AI-Tutoring in Software Engineering Education

Experiences with Large Language Models in Programming Assessments

Eduard Frankford
eduard.frankford@uibk.ac.at
University of Innsbruck
Innsbruck, Austria

Clemens Sauerwein
clemens.sauerwein@uibk.ac.at
University of Innsbruck
Innsbruck, Austria

Patrick Bassner
patrick.bassner@tum.de
Technical University of Munich
Munich, Germany

Stephan Krusche
krusche@tum.de
Technical University of Munich
Munich, Germany

Ruth Breu
ruth.breu@uibk.ac.at
University of Innsbruck
Innsbruck, Austria

ABSTRACT

With the rapid advancement of artificial intelligence (AI) in various domains, the education sector is set for transformation. The potential of AI-driven tools in enhancing the learning experience, especially in programming, is immense. However, the scientific evaluation of Large Language Models (LLMs) used in Automated Programming Assessment Systems (APASs) as an AI-Tutor remains largely unexplored. Therefore, there is a need to understand how students interact with such AI-Tutors and to analyze their experiences.

In this paper, we conducted an exploratory case study by integrating the GPT-3.5-Turbo model as an AI-Tutor within the APAS Artemis. Through a combination of empirical data collection and an exploratory survey, we identified different user types based on their interaction patterns with the AI-Tutor. Additionally, the findings highlight advantages, such as timely feedback and scalability. However, challenges like generic responses and students' concerns about a learning progress inhibition when using the AI-Tutor were also evident. This research adds to the discourse on AI's role in education.

KEYWORDS

Programming Education, Automated Programming Assessment Systems, Artificial Intelligence, ChatGPT, OpenAI, ChatBots

1 INTRODUCTION

The recent rise of artificial intelligence (AI) has resulted in transformative changes across various sectors. In healthcare, AI has enabled advanced diagnostics and personalized treatments [9]. In finance, algorithmic trading and fraud detection have been revolutionized [7] and the automotive industry is on the brink of a new era with the development of autonomous vehicles [34]. We have seen first applications of AI in the educational sector through Intelligent Tutoring Systems (ITS) [4]. ITS offer personalized learning experiences, yet their reliance on limited training data confines their applicability to specific scenarios [4]. This limitation not only escalates development costs but also restricts the scope and depth of feedback, thus hindering their broader adoption in diverse educational contexts.

Recently, with the introduction of ChatGPT, we have entered the age of accessible generative AI (GenAI) and large language models (LLMs). LLMs are trained on vast amounts of diverse data and can therefore generate nuanced, comprehensive, and context-aware feedback [21]. Beyond just unit test feedback, LLMs like OpenAI's GPT-3.5-Turbo or GPT-4 have the potential to recognize a broader spectrum of student mistakes and offer tailored guidance. Such capabilities can bridge the gap in the shortcomings of traditional ITSs and expand the horizon for feedback mechanisms within programming education. While the integration of LLMs into various tools and sectors is well-documented, its specific application in programming education, especially, in the form of an AI-Tutor within APASs, remains mostly unexplored.

To address this gap, we seek to address the following research questions:

(1) **RQ1:** What is the nature of student interaction with Automated Programming Assessment Systems when facilitated by an AI-Tutor?
(2) **RQ2:** How do students experience AI driven feedback in Automated Programming Assessment Systems?
(3) **RQ3:** What are the lessons learned after implementing and operating an AI-Tutor within an Automated Programming Assessment System?

As a first step, the primary objective is to explore the effectiveness and implications of integrating an AI-Tutor based on GenAI, specifically OpenAI's GPT-3.5-Turbo model, into an APAS [13]. This approach combines empirical data collection with an exploratory survey. As part of the empirical investigation we closely monitored the AI-tutor's usage, student interactions, code submissions, and feedback timings. Additionally, we analyzed code changes between submissions to understand student engagement patterns with the AI-Tutor. The preliminary findings suggest that the AI-Tutor offers unique benefits, but there is still a long way to fully optimize the student learning experience.

The remainder of this paper is structured as follows: Section 2 provides an overview of related work. Section 3 elaborates on the research techniques. Section 4 presents the main findings of this study, which are further discussed in Section 5. Section 6 outlines potential constraints of this study, and

we conclude in Section 7, summarizing the main insights and reflecting on the broader implications of this research.

2 RELATED WORK

ITSs have long been a subject of interest in the realm of programming education [1]. These systems are generally designed to deliver instructional content in a way that is tailored to individual learners, adapting to a student's needs [4]. There have been experiments proving that these systems show similar effects like human tutoring [18]. As a result, many ITSs have been created for programming education [1, 3, 10]. Adaptive or intelligent feedback is a common feature, but this feedback is mainly generated by extensive unit testing [4].

Beside unit testing, the application of machine learning to emulate human feedback is no recent advancement, as the first chat-bot has been introduced over 50 years ago [2]. Since then, these chat-bots have become more and more intelligent [17, 32]. However, they are normally trained on questions the creators expect users to ask, but this is changing with the introduction of ChatGPT [5].

Rudolph et al. did one of the first extensive literature reviews on ChatGPT and focused on its relevance for higher education, especially on student assessment, student learning and teaching [26]. They found that with ChatGPT it is now possible to simulate the assistance provided by a tutor, such as providing personalised assistance in solving problems. Furthermore, Ray focused on the applications of ChatGPT across various domains and found among other things that it has potential in personalizing learning, by analyzing data on students' learning preferences, strengths, and weaknesses [25]. Kasneci et al. discuss the opportunities and challenges when using generative AI tools like ChatGPT in education [12]. They point out the opportunity to provide personalized feedback to students.

Literature also already documents the effective use of ChatGPT in improving source code. For example, Surameery and Shakor explored the use of ChatGPT to solve programming bugs [30]. To be precise, they examined how they can leverage the model to provide debugging assistance, bug prediction and bug explanation to help solve programming problems. They conclude that ChatGPT can play an important role in solving programming bugs, but it is not a perfect solution and should be seen as an additional debugging tool. Sobania et al. analyzed the automatic bug fixing performance of ChatGPT using the bug fixing benchmark set, QuixBugs [28]. They found that ChatGPT's bug fixing performance is notably better than other state of the art approaches being able to solve 31 out of 40 bugs. Other researchers, like Ouh et al. and Tian et al., conducted empirical analyses of ChatGPT's potential as a programming assistant focusing on code generation, program repair, and code summarization [22, 31]. Tian et al. found that ChatGPT can hint surprisingly well to the original intention behind what a correct version of a program should look like [31].

Pardos and Bhandari concentrated on comparing the efficacy of hints authored by human tutors and hints generated by ChatGPT for elementary and intermediate Algebra [23]. They found that 79% of hints produced by ChatGPT passed a manual quality test. Additionally, Lo conducted research to decide on how ChatGPT performs in different subject domains [21] and found that ChatGPT overall performance regarding programming was outstanding to satisfactory [29]. However, regarding "Software Testing", it was able to answer 55.6% of the questions partially correctly [11].

Industry has also recognized the value of generative AI, with EdTech organizations developing AI-based solutions to help students with their coursework and giving ideas for lessons to educators [16]. Kshetri found that Quizlet launched an AI-Tutor Q-Chat, which combines ChatGPT with Quizlet's educational content library [16]. Furthermore, Khan Academy also started using AI to create a chat-bot, based on the GPT-4 model, with the goal in mind that students can use it to ask for assistance without the tool revealing the solution, but helping them solve the exercise [16].

The related work section shows that existing literature has explored traditional ITSs and the general capabilities of ChatGPT in various domains, and industry has already implemented sophisticated tools using generative AI. However, an exploratory understanding of its practical application as an AI-Tutor within APASs is still missing. The empirical studies have primarily focused on the model's ability to debug, generate code, and provide hints, and therefore, were able to state that large language models can be used for specific tasks, like tutoring. However, none have really implemented such a system and therefore, the real student experience, interaction patterns, and perceptions when using ChatGPT as an AI-Tutor in APASs have not been scientifically investigated. This study seeks to offer a scientific evaluation on the integration a GPT-based AI-Tutor in APASs, demonstrating that its tutoring capabilities, as proposed in literature, can be realized in practice.

3 METHODOLOGICAL APPROACH

This study is guided by a set of research questions (RQ1, RQ2 and RQ3) that have been defined in Section 1. To address these research questions, we implemented a three-stage methodological approach:

(1) **Integration of the AI-Tutor within an APAS:** Initial integration into the Artemis platform [15, 19].
(2) **Practical application by students:** Students solved a specific programming task on the platform.
(3) **Exploratory survey:** A survey targeting students of the "Introduction to Programming" course at the University of Innsbruck to collect their experiences with the AI-Tutor.

The following sections explain the details of each of these stages.

3.1 Integration of the AI-Tutor within an APAS

We have implemented the AI-Tutor to collect data. This included developing a prototype that integrates the APAS Artemis [14, 15] with the GPT-3.5-Turbo model of OpenAI. We have chosen Artemis as the APAS for this study because of several reasons:

(1) **Open Source**: Artemis is available as an open source project on GitHub, which makes this research reproducible.[1]

(2) **Functional Scope**: Artemis provides all the basic features necessary for an APAS, including automatic exercise evaluation via test driven feedback, which improves the external validity of the findings [27].

(3) **Online Editor**: Artemis allows students to solve exercises online via a built-in code editor. This made the implementation of the AI-Tutor and data collection easier.

(4) **Large User Base**: Artemis is used by more than ten different universities, like the TU Munich and University of Innsbruck, and is therefore used by thousands of students every semester. Therefore, improving the Artemis APAS is directly beneficial for a large user base.

Before the integration of the AI-Tutor the workflow to use Artemis for programming exercises was the following [15]:

(1) **Instructors prepare an exercise**: Mainly involves the creation of an exercise description, the creation of a template file, the creation of a sample solution and the creation of unit tests to test the code submitted by the students.

(2) **Students solve an exercise**: Students write code to solve the problem statement using the integrated online editor offered by the platform. When students submit a solution attempt, the code of the submission is stored in a version control system. For this study, it was GitLab.

(3) **System returns feedback**: For each submission, a build pipeline is triggered that executes the test cases written by the instructors and returns the test results with individual messages as feedback to the students.

The integration of the AI-Tutor extended this workflow by an additional possibility to request feedback from the AI-Tutor. This extended workflow has been depicted in Figure 1.

In Figure 2, the Artemis code editor is shown with the new possibility to request AI feedback by clicking the "View AI Feedback" button displayed on the top right.

Once this request is send to the server, the current solution of the student, the exercise description and sample solution are retrieved from the APAS' database and an API-call to the OpenAI servers containing the following information is sent:
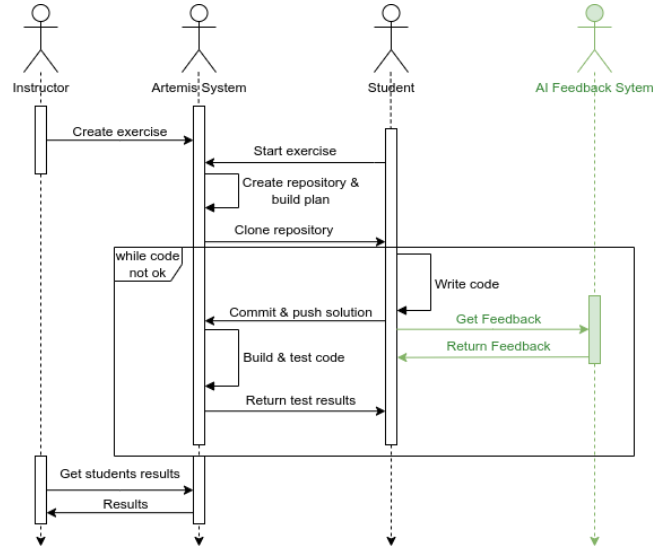
---

[1] https://github.com/ls1intum/Artemis

Figure 1: Sequence diagram of the usage workflow of the Artemis system extended by the AI-Tutor functionality.
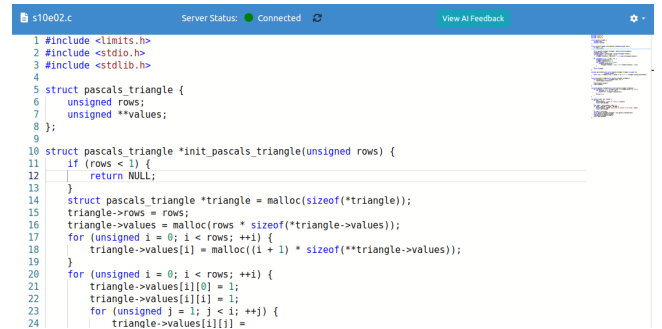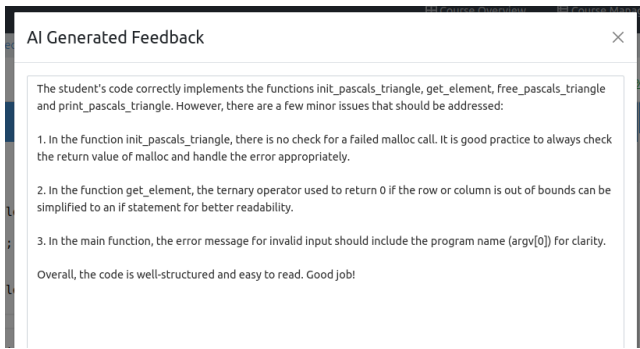

Figure 2: Artemis code editor with the button to "View AI Feedback".

(1) **Model**: The requested LLM, which is in our case GPT-3.5-Turbo.

(2) **Message**: The prompt which the LLM should take into consideration. This prompt can be seen in Listing 1.

(3) **Temperature**: We set the temperature at 0.7 to balance predictability and creativity in the LLM's responses. This level ensures relevant feedback with sufficient variability for exploring diverse solutions.[2]

Once having sent this API-call to the OpenAI servers, we extracted the response of the LLM and displayed it in a pop-up window without any further modifications. This can be seen in Figure 3. All the code files needed to adapt Artemis to display this possibility can be found in Figshare[3].

---

[2] https://platform.openai.com/docs/guides/text-generation/how-should-i-set-the-temperature-parameter

[3] https://figshare.com/s/636a9c5ff8f2c8315f26

Figure 3: AI Generated Feedback displayed in a pop up.

The language model receives the following prompt depicted in Listing 1:

Listing 1: GPT-3.5-Turbo Prompt

```
Act as a programming tutor and give informal
feedback in language to the student.
The exercise description is the following:
    description
The students code looks like that at the moment
    : current
Do not provide a code solution.
The optimal solution should look like that:
    solution
Important: Do not provide code.
```

The prompt starts with the main instruction to tell the LLM to act as a tutor. `language` is the current selected language in Artemis. This value can either be English or German.

`description` is the the task to be solved by the student. For this study the students had to implement Pascal's triangle and the task was to implement the functions to generate, display and release the memory for a portion of Pascal's triangle [8]. In addition to the task description, the students were given a starting template with method stubs to start the exercise with.

We have chosen the Pascal's triangle exercise to test the AI-Tutor because:

(1) **Foundational programming constructs**: The implementation of Pascal's triangle touches upon many foundational programming concepts such as loops, conditionals, arrays, and in some languages, dynamic memory management. If an AI can give valuable feedback on this exercise, it indicates its capability to understand and instruct on tasks involving the foundational concepts mentioned before.

(2) **Algorithmic thinking**: The process for creating Pascal's triangle involves iterative and recursive thinking. This showcases the AI's capability to handle a diverse range of algorithmic challenges as many other algorithm problems involve similar patterns of thought.

(3) **Concept overlap**: Many problems in computer science and mathematics share concepts with Pascal's triangle, e.g., the binomial expansion and combinatorics. A successful tutoring here indicates the AI's potential to generalize its capabilities to related problems.

(4) **Versatility in problem complexity**: Pascal's triangle can be approached in multiple ways. If the AI-tutor can manage the range of solutions for this problem, it suggests its robustness in tutoring exercises with different levels of complexity.

(5) **Debugging and problem solving**: Common mistakes are possible in implementing Pascal's triangle. An AI-tutor's ability to diagnose and correct these signifies its potential to generalize this capability to other programming challenges.

The last two parts of the prompt, `current` and `solution`, represent the current solution of the student and the optimal solution defined by the exercise creator, respectively.

To integrate the AI-Tutor we have chosen to not enable direct interactions with the model because of the following reasons:

(1) **Easier usability**: The assumption was that predetermining the prompt would simplify the user experience. By eliminating the chat-bot-style interaction, we sidestepped the necessity for students to formulate a question, thus streamlining their interactions.

(2) **Controlled environment**: A predefined interaction model provides a more controlled setting, thereby simplifying measures taken to avoid students from receiving the solution for the exercise via prompt engineering [20].

(3) **Quality assurance**: With a static model, we were able to ensure that the AI-Tutor offers consistent pedagogically sound feedback, which is in line with the course's learning objectives.

(4) **Data privacy**: Direct interactions could inadvertently lead students to input sensitive or personal information. A static model minimizes this risk, adhering better to data privacy standards.

(5) **Resource efficiency**: Direct, dynamic interactions with the system may consume more resources, because the chat history should be given as context to enable meaningful conversations. Therefore, leading to higher costs as more tokens are used.

(6) **Reduction of over-reliance**: By limiting direct interactions, students were encouraged to think critically and not over-rely on the AI-Tutor for every minor query or challenge.

When the student presses the button "View AI Feedback" we store the following information in the database:

(1) **Code**: The current solution of the student.
(2) **Feedback**: The feedback returned by the LLM.
(3) **User**: A user identifier to identify each request.
(4) **File**: The file on which the student is working on.
(5) **Timestamp**: The current time.

3.2 Exploratory Survey

We selected students from the "Introduction to Programming" tutorial as subjects for the experiment. This tutorial is part of the Bachelor in Computer Science curriculum at the University of Innsbruck and teaches first year students the basics in the programming language C. For this experiment a total of 23 students actively participated. While this may seem like a modest sample size, it's important to note that the qualitative nature of this analysis allowed for a more in-depth understanding of individual experiences, making the size not only manageable but also advantageous and given their recent interactions with traditional, human tutors, these students were especially appropriate subjects for assessing an AI-Tutor.

Prior to the data collection and survey implementation, we introduced the students to the possibility of receiving feedback from the newly implemented AI-Tutor via Artemis.

In the subsequent tutorial, the students were tasked with solving the, before described, Pascal's Triangle task [8]. They had one week to solve the task. While they were solving the exercise they were free to choose whether they used the new AI-Feedback functionality or not. However, when they pressed the "View AI-Feedback"-button we stored the "AI Feedback data" (Code at feedback time, feedback returned by the LLM, User, File and Timestamp) in the database and when they submitted their current code to the Artemis system their current solution, test results and timestamp were saved in the version control system connected to Artemis.

Finally, for the next course, we allotted approximately 15 minutes for the students to complete a questionnaire. In the survey we asked the following questions based on the Technology Acceptance Model (TAM)[6]. TAM is a theoretical model that includes two primary factors that determine an individual's intention to use a technology: (1) Perceived Ease of Use (PEOU) and (2) Perceived Usefulness (PU). The model has been widely adopted in various fields to understand and predict the acceptance of newly implemented features.

(1) **I find the AI-Tutor easy to use**: This is directly related to the PEOU dimension of TAM. It seeks to collect the respondents' perceptions about the ease of interface and interaction with the AI-Tutor.

(2) **Using the AI-Tutor for my tasks enables me to accomplish the tasks more quickly**: This question is mainly about the efficiency offered by the AI-Tutor, which can be seen as a subset of PU as it implies the benefit of time-saving.

(3) **Using the AI-Tutor improves my performance**: This touches on the PU dimension by gauging whether the users feel they perform better in their tasks due to the AI-Tutor.

(4) **Using the AI-Tutor for my tasks increases my productivity**: Again, this is a question about PU. By increasing productivity, the AI-Tutor is seen as adding value to the user's.

(5) **Using the AI-Tutor makes it easier to do my tasks**: This question is about both PEOU and PU.

On one hand, it assesses ease of task accomplishment (PEOU), and on the other, it speaks to the utility value of the AI-Tutor (PU).

(6) **I find the AI-Tutor useful**: This is a direct reflection of the PU dimension, asking the respondent to evaluate the overall usefulness of the AI-Tutor.

To obtain additional feedback, we asked the following open questions:

(1) What challenges did you encounter when utilizing the AI-Tutor?

(2) Do you have any further suggestions on how the AI-Tutor could be improved?

Both questions aim to uncover specific difficulties or obstacles that users have faced, helping to identify specific areas for improvement in the design or functionality of the AI-Tutor. Lastly, we asked questions about their demographics, including their highest degree, their current semester and their programming experience.

3.3 Data Analysis

The data analysis involved first the combination of two datasets: The data saved when AI-Feedback was requested and the data saved when the students submitted their solutions. The "AI Feedback" dataset provided insights into the code at feedback time, feedback from the AI model, user details and timestamps. The student submissions included the code at submission time, the test results and the respective timestamps.

For accuracy, students who did not solve the exercise and did not engage with the AI-Tutor were excluded from the qualitative analysis allowing for an assessment of a total of 12 participants. This analysis was designed to identify patterns, and insights from the student responses, ensuring a comprehensive understanding of their experiences with the AI-Tutor. This analysis included:

(1) **Temporal Coding:** Students' submissions and feedback request times were identified and marked in different colors to identify interaction patterns of students.

(2) **Thematic Coding:** Students' responses from the open-ended questions were initially read and re-read to identify common themes and patterns.

(3) **Theme Development:** The patterns were grouped under broader thematic categories, and a narrative was constructed around each theme. This involved interpreting the data within the context of this study's research questions.

Key insights derived from this qualitative analysis were essential in understanding the intricacies of the student interactions and experiences.

4 RESULTS

In the following, we present the results of the conducted analysis. We divided this section into three subsections, each addressing a specific research question.

4.1 Student Interaction

Overall, the following interaction patterns emerged from the analysis of the data. Four students neither made submissions to Artemis nor sought feedback from the AI-Tutor. One student made a single submission to Artemis without asking for any feedback from the AI-Tutor. In contrast, a different student sought feedback from the AI-Tutor once, yet did not submit anything to Artemis. Two students made several submissions to Artemis without seeking feedback from the AI-Tutor. Different patterns were observed, with one student displaying each of the following behaviors: making a single submission to Artemis and seeking feedback from the AI-Tutor once, making multiple submissions to Artemis and asking for feedback from the AI-Tutor once, and making a single submission to Artemis while seeking feedback from the AI-Tutor multiple times. It is particularly noteworthy that 12 students worked intensively with both systems, uploaded numerous submissions to Artemis and frequently asked for feedback from the AI tutor.

Considering its significance, we primarily focus on the behavior of the 12 students who exhibited high interaction rates with both Artemis and the AI-Tutor. Figure 4 illustrates the timestamps when the students asked the AI-Tutor or submitted their solution to the APAS. On the Y-Axis, each line corresponds to a student and the X-Axis can be interpreted as a timeline starting with 2023-05-23 and ending with 2023-05-29. The red points in this figure indicate the time at which a student submitted their code to APAS. indicate the exact time at which a student requested feedback from the AI-Tutor. This figure shows that there are mainly two different ways in which students interact with the AI tutor. Based on this timeline, we were able to derive two user personas.

4.1.1 Continuous Feedback - Iterative Ivy.
Iterative Ivy represents students who utilize the AI-Tutor intensively before their initial submission to the APAS. These students often begin without a complete solution and turn to the AI-Tutor for guidance on understanding and solving the exercise. The AI-Tutor, in its capacity, guides through specific instructions encompassing aspects like function implementation, memory management, and value calculation. Over multiple feedback cycles, students refine their solutions. When the AI-Tutor's feedback shifts towards minor optimizations, students tend to transition to submitting their work to the APAS, aiming for a perfect score.

4.1.2 Alternating Feedback - Hybrid Harry.
Hybrid Harry exemplifies students who alternate between the AI-Tutor feedback and APAS submissions throughout their coding process. Typically, they begin their tasks by seeking initial insights from the AI-Tutor even before submitting a solution. Some send repeated requests for feedback on the same code segment, indicating potential uncertainties or the need for more explicit guidance. These students tend to submit their work to the APAS after establishing a foundation of their

code. Notably, the AI-Tutor recognized incomplete or non-functional implementations, which students corrected after being told so by the AI-Tutor.

> **Main Findings for RQ 1**
>
> We identified two user personas: (1) Continuous Feedback - Iterative Ivy, who relied mainly on AI feedback before final submissions to APAS, and (2) Alternating Feedback - Hybrid Harry, who alternately used the AI-Tutor and APAS submissions throughout the process.

4.2 Student Experience

Figure 5 represents the distribution of user responses based on the questionnaire defined in Section 3. The responses are presented as horizontal stacked bars. Each bar represents a different statement, and the segments of the bar represent the proportion of responses for each level of agreement. The position of the bars along the X-axis reflects the average sentiment of the responses, ranging from negative on the left to positive on the right. The zero point serves as a reference for interpreting Figure 5. If the majority of a bar lies to the left of this point, it generally indicates a more negative sentiment. Conversely, if it is situated to the right of the zero point, the sentiment is predominantly positive. Like this, the figure allows us to easily visualize how users perceive the AI-Tutor and its benefits. Examining this figure we found that reactions were mixed, ranging from positive to equally negative. However, the polarized responses appear to neutralize one another, resulting in a largely neutral overall response.

Mapping the Likert scale from -3 to +3, using 0 as a neutral midpoint. The average sentiment for each answer is the following:

- I find the AI Tutor easy to use: Somewhat Agree (1.29).
- Using the AI-Tutor for my tasks enables me to accomplish the tasks more quickly: Neutral (-0.29).
- Using the AI-Tutor improves my performance: Neutral (-0.43).
- Using the AI-Tutor for my tasks increases my productivity: Neutral (-0.43).
- Using the AI-Tutor makes it easier to do my tasks: Neutral (0.14).
- I find the AI-Tutor useful: Neutral (0.14).

Only the statement "I find the AI Tutor easy to use" received an other than neutral average response, indicating a mild agreement.

Regarding the open questions about challenges encountered while using Artemis and its AI-Tutor, student feedback consistently touched on several main themes:

(1) **Desire for Greater Specificity:** The AI-Tutor's responses were perceived as too generic. Students preferred more context-specific feedback pointing directly to improvement areas in the code.
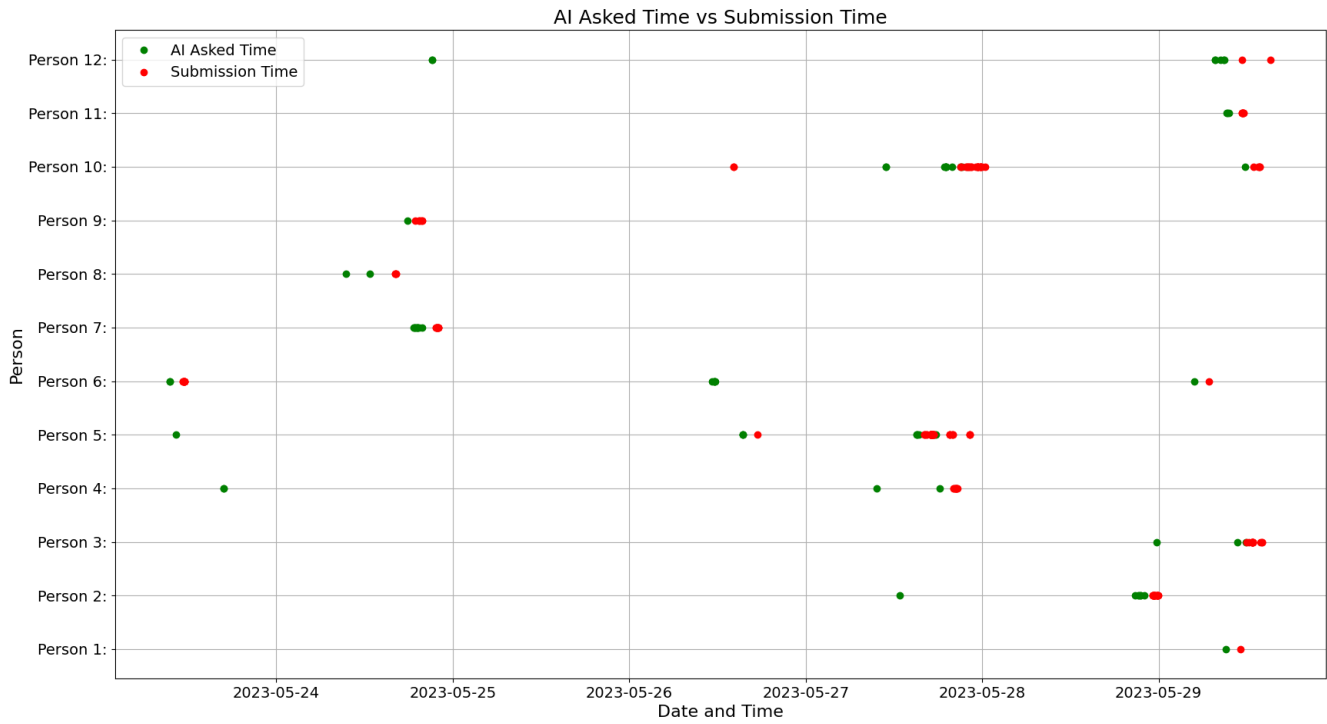
Figure 4: This figure shows the times when each student asked the AI-Tutor or submitted a code solution to the APAS.
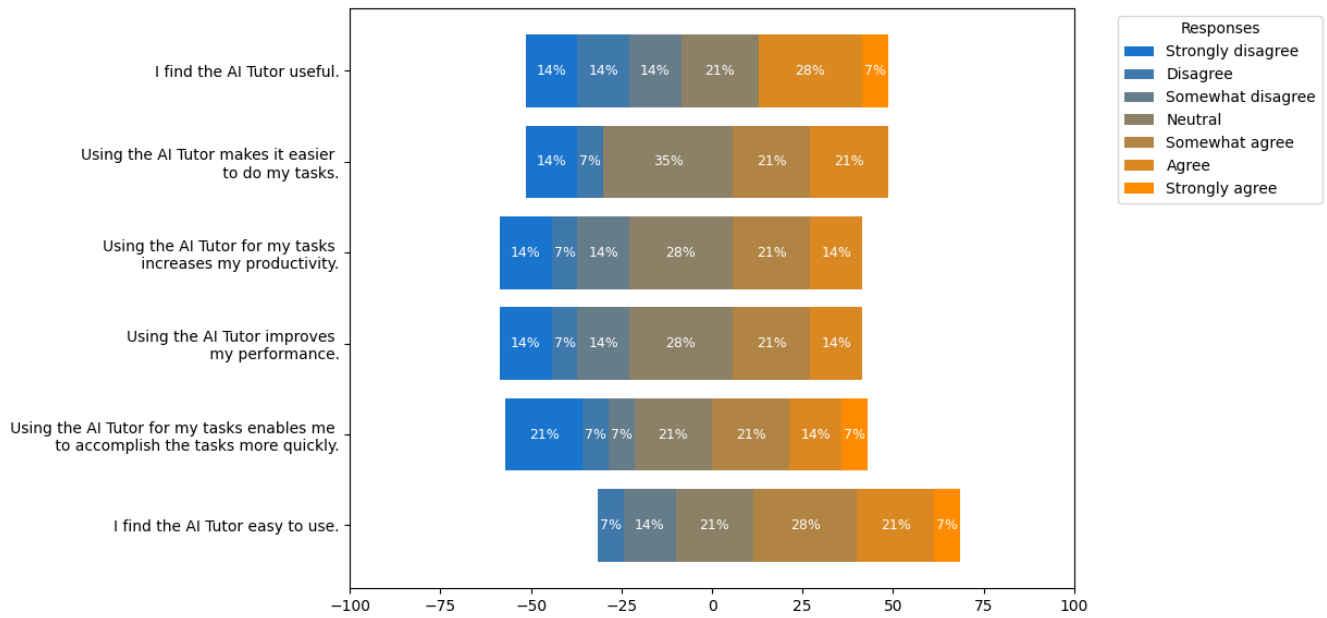

Figure 5: Students' satisfaction with the AI Tutor.

(2) **Request for Increased Interactivity and Interface Concerns:** Students expressed the wish for enhanced interactive capabilities with the AI-Tutor, such as the ability to ask follow-up questions after initial feedback. Additionally, the interface was criticized because there was no possibility to see old feedback because once the feedback window was closed one could only request new feedback.

(3) **Demand for Concrete Examples:** To supplement the written feedback, students believed that concrete code examples would help them interpreting the AI's suggestions.

(4) **Apprehension about Learning Inhibition:** Some students feared that using the AI-Tutor might lead to over-reliance which would slow down their learning progress.

In terms of general feedback, students mentioned the system's potential, but also that it is notably perceived as an early-stage prototype. Furthermore, they compared its current utility to rudimentary software aids, expressing hope for more refined, context-aware feedback in future iterations.

Main Findings for RQ 2

Some students found the system useful others stated the opposite resulting in an overall neutral result regarding the TAM. However, answers to the open-questions revealed that students, which gave mostly negative responses found the feedback to be too generic and lacking concrete examples.

4.3 Lessons Learned

The practical integration of a large language model into an APAS offered valuable insights into the system's strengths and weaknesses. Through this experience, several key lessons and actionable insights can be derived.

A key lesson learned is that the AI-Tutor exhibits the capacity for real-time, personalized feedback provision. We have found that the system tends to return a more high level explanation of the task, if the students had not yet written a lot of code. When the student has already written much code that is mostly correct, then the system tends to start giving recommendations on how to improve the code quality. For example, to add comments explaining the code or to change ternary operators to if-statements for better readability. An other surprising insight is that the AI-Tutor was able to give feedback on logical and semantic issues. We found that if students had defined wrong boundary conditions to terminate a loop, then the AI-Tutor recognized this and proposed to the student to change this condition. This immediate feedback helped students to quickly correct their errors and thereby mitigating the acquisition of poor coding practices. Additionally, the system's inherent ability to serve feedback to a large, diverse student population in real-time underscores its applicability in large-scale educational contexts, particularly in Massive Open Online Courses (MOOCs).

However, we also learned that AI-Tutoring does not come without its challenges. The AI-Tutor, while efficient, occasionally delivered only general feedback, which means that there's room to refine its responses for more detailed, code-specific guidance. Analyzing the feedback provided by the AI-Tutor we found that from 75 feedback requests 55(66.6%) were useful and 20(26.6%) were categorized as not useful. Among the 20 not useful responses we found that three revealed the solution of the exercise to be solved, four answers were hallucinations and 13 were too general to be helpful in the students situation. Mostly, if the answers were too general, we found that the AI-Tutor explained the exercise to the student even-though his or her solutions was already very sophisticated. The hallucinations were mainly about the AI-Tutor stating that a function looks well implemented, although there was no student implementation there yet or that a function should be implemented that was already implemented by the student.

Additionally, as the existing system did not allow any interaction, we learned that enhancing its capability to address follow-up queries would improve the learning experience. Regarding the operational dependency, we found that downtimes in the API could jeopardize the tutor's functionality.

We also found that it is important to address students' over-reliance concerns, encouraging them to use the AI-Tutor without the fear that their learning progress might be hindered. Additionally, the feedback quality might get compromised due to context limits of models like GPT-3.5-Turbo. Exploring ways to manage this limitation effectively will be beneficial. Despite careful prompt crafting, there were instances where the AI-Tutor revealed solutions. Ensuring that the model maintains adherence to the guidelines is a pivotal lesson.

Main Findings for RQ 3

Implementing an AI-Tutor in an APAS showed that AI can support human tutors, allowing them to focus on deeper personal interactions. However, improvements are needed, as the AI's feedback was effective only 66.6% of the time, being too generic, revealing solutions, or incorrect. Additionally, some students worry that using the AI-Tutor may slow their learning progress.

---

5 DISCUSSION

In this study we found mainly two usage strategies adopted by students when interacting with the AI-Tutor. It is important to understand these user personas, as they provide insights that can inform the design of AI-powered educational systems to be able to fit different learning styles and strategies. First of all, we defined the user persona called Iterative Ivy. Users assigned to this persona first used the AI-Tutor intensively and only when the solution was already very advanced started to submit their solutions to the system. This approach seems to favor a traditional, linear programming strategy: (1) Comprehending the problem, (2) Writing

a complete solution, and then (3) Validating the solution. By seeking continuous feedback from the AI-Tutor, these students ensured they were progressing on the right track before submitting their final solution. This finding suggests that AI-Tutors are beneficial to students who prefer to seek guidance and validation throughout their learning process, rather than just at the end. However, a potential concern here could be an over-reliance on the AI-Tutor. The continuous feedback-seeking behavior may stem from uncertainty or lack of confidence, which needs to be addressed in further pedagogical planning. Secondly, we defined the user persona Hybrid Harry. Whose strategy contrasts sharply with Iterative Ivy's approach. Users assigned to this persona alternated between seeking AI-Tutor feedback and submitting solutions to the APAS. These students opted for an iterative learning approach, which represents a more agile programming practice. This suggests that AI-Tutors and APAS can facilitate active, self-regulated learning. However, the risk here lies in the potential for students to rely too heavily on the test feedback to guide their work, which could inadvertently lead to a trial-and-error approach to solve an exercise, rather than understanding the core principles.

Furthermore, given the responses to the TAM questions we found that students have mixed feelings regarding the usefulness of the AI-Tutor. While some students do not appreciate the help of the AI-Tutor others do appreciate it. It is important to identify the exact reasons why this is the case, but a first analysis indicates that the main reasons for the negative responses are user interface and prompt related. Students who responded negatively to the TAM questions also complained about the inability to ask follow-up questions and that the system did not return code examples or that the feedback is too general. The problem regarding the inability to ask follow up questions can be solved by changing to a chat-bot based system. The second problem regarding the missing code examples can be addressed by changing the prompt to allow code examples as responses. As a result, it is reasonable to conclude that GPT-3.5-Turbo can be successfully used as a language model behind an AI-Tutor.

Addressing fears that AI-Tools may inhibit learning success is also crucial. Reiterating the tool's purpose to supplement rather than supplant traditional learning methods may decrease such concerns.

Additionally, the use of AI-Tutors in programming education, as seen in this research, presents a unique set of learned lessons. Regarding ITS we have found that LLMs offer a distinct adaptability advantage, because in conventional ITS, altering feedback mechanisms often demands intricate changes in the system's codebase, which can be time-consuming and resource-intensive. However, with LLMs, modifications are primarily done through prompt engineering. Given their vast training data, refining or adjusting the prompts can quickly adapt the feedback the model provides, without needing to change its internal mechanics. This allows educators to swiftly adapt to changing educational needs or methodologies.

Among the advantages of AI-Tutors over traditional human tutors, the promptness of feedback provided by AI-Tutors stands out as a game-changer. The capacity to instantly identify and correct errors can be vital for students learning programming, because quick feedback can minimize the propagation of misunderstandings and bad coding practices. Moreover, the scalability and cost-effectiveness of an AI-Tutor makes it a compelling choice, especially in resource-limited settings or with a large student base.

However, these benefits come with their share of challenges. An aspect of the AI-Tutor that needs attention is its occasional inclination to "hallucinate", producing responses that might not be entirely accurate or relevant. In this study, this primarily manifested when a student had already implemented a correct solution, resulting in the LLM sometimes advising the student to refine a program that was already functioning correctly. A potential mitigation strategy could involve integrating the AI feedback with unit test results. This would inform the student when their solution meets all criteria, signaling that subsequent AI feedback may not be entirely accurate.

The dependence on API availability and the inherent token limit of models like GPT-3.5-Turbo add an additional layer of complexity. Any change or downtime in the API can hinder the AI-Tutor's operations, and the context limitation imposed by the token limit can affect the quality of feedback, especially for more complex or lengthy code submissions.

A more psychological perspective brings forth concerns about the impact of AI-Tutors on students' learning progress and the lack of a personal touch. An over-reliance on the AI-Tutor might impede students from developing their problem-solving skills, as they might rely too heavily on instant feedback rather than trying to debug and solve problems themselves. Additionally, the impersonal nature of an AI-Tutor might make the learning experience less engaging and less adaptive to individual student needs, which could affect motivation and learning outcomes.

These challenges should not overshadow the immense potential that an AI-Tutor holds. By addressing the issues mentioned, we can create more sophisticated and effective AI-Tutors that can significantly enhance the educational experience and outcomes for students learning programming. The journey towards optimizing AI-Tutors for programming education is still in progress, but the destination seems promising.

6 LIMITATIONS

In this paper we mainly considered the following four categories of validity, also used by [33]: (1) Construct validity, (2) Reliability, (3) Internal validity and (4) External validity.

6.1 Construct Validity

The research questions were defined using the PICOC system [24]. The PICOC framework provides a systematic way to formulate research questions by emphasizing five elements: Population, Intervention, Comparison, Outcome, and Context.

This structured approach ensures that research questions are both comprehensive and relevant. For this study:

(1) **RQ1** primarily addresses the Population, Intervention, Outcome, and Context by examining the nature of student interactions within the specific context of an APAS assisted by an AI-Tutor.

(2) **RQ2** focuses on the Population, Intervention, and Outcome by probing the students' experiences with AI-driven feedback when guided by the AI-Tutor.

(3) **RQ3** encompasses all the PICOC elements, especially Context, by analyzing the broader lessons learned from deploying an AI-Tutor within the APAS environment.

The research questions were further refined through discussions with several experts in the field to ensure alignment with the topic of interest. Leveraging the PICOC system as a foundation, coupled with the structured data collection approach and exploratory survey, facilitated a thorough answering of RQ1–3.

6.2 Reliability

We conducted a systematic data collection and analysis approach, as detailed in Section 3. Therefore, the process is both transparent and reproducible. However, it's crucial to note that the use of GPT-3.5-Turbo introduced a variable element. Given the nature of LLMs, not every prompt produces identical responses on different occasions. As a result, while the core structure and methodology can be reproduced, there may be slight variations in the responses generated by the model across different replications of the experiment.

6.3 External Validity

One potential limitation in this domain arises from the fact that we integrated the AI-Tutor only into Artemis. However, a systematic comparison of various APASs confirmed that Artemis' basic functionalities are echoed in many other APASs, deeming it a representative system [27]. Additionally, the integration of the AI-Tutor can be done platform independent, because the approach stays the same, as it should be possible on all APAS to integrate a pop-up window that displays the results of the REST API calls.

Another potential threat to external validity is the use of a GPT model as the foundation for the AI-Tutor. While the used model is a state-of-the-art LLM and exhibits advanced conversational abilities, it might not perfectly mimic every possible LLM's behavior. Nonetheless, given that the model is based on the same foundational architectures as most other prevalent LLMs, and shares many of their characteristics and capabilities, we argue that the findings related to GPT-3.5-Turbo can largely be extrapolated to other similar models. It serves as a representative example, providing insights that are likely applicable across various LLMs.

Furthermore, the total number of participants can influence the external validity. In this study 23 students from the course "Introduction to Programming" participated. This sample size is too small to conduct a statistically significant

quantitative analysis. As a result, we decided to focus on a qualitative analysis and report the experiences of implementing and operating an AI-Tutor. This allows for a deeper exploration of the students' experiences and behaviors when using the system. Last but not least, given the students' recent engagements with traditional human tutors, they were especially well-suited to evaluate the AI-Tutor.

6.4 Internal Validity

This study study largely leaned on qualitative analysis, which can sometimes introduce subjective bias. Nevertheless, the methodological rigor employed aimed to minimize such biases. The detailed procedures involved in the qualitative analysis have been outlined in the research methodology section. By closely following these methodological steps, we have aimed to ensure that the findings are both credible and trustworthy.

7 CONCLUSION

In this study of integrating the model behind ChatGPT as an AI-Tutor into the Artemis APAS, we uncovered both the immense potential and challenges of such an application. While the AI-Tutor offered advantages like timely feedback and scalability, its limitations were apparent. These included occasionally generic feedback, lack of interactive dialog, operational vulnerabilities related to API availability, potential over-reliance by students, the absence of a human touch, and technical constraints like context limits.

The vast potential of AI-Tutors in programming education is undeniable, but careful implementation and ongoing refinement are essential. This exploration underscores the need for more research in this domain, balancing technological progress with the irreplaceable human aspect of education.

Future work should focus on enhancing AI-Tutor's feedback specificity, interactivity improvements, user interface refinements, and addressing the token limit and prompt engineering challenges. The potential exploration of more powerful models like GPT-4 may further improve the feedback quality. This study's findings serve as a foundation for continued research in this innovative intersection of AI and education.

8 DATA AVAILABILITY

The data supporting the findings of this study are openly available in Figshare [4]. The dataset comprises the following:

(1) **Data Analysis ANONYM.xlsx**:
   (a) *Sheet 1*: Contains extracted data from the database, such as Code, Feedback, User, Time, as well as various descriptive statistics, detailing, for instance, the frequency with which each user consulted the AI-Tutor, submissions to the APAS, and the final score.
   (b) *Sheet 2*: Houses the responses from the qualitative survey.
   (c) *Sheet 3*: Features an analysis that groups submissions and the state of the code when querying the

---

[4] https://figshare.com/s/636a9c5ff8f2c8315f26

AI-Tutor. It assesses the quality of the feedback and observes code alterations post-feedback.

(2) **Student submissions to the version control system**: Comprises multiple anonymized folders, each storing the code a student uploaded to the system. This code is augmented at the end with annotations detailing if the student had previously consulted the AI-Tutor and, if so, the associated timestamps.

(3) **Artemis-Files**: Contains all essential files to be integrated into your public Artemis project to activate the AI-Tutor functionality. These files are designed to work with the open-source Artemis project.[5] A comprehensive repository has not been released due to challenges associated with its anonymization.

It is essential to note that all personal identifiers have been removed to maintain confidentiality and adhere to data protection principles.

9 ACKNOWLEDGMENTS

REFERENCES

[1] J. R Anderson and E. Skwarecki. 1986. The automated tutoring of introductory computer programming. *Commun. ACM* 29, 9 (sep 1986), 842–849. https://doi.org/10.1145/6592.6593

[2] Luka Bradeško and Dunja Mladenić. 2012. A survey of chatbot systems through a loebner prize competition. In *Proceedings of Slovenian language technologies society eighth conference of language technologies*, Vol. 2. sn, 34–37.

[3] PL Brusilovsky. 1992. Intelligent tutor, environment and manual for introductory programming. *Educational & Training Technology International* 29, 1 (1992), 26–34.

[4] Tyne Crow, Andrew Luxton-Reilly, and Burkhard Wuensche. 2018. Intelligent tutoring systems for programming education. In *Proceedings of the 20th Australasian Computing Education Conference*. ACM. https://doi.org/10.1145/3160489.3160492

[5] Marian Daun and Jennifer Brings. 2023. How ChatGPT Will Change Software Engineering Education. In *Proceedings of the Conference on Innovation and Technology in Computer Science Education V. 1*. 110–116.

[6] Fred D. Davis. 1985. *A Technology Acceptance Model for Empirically Testing New End-User Information Systems: Theory and Results*. Ph. D. Dissertation. Massachusetts Institute of Technology, Sloan School of Management.

[7] Awishkar Ghimire, Surendrabikram Thapa, Avinash Kumar Jha, Surabhi Adhikari, and Ankit Kumar. 2020. Accelerating business growth with big data and artificial intelligence. In *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*. IEEE, 441–448.

[8] Andreas M Hinz. 1992. Pascal's Triangle and the Tower of Hanoi. *The American mathematical monthly* 99, 6 (1992), 538–544.

[9] Dean Ho, Stephen R Quake, Edward RB McCabe, Wee Joo Chng, Edward K Chow, Xianting Ding, Bruce D Gelb, Geoffrey S Ginsburg, Jason Hassenstab, Chih-Ming Ho, et al. 2020. Enabling technologies for personalized and precision medicine. *Trends in biotechnology* 38, 5 (2020), 497–518.

[10] Jay Holland, Antonija Mitrovic, and Brent Martin. 2009. J-LATTE: a Constraint-based Tutor for Java. (2009).

[11] Sajed Jalil, Suzzana Rafi, Thomas D LaToza, Kevin Moran, and Wing Lam. 2023. Chatgpt and software testing education: Promises & perils. In *2023 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 4130–4137.

[12] Enkelejda Kasneci, Kathrin Sessler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, Stephan Krusche, Gitta Kutyniok, Tilman Michaeli, Claudia Nerdel, Jürgen Pfeffer, Oleksandra Poquet, Michael Sailer, Albrecht Schmidt, Tina Seidel, Matthias Stadler, Jochen Weller, Jochen Kuhn, and Gjergji Kasneci. 2023. ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and Individual Differences* 103 (apr 2023), 102274. https://doi.org/10.1016/j.lindif.2023.102274

[13] Hieke Keuning, Johan Jeuring, and Bastiaan Heeren. 2016. Towards a systematic review of automated feedback generation for programming exercises. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*. 41–46.

[14] Stephan Krusche. 2021. *Interactive learning - A scalable and adaptive learning approach for large courses*. Habilitation. Technische Universität München.

[15] Stephan Krusche and Andreas Seitz. 2018. Artemis: An automatic assessment management system for interactive learning. In *Proceedings of the 49th ACM technical symposium on computer science education*. 284–289.

[16] Nir Kshetri. 2023. The Economics of Generative Artificial Intelligence in the Academic Industry. *Computer* 56, 8 (aug 2023), 77–83. https://doi.org/10.1109/mc.2023.3278089

[17] Mohammad Amin Kuhail, Nazik Alturki, Salwa Alramlawi, and Kholood Alhejori. 2023. Interacting with educational chatbots: A systematic review. *Education and Information Technologies* 28, 1 (2023), 973–1018.

[18] James A Kulik and JD Fletcher. 2016. Effectiveness of intelligent tutoring systems: a meta-analytic review. *Review of educational research* 86, 1 (2016), 42–78.

[19] Matthias Linhuber, Jan Philip Bernius, and Stephan Krusche. 2023. Constructive Alignment in Modern Computing Education: An Open-Source Computer-Based Examination System. In *23nd Koli Calling International Conference on Computing Education Research* (Koli, Finland) *(Koli '23)*. https://doi.org/10.35542/osf.io/nmpf6

[20] Yi Liu, Gelei Deng, Zhengzi Xu, Yuekang Li, Yaowen Zheng, Ying Zhang, Lida Zhao, Tianwei Zhang, and Yang Liu. 2023. Jailbreaking chatgpt via prompt engineering: An empirical study. *arXiv preprint arXiv:2305.13860* (2023).

[21] Chung Kwan Lo. 2023. What is the impact of ChatGPT on education? A rapid review of the literature. *Education Sciences* 13, 4 (2023), 410.

[22] Eng Lieh Ouh, Benjamin Kok Siew Gan, Kyong Jin Shim, and Swavek Wlodkowski. 2023. ChatGPT, Can You Generate Solutions for my Coding Exercises? An Evaluation on its Effectiveness in an undergraduate Java Programming Course. *arXiv preprint arXiv:2305.13680* (2023).

[23] Zachary A Pardos and Shreya Bhandari. 2023. Learning gain differences between ChatGPT and human tutor generated algebra hints. *arXiv preprint arXiv:2302.06871* (2023).

[24] Mark Petticrew and Helen Roberts. 2008. *Systematic reviews in the social sciences: A practical guide*. John Wiley & Sons.

[25] Partha Pratim Ray. 2023. ChatGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope. *Internet of Things and Cyber-Physical Systems* (2023).

[26] Jürgen Rudolph, Samson Tan, and Shannon Tan. 2023. ChatGPT: Bullshit spewer or the end of traditional assessments in higher education? *Journal of Applied Learning and Teaching* 6, 1 (2023).

[27] Clemens Sauerwein, Simon Priller, Martin Dobiasch, Stefan Oppl, Michael Felderer, and Ruth Breu. 2023. Lecturers' and Students' Experiences with an Automated Programming Assessment System. (2023).

[28] Dominik Sobania, Martin Briesch, Carol Hanna, and Justyna Petke. 2023. An analysis of the automatic bug fixing performance of chatgpt. *arXiv preprint arXiv:2301.08653* (2023).

[29] Petra Stutz, Maximilian Elixhauser, Judith Grubinger-Preiner, Vivienne Linner, Eva Reibersdorfer-Adelsberger, Christoph Traun, Gudrun Wallentin, Katharina Wöhs, and Thomas Zuberbühler. 2023. Ch (e) atgpt? an Anecdotal Approach on the Impact of Chatgpt on Teaching and Learning Giscience. *Preprint) https://doi. org/10.35542/osf. io/j3m9b* (2023).

[30] Nigar M Shafiq Surameery and Mohammed Y Shakor. 2023. Use chat gpt to solve programming bugs. *International Journal*

---

of Information Technology & Computer Engineering (IJITC) ISSN: 2455-5290 3, 01 (2023), 17–22.

[31] Haoye Tian, Weiqi Lu, Tsz On Li, Xunzhu Tang, Shing-Chi Cheung, Jacques Klein, and Tegawendé F Bissyandé. 2023. Is ChatGPT the Ultimate Programming Assistant–How far is it? arXiv preprint arXiv:2304.11938 (2023).

[32] Rainer Winkler and Matthias Söllner. 2018. Unleashing the potential of chatbots in education: A state-of-the-art analysis. In Academy of Management Proceedings, Vol. 2018. Academy of Management Briarcliff Manor, NY 10510, 15903.

[33] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. 2012. Experimentation in software engineering. Springer Science & Business Media.

[34] Caiming Zhang and Yang Lu. 2021. Study on artificial intelligence: The state of the art and future prospects. Journal of Industrial Information Integration 23 (2021), 100224.