker. Large language models (gpt) for automating feedback on programming assignments, 2023.
[10] Inbal Shani. Survey reveals AI's impact on the developer experience | The GitHub Blog. *GitHub Blog*, June 2023. URL https://github.blog/2023-06-13-survey-reveals-ais-impact-on-the-developer-experience/#methodology.
[11] R. Budhiraja, I. Joshi, et al. "it's not like jarvis, but it's pretty close!" - examining chatgpt's usage among undergraduate students in computer science. In *ACE*, 2024.
[12] E. Frankford, I. Höhn, C. Sauerwein, and R. Breu. A survey study on the state of the art of programming exercise generation using large language models. *arXiv*, 2024.
[13] Y. Yao, J. Duan, K. Xu, Y. Cai, Z. Sun, and Y. Zhang. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing*, page 100211, 2024.
[14] A. Fan, B. Gokkaya, M. Harman, M. Lyubarskiy, S. Sengupta, S. Yoo, and J. Zhang. Large language models for software engineering: Survey and open problems. In *ICSE*, 2023.
[15] M. Vierhauser, I. Groher, T. Antensteiner, and C. Sauerwein. Towards integrating emerging ai applications in se education. In *CSEE&T*, 2024.
[16] D. Cambaz and X. Zhang. Use of ai-driven code generation models in teaching and learning programming: a systematic literature review. In *SIGCSE*, 2024.
[17] B. A. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering. Technical report, 2007.
[18] M. Abolnejadian, S. Alipour, and K. Taeb. Leveraging chatgpt for adaptive learning through personalized prompt-based instruction: A cs1 education case study. In *CHI*, 2024.
[19] V. Agarwal, M. Krishan Garg, S. Dharmavaram, and D. Kumar. "which llm should i use?": Evaluating llms for tasks performed by undergraduate computer science students, 2024.
[20] Anishka and A. Mehta. Can chatgpt play the role of a teaching assistant in an introductory programming course?, 2023.

[21] C. Arora, U. Venaik, P. Singh, S. Goyal, J. Tyagi, S. Goel, U. Singhal, and D. Kumar. Analyzing llm usage in an advanced computing class in india, 2024.

[22] I. Aviv, M. Leiba, H. Rika, and Y. Shani. The impact of chatgpt on students' learning programming languages. In *CHI*, 2024.

[23] I. Azaiz, O. Deckarm, and S. Strickroth. Ai-enhanced auto-correction of programming exercises: How effective is gpt-3.5? *iJEP*, 2023.

[24] I. Azaiz, N. Kiesler, and S. Strickroth. Feedback-generation for programming exercises with gpt-4, 2024.

[25] N. P. Bakas, M. Papadaki, E. Vagianou, I. Christou, and S. A. Chatzichristofis. *Integrating LLMs in Higher Education, Through Interactive Problem Solving and Tutoring: Algorithmic Approach and Use Cases*. Springer Nature Switzerland, 2024.

[26] R. Balse, V. Kumar, P. Prasad, and J. Madathil Warriem. Evaluating the quality of llm-generated explanations for logical errors in cs1 student programs. In *COMPUTE*, 2023.

[27] R. Balse, B. Valaboju, S. Singhal, J. Madathil Warriem, and P. Prasad. Investigating the potential of gpt-3 in providing feedback for programming assessments. In *ITiCSE*, 2023.

[28] B. A. Becker, P. Denny, and J. Finnie-Ansley. Programming is hard - or at least it used to be: Educational opportunities and challenges of ai code generation. In *SIGCSE*, 2023.

[29] J. Berrezueta-Guzman and S. Krusche. Recommendations to create programming exercises to overcome chatgpt. In *CSEE&T*, 2023.

[30] U. Ali Bukar, M. Shohel Sayeed, S. Fatimah Abdul Razak, S. Yogarayan, O. Ahmed Amodu, and R. Azlina Raja Mahmood. Text analysis on early reactions to chatgpt as a tool for academic progress or exploitation. *SN Computer Science*, 2024.

[31] C. Cao. Scaffolding cs1 courses with a large language model-powered intelligent tutoring system. In *IUI*, 2023.

[32] B. Pereira Cipriano and P. Alves. Gpt-3 vs object oriented programming assignments: An experience report. In *ITICSE*, 2023.

[33] B. Pereira Cipriano and 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 *ITICSE*, 2024.

[34] A. Del Carpio Gutierrez and P. Denny. Evaluating automatically generated contextualised programming exercises. In *SIGCSE*, 2024.

[35] A. Dengel, R. Gehrlein, and D. Fernes. Qualitative research methods for large language models: Conducting semi-structured interviews with chatgpt and bard on computer science education. *Informatics*, 2023.

[36] P. Denny, S. MacNeil, J. Savelka, L. Porter, and A. Luxton-Reilly. Desirable characteristics for ai teaching assistants in programming education. *arXiv*, 2024.

[37] P. Denny and J. Leinonen. Prompt problems: A new programming exercise for the generative ai era. In *SIGCSE*, 2024.

[38] J. Doughty, Z. Wan, et al. A comparative study of ai-generated (gpt-4) and human-crafted mcqs in programming education. In *ACE*, 2024.

[39] I. Drori, S. J. Zhang, R. Shuttleworth, and S. Zhang. From human days to machine seconds: Automatically answering and generating machine learning final exams. In *KDD*, 2023.

[40] M. E. Ellis, K. Mike Casey, and G. Hill. Chatgpt and python programming homework. *DSJIE*, 2024.

[41] A. Fan and H. Zhang. Exploring the potential of large language models in generating code-tracing questions for introductory programming courses. In *EMNLP*, 2023.

[42] J. Carlos Farah, S. Ingram, B. Spaenlehauer, F. Kim-Lan Lasne, and D. Gillet. *Prompting Large Language Models to Power Educational Chatbots*. Springer Nature Singapore, 2023.

[43] T. Haoran Feng, P. Denny, B. Wuensche, A. Luxton-Reilly, and S. Hooper. More than meets the ai: Evaluating the performance of gpt-4 on computer graphics assessment questions. In *ACE*, 2024.

[44] A. S. Fernandez and K. A. Cornell. Cs1 with a side of ai: Teaching software verification for secure code in the era of generative ai. In *SIGCSE*, 2024.

[45] E. Frankford, C. Sauerwein, P. Bassner, S. Krusche, and R. Breu. Ai-tutoring in software engineering education. In *ICSE-SEET*, 2024.

[46] A. Pimenta Freire, P. Christina Figueira Cardoso, and A. de Lima Salgado. May we consult chatgpt in our human-computer interaction written exam? an experience report after a professor answered yes. In *IHC*, 2024.

[47] A. Garg and R. Rajendran. The impact of structured prompt-driven generative ai on learning data analysis in engineering students. In *CSEDU*, 2024.

[48] C. Grévisse. Comparative quality analysis of gpt-based multiple choice question generation. In *Applied Informatics*, 2024.

[49] C. Grévisse, M. Angeliki S. Pavlou, and J. G. Schneider. Docimological quality analysis of llm-generated multiple choice questions in computer science and medicine. *SN Computer Science*, 2024.

[50] S. Gumina, T. Dalton, and J. Gerdes. Teaching it software fundamentals: Strategies and techniques for inclusion of large language models: Strategies and techniques for inclusion of large language models. In *SIGITE*.

[51] P. Haindl and G. Weinberger. Students' experiences of using chatgpt in an undergraduate programming course. *IEEE Access*, 2024.

[52] K. Hanifi, O. Cetin, and C. Yilmaz. On chatgpt: Perspectives from software engineering students. In *QRS*.

[53] M. Hoq and Y. Shi. Detecting chatgpt-generated code submissions in a cs1 course using machine learning models. In *SIGCSE*, 2024.

[54] I. Hou, S. Mettille, O. Man, Z. Li, C. Zastudil, and S. MacNeil. The effects of generative ai on computing students' help-seeking preferences. In *ACE*, 2024.

[55] S. Jacobs and S. Jaschke. Evaluating the application of large language models to generate feedback in programming education, 2024.

[56] H. Jin, S. Lee, H. Shin, and J. Kim. Teach ai how to code: Using large language models as teachable agents for programming education. In *CHI*, 2024.

[57] M. Jordan, K. Ly, and A. Gerald Soosai Raj. Need a programming exercise generated in your native language? chatgpt's got your back: Automatic generation of non-english programming exercises using openai gpt-3.5. In *SIGCSE*, 2024.

[58] I. Joshi, R. Budhiraja, H. Dev, J. Kadia, M. Osama Ataullah, S. Mitra, H. D. Akolekar, and D. Kumar. Chatgpt in the classroom: An analysis of its strengths and weaknesses for solving undergraduate computer science questions. In *SIGCSE*, 2024.

[59] G. Jošt, V. Taneski, and S. Karakatič. The impact of large language models on programming education and student learning outcomes. *Applied Sciences*, 2024.

[60] B. Jury, A. Lorusso, J. Leinonen, P. Denny, and A. Luxton-Reilly. Evaluating llm-generated worked examples in an introductory programming course. In *ACE*, 2024.

[61] O. Karnalim, H. Toba, and M. Christianti Johan. Detecting ai assisted submissions in introductory programming via code anomaly. *EAIT*, 2024.

[62] M. Kazemitabaar, J. Chow, and C. Ka To Ma. Studying the effect of ai code generators on supporting novice learners in introductory programming. In *CHI*, 2023.

[63] M. Kazemitabaar, R. Ye, X. Wang, A. Zachary Henley, P. Denny, M. Craig, and T. Grossman. Codeaid: Evaluating a classroom deployment of an llm-based programming assistant that balances student and educator needs. In *CHI*, 2024.

[64] N. Kiesler and D. Schiffner. Large language models in introductory programming education: Chatgpt's performance and implications for assessments, 2023.

[65] N. Wook Kim, H. Ko, G. Myers, and B. Bach. Chatgpt in data visualization education: A student perspective. *arXiv preprint arXiv:2405.00748*, 2024.

[66] V. D. Kirova and C. S. Ku. Software engineering education must adapt and evolve for an llm environment. In *SIGCSE*, 2024.

[67] T. Kosar, D. Ostojić, Y. David Liu, and M. Mernik. Computer science education in chatgpt era: Experiences from an experiment in a programming course for novice programmers. *Mathematics*, 2024.

[68] C. Koutcheme. *Training Language Models for Programming Feedback Using Automated Repair Tools*. Springer, 2023.

[69] T. Krüger and M. Gref. Performance of large language models in a computer science degree program. In *AI4AI*, 2024.

[70] N. Ashok Kumar and A. Lan. Using large language models for student-code guided test case generation in computer science education, 2024.

[71] K. Kuramitsu, Y. Obara, M. Sato, and M. Obara. Kogi: A seamless integration of chatgpt into jupyter environments for programming education. In *SPLASH-E*, 2023.

[72] S. Lau and P. Guo. From "ban it till we understand it" to "resistance is futile": How university programming instructors plan to adapt as more students use ai code generation and explanation tools such as chatgpt and github copilot. In *ICER*, 2023.

[73] J. Leinonen, P. Denny, and S. MacNeil. Comparing code explanations created by students and large language models. In *ITiCSE*, 2023.

[74] J. Li and P. Håkon Meland. Evaluating the impact of chatgpt on exercises of a software security course, 2023.

[75] H. Li. The potential of large language models as tools for analyzing student textual evaluation: A differential analysis between cs and non-cs students. In *CEI*, 2023.

[76] M. Liffiton, B. E Sheese, J. Savelka, and P. Denny. Codehelp: Using large language models with guardrails for scalable support in programming classes. In *Koli Calling*, 2024.

[77] M. Liu and F. M'Hiri. Beyond traditional teaching: Large language models as simulated teaching assistants in computer science. In *SIGCSE*, 2024.

[78] R. Liu, C. Zenke, and C. Liu. Teaching cs50 with ai: Leveraging generative artificial intelligence in computer science education. In *SIGCSE*, 2024.

[79] W. Lyu, Y. Wang, T. Rachel Chung, Y. Sun, and Y. Zhang. Evaluating the effectiveness of llms in introductory computer science education: A semester-long field study. *arXiv*, 2024.

[80] Q. Ma, H. Shen, K. Koedinger, and S. Wu. How to teach programming in the ai era? using llms as a teachable agent for debugging. In *AIED*, 2024.

[81] S. MacNeil, A. Tran, and A. Hellas. Experiences from using code explanations generated by large language models in a web software development e-book. In *SIGCSE*, 2023.

[82] E. D. Manley, T. Urness, A. Migunov, and M. Alimoor Reza. Examining student use of ai in cs1 and cs2. *J. Comput. Sci. Coll.*, 2024.

[83] N. C. Mendonça. Evaluating chatgpt-4 vision on brazil's national undergraduate computer science exam. *ACM Trans. Comput. Educ.*, 2024.

[84] S. Nguyen, H. McLean Babe, Y. Zi, A. Guha, C. Jane Anderson, and M. Q Feldman. How beginning programmers and code llms (mis)read each other. In *CHI*, 2024.
[85] P. Oli, R. Banjade, J. Chapagain, and V. Rus. Automated assessment of students' code comprehension using llms, 2024.
[86] G. Oosterwyk, P. Tsibolane, P. Kautondokwa, and A. Canani. Beyond the hype: A cautionary tale of chatgpt in the programming classroom, 2024.
[87] M. Sheinman Orenstrakh, O. Karnalim, C. Anibal Suarez, and M. Liut. Detecting llm-generated text in computing education: A comparative study for chatgpt cases, 2023.
[88] A. Padiyath, X. Hou, A. Pang, D. Viramontes Vargas, X. Gu, T. Nelson-Fromm, Z. Wu, M. Guzdial, and B. Ericson. Insights from social shaping theory: The appropriation of large language models in an undergraduate programming course. *arXiv*, 2024.
[89] M. J. Parker, C. Anderson, C. Stone, and Y. Oh. A large language model approach to educational survey feedback analysis. *IJAIED*, 2024.
[90] S. R. Piccolo and P. Denny. Evaluating a large language model's ability to solve programming exercises from an introductory bioinformatics course. *PLOS Computational Biology*, 2023.
[91] K. Prakash, S. Rao, R. Hamza, J. Lukich, V. Chaudhari, and A. Nandi. Integrating llms into database systems education. In *DataEd*, 2024.
[92] J. Prather, P. Denny, J. Leinonen, B. A. Becker, I. Albluwi, M. Craig, H. Keuning, N. Kiesler, T. Kohn, A. Luxton-Reilly, S. MacNeil, A. Petersen, R. Pettit, B. N. Reeves, and J. Savelka. The robots are here: Navigating the generative ai revolution in computing education. In *ITiCSE-WGR*, 2023.
[93] J. Prather, P. Denny, J. Leinonen, D. H. Smith, B. N. Reeves, S. MacNeil, B. A. Becker, A. Luxton-Reilly, T. Amarouche, and B. Kimmel. Interactions with prompt problems: A new way to teach programming with large language models, 2024.
[94] B. Qureshi. Chatgpt in computer science curriculum assessment: An analysis of its successes and shortcomings. In *ICSLT*, 2023.
[95] N. Raihan, D. Goswami, S. Sayara Chowdhury Puspo, C. Newman, T. Ranasinghe, and M. Zampieri. Cseprompts: A benchmark of introductory computer science prompts. In *ISMIS*, 2024.
[96] J. Rajala, J. Hukkanen, M. Hartikainen, and P. Niemelä. "call me kiran" – chatgpt as a tutoring chatbot in a computer science course. In *Mindtrek*, 2023.
[97] S. Rasnayaka, G. Wang, R. Shariffdeen, and G. Neelakanta Iyer. An empirical study on usage and perceptions of llms in a software engineering project, 2024.
[98] M. Reiche and J. L. Leidner. *Bridging the Programming Skill Gap with ChatGPT: A Machine Learning Project with Business Students*. Springer Nature Switzerland, 2024.
[99] R. Rodriguez-Echeverría, J. D. Gutiérrez, and J. M. Conejero. Analysis of chatgpt performance in computer engineering exams. *IEEE-RITA*, 2024.
[100] M. Sánchez and A. Herrera. *Assessing ChatGPT's Proficiency in CS1-Level Problem Solving*. Springer Nature Switzerland, 2023.
[101] S. Sarsa, P. Denny, A. Hellas, and J. Leinonen. Automatic generation of programming exercises and code explanations using large language models. In *ICER*, 2022.
[102] F. Sarshartehrani, E. Mohammadrezaei, M. Behravan, and D. Gracanin. Enhancing e-learning experience through embodied ai tutors in immersive virtual environments: A multifaceted approach for personalized educational adaptation. In *CHI*, 2024.
[103] J. Savelka, P. Denny, M. Liffiton, and B. Sheese. Efficient classification of student help requests in programming courses using large language models, 2023.
[104] J. Savelka, A. Agarwal, and M. An. Thrilled by your progress! large language models (gpt-4) no longer struggle to pass assessments in higher education programming courses. In *ICER*.
[105] J. Savelka, A. Agarwal, C. Bogart, and M. Sakr. From gpt-3 to gpt-4: On the evolving efficacy of llms to answer multiple-choice questions for programming classes in higher education. In *Computer Supported Education*, 2024.
[106] A. Scholl, D. Schiffner, and N. Kiesler. Analyzing chat protocols of novice programmers solving introductory programming tasks with chatgpt, 2024.
[107] J. S. Sharpe, R. E. Dougherty, and S. J. Smith. Can chatgpt pass a cs1 python course? *J. Comput. Sci. Coll.*, 2024.
[108] B. Sheese, M. Liffiton, J. Savelka, and P. Denny. Patterns of student help-seeking when using a large language model-powered programming assistant. In *ACE*, 2024.
[109] T. Song, Q. Tian, Y. Xiao, and S. Liu. Automatic generation of multiple-choice questions for cs0 and cs1 curricula using large language models. In *Computer Science and Education*, 2024.
[110] A. Sterbini and M. Temperini. Automated analysis of algorithm descriptions quality, through large language models. In *ITS*, 2024.
[111] A. Strzelecki, K. Cicha, M. Rizun, and P. Rutecka. Acceptance and use of chatgpt in the academic community. *EAIT*, 2024.
[112] N. Binh Duong TA, H. Gia Phuc NGUYEN, and G. Swapna. Exgen: Ready-to-use exercise generation in introductory programming courses. In *ICCEC*. Asia-Pacific Society for Computers in Education, 2023.
[113] A. Tran, K. Angelikas, E. Rama, and C. Okechukwu. Generating multiple choice questions for computing courses using large language models. In *FIE*, 2023.
[114] X. Tu, J. Zou, W. J Su, and L. Zhang. What should data science education do with large language models. *arXiv*, 2023.
[115] A. Vadaparty, D. Zingaro, D. H. Smith, M. Padala, C. Alvarado, J. Gorson Benario, and L. Porter. Cs1-llm: Integrating llms into cs1 instruction, 2024.
[116] V. Venkatesh, V. Venkatesh, and V. Kumar. Evaluating copilot on cs1 code writing problems with suppressed specifications. In *COMPUTE*, 2023.
[117] H. Wan, H. Luo, M. Li, and X. Luo. Automated program repair for introductory programming assignments. *IEEE TLT*, 2024.
[118] T. Wang and D. Díaz. Exploring the role of ai assistants in computer science education: Methods, implications, and instructor perspectives. In *VL/HCC*, 2023.
[119] H. Wang, P. Qiang, H. Tan, and J. Hu. Enhancing image comprehension for computer science visual question answering. In *CVPR*, 2024.
[120] J. Wolfer. A qualitative assessment of chatgpt generated code in the computer science curriculum. In *Towards a Hybrid, Flexible and Socially Engaged Higher Education*, 2024.
[121] R. Xiao, L. Han, X. Zhou, J. Wang, N. Zong, and P. Zhang. Qacp: An annotated question answering dataset for assisting chinese python programming learners, 2024.
[122] C. Zastudil and M. and Rogalska. Generative ai in computing education: Perspectives of students and instructors. In *(FIE)*, 2023.
[123] Z. Zhang and Z. Dong. Students' perceptions and preferences of generative artificial intelligence feedback for programming, 2023.
[124] D. Zhang, Q. Cao, Y. Guo, and L. Wang. Assistant teaching system for computer hardware courses based on large language model. In *Computer Science and Education*, 2024.
[125] J. Bien and G. Mukherjee. Generative ai for data science 101: Coding without learning to code. *arXiv*, 2024.
[126] I. Estévez-Ayres, P. Callejo, and M. Ángel Hombrados-Herrera. Evaluation of llm tools for feedback generation in a course on concurrent programming. *IJAIED*, 2024.
[127] E. F. Gehringer, J. George Wang, and S. Kumar Jilla. Dual-submission homework in parallel computer architecture: An exploratory study in the age of llms. In *WCAE*, 2024.
[128] S. Jalil, S. Rafi, T. D. LaToza, K. Moran, and W. Lam. Chatgpt and software testing education: Promises & perils. In *ICSTW*, 2023.
[129] Y. Shen and X. Ai. Implications of chatgpt for data science education. In *SIGCSE*, 2024.
[130] Y. Zheng. Chatgpt for teaching and learning: An experience from data science education. In *SIGITE*, 2023.
[131] S. Grover. Teaching ai to k-12 learners: Lessons, issues, and guidance. In *SIGCSE*, 2024.
[132] M. Kazemitabaar and X. Hou. How novices use llm-based code generators to solve cs1 coding tasks in a self-paced learning environment. In *Koli Calling*, 2024.
[133] O. Petrovska, L. Clift, F. Moller, and R. Pearsall. Incorporating generative ai into software development education. In *CEP*, 2024.
[134] Z. Ahmed, S. Sadat Shanto, and A. Islam Jony. Potentiality of generative ai tools in higher education: Evaluating chatgpt's viability as a teaching assistant for introductory programming courses. *STEM Education*, 2024.
[135] L. Roest, H. Keuning, and J. Jeuring. Next-step hint generation for introductory programming using large language models. In *ACE*, 2024.
[136] B. Cipriano and P. Alves. Gpt-3 vs object oriented programming assignments: An experience report. In *ITICSE*, 2023.
[137] S. Mezzaro, A. Gambi, and G. Fraser. An empirical study on how large language models impact software testing learning. In *EASE*, 2024.
[138] B. Cowan, Y. Watanobe, and A. Shirafuji. Enhancing programming learning with llms: Prompt engineering and flipped interaction. In *ASSE*, 2024.
[139] H. McLean Babe, S. Nguyen, and Y. Zi. Studenteval: A benchmark of student-written prompts for large language models of code, 2023.
[140] T. Brown, B. Mann, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 2020.
[141] Mark Chen, Jerry Tworek, et al. Evaluating large language models trained on code. *arXiv*, 2021.
[142] J. Achiam, S. Adler, S. Agarwal, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
[143] R. Li, L. B. Allal, Y. Zi, N. Muennighoff, D. Kocetkov, C. Mou, M. Marone, C. Akiki, J. Li, J. Chim, et al. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*, 2023. URL https://arxiv.org/pdf/2305.06161.pdf.
[144] Z. Feng, D. Guo, et al. Codebert: A pre-trained model for programming and natural languages. In *EMNLP*, 2020.
[145] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.
[146] G. Penedo, Q. Malartic, and D. Hesslow. The refinedweb dataset for falcon llm: Outperforming curated corpora with web data, and web data only, 2023.
[147] The MosaicML NLP Team. Mpt-30b: Raising the bar for open-source foundation models, 2023.