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 the development of 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 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 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. 2026. Fast and Fleeting: Evaluating ChatGPT's Impact on Students' Computational Thinking Skills. In . ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Computational thinking (CT) involves expressing solutions in structured and executable forms, decomposing complex problems, reasoning at multiple levels of abstraction, and automating procedures via algorithmic design; skills that are foundational to the design and implementation of robust systems [37]. In addition to programming skills, successful software engineers need to master computational thinking [15]. While computer science and software engineering curricula typically aim 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 [21], 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 [14, 28]. However, the influence of generative AI on computational thinking among these students remains underexplored. Given concerns about students' over-reliance on these tools [13], it is critical to determine whether generative AI use impedes the development of competencies central to effective software engineering.

To address this gap, we conduct a controlled experiment in which participants complete object-oriented design and programming (OOP) tasks that naturally elicit computational thinking, specifically 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 design the study's tasks to operationalize abstraction and decomposition. Participants are asked to translate a solution description into an object-oriented design by identifying appropriate classes and attributes, which reflects abstraction. They also need to decompose the problem into key entities, then further break down each entity's behavior into methods. Quantitatively, we compare task completion time and solution correctness across the learning, testing, and retention phases. We treat time and correctness as indicators of short-term task performance, and we use retention performance as an indicator of longer-term learning. While time and correctness do not directly measure computational thinking, they provide observable outcomes of performance on tasks designed to represent abstraction and decomposition. 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 qualitatively investigate participants' interactions with ChatGPT and their prompting patterns. We balance our

Conference'17, Washington, DC, USA
2026. ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

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.

Based on data from 21 students, 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 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 their actual performance over time, reflecting a decline in self-awareness of skill level. In contrast, the control group's self-awareness improved. Based on our results, we provide suggestions for effectively integrating AI tools in software engineering courses.

2 Terminology and Related Work

Recent work has shown a growing use of generative AI (GenAI) tools like ChatGPT in computing education. Hanifi et al. [11] surveyed 113 CS and SE students and found that most used ChatGPT in their projects and were generally satisfied, despite concerns about hallucinations and a limited understanding of how the tool works. Hammer et al. [10] found similar results, with students seeing ChatGPT as a helpful "multi-task solver." Prather et al. [26] surveyed both students and instructors and concluded that more research is needed on how GenAI can best support learning.

Other studies focus more on the impact of using GenAI tools, reporting mixed results. Kazemitabaar et al. [14] found that code-generation tools helped novice programmers complete tasks faster and with better performance, without impairing their ability to modify code manually. Xue et al. [38] observed that while ChatGPT can support UML problem-solving, it may not lead to better learning outcomes and could discourage students from exploring other learning resources.

While the above studies suggest that ChatGPT has become "a student's best friend", other studies also reveal potential concerns. Rahe and Maalej [28] 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 it 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. [27] 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. [31] 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 a lower understanding of the generated code, indicating that Copilot helps with productivity but may reduce deep code comprehension. Penney et al. [24] 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.

Ghimire and Edwards [8] analyzed prompts, AI replies, and keystroke logs from CS1 assignments and found that students asked for debugging and conceptual help more often than they asked the AI to write code, while also frequently copying AI outputs into their submissions. Fenu et al. [7] similarly examined student interactions with AI support in a programming training setting, showing that prompting guidance shifted students toward more goal-driven, optimization-oriented prompting rather than passive prompting.

Broader qualitative studies echo these themes found in the previous, more investigative studies. Chugh et al. [6] reported that students viewed GenAI as a "study buddy" that improved their learning but raised concerns about over-reliance and accuracy. Zönnchen et al. [39] showed that GenAI worked well for simple tasks but struggled with complex ones, often generating semantic errors. Aruleba et al. [1] warned that while LLMs can support learning, they can also encourage shallow understanding if not guided properly. Philbin and Sentance [25] found that students mostly use GenAI in passive ways that replicate traditional practices, rather than promoting creativity or critical thinking.

Accordingly, while prior studies show that students perceive ChatGPT as helpful and time-saving, they also reveal concerns about reliance, shallow understanding, and limited long-term benefits. However, most of the above studies focus on programming tasks with limited empirical work studying how GenAI tools affect specific computational thinking skills and knowledge retention. *Computational thinking (CT)* [37] describes a set of thinking skills and approaches that are built around the "power and limits" of the computing process [37]. These are essential skills that software engineers develop to formulate solutions, beyond programming. CT skills include abstraction, decomposition, recursive thinking, algorithmic thinking, and the ability to generalize and transfer solutions to a wide variety of problems [37].

Our study addresses this gap by evaluating how ChatGPT impacts students' ability to apply abstraction and decomposition skills over time. We focus on *abstraction* and *decomposition* as two representative computational thinking skills. Abstraction is the process of focusing on the essential parts of a problem, highlighting relevant features while hiding complex implementation details. It supports clearer modeling and helps software engineers reason about systems at multiple levels. Decomposition is the process of breaking down large or complex problems into smaller, manageable parts that can be developed or solved independently [37]. We investigate this problem through a controlled experiment, unlike relevant work that primarily analyzes submitted assignments (e.g., [38]).

When considering how a skill is learned, there is a distinction between two types of learning transfer: *short-term* and *long-term*. *Short-term transfer* (or near transfer) refers to applying newly acquired knowledge shortly after instruction [3, 9, 12, 20, 33]. *Long-term transfer* (or far transfer) involves applying knowledge after a delay—such as a week or more—indicating deeper understanding and retention. Our experiment is designed to evaluate both; we assess short-term transfer through in-session learning and testing tasks, and long-term transfer through a follow-up retention test a week later.

3 Experiment Setup

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 Participant Recruitment & Group Assignment

3.2.1 Inclusion Criteria. We recruit students from the New York University Abu Dhabi. 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. In other words, these students have sufficient programming skills but have no in-depth practice of abstraction and decomposition. Since Python is the programming language of instruction in the introductory computer science course, we use it for the tasks in our study. Accordingly, as part of our inclusion screening, we additionally make sure that students self-declare Python efficiency.

3.2.2 Recruitment and Screening. We announce the study by posting flyers in different locations on campus where undergraduate students usually congregate and by advertising in relevant first-year courses. Interested students fill out an online screening survey that asks about their programming background, their grade in the introductory programming course, whether they were enrolled in or have completed more advanced courses, their proficiency in various languages, including Python, and their choice of GenAI tool, as well as the tasks they use it for in their coursework. We filter sign-ups based on the inclusion criteria and reach out to eligible participants for scheduling.

3.2.3 Experimental Block Assignment. We use pairwise matching followed by balanced random assignment [5, 18] to ensure that the participants in the two groups are similar in terms of their programming experience and background. We first form pairs of participants with similar backgrounds based on their years of programming, self-reported Python proficiency, and grade in the Introduction to Programming course. Within each pair, we then randomly assigned

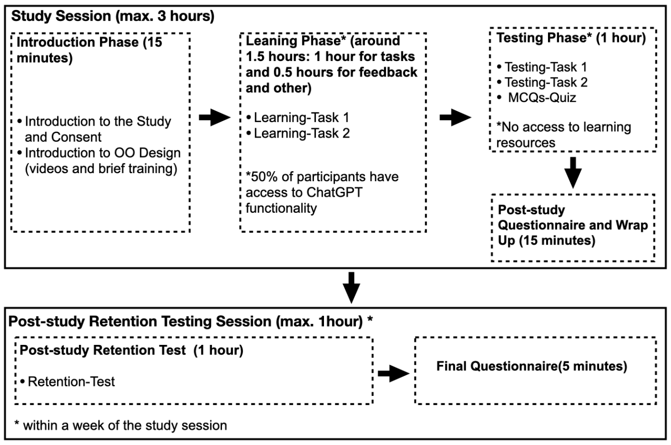

Study Session (max. 3 hours)
Introduction Phase (15 minutes)
• Introduction to the Study and Consent
• Introduction to OO Design (videos and brief training)
Leaning Phase* (around 1.5 hours: 1 hour for tasks and 0.5 hours for feedback and other)
• Learning-Task 1
• Learning-Task 2
*50% of participants 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.

one participant to the experiment group and the other to the control group. Participants without a matched pair are randomly assigned to either group while keeping group sizes as balanced as possible. We use pairs to balance groups at assignment, however we conduct the analysis at the group level.

3.2.4 Compensation and Ethical Considerations. As compensation for participants' time, we offer each participant a 100 AED Amazon gift card. We received approval from our Institutional Review Board (IRB), which considers the study to pose minimal risk to participants. At the beginning of the study, we obtain informed consent from each participant. We inform students of the study's purpose and that they can withdraw or take a break at any time during the sessions. We anonymize all data collected during the experiment.

3.3 Experiment Design

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 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, intended to mimic a self-study session followed by an assessment. We next provide an overview of these two sessions, with Section 3.3.3 detailing the specific tasks used.

The study session can last up to three hours. At the beginning of the study session (*introduction phase*), the researcher obtains the participant's consent and provides a brief overview of 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

Table 1: List of Post-study Questionnaire Questions

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

an example and an educational video on abstraction, as well as another video on writing Python classes.

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[1] 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 completes each learning task, we show them the answer key as 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 confidence and experience with the tasks. The experiment group also 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 [14]. Participants can sign up for the retention test anytime during the week following their study session, with a minimum gap of two days (for participants who do their study session on Friday and choose the following Monday for the retention 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 Independent and Dependent Variables. Our *independent variable* is the type of tool access provided during the learning phase; participants in the experiment group had access to ChatGPT, whereas participants in the control group were limited to traditional web search. Our *dependent variables* are the student's solution correctness based on an answer key and a predefined rubric, time to complete the tasks, their confidence in their solution, and their skills retention. 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.

3.3.2 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.

3.3.3 Tasks. Leveraging the authors' combined extensive experience in teaching software engineering and OOP, we design 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 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 requires 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

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

Table 2: Participants Demographics

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

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[2].

3.3.4 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.

3.3.5 Pilot Study. We first run 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[29] for data that is not normally distributed and a T-Test [29] for normally distributed data. We use the chi-squared test for categorical data [29]. 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. Python proficiency is self-reported on a 5-point scale collected in the screening survey. 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].

Over half of the participants (54.2%) selected ChatGPT as their preferred GenAI tool, which was the tool used in our experiment. The remaining participants chose other tools, such as Claude and Perplexity, but none were selected by more than 10% of the participants.

Based on our pair-matching design to balance the participants' backgrounds across the two groups, we were able to balance 9 pairs of the 21 participants (18 participants), then randomly assigned 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 no significant differences between the groups regarding

[2]https://figshare.com/s/859073b2e4eaa338ff68

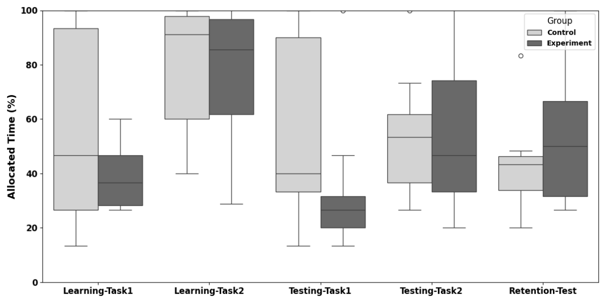


Figure 2: Time Taken Per Task by Group.

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 backgrounds of the two groups.

We note that, across participants in each group, the median number of days between the study session and the retention test is 7 days. In the final survey, where we ask students whether they prepared for the retention test, only one participant mentions studying related materials (i.e., documentation and tutorials), while the rest report no preparation. In order to avoid contamination concerns, we explicitly instructed participants not to share or discuss the tasks or the retention test with any of their peers.

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-Task1 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 the 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). However, we note that none of the time differences we observe above are statically significant.

Table 3: Correctness Transitions of Participants. ↑ indicates improvement in correctness category, ↓ indicates a drop, while ~ indicates maintained correctness.

Group	PID	Initial	Learning-Task2 →Testing-Task2	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	↓	~

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) [19], 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 (ICC), which ranges from 0 to 1, with 1 indicating perfect agreement and 0 indicating no agreement [32]. The ICC captures 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.

4.3.1 *Group-level analysis.*

Figure 3 shows the distribution of correctness scores across all tasks for the two participant groups.

We find that the experiment group scores higher than the control group on both Learning-Task1 and Learning-Task2, with statistically significant differences. For Learning-Task1, a Mann-Whitney U test yields $p = 0.0210$ with a medium effect size (rank-biserial $r = 0.4545$). For Learning-Task2, an independent-samples t-test yields $p = 0.0080$ with a large effect size (Cohen's $d = 1.2940$). We

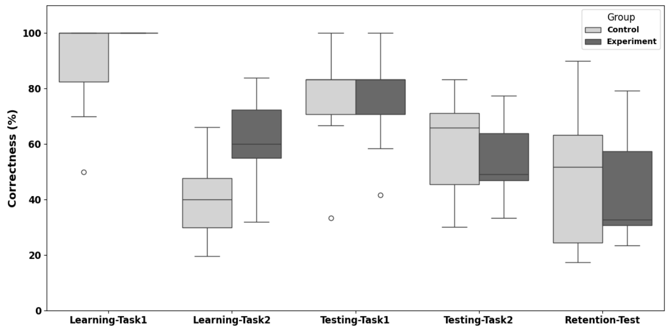


Figure 3: Task Solution Correctness Per Task by Group.

used the Shapiro–Wilk test to assess normality for both tasks. 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 those without access (control group). However, once access was removed during testing and retention, the control group achieved equal scores on the easier testing task and higher scores on the more difficult testing task and the retention test.

4.3.2 Individual-level analysis. Comparing distributions helps us understand how the two groups performed overall, but it does not provide insights into individual student journeys. 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%). Table 3 summarizes the transitions.

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 and 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 Learning-Task2 to the Retention-Test, we also hope to see increased or maintained scores to exhibit long-term learning transfer. For the control group, we find this

Table 4: Combining Confidence in (1) Class Design and (2) Attributes and Methods Design Confidence Ratings into a single Design Confidence rating

Confidence 1	Confidence 2	Design Confidence
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 or Somewhat) Confident	Not Confident (Very or Somewhat)	Neutral
Somewhat Unconfident	Somewhat Unconfident	Low - Not Confident
Very Unconfident	Somewhat Confident	Low - Not Confident
Very Unconfident	Very Unconfident	Low - Not Confident

observation to hold, with 50% of the participants maintaining their performance, 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. Only approximately 33% maintained their performance, while 11% improved.

Observation 3: At the individual student level, we observe that the control group exhibits more long-term transfer (i.e., between Learning-Task2 and Retention-Test): experiment group (56% declined, 33% maintained, 11% improved) vs. control group (10% declined, 50% maintained, 40% improved).

4.4 Perceived Complexity and Confidence

Recall that in the post-study questionnaire, we ask the participants a series of Likert-scale [16] 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). These subjective ratings allow us to relate students' self-reported confidence to their actual performance, in line with prior work showing that the accuracy of self-evaluation (i.e., calibration of confidence to performance) is a strong predictor of exam success and course achievement, and therefore an important component of effective learning [22, 35].

4.4.1 Common questions. Figure 4 compares the answers to the common questions between the two groups. The answers of each group are shown on a stacked bar chart that shows the percentage of the answers for each rating on the Likert scale.

To measure participants' confidence in their design, we asked them two questions, one about their confidence in the class design and another about the attributes and methods design. For easier reporting and analysis, we combine both responses into a single design confidence category: High (Confident), Neutral, or Low (Not Confident), following the labels shown in Table 4. We apply the same approach for analyzing how challenging they found the task.

In terms of confidence in their design and confidence in applying abstraction and decomposition in the future, we find that both groups have comparable confidence. Around 30% of the experiment group feel confident in their design, 40% were neutral, and 30% are not confident. In comparison, 27% of the control group feel confident in their design, 45% are neutral, and 27% are not confident.

Regarding their confidence in applying abstraction and decomposition in the future, around 73% of the control group report a degree of confidence, and around 9% are neutral, while around 18% report a degree of non-confidence. As for the experiment group, around 70% report a degree of confidence, 10% are neutral, while 20% report a degree of non-confidence.

Regarding task difficulty, the two groups are comparable, but the experiment group has more participants who perceive it as not challenging than the control group (around 27% of participants found it to be easy vs. 9% of the control group). In general, most participants in both groups are neutral (82% of control and 50% of experiment). Only 9% of each group found the task challenging.

Observation 4: Both groups demonstrate comparable levels of confidence in their solutions and future skills, as well as similar perceptions of the task's level of challenge. However, we note that more participants in the experiment group find the tasks not to be challenging (27% vs.9%).

4.4.2 *Experiment group's specific questions.*

In terms of using ChatGPT for the given tasks, 80% of the experiment group participants find it easy or somewhat easy, while 20% find it somewhat difficult.

Regarding their confidence in the solutions provided by ChatGPT, 70% of the experiment group report a degree of confidence, 20% report a degree of non-confidence, while 10% are neutral. Regarding their confidence in their skills without ChatGPT help, 60% of the experiment group reported a degree of confidence, while the remaining 40% report a degree of non-confidence.

When asked about whether ChatGPT helped them save time solving tasks, around 90% report a degree of agreement, while only 10% report that it took more time to complete the tasks with ChatGPT. We find that around 65% agree that ChatGPT helps identify classes and attributes and methods, while around 30.0% were neutral. Only 5% indicate that it did not help.

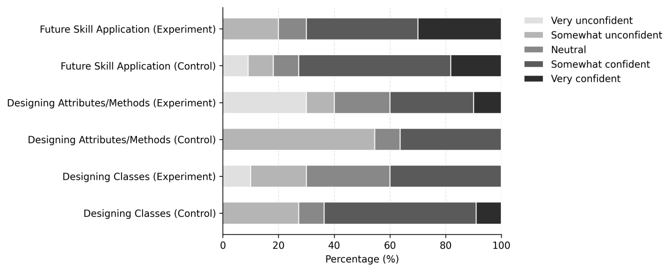
Interestingly, when asked if they thought ChatGPT helped improve their ability to think about abstraction and decomposition, the majority of participants are either neutral (40%) or disagree (40%) while only 20% agree. On the other hand, when asked if they thought ChatGPT would help them solve similar tasks in the future, the majority of participants agree (90%) while the remaining 10% are neutral. The majority of participants (60%) also agree that they would recommend ChatGPT to their peers, while the remaining 40% are neutral.

In terms of how challenging they found manually abstracting and decomposing without ChatGPT help, around 50% found it challenging, while around 40 % were neutral. Only 10 % found it somewhat easy.

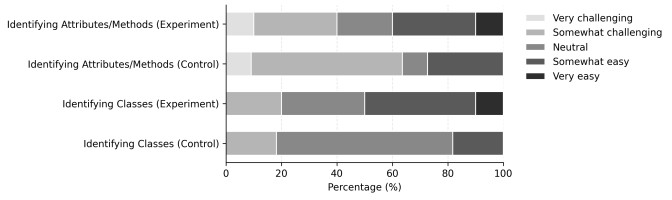
Observation 5: The experiment group find ChatGPT easy to use and time-saving, and also have high confidence in its solutions. While half the participants find it challenging to do these tasks manually, only 20% of participants agree that ChatGPT helps improve their ability to think about abstraction 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 whether they truly understood the concepts. On the other hand, we also compare the students' Confidence in Future Skills with their Retention Test Score to see whether students can accurately estimate the knowledge they have gained.



(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

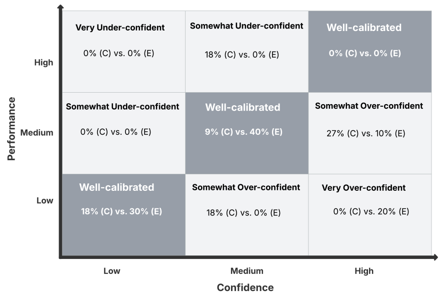
Figure 4: Control group vs. experiment group responses to post-study questionnaire

We categorize the correctness scores into three levels: High (80% or more), Medium (50-79%), and Low (<50%). We categorize the confidence as per the 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 used to categorize participants based on this matching, and summarize the results in Figure 5.

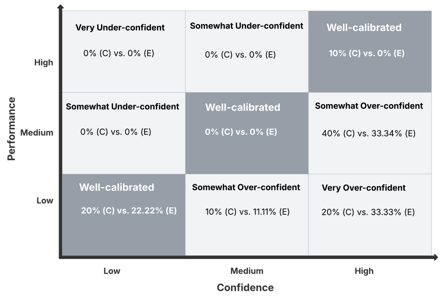
When comparing testing solution score and confidence in that solution (Figure 5a), we find that the experiment group is dominated by calibrated participants (70% of participants as opposed to 27% for the control group). On the other hand, Figure 5b shows that both groups were overconfident in their future skills.

However, when comparing between Retention Test Confidence and Retention Test Score (Figure 5c), we start seeing the control group becoming more aware of their current performance, where 50% are calibrated(as opposed to only 27% during testing).

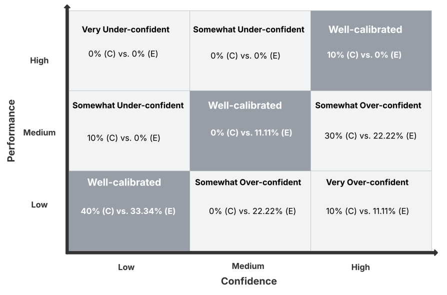
While the experiment group showed good awareness of the testing solutions' performance, they lose this awareness when it



(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 5: Comparison of Actual Performance vs. Reported Confidence. Percentages show the proportion of Control (C) and Experiment (E) participants in each category. Matching confidence to performance gives participant labels (calibrated when confidence aligns with performance).

comes to their confidence in the retention test solutions. We also start to see some participants reporting low confidence, which we did not observe in their confidence ratings after the study session.

Observation 6: When comparing confidence with actual performance, participants in the experiment group were better calibrated during the study phase (70% vs. 27%). However, this awareness declined later, and they became overconfident about their future skills. By the final questionnaire, some lost confidence, while the control group showed improved calibration.

4.5 Interaction with ChatGPT and Prompting Patterns

We now report on the interactions of the experiment group with ChatGPT. We analyze students' prompts to better understand how they use ChatGPT during the learning phase, which can potentially give us insights into how different usage can possibly affect learning. We analyze the screen recordings as well as the prompts' content to codify the prompts into categories. We use close coding [30], using a pre-defined set of prompt categories from prior work [28, 38]. We manually codify the prompts, with one author coding while a second author reviews the results. Each prompt was assigned one category, as the prompts generally expressed a single intent. For example, prompts that copied the full problem statement and explicitly asked for solutions were labeled Direct Solution. This category also included prompts that asked ChatGPT to provide all classes or attributes and methods, as well as prompts that pasted partial code and asked ChatGPT to complete the solution. A prompt like "Format of defining a class" (P28) is considered as Syntax assistance, while "Explain the enrollment function¿' (P27) is categorized as Explanatory Inquiry.

We summarize the observed prompting patterns and their frequencies in Table 5. 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 categories are "Code Correction (Fix)" and "Miscellaneous (Others)". Overall, our findings match those previously found in programming studies [28] where the most common category was also the "Solve" pattern.

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 (Obs. 1 and Obs. 2), suggesting 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). Experiment group students had favorable views on the use of ChatGPT, and were more calibrated in their confidence for the study tasks, but overconfident in their future application of these skills and in assessing their retention test performance (Obs. 4, Obs. 5, and Obs. 6). The most common prompts were to get direct answers or explanations, indicating cognitive offloading to the tool (Obs. 7).

Overall, ChatGPT gave students an initial added advantage: it saved time and helped more participants perceive the tasks as less challenging, which could make them more open to engaging with

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%

these tasks [34]. However, given the change in results after removing AI-assistance, we conclude that ChatGPT was used as a "crutch" rather than a learning tool, providing fast but fleeting help.

Take Home Message 1 - Early Integration Plus Reflection. The added initial advantages that we observe 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 students gain deep insights into their competencies (and incompetencies). Having their incompetencies acknowledged and brought to their consciousness will help students gain mastery of the needed computational skills [17]. This recommendation aligns with the long-established role of formative assessment, which focuses on feedback, in learning, where providing timely feedback and opportunities for reflection has been shown to improve student learning [4, 23, 36].

Take Home Message 2 - Prompting Techniques for Better Learning. From the prompting patterns (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, it still has a sense of reliance on the tool. We can observe that partial guidance is underutilized and could be improved. Educators need to instruct students to use more partial guidance prompts to scaffold their learning. They also need to incorporate prompting exercises in their teaching to help students learn proper prompting that can help their 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.

6 Threats to Validity

Construct Validity. We focus only on decomposition and abstraction as two key representative aspects of computational thinking. While computational thinking is hard to measure objectively in a study setting, object-oriented design tasks exemplify key 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 by using a pair-matched design followed by random assignment, and our statistical analysis confirms that there are no significant differences between the two groups.

A given problem can be designed/abstracted in different ways, making marking subjective. To overcome this, we develop a detailed rubric that aligns with the ideal solution for the tasks; however, we also provide guidelines for graders to realistically consider what is expected of students at this early stage of learning. The rubrics focus more on the design aspects rather than the exact syntax. Two independent graders grade every submission and meet to resolve any score disagreements. 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 in which students have broader access to resources. 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 Conclusion

This paper investigates whether access to ChatGPT during learning affects students' short- and long-term acquisition of computational-thinking skills. We focus on decomposition and abstraction through a controlled experiment. The results show that ChatGPT provided fast but fleeting advantages; students completed tasks faster, found them less challenging, and showed better initial performance awareness. However, these benefits did not last. The findings suggest that while ChatGPT can support learning, it should be paired with reflection and guided prompting to ensure lasting impact.

Acknowledgments

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] John D Bransford, Ann L Brown, Rodney R Cocking, et al. 2000. *How people learn*. Vol. 11. Washington, DC: National academy press.

[5] 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.

[6] 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.

[7] Gianni Fenu, Roberta Galici, Mirko Marras, and Diego Reforgiato. 2024. Exploring student interactions with AI in programming training. In *Adjunct Proceedings of the 32nd ACM Conference on User Modeling, Adaptation and Personalization*. 555–560.

[8] Aashish Ghimire and John Edwards. 2024. Coding with ai: How are tools like chatgpt being used by students in foundational programming courses. In *International Conference on Artificial Intelligence in Education*. Springer, 259–267.

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

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

[16] Rensis Likert. 1932. A technique for the measurement of attitudes. *Archives of psychology* (1932).

[17] 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.

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

[19] 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

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

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

[22] Jennifer L Osterhage, Ellen L Usher, Trisha A Douin, and William M Bailey. 2019. Opportunities for self-evaluation increase student calibration in an introductory biology course. *CBE—Life Sciences Education* 18, 2 (2019), ar16.

[23] Leanne Owen. 2016. The Impact of Feedback as Formative Assessment on Student Performance. *International journal of teaching and learning in higher education* 28, 2 (2016), 168–175.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

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

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

[31] 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.

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

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

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

[35] Italo Testa, Silvia Galano, and Oreste Tarallo. 2023. The relationships between freshmen's accuracy of self-evaluation and the likelihood of succeeding in chemistry and physics exams in two STEM undergraduate courses. *International Journal of Science Education* 45, 5 (2023), 358–382.

[36] Jagadeeswaran Thangaraj, Monica Ward, and Fiona O'Riordan. 2023. A Systematic Review of Formative Assessment to Support Students Learning Computer Programming. In *4th International Computer Programming Education Conference (ICPEC 2023)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 7–1.

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

[38] 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.

[39] 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.