Fast and Fleeting: Evaluating ChatGPT's Impact on Students' Computational Thinking Skills

1st May Mahmoud
New York University Abu Dhabi
Abu Dhabi, United Arab Emirates
m.mahmoud@nyu.edu

2nd Eric Asare
New York University Abu Dhabi
Abu Dhabi, United Arab Emirates
eric.asare@nyu.edu

3rd Nisa Shahid
New York University Abu Dhabi
Abu Dhabi, United Arab Emirates
ns5376@nyu.edu

4th Nourhan Sakr
The American University in Cairo
Cairo, Egypt
n.sakr@columbia.edu

5th Sarah Nadi
New York University Abu Dhabi
Abu Dhabi, United Arab Emirates
sarah.nadi@nyu.edu

Abstract

With the growing accessibility of generative AI tools, students are increasingly using them in coursework. However, their impact on developing core computational thinking skills remains unclear. This paper investigates whether access to ChatGPT during learning affects students' short- and long-term transfer of two essential computational thinking skills, specifically, decomposition and abstraction. In a controlled experiment, we divide students into two groups; one with ChatGPT assistance during the learning phase and one using traditional resources. Participants complete learning tasks and testing tasks in a study session, followed by a retention test within a week. We measure task completion time and solution correctness. We also collect post-study questionnaire responses to assess confidence, perceived task difficulty, and experience with ChaGPT. We analyze screen recordings and ChatGPT logs to understand usage patterns. Our findings show ChatGPT provided fast but fleeting advantages, helping students complete tasks faster, perceive them as less challenging, and have better initial awareness of their performance. However, these gains did not carry over for long-term, suggesting that the tool acted more as a crutch than a lasting learning aid.

Keywords

AI4SE, Computer Science Education, Software Engineering Education, Computational Thinking, Generative AI

ACM Reference Format:
1st May Mahmoud, 2nd Eric Asare, 3rd Nisa Shahid, 4th Nourhan Sakr, and 5th Sarah Nadi. 2025. Fast and Fleeting: Evaluating ChatGPT's Impact on Students' Computational Thinking Skills. In . ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Computational thinking (CT) entails expressing solutions in structured, executable forms; decomposing complex problems, reasoning across multiple levels of abstraction, and automating procedures via algorithmic design—skills that are foundational to the design and implementation of robust systems [30]. In addition to programming skills, successful software engineers need to master computational thinking [12]. While computer science and software engineering curricula typically try to cultivate these skills, the rise of generative AI raises a key question for education: *how does its usage influence students' development of computational thinking skills?*

With the spread of generative AI tools such as ChatGPT [17], students increasingly leverage them to expedite coursework completion. Prior work on generative AI in software engineering and computer science education has largely examined effects on programming performance, especially in introductory courses [11, 22]. However, the influence of generative AI on computational thinking among these students remains underexplored. Given concerns about students' over-reliance on these tools [10], it is critical to determine whether generative AI use impedes the development of competencies central to effective software engineering.

To address this gap, we design a controlled experiment using object-oriented design and programming (OOP) tasks, a natural setting for computational thinking, as they require abstraction and decomposition in system modeling. In addition to immediate task completion and correctness, we examine whether exposure to generative AI affects *retention* of these skills, assessed via a delayed, post-test without AI support.

Our main research question is: *does having access to ChatGPT when learning to solve object-oriented design tasks help or hinder students' ability to learn abstraction and decomposition skills?*

We address this research question by analyzing both quantitative and qualitative data. We analyze task completion time and solution correctness across the learning phase, testing phase, and retention-test. We examine the participants' confidence in their solutions, the complexity of the tasks, and their confidence in applying these skills in future problems. We also investigate the participants' interaction with ChatGPT and their prompting patterns. We balance our participants across two groups: a control group that only has access to web search with no AI-assistance, and an experiment group that has access to ChatGPT during the learning phase.

Our results show that students with access to ChatGPT completed the initial tasks faster, saving about 8.89% in median completion time across the learning and testing phases. However, this advantage disappeared in the retention test, where they took 6.67% longer than the control group. While the experiment group achieved higher correctness during learning, the control group matched or

Conference'17, Washington, DC, USA
2025. ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

outperformed them in later testing and retention tasks. On an individual level, the experiment group performance declined over time, whereas the control group showed gradual improvement. Although participants using ChatGPT viewed it positively, their confidence became less aligned with actual performance over time, in contrast to the control group, whose self-awareness improved.

2 Background

Computational thinking (CT) was first introduced by Jeannette Wing [30], describing a set of thinking skills and approaches that are built around the "power and limits" of the computing process [30]. These are essential skills that software engineers develop in order to formulate solutions. Computational thinking skills consist of a subset of skills, among which are abstraction, decomposition, recursive thinking, algorithmic thinking, and the ability to generalize and transfer solutions to a wide variety of problems [30].

In our work, we focus only on abstraction and decomposition as two representative computational skills. *Abstraction* is the process of focusing on the essential parts of a problem, showing the essential, relevant features while hiding complex implementation details. Abstraction allows clearer modeling and representation of complex systems, making a problem more tractable [30]. It also enables software engineers to reason about systems at multiple levels of detail. *Decomposition* focuses on breaking down large and complex problems or systems into smaller, more manageable parts that can be more easily understood, developed, or solved independently [30].

Generative Artificial Intelligence (GenAI) is a type of artificial intelligence where models create new content such as text, code, or images in response to a prompt. GenAI is built on *Large Language Models (LLMs)*, which are deep neural networks based on the transformer architecture [29], trained on large amounts of data from several sources. LLMs may be used through an interactive chat interface such as OpenAI's ChatGPT[1],Google's Gemini[2], and Anthropic's Claude[3], allowing users to interact with the LLM using natural language prompts.

When considering the process of learning a skill, we need to distinguish between two types of learning: short-term and long-term transfer. *Short-term transfer of learning* deals with the application of a newly acquired skill or knowledge shortly after the learning experience or instruction (also referred to as near transfer)[3, 6, 9, 16, 27]. *Long-term transfer of learning* refers to the application of skills and knowledge after a significant delay, for example after a week or a month of learning, indicating a deeper and more stable skill (also referred to as far transfer) [3, 6, 9, 16, 27]. In this paper, we consider both aspects of learning in our experiment. To consider the short-term transfer of learning, we test students for their learning within the study session. We then invite students to take a retention test within a week of their session to assess the long-term transfer of learning. We discuss more details of the experiment design in Section 3.

[1]https://chatgpt.com/
[2]https://gemini.google.com/
[3]https://claude.ai/

3 Experiment Overview

3.1 Purpose

The purpose of our experiment is to investigate the effect of using GenAI tools such as ChatGPT when students learn to decompose and abstract a given natural language problem statement into classes, as opposed to using traditional online resources. We use ChatGPT as it was the most frequently used GenAI tool among our participants.

3.2 Recruitment and Participants

3.2.1 Inclusion Criteria. We recruited students from one university[4]. We consider students with sufficient programming skills and limited knowledge of advanced topics like software design and engineering in our inclusion criteria. We define our inclusion criteria to be limited to students who have taken the introductory programming course (either from the Computer Science or Computer Engineering programs) and have not yet taken more advanced courses that go into depth about topics like object-oriented programming and software engineering.

3.2.2 Recruitment. We announced the study by posting flyers in different locations on campus where undergraduate students usually congregate and by advertising in specific introductory and elective courses. Interested students filled out an online screening survey where we asked about their programming background, their grades in the introductory programming courses, whether they were enrolled in or have completed more advanced courses, their proficiency in various languages including Python, and their use of generative AI in their coursework. We filter sign-ups based on the inclusion criteria and reach out to eligible participants to schedule a study session. As compensation for their time, we offer each participant an Amazon gift card of a suitable amount approved by the ethics board[5].

3.2.3 Ethical Considerations. We applied and obtained our university's ethics board's approval. The study is considered to pose minimal risk to participants. We get informed consent from each participant at the beginning of the study. We inform the students of the purpose of the study, and that they can withdraw at any time and take a break at any time during the session. We ensure that the data collected is anonymized. We also conducted a pilot study to confirm that the session was not too long and the tasks were not too complex to avoid fatigue.

3.3 Experiment Setup

We design our experiment as a between-subjects controlled experiment where each participant receives only one treatment. We divide participants into two groups. One group has access to ChatGPT (*experiment group*) while the other does not (*control group*), but has access to Google web search with no AI functionality. We illustrate the workflow of our experiment in Figure 1. The experiment consists of a study session and a post-study retention test session, with the intention of mimicking a self-study session followed by an assessment. We next describe the setup of these two sessions,

[4]university name will be disclosed in the camera-ready version to adhere to double anonymous review requirements.
[5]exact amount and currency will be disclosed in the camera-ready version.

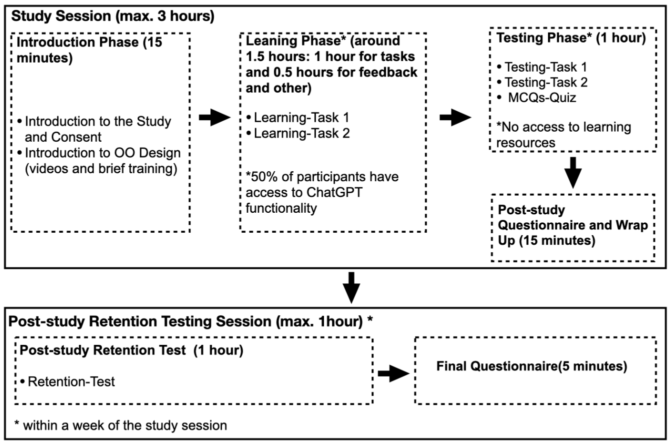

Study Session (max. 3 hours)
Introduction Phase (15 minutes)
• Introduction to the Study and Consent
• Introduction to OO Design (videos and brief training)
Leaning Phase* (around 1.5 hours: 1 hour for tasks and 0.5 hours for feedback and other)
• Learning-Task 1
• Learning-Task 2
*50% of participants have access to ChatGPT functionality
Testing Phase* (1 hour)
• Testing-Task 1
• Testing-Task 2
• MCQs-Quiz
*No access to learning resources
Post-study Questionnaire and Wrap Up (15 minutes)
Post-study Retention Testing Session (max. 1hour) *
Post-study Retention Test (1 hour)
• Retention-Test
Final Questionnaire(5 minutes)
* within a week of the study session

Figure 1: Experiment Workflow: Participants sit through a study session for a maximum of 3 hours, followed by a retention test session, within a week of the study session. In the study session, they go through an introductory phase, a learning phase where participants in the experimental group have access to ChatGPT, and then a testing phase and a post-study questionnaire.

then Section 3.3.4 provides more information on the specific tasks used. We chose Python as the target language for the tasks, as it is the one taught in the introductory computer science course.

The study session lasts for three hours. At the beginning of the study session (*introduction phase*), the researcher starts by getting the participant's consent and giving them a brief on the study setup. The researcher informs the participant that they need to approach the tasks with the aim of learning the skills. They also inform participants that they will do testing tasks and a retention test later. The researcher then gives the participant a brief lecture on decomposition and thinking in abstraction. This lecture includes an example and an educational video on abstraction, as well as another video on writing classes in Python.

The participant then moves on to the *learning phase*, where they solve two learning tasks. We share the tasks with the participant as a Google Colab[6] notebook, since it facilitates online sharing and storage. For participants in the experimental group, we have ChatGPT open in a split-screen view. We use our own ChatGPT account and clear the account data before each study session. We set the ChatGPT model to GPT-4o. After a participant is done with each of the learning tasks, we show them the answer key to these tasks as a form of feedback. This mimics self-study activities, where students would solve exercises and check their answers against a model answer. At the end of the learning phase, we export and store the chat logs from the ChatGPT session.

Once done, the participant moves on to the *testing phase* where they (1) attempt two testing tasks on their own (regardless of which treatment they receive) and (2) answer a Multiple-Choice Question Quiz (MCQ-Quiz) without any assistance. At the end of the study session, the participant completes a post-session questionnaire, which collects their confidence in their solution and information about their experience solving the tasks. Both groups answer the same questions about their experience with the tasks, but the experiment group answers additional questions regarding their interaction with ChatGPT.

Within a week, the participant takes a *post-study retention test*, in which they work on a task similar to the one they solved in the study session, but without access to any assistive tools. We chose the week time frame for the retention test instead of longer periods for the feasibility of implementation, and in line with previous retention studies [11]. Participants can sign up for the retention test anytime during the week following their initial study session. Sign-ups open on the Monday immediately after their session week, ensuring at least a two-day gap between the original session and the retention test (e.g., participants attending a session on Friday will have the weekend as a minimum gap before the test). At the end of the retention test, the participant takes a brief final questionnaire that asks them about their confidence in their retention test solution, and whether they prepared for the test.

3.3.1 Experimental Block Assignment. We use a *matched-pair design* [4, 14] to ensure that the participants in the two groups are similar in terms of their programming experience and background. We match participants into pairs based on their years of programming, Python proficiency, and grade in the Introduction to Programming course. Each paired participant is then randomly assigned to the experiment or control group. Participants without a matched pair are randomly assigned to either the control or experiment group to balance the number of participants in each group.

3.3.2 Independent and Dependent Variables. Our *independent variable* in the experiment is access to ChatGPT during the learning phase of the study. Our *dependent variables* are the student's solution correctness based on an answer key and a predefined rubric (scored out of a maximum of 116 points), time to complete the tasks, their confidence in their solution, and their skills retention (graded retention test based on an answer key and a pre-defined rubric with a maximum of 60 points). To avoid researcher bias affecting the grades, we have two researchers grade the tasks and discuss any disagreements. We report agreement rates in our results. To come up with a single score for each task, we calculate the mean of the two graders' scores and consider it the final score.

3.3.3 Hypothesis. Our H_o (null hypothesis) states that there is no significant difference in task completion time or scores between the experiment and control groups. The H_A (alternative hypothesis) is that the ChatGPT-assisted group will differ significantly in these measures.

3.3.4 Tasks. We have designed several tasks to teach and test participants' decomposition and abstraction skills. In the learning phase, the participant first solves a simple task, **Learning-Task1**, to learn the simple abstraction and decomposition involved in creating a class in Python and adding attributes and methods. In this first task, we provide the participant with the description of a simple `Car` class and ask them to implement it in Python as per the description. We allocate a maximum of 15 minutes for Learning Task 1. Then, the participant works on **Learning-Task2**, which is the main design task. The participant works on decomposing and abstracting a problem statement of a university student information system into classes with attributes and methods, and implementing

[6]https://colab.research.google.com/

QID-Topic	Question	Group Asked
Q1-Identifying Classes	How challenging did you find identifying relevant classes for the system?	Control & Experiment
Q2-Identifying Attributes and Methods	How challenging did you find identifying relevant attributes and methods of each class?	Control & Experiment
Q3-Designing Classes	How confident were you in designing each class for the system?	Control & Experiment
Q4-Designing Attributes and Methods	How confident were you in designing the attributes and methods for each class?	Control & Experiment
Q5-Abstraction Improvement	Did the task help you improve your ability to abstract a problem by identifying relevant components (classes)?	Control & Experiment
Q6-Future Skill Application	Do you feel confident that you can apply the skills learned in this task to solve a similar problem in the future?	Control & Experiment
Q7-Ease of Using ChatGPT	How easy was using ChatGPT to complete the task?	Experiment
Q8-ChatGPT Solutions Confidence	How confident were you in the solutions provided by ChatGPT?	Experiment
Q9-Time Saved with ChatGPT	Did ChatGPT help you save time when completing the task?	Experiment
Q10-ChatGPT Helped Identify Classes	"Using ChatGPT helped me identify relevant classes more effectively." Do you agree with this statement?	Experiment
Q11-ChatGPT Helped Design Attributes/Methods	"ChatGPT assisted me in designing class attributes and methods." Do you agree with this statement?	Experiment
Q12-ChatGPT Improved Abstraction	"I feel that ChatGPT improved my ability to think about abstraction in problem-solving." Do you agree with this statement?	Experiment
Q13-Confidence Without ChatGPT	Do you feel confident you could design similar classes and relationships without ChatGPT in the future?	Experiment
Q14-Challange Manually Abstracting	How challenging was manually abstracting the problem into classes after using ChatGPT?	Experiment
Q15-ChatGPT's Future Help	Do you believe using ChatGPT will help you solve similar tasks using classes and objects in the future?	Experiment
Q16-Recommend ChatGPT	"I recommend using ChatGPT to solve programming tasks that require abstraction skills to my peers." Do you agree with this statement?	Experiment

Table 1: List of Post-study Questionnaire Questions

the skeleton of each class in Python, including the attributes and methods. We allocate a maximum of 45 minutes for the second task.

In the testing phase, both groups work on **Testing-Task1**, first modifying the `Car` class from Learning-Task1 to add a `mileage` attribute and two methods to update and increment it. In the second task **Testing-Task2**, the participant works on modifying a basic implementation of the university information system from Learning-Task2 to add some new requirements. The task requires the student to update the system to keep track of grades and student GPA (which was not included in the learning task requirements). This will require students to decide on adding new classes, as well as editing already available classes to account for this new requirement. Participants also answer (**MCQs-Quiz**), which focuses on object-oriented design. Some questions ask about general concepts of objects and classes, while others provide a short system description and ask them to choose from a list of classes or attributes to best abstract the system. Finally, the participant fills out a post-session questionnaire that helps us understand their confidence in their solution and gain more insights into their experience solving the tasks. We list the questions of the questionnaire in Table 1.

For the **Retention-Test**, we ask participants to attempt a similar design task to Learning-Task2 and Testing-Task2, where they design an event management system, but without access to any assistance. We provide all details regarding the tasks, MCQs, and study workflow in our online resource, which includes our replication package, analysis scripts, and anonymized data: https://figshare.com/s/859073b2e4eaa338ff68.

3.3.5 Data Collection. We collect the following data from each participant: (1) *Time taken* to complete each task (Each task from the study session and post-study retention test session). (2) The *correctness* of each task solution based on a pre-defined rubric. (3) The *answers to MCQs-Quiz*. (4) Answer to the *post-study session questionnaire*. (5) *Screen recording* of each session with no audio or camera capture of the participants. (6) *Chat log history* with ChatGPT if the student was in the experiment group.

3.3.6 Pilot Study. We ran two pilot study sessions to assess the study session's duration and the complexity of the tasks. The pilot helped us confirm that the study workflow and data collection process were effective. We used our observations and findings from the pilot study to add more instructions to clarify the tasks' instructions. The data from the pilot study are not included in the final data reported in this paper.

4 Results

We recruited 21 participants for the study, 11 participants for the control group, and 10 participants for the experiment group. However, one participant from the control group and one from the experiment group did not complete the retention test, leaving us with data from 19 participants for the retention test. We first describe participant demographics and the group assignment before presenting our results in terms of task completion time and task correctness. We then discuss the post-study questionnaire reporting on the participants' perceived complexity of the tasks and confidence. Finally, we report our analysis of participants' interactions with ChatGPT and their prompting patterns.

To report statistically significant differences, we use a Mann-Whitney U Test[23] for data that is not normally distributed and a T-Test [23]d for normally distributed data. We use the chi-squared test for categorical data [23]. We set our p-value to be <= 0.05 for statistical significance.

4.1 Demographics and Group Assignments

We summarize the participants' demographics in Table 2. Our sample's demographics align well with the typical profile of novice Computer Science or Software Engineering students who are still developing their computational thinking skills. The majority of participants are in their early academic years (freshmen and sophomores constitute around 71% of the participants), and most of them are enrolled in CS or related fields. The majority have been programming for a few years (around 75% of the participants have been programming for 2 years or less). The gender distribution is consistent with what is observed for a computer science students population (around 62.5% of the participants are male, while 37.5% are female) [2]. The majority of participants (54.2%) listed ChatGPT as their top choice of GenAI tool, which is the GenAI tool we use in our experiment.

Based on our matched pair design to balance the participants' background across the two groups, we were able to balance 9 pairs of the 21 participants (18 participants), then randomly assigned

PID	Age	Gender	Year of Study	Field	Years Prog.	Python Prof.	Top GenAI
P1	19	Male	Freshman	Undeclared intending CS	less than 1 year	3	ChatGPT (OpenAI)
P2	19	Female	Freshman	Computer Science	less than 1 year	4	GitHub Copilot
P3	18	Female	Freshman	Computer Science	less than 1 year	3	ChatGPT (OpenAI)
P4	19	Male	Freshman	Computer Science	less than 1 year	4	Claude
P5	20	Male	Sophomore	Computer Science	1-2 years	3	ChatGPT (OpenAI)
P6	19	Male	Sophomore	Undeclared intending EE	More than 3 years	4	ChatGPT (OpenAI)
P7	20	Male	Junior	Data Science and mathematics	1-2 years	3	ChatGPT (OpenAI)
P8	-	Male	Senior	Minor CS	1-2 years	3	GitHub Copilot
P9	21	Female	Senior	Minor CS	less than 1 year	2	ChatGPT (OpenAI)
P10	20	Male	Sophomore	Computer Engineering	More than 3 years	3	Cursor
P11	21	Male	Sophomore	Computer Engineering	less than 1 year	3	ChatGPT (OpenAI)
P12	22	Male	Senior	Minor CS	2-3 years	4	ChatGPT (OpenAI)
P13	20	Male	Sophomore	Computer Science	1-2 years	3	ChatGPT (OpenAI)
P14	20	Female	Junior	Electrical Engineering	1-2 years	3	Claude
P15	19	Female	Sophomore	Computer Science	1-2 years	2	Perplexity
P16	19	Female	Sophomore	Minor CS	1-2 years	3	ChatGPT (OpenAI)
P17	18	Male	Freshman	Computer Science	More than 3 years	5	Perplexity
P18	19	Male	Freshman	Computer Science	less than 1 year	4	ChatGPT (OpenAI)
P19	23	Male	Senior	Minor CS	1-2 years	4	ChatGPT (OpenAI)
P20	18	Male	Freshman	Computer Science	2-3 years	3	-
P21	19	Male	Freshman	Computer Science	More than 3 years	3	Deepseek

Table 2: Participants Demographics

each participant from the pair to a group. For the remaining 3 unmatched participants, we randomly assigned them to either group to balance group sizes; two ended up in the control group and one in the experiment group. We use a Chi-squared test to confirm that there are significant differences between the groups regarding gender (p=1.000), year of study (p=0.628), and years of programming (p=0.628). Using a Mann-Whitney U Test, we also confirm no differences between the groups in regards to their proficiency in Python (p=0.285) and their grade in Introduction to Computer Science (p=0.320). Overall, we do not find any statistically significant difference between the background of the two groups.

4.2 Completion Time

Figure 2 shows the distribution of time taken to complete each task across the two participant groups. Since the allocated time for each task is different, we show the proportion of allocated time taken by each participant for easier comparison.

Overall, we find that the experiment group tends to finish the tasks faster than the control group, generally taking less of the allocated time for the tasks. For example, we can see a clear difference in completion time in Learning-Task1 where the experiment group finished much faster than the control group (36.67% vs 46.67% of the allocated time). Recall that in Learning-Task 1 and Learning-Task2, participants in the experiment group had access to ChatGPT to perform the tasks. Learning-Task1 was particularly simple and explains why almost all students in the experiment group finished very quickly. On the other hand, while the experiment group also finishes tasks faster in the somewhat more challenging Learning-Task2, the difference between the two groups is smaller (85.56% vs 91.11% of the allocated time).

We see the same patterns for Testing-Task1 and Testing-Task2, where the experiment group finishes the easier Testing-Task1 much faster than the control group, while the difference is less in Testing-Task2. Recall that both groups solve these testing tasks with no assistance. On average the experiment group saved around 9% of allocated time in the study sessions tasks.

When it comes to the retention test, we observe the opposite results. We find that the control group took on average *less* time than the experiment group (median of 43.33% of allocated time for control vs. 50.00% for experiment), taking around 7.00% more time. However, none of the differences in time taken that we observe above are statistically significant.

Observation 1: During the study session, students with access to ChatGPT (experiment group) finished all tasks faster than students without access (control group), saving on average around 9% of the allocated time. However, during the retention test, students who did not have access to ChatGPT during learning (control group) finished the tasks faster than the experiment group (experiment group took on average 7% more of the allocated time).

4.3 Task Correctness

Recall that we have two different authors grading each task from the study session as well as the retention test. To compare the different scores, we calculate the mean absolute difference (MAD) [15], which is the average of the absolute differences between the two graders' scores for each task, where each score is a percentage (i.e., normalized by the total possible points). Overall, the two graders were in high agreement across tasks, with a mean absolute difference of 2.092% points out of the total possible points (100%). To measure agreement, we also calculate the Intraclass Correlation Coefficient where the value ranges from 0 to 1, where 1 means perfect agreement and 0 means no agreement [26]. The ICC captures both

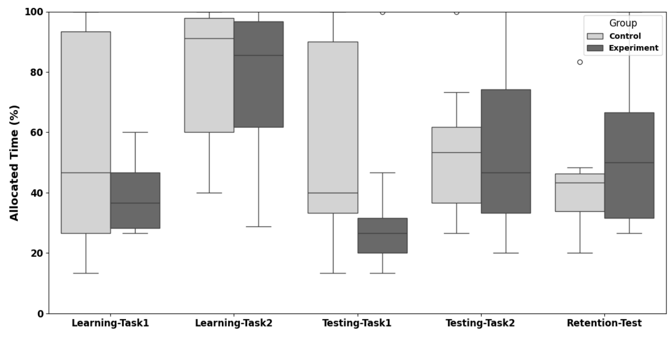


Figure 2: Box Plot of Time Taken Per Task by Group. The experimental group shows faster task completion for the tasks in the learning and testing phases, while the control group completed the Retention-Test in less time.

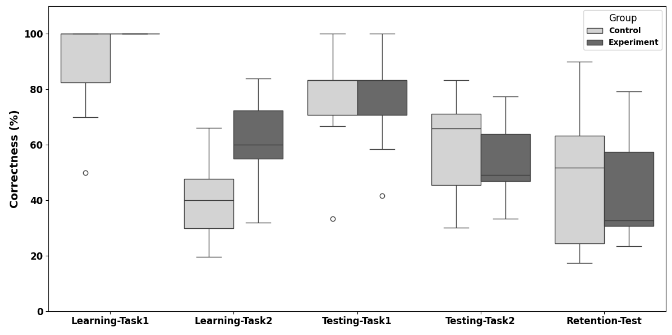


Figure 3: Box Plot of Tasks Solution Correctness Per Task by Group. The experimental group had higher performance on the learning tasks but lost this advantage in the main testing task (Testing-Task2), where the control group outperformed them. The control group also had better performance in the Retention-Test.

how much raters differ in their scores and how much disagreement exists between the raters. The resulting ICC value of 0.969 indicates strong agreement between the two graders.

Figure 3 shows the distribution of correctness scores across all tasks for the two participant groups. We find that the experiment group has better scores than the control group for both Learning-Task1 and Learning-Task2, with a statistically significant difference (p=0.0210 for Learning-Task1 using Mann-Whitney U Test and p=0.0080 for Learning-Task2 using T-Test). For Testing-Task1, both groups show almost the same distribution and have the same median correctness of 83.33%. However, we find that in Testing-Task2 (which is the main design task), the control group has a much higher median correctness score than the experiment group (65.83% for control vs. 49.17% for experiment), but this difference is not statistically significant.

We observe the same trend in the retention test correctness scores in the right-most plot of Figure 3. We find that the control group achieves a higher median correctness score compared to the experiment group (51.83% for control vs. 32.67% for experiment), but the difference is not statistically significant.

Observation 2: During the study session, students with access to ChatGPT (experiment group) obtained higher correctness scores than students without access (control group). However, once access is removed during testing and retention, the control group achieved equal scores on the easier testing task and higher scores in the more difficult testing task, as well as the retention test.

Comparing distributions helps us understand how the two groups performed overall, but it does not give us insights into the individual student journey. To consider this aspect, we analyze the results at the individual student level to understand correctness consistency among the different tasks. We do this analysis for the 19 participants who finished their retention test (10 for the control group, and 9 for the experiment group). We categorize correctness scores into three levels: High (Correctness score of 80% or more), Medium (50-79%), and Low (<50%). We also summarize the individual transitions in Table 3.

We first discuss the transition from Learning-Task2 to Testing-Task2. Ideally, if students exhibit short-term learning transfer, they would either maintain or increase their scores between Learning-Task2 to Testing-Task2. For the experiment group, we find that 56% of the experiment participants did worse in Testing-Task2, 22% remained at the same level, and only 22% improved from Low in Learning-Task2 to Medium in Testing-Task2. Meanwhile, we see more promising transitions for the control group. Specifically, 30% of the control participants improved their performance (20.00% went from Low to High and 10.00% moved from Low to Medium), while the remaining 70% maintained their correctness performance.

Considering the transition from Testing-Task2 to the Retention-Test, we also hope to see increased or maintained scores to exhibit more long-term learning transfer. For the control group, we find this observation to hold, with 50% of the participants maintaining their performance and 40% of the participants increasing in performance, and only 10% of the participants dropping in performance. In contrast, for the experiment group, we find that the majority (around 56%) of the participants declined in performance. In addition, around 33% maintained their performance while only 11% improved.

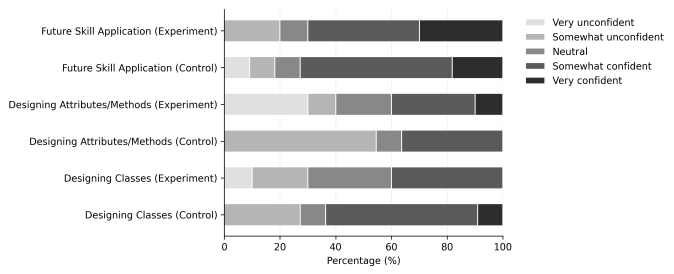
Observation 3: At the individual-student level, we observe that students in the experiment group tend to decline in task correctness across the progressive design tasks (around 56% decline in performance between Learning-Task2 and Retention-Test). On the other hand, students in the control group tend to show greater improvement across these tasks (around 40% improved in performance, while 50% maintained their performance between Learning-Task2 and Retention-Test).

4.4 Perceived Complexity and Confidence

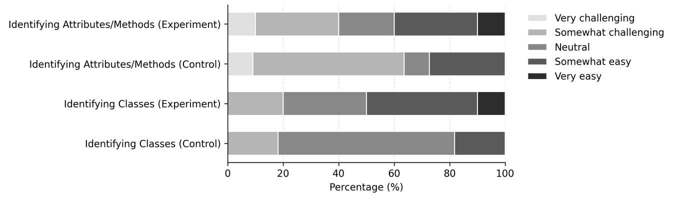
Recall that in the post-study questionnaire, we ask the participants a series of Likert-scale [citation needed] questions regarding their confidence in their solutions, the complexity of the tasks, and their confidence in applying these skills in future problems (questions listed in Table 1). Each question has a 5-point scale, ranging from strongly agree to strongly disagree (for agreement questions) or strongly confident to very unconfident (for confidence questions).

Group	PID	Initial	Learning-Task2 →Testing-Task2	Testing-Task2 →Retention-Test	Learning-Task2 →Retention-Test
Control	P3	Low	↑	↓	↑
Control	P6	Low	↑	↓	↑
Control	P7	Low	~	↑	↑
Control	P10	Medium	~	↓	↓
Control	P11	Low	~	~	~
Control	P12	Low	~	~	~
Control	P15	Low	~	~	~
Control	P16	Medium	~	~	~
Control	P17	Medium	~	↑	↑
Control	P19	Low	↑	↓	~
Experiment	P1	Medium	~	↓	↓
Experiment	P2	Low	↑	~	↑
Experiment	P5	Medium	~	~	~
Experiment	P22	Medium	↓	~	↓
Experiment	P9	Low	↑	↓	~
Experiment	P13	Medium	↓	~	↓
Experiment	P14	Medium	↓	~	↓
Experiment	P20	High	↓	↓	↓
Experiment	P21	Medium	↓	↑	~

Table 3: Performance Transitions of Participants. The Control group shows more participants improving or maintaining their performance from Learning-Task2 to Retention-Test, while the Experiment group shows a tendency to decline.



(a) Comparison regarding confidence in designing classes, attributes and methods, and future skills

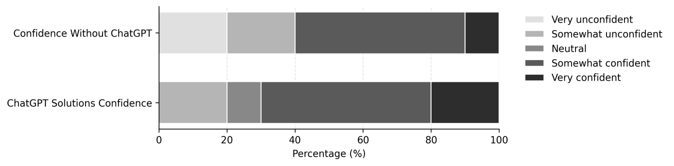


(b) Comparison regarding challenge in identifying relevant classes, attributes and methods of each class

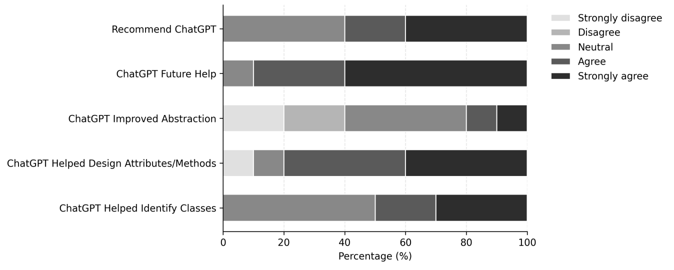
Figure 4: Comparison of Survey Responses between the Control and Experimental Groups for the post-study questionnaire

We now analyze the answers of both groups to understand differences in perceptions of task complexity and confidence.

4.4.1 Comparison between the groups in the post-study questionnaire. Figure 4 compares the answers to the common questions between the two groups. The answers of each group are shown on



(a) Confidence participants' skills to manually work on abstraction and decomposition without ChatGPT help, and confidence in ChatGPT solution



(b) Agreement with statement about ChatGPT use in the tasks

Figure 5: Survey Responses for the experimental group's specific post-study questions

Table 4: Rules for Aggregating Confidence in Class Design and Attributes/Methods Design Confidence Ratings into a Desgin Confidence Category

Confidence 1	Confidence 2	Aggregated Confidence Category
Very Confident	Very Confident	High - Confident
Very Confident	Somewhat Confident	High - Confident
Somewhat Confident	Somewhat Confident	High - Confident
Neutral	Neutral	Neutral
Neutral	Confident or Not Confident	Neutral
(Very and Somewhat) Confident	Not Confident (Very and Somewhat)	Neutral
Somewhat Unconfident	Somewhat Unconfident	Low - Not Confident
Very Unconfident	Somewhat Confident	Low - Not Confident
Very Unconfident	Very Unconfident	Low - Not Confident

a stacked bar chart that shows the percentage of the answers for each rating on the Likert scale.

For the confidence in their design, we ask participants one question regarding their confidence in the classes design, and one question for their confidence in the attributes and methods design. For easy of reporting and analysis, we aggregate the two answers into one design confidence category (High (Confident), Neutral, Low (Not Confident)). We follow the label as in table 4. We do the same for how challenging the find the task.

In terms of confidence in their design and confidence in applying abstraction and decomposition in the future, both groups had comparable confidence. Around 30% of the experiment group felt confident in their design, 40% were neutral, and 30% were not confident. In comparison, 27% of the control group felt a confident in their design, 45% were neutral, and 27% were not confident. Regarding their confidence in applying abstraction and decomposition in the future, around 73% of the control group reported a degree of confidence, and around 9% were neutral, while around 18% reported a degree of non-confidence. As for the experiment group, around

70% reported a degree of confidence, 10% were neutral, while 20% reported a degree of non-confidence.

Regarding how challenging the tasks are, the two groups where comparable but the experiment group had more participants perceiving it as not challenging than the control group (around 27% of participants found it to be easy vs. 9% of the control group). While 82% of the control group were neutral vs 50% of the experiment group. Only around 9% of both groups found it to be challenging.

Observation 4: When comparing groups regarding their confidence in their solution and future skills, as well as how challenging the tasks are, we find them to have somewhat comparable results. However more participants in the experiment group find the tasks to be not challenging (27% vs.9%).

4.4.2 *Experiment group's specific questions.*

Figure 5 shows the results of the questions specific to the experiment group. In terms of using ChatGPT for the given tasks, around 80.0% of participants found it easy or somewhat easy, while the remaining 20.0% found it somewhat difficult.

Regarding their confidence in the solutions provided by ChatGPT, around 70.0% of the experiment group reported a degree of confidence, and around 20.0% of the experiment group reported a degree of non-confidence, while 10.0% were neutral. Regarding their confidence in their skills without ChatGPT help, around 60.0% of the experiment group reported a degree of confidence, while the remaining 40.0% of the experiment group reported a degree of non-confidence (Figure 5a).

When asked about whether ChatGPT helped them save time solving tasks, around 90.0% reported a degree of agreement, while only 10.0% reported that it took more time to complete the tasks with ChatGPT.

From Figure 5b, we find that around 65.0% agreed that ChatGPT helps in identifying classes, and in identifying attributes and method, while around 30.0% where neutral. Only around 5.0% disagreed that it did not help.

Interestingly, when asked if they think ChatGPT helps improve their ability to think about abstraction and decomposition, the majority of participants either were neutral (40.0%) or disagreed (40.0%) while only 20.0% agreed. On the other hand, when asked if they think ChatGPT will help them solve similar tasks in the future, the majority of participants agreed (around 90.0%) while the remaining 10.0% were neutral. When asked if they recommend ChatGPT to their peers, the majority of participants agreed (around 60.0%) and around 40.0% were neutral and no one disagreed.

In terms of how challenging they found manually abstracting and decomposing without ChatGPT help, around 50.0% found it challenging, while around 40.0 % were neutral. Only 10.0 % found it somewhat easy.

Observation 5: The experiment group find ChatGPT easy to use and time-saving, and also have high confidence in its solutions. While half the participants find it challenging to do these tasks manually, only 20% of participants agree that ChatGPT helps improve their ability to think about abstraction or decomposition.

4.4.3 *Confidence vs. Performance.*

Recall that we ask participants for three confidence measures: one for their solutions for the testing task (*Testing Solution Confidence*), one for their confidence in future skills (at least a week before their retention test) (*Confidence in Future Skills*), and one for their confidence in their retention test solutions (*Retention Test Solution Confidence*). Accordingly, we compare their Testing Solution Confidence with their Testing Solution Score and their Retention Test Solution Confidence with their Retention Test Solution Score. These comparisons help us evaluate the students' self-assessment abilities, i.e., whether they can accurately judge when they performed well or poorly and wether they have truly understood the concepts. On the other hand, we can compare the students' Confidence in Future Skills with their Retention Test Score to see whether students can accurately estimate the knowledge they have gained.

We categorize the participants' correctness scores into three levels: High (80% or more), Medium (50-79%), and Low (<50%). We categorize the confidence as per rules in Table 4. We then compare confidence scores with correctness scores to determine how much their confidence matches their actual performance. A calibrated participant is a participant whose confidence matches their actual performance. We indicate the different labels we use to categorize participants based on this matching, and summarize the results in Figure 6

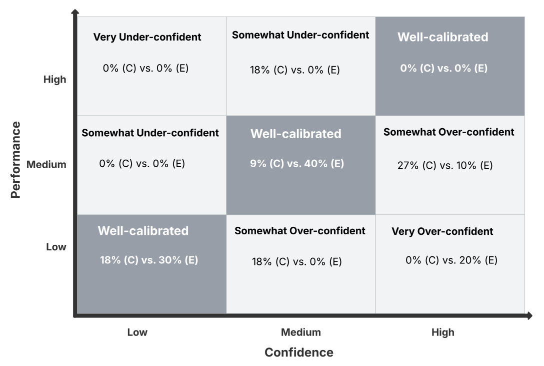
Analyzing this results, we can see that the experiment group have the majority of their participants as calibratedparticipants (70% of participants as opposed to 27% for control group) when comparing their testing solution score and their confidence in their solution. However, when comparing their retention score to their confidence in their future skills, we find that both groups were overconfident in their skills.

However, when comparing between Retention Test Confidence and Retention Test Score, we find that around 50% of students in the control group are calibrated. We start seeing the control group becoming more aware of their current performance. While, the experiment group lose their awarness from the testing solutions reporting only around 44% as calibrated, and we start seeing underconfident participants which we do not observe in the confidence in testing solution or future skills when questioned after the study session.

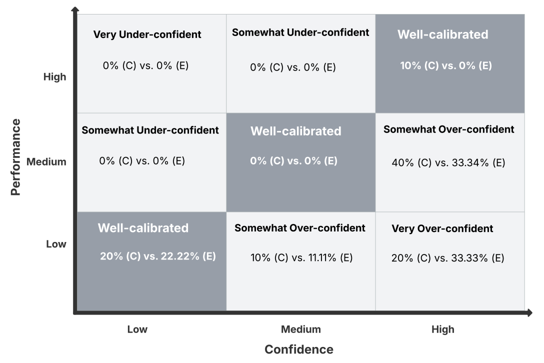
Observation 6: When comparing confidence with actual performance, the experiment group showed better alignment (70% vs. 27%) during the study tasks. However, this awareness declined later, as they became overconfident about their future skills, similar to the control group. By the final questionnaire, some experiment group students lost confidence, unlike earlier responses, while more control group students became better aligned between confidence and performance.

4.5 Interaction with ChatGPT and Prompting Patterns

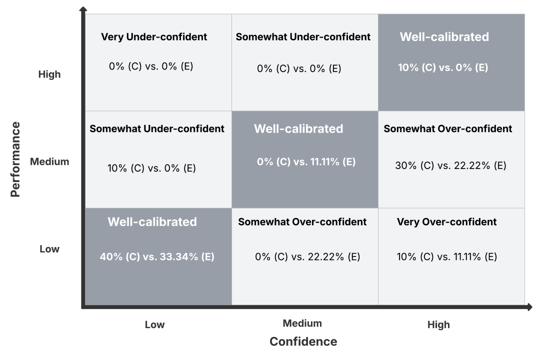
We now report on the experiment's group interactions with ChatGPT. We analyze the screen recordings as well as the prompts' content to codify the prompts into categories. We used close coding [24], using a pre-defined set of prompt categories from prior



(a) Testing Solution Score vs. Testing Solution Confidence



(b) Retention Test Solution Score vs. Confidence in Future Skills



(c) Retention Test Solution Score vs. Retention Test Solution Confidence.

Figure 6: Comparison of Actual Performance vs. Reported Confidence. Percentages show proportion of Control (C) participants and Experiment (E) participants in each category.

work [22, 31]. We manually codify the prompts, with one author coding while a second author reviews the results.

We summarize the prompting patterns we observed in Table 5, including how often we observe each pattern. We find that the most common prompting category is the "Direct Solution (Solve)" pattern, where participants request a complete solution or a ready-to-use code snippet by directly pasting the problem statement (27%), followed by the "Explanatory Inquiry (Explain)" pattern (23%). The least common category is the "Code Correction (Fix)" and the "Miscellaneous (Others)" pattern. Overall, our findings match those previously found in programming studies [22] where the most common category was also the "Solve" pattern.

Table 5: ChatGPT prompting pattern categories, including percent of prompts belonging to each category (frequency)

Pattern Category	Description	Frequency
Direct Solution (Solve)	Prompts where users request a complete solution or a ready-to-use code snippet by directly pasting the problem statement.	27%
Explanatory Inquiry (Explain)	Prompts that ask for detailed explanations regarding some aspects of the problem and proposed solutions.	23%
Partial Guidance (Hint)	Prompts that ask for hints or partial guidance rather than a full solution—helping users move forward incrementally without handing over the entire answer.	18%
Syntax Assistance (Syntax)	Prompts focusing on verifying or correcting the syntax, formatting, or structure of code to ensure adherence to language standards.	16%
Modification (Modify)	Prompts where users ask for modification to a given code or solution.	9%
Code Correction (Fix)	Prompts in which users provide an error message or problematic code and ask ChatGPT to generate a corrected version.	5%
Miscellaneous (Others)	Any prompts that do not clearly fit into the above categories, such as incomplete queries.	2%

Observation 7: The most common prompting category across the study tasks is the "Direct Solution (Solve)", where around 27% of the prompts are in this category, followed by the "Explantory Inquiry (Explain)" pattern (23%).

5 Discussion and Implications

ChatGPT provides fast but fleeting advantages. Students who had access to ChatGPT during the learning phase completed tasks faster and scored higher at that stage. However, this advantage does not carry over to the later phases (Observation 1 and Observation 2). This may indicate that any effect of the tool does not last long-term. On an individual level, these students' performance tends to decline as they progress through the tasks, whereas students in the control group tend to improve (Observation 3). The experiment group students had favorable views on the use of ChatGPT, and more students found the tasks to be less challenging than the control group (Observation 4 and Observation 5). They also had better aligned confidence in their performance for the study tasks. In contrast, when it comes to their future skills, they lose this awareness and become overconfident.While some start to lose their confident when assessing their confidence in their solution after the retention test (Observation 6). The most common prompts used by students were to get direct answers or explanations, which indicates a level of cognitive offloading to the tool ((Observation 7).

Overall, the results show that ChatGPT gave the students an initial added advantage. It helped them save time. It also helped more participants to perceive the tasks to be less challenging than the control group. This perception could potentially help in students being more open to working on these tasks [28].

However, given the above change in results after removing AI-assistance, we conclude that ChatGPT was used as a "crutch" rather than a learning tool. In other words, ChatGPT provided fast but fleeting help.

Next, we reflect on these implications and discuss two take-home messages that can help educators better integrate generative AI into teaching and learning, and help students preserve the AI-assistance tool gains in the long term.

Take Home Message 1 - Early Integration Plus Reflection. Educators need to incorporate more self-reflection and more critical thinking in their assignments, which gets the students to reflect more on their skills. The added initial advantages we observed indicate that educators should integrate AI assistance early in the learning process rather than oppose it. However, they should incorporate checkpoints within the semester for students to assess their own skills without the help of generative AI. These checkpoints will give students feedback on their actual competencies. To do this educators should integrate pre-and post-assessment questionnaires on the students' confidence and perception of their skills. Comparing their confidence to their actual performance on any course deliverables can help student gain deep insights into their competencies (and in-competencies). Having their in-competences acknowledged and brought to their consciousness will help students gain mastery of the needed computational skills [13].

Take Home Message 2 - Prompting Techniques for Better Learning. From the prompting techniques (Observation 7), we can observe patterns that are indicative of cognitive offloading to the AI tool. While explanation prompting does indicate some level of the students' attempt to learn, but this still has a sense of reliance on the tool. We can observe that partial guidance is underutilized and could be improved. Educators need to advocate for students to use more partial guidance prompts to scaffold students' learning. These prompting techniques will get students to share the cognitive load with the tool rather than asking for a ready-made solution. Thus, using the tool as a scaffold to their learning rather than a crutch. Educators can also incorporate prompting exercises in which students are given a task, with the purpose of having them think through how they will formulate the prompt for the generative AI tool and reflect on their prompting techniques.

6 Threats to Validity

Construct Validity. We use decomposition and abstraction as proxies for computational thinking. Though not the only components, they are key aspects. While computational thinking is hard to measure objectively in a study setting, we believe object-oriented design tasks exemplify these skills and offer a concrete, feasible way to assess students' understanding and application. Future studies can augment our results by studying other aspects of computational thinking.

Internal Validity. Variations in the students' prior knowledge can impact the results. We mitigate this confounding factor using a matched-pairs design. With this design, we try to balance the two groups with regard to their background, in order to minimize the effect of background variations. We find that there is indeed no statistically significant difference between the groups regarding the background variables, showing that our matched pair design was effective in balancing the participants' background across the two groups.

A given problem can be designed/abstracted in different ways, making marking subjective. To overcome this, we developed a detailed rubric that aligns with the tasks' ideal solution; however, we provided guidelines for graders to realistically consider what is expected of students at this early stage of learning. The rubrics focused more on the design aspects rather than exact syntax. Two independent graders graded every submission and met to reconcile any disagreements in score. In addition, in our analysis, we focus on changes in performances (so drops and gains between experiment groups) rather than the absolute numerical values themselves. Finally, we measure the participants' confidence using self-reported Likert-scale responses, which can be subjective.

External Validity. We conduct the study in a controlled environment, which may not reflect real-world scenarios where students have broader resource access. The tasks are limited in scope and may not capture full system complexity, but such constraints ensure feasibility without overburdening students during the term. We also use one week as the retention test time frame, which may not be long enough to capture the full effect; however, testing in a longer time frame would not be feasible given the availability of participants. Future work with longer time frames will help us better understand the longer-term effects of using GenAI in learning.

Statistical Conclusion Validity. One limitation of our study is the relatively small number of participants, which may have limited our ability to observe more statistically significant differences, even though the observations are practically meaningful.

7 Related Work

Hanifi et al. [8] surveyed 113 computer science and software engineering students about their views on ChatGPT's usefulness, trust, and challenges. Most students reported using ChatGPT in their projects and were satisfied with it, but only about half understood how the technology works. Around 90% experienced hallucinations, where ChatGPT gave incorrect or nonsensical answers. Hammer et al. [7] found similar results, most students used ChatGPT in academic tasks, viewed it as helpful, and often saw it as a "multi-task solver." Prather et al. [20] surveyed students and instructors on their views of LLMs in computing education, and called for more research into how to best use GenAI in learning.

With this growing reliance of students on generative AI tools like ChatGPT, researchers began to study their effect on students' performance and learning. Kazemitabaar et al. [11] examines how code generation tools impact novice programmers' performance [11]. The results indicate that such tools help programmers complete tasks faster and with higher performance without impairing their ability to perform manual code modifications. Xue et al. [31] look into the use of generative AI by students as they solve a UML task. The authors note that using ChatGPT does not necessarily have a significant impact on students' learning performance and that students tend not to explore other traditional resources once they start using ChatGPT.

While the above studies suggest that ChatGPT has become "a student's best friend", there are other studies that also reveal potential concerns. Rahe and Maalej [22] investigated how programming students interact with ChatGPT during coding tasks. The authors observed that students often fall into a loop in which they repeatedly submit flawed AI-generated code and then ask the bot for corrections, rather than using the tool to understand the code and their mistakes. This may suggest that students over-rely on the tool without critically thinking about the task at hand.

Rachmat et al. [21] studied how ChatGPT-like chatbots affect reflective thinking in an introductory programming course using a mixed-methods design. While the quantitative results showed no significant improvement, qualitative data revealed that students using structured prompts showed deeper reasoning and better self-questioning. This suggests chatbots can support reflective learning when used as guided partners rather than direct answer providers. Shihab et al. [25] conducted a within-subjects experiment on GitHub Copilot in brownfield programming tasks. Students completed tasks faster and spent less time coding or searching online, but many reported lower understanding of the generated code, indicating that Copilot helps with productivity but may reduce deep code comprehension. Penney et al. [18] compared learning with ChatGPT versus human tutors by novice programmers using mixed-method research. Results showed no significant difference in learning gains, but students felt more comfortable asking ChatGPT questions. However, they used short, unrefined prompts, while human-tutored students asked more reflective questions. The study highlights ChatGPT's role in supporting low-pressure inquiry, but with limited deep learning, stressing the need for AI literacy and thoughtful prompting. Chugh et al. [5] conducted a multi-university qualitative study on how ICT students use GenAI tools like ChatGPT. Students described GenAI as a useful and time-saving "study buddy" that improved their learning experience, but also raised concerns about accuracy, over-reliance, and ethics.

Zönnchen et al. [32] reviewed studies on the use of ChatGPT in teaching and learning software engineering. They found that generative AI tools work well for simple programming tasks but are less effective for complex ones, often producing more logical (semantic) errors than syntax errors. They highlight that evaluating AI-generated code critically may become more important than writing syntactically correct code. Aruleba et al. [1] conducted a systematic review focusing on engagement and pedagogy. They found that LLMs can support debugging and learning but may cause over-reliance and shallow understanding if not properly guided. Philbin and Sentance [19] also performed a systematic literature review and reported that generative AI often replicates existing teaching methods, with students using AI mostly in passive or assistive ways,such as debugging or generating explanations,rather than for creative problem-solving.

8 Conclusion

This paper presented our research that investigates whether access to ChatGPT during the learning process influences students' short- and long-term acquisition of essential computational-thinking skills. We ran a controlled experiment that focuses on decomposition and abstraction. From our results, we concluded that ChatGPT provided fast but fleeting advantages, helping students complete tasks faster, perceive them as less challenging, and have better initial awareness of their performance. However, these gains did not carry over to later phases, suggesting that the tool acted more as a crutch than a lasting learning aid. Students tended to rely on direct solutions, indicating cognitive offloading rather than skill development. Educators should therefore integrate generative AI early but pair it with reflection and guided prompting to preserve learning benefits in the long term.

Acknowledgements

The authors acknowledge using generative AI tools like ChatGPT for editorial purposes only, and for helping with writing data analysis scripts and to occasionally summarize original text to fit space limitations.

References

[1] Kehinde Aruleba, Solomon Sunday Oyelere, George Rabeshi Obaido, and Ismaila Temitayo Sanusi. 2025. A Scoping Review of Student and Educator Engagement with Large Language Models in Introductory Programming Education. In *Proceedings of the 2025 Conference on UK and Ireland Computing Education Research*. 1–8.

[2] Computing Research Association. 2023. *2023 CRA Taulbee Survey Report*. Technical Report. Computing Research Association. https://cra.org/wp-content/uploads/2024/05/2023-CRA-Taulbee-Survey-Report.pdf 53rd annual survey of degree production, enrollment, and employment in North American computing departments.

[3] Susan M Barnett and Stephen J Ceci. 2002. When and where do we apply what we learn?: A taxonomy for far transfer. *Psychological bulletin* 128, 4 (2002), 612.

[4] Miriam Bruhn and David McKenzie. 2009. In pursuit of balance: Randomization in practice in development field experiments. *American economic journal: applied economics* 1, 4 (2009), 200–232.

[5] Ritesh Chugh, Darren Turnbull, Sangeetha Kutty, F Sabrina, MM Rashid, A Morshed, S Azad, S Kaisar, and S Subramani. 2025. Generative AI as a learning assistant in ICT education: student perspectives and educational implications. *Education and Information Technologies* (2025), 1–36.

[6] Shiva Hajian. 2019. Transfer of learning and teaching: A review of transfer theories and effective instructional practices. *IAFOR Journal of education* 7, 1 (2019), 93–111.

[7] Sabine Hammer, Sarah Ottinger, Benedikt Zönnchen, Michel Hohendanner, Martin Hobelsberger, and Veronika Thurner. 2024. ChatGPT in Higher Education: Perceptions of Computer Science-Related Students. In *2024 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 01–08.

[8] Khadija Hanifi, Orcun Cetin, and Cemal Yilmaz. 2023. On chatgpt: Perspectives from software engineering students. In *2023 IEEE 23rd International Conference on Software Quality, Reliability, and Security (QRS)*. IEEE, 196–205.

[9] Beryl Hesketh. 1997. Dilemmas in training for transfer and retention. *Applied Psychology* 46, 4 (1997), 317–339.

[10] Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, et al. 2023. ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and individual differences* 103 (2023), 102274.

[11] Majeed Kazemitabaar, Justin Chow, Carl Ka To Ma, Barbara J Ericson, David Weintrop, and Tovi Grossman. 2023. Studying the effect of AI code generators on supporting novice learners in introductory programming. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–23.

[12] Irene Lee, Fred Martin, and Katie Apone. 2014. Integrating computational thinking across the K–8 curriculum. *Acm Inroads* 5, 4 (2014), 64–71.

[13] Marsha C Lovett, Michael W Bridges, Michele DiPietro, Susan A Ambrose, and Marie K Norman. 2023. *How learning works: Eight research-based principles for smart teaching*. John Wiley & Sons.

[14] I Scott MacKenzie. 2024. Human-computer interaction: An empirical research perspective. (2024).

[15] Kenneth O. McGraw and S. P. Wong. 1994. The Descriptive Use of Absolute Differences between Pairs of Scores with a Common Mean and Variance. *Journal of Educational Statistics* 19, 2 (1994), 103–110. http://www.jstor.org/stable/1165141

[16] Timothy J Nokes-Malach and Jose P Mestre. 2013. Toward a model of transfer as sense-making. *Educational Psychologist* 48, 3 (2013), 184–207.

[17] OpenAI. [n. d.]. ChatGPT. https://chat.openai.com. Accessed: 2025-04-16.

[18] Jacob Penney, Pawan Acharya, Peter Hilbert, Priyanka Parekh, Anita Sarma, Igor Steinmacher, and Marco Aurelio Gerosa. 2025. Outcomes, Perceptions, and Interaction Strategies of Novice Programmers Studying with ChatGPT. In *Proceedings of the 7th ACM Conference on Conversational User Interfaces*. 1–15.

[19] Carrie Anne Philbin and Sue Sentance. 2025. Student Use and Teacher Practice: A Scoping Review of Generative AI in Computing Education Using PICRAT. In *Proceedings of the 2025 Conference on UK and Ireland Computing Education Research*. 1–7.

[20] James Prather, Paul Denny, Juho Leinonen, Brett A Becker, Ibrahim Albluwi, Michelle Craig, Hieke Keuning, Natalie Kiesler, Tobias Kohn, Andrew Luxton-Reilly, et al. 2023. The robots are here: Navigating the generative ai revolution in computing education. In *Proceedings of the 2023 working group reports on innovation and technology in computer science education*. 108–159.

[21] Agatha Rachmat, Craig Watterson, and Karsten Lundqvist. 2025. The Impact of Chatbots on Students' Reflective Thinking in Introductory Programming Course. In *2025 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 1–10.

[22] Christian Rahe and Walid Maalej. 2025. How Do Programming Students Use Generative AI? *Proceedings of the ACM on Software Engineering* 2, FSE (2025), 978–1000.

[23] Sheldon M Ross. 2017. *Introductory statistics*. Academic Press.

[24] Johnny Saldaña. 2021. The coding manual for qualitative researchers. (2021).

[25] Md Istiak Hossain Shihab, Christopher Hundhausen, Ahsun Tariq, Summit Haque, Yunhan Qiao, and Brian Wise Mulanda. 2025. The Effects of GitHub Copilot on Computing Students' Programming Effectiveness, Efficiency, and Processes in Brownfield Coding Tasks. In *Proceedings of the 2025 ACM Conference on International Computing Education Research V. 1*. 407–420.

[26] Patrick E Shrout and Joseph L Fleiss. 1979. Intraclass correlations: uses in assessing rater reliability. *Psychological bulletin* 86, 2 (1979), 420.

[27] P Robert-Jan Simons. 1999. Transfer of learning: Paradoxes for learners. *International journal of educational research* 31, 7 (1999), 577–589.

[28] Piers Steel. 2007. The nature of procrastination: a meta-analytic and theoretical review of quintessential self-regulatory failure. *Psychological bulletin* 133, 1 (2007), 65.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[30] Jeannette M Wing. 2006. Computational thinking. *Commun. ACM* 49, 3 (2006), 33–35.

[31] Yuankai Xue, Hanlin Chen, Gina R Bai, Robert Tairas, and Yu Huang. 2024. Does chatgpt help with introductory programming? an experiment of students using chatgpt in cs1. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training*. 331–341.

[32] Benedikt Zönnchen, Veronika Thurner, and Axel Böttcher. 2024. On the Impact of ChatGPT on Teaching and Studying Software Engineering. In *2024 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 1–10.