Self-Regulation, Self-Efficacy, and Fear of Failure Interactions with How Novices Use LLMs to Solve Programming Problems

Lauren E. Margulieux
Georgia State University
Atlanta, Georgia, USA
lmargulieux@gsu.edu

James Prather
Abilene Christian University
Abilene, Texas, USA
james.prather@acu.edu

Brent N. Reeves
Abilene Christian University
Abilene, Texas, USA
brent.reeves@acu.edu

Brett A. Becker
University College Dublin
Dublin, Ireland
brett.becker@ucd.ie

Gozde Cetin Uzun
Georgia State University
Atlanta, Georgia, USA
gcetin1@student.gsu.edu

Dastyni Loksa
Towson University
Towson, Maryland, United States
dloksa@towson.edu

Juho Leinonen
Aalto University
Espoo, Finland
juho.2.leinonen@aalto.fi

Paul Denny
University of Auckland
Auckland, New Zealand
paul@cs.auckland.ac.nz

ABSTRACT

We explored how undergraduate introductory programming students naturalistically used generative AI to solve programming problems. We focused on the relationship between their use of AI to their self-regulation strategies, self-efficacy, and fear of failure in programming. In this repeated-measures, mixed-methods research, we examined students' patterns of using generative AI with qualitative student reflections and their self-regulation, self-efficacy, and fear of failure with quantitative instruments at multiple times throughout the semester. We also explored the relationships among these variables to learner characteristics, perceived usefulness of AI, and performance. Overall, our results suggest that student factors affect their baseline use of AI. In particular, students with higher self-efficacy, lower fear of failure, or higher prior grades tended to use AI less or later in the problem-solving process and rated it as less useful than others. Interestingly, we found no relationship between students' self-regulation strategies and their use of AI. Students who used AI less or later in problem-solving also had higher grades in the course, but this is most likely due to prior characteristics as our data do not suggest that this is a causal relationship.

CCS CONCEPTS

• **Social and professional topics** → **Computer science education**; **CS1**.

KEYWORDS

artificial intelligence; Copilot; CS1; fear of failure; generative AI; introductory programming; large language models; LLMs; metacognition; self-efficacy; self-regulation; self-regulated learning

ITiCSE 2024, July 8–10, 2024, Milan, Italy

ACM ISBN 979-8-4007-0600-4/24/07.
https://doi.org/10.1145/3649217.3653621

ACM Reference Format:
Lauren E. Margulieux, James Prather, Brent N. Reeves, Brett A. Becker, Gozde Cetin Uzun, Dastyni Loksa, Juho Leinonen, and Paul Denny. 2024. Self-Regulation, Self-Efficacy, and Fear of Failure Interactions with How Novices Use LLMs to Solve Programming Problems. In *Proceedings of the 2024 Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2024), July 8–10, 2024, Milan, Italy*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3649217.3653621

1 INTRODUCTION

Generative AI (GenAI) is poised to drastically alter programming education [14]. Some believe that GenAI will negatively impact programming education, including warnings that GenAI will render programming as we currently know it to be obsolete [44]. Conversely, many believe that programming will flourish due to GenAI. For instance, the makers of Github Copilot (a GenAI code completion tool) recently proclaimed that the tool caused a massive increase in productivity across all developers who were using it[1]. Many instructors have had similar opinions – some embracing GenAI and some rejecting it or banning its use [26].

Although early research was quick to show the capabilities of GenAI with regard to introductory programming assignments and exams [15, 16], we have much to learn about how GenAI impacts student learning [4]. If GenAI tools can answer student questions [19, 28], interpret unclear syntax error messages [27], and write whole blocks of code from scratch [38], it has the potential to scaffold student learning like never before—or remove all critical thinking from the curriculum.

Two important factors in student learning are metacognition and self-efficacy [32]. Several proposals for using GenAI to support metacognition and self-regulation have been published in the past year [36, 41], but none have measured how the use of GenAI impacts these critical skills. This paper reports on a mixed-methods, repeated measures study that explores the use of GenAI tools in an undergraduate introductory programming course. In particular, this

[1] https://www.zdnet.com/article/microsofts-github-copilot-pursues-the-absolute-time-to-value-of-ai-in-programming/

study examines student motivations for using GenAI, student patterns of use, and how GenAI use relates to students' self-regulation, self-efficacy, and fear of failure while learning programming. Our research questions were:

- **RQ1:** How do novice programmers use GenAI tools to solve programming problems?
- **RQ2:** How does the use of AI-generated solutions relate to students' self-regulation and self-efficacy in an introductory programming course?
- **RQ3:** How does fear of failure interact with students' use of AI, self-regulation, and self-efficacy?

2 RELATED WORK

2.1 GenAI in Programming Education

Programming education is already being impacted by GenAI [3], with many opportunities and challenges ahead [4]. However, given that GenAI only broke into the mainstream with the release of ChatGPT a little over a year ago, empirical evidence is still lacking while educators must consider or implement big changes. Zingaro & Porter authored a book, "Learn AI-Assisted Python Programming with GitHub Copilot and ChatGPT" [35] that Porter used in his CS1 class in Sept. 2023. In an interview, Porter articulated the role of GenAI as that of an unavoidable catalyst for change with the potential to lower the barrier for learning to program and bring broader and more diverse professionals to industry, in part by addressing the so-called "hidden curriculum problem". GenAI might help overcome these barriers by scaffolding student learning. Tools such as Copilot provide syntactically correct code completions [12], and are proficient in explaining, and often eliminating, syntax errors [27]. GenAI can also impact help-seeking behaviors [20], and students find GenAI worked examples useful for their learning [22]. However, these findings are based on expert evaluations or student perceptions. Some empirical evaluations are starting to appear, such as work on using GenAI to interpret cryptic syntax errors that found student subsequent error rates decreased [42, 43]. While not measuring learning, it is a step toward understanding how GenAI can be used to scaffold student learning.

2.2 Metacognition & Self-Regulation in Programming Education

Metacognition and self-regulation are important factors in learning that improve academic motivation and performance [2, 34, 37, 46]. Metacognition, as explained by Flavell, describes a learner's knowledge of their cognitive abilities and strategies [17], and self-regulation describes a learner's cyclical process of setting goals, monitoring their progress, and adjusting their behavior to achieve those goals [2, 46]. Additionally, Pintrich suggested that the process of self-regulated learning is influenced by prior knowledge, motivation, behavior, and context [32, 33]. Besides these foundational theories, there are two metacognitive theories in the CS domain relevant to learning with AI. Xie et al.'s theory of instruction for introductory programming skills suggests incremental instruction on four skills: 1) Tracing code, 2) Writing correct syntax, 3) Understanding templates and their use, and 4) Using templates for solving problems. According to the theory, sequencing and providing explicit instruction for these skills support students [32, 45]. The second theory, Loksa et al.'s theory of programming problem-solving, suggests teaching programming problem-solving through six stages: 1) Reinterpreting problem prompts, 2) Searching for analogous problems, 3) Searching for solutions, 4) Evaluating potential solutions, 5) Implementing a solution, and 6) Evaluating the implemented solution. Loksa et al. suggest learners define their problem-solving stage when they ask for help [31, 32]. AI tools can support students in each of these skills and stages.

Research shows self-regulation skills affect learning in CS courses. For example, students who frequently used metacognitive strategies typically have higher grades in CS1 courses (e.g., [6]) while students without metacognitive control are more likely to fail to understand and implement problems (e.g., [18]). Some of the most popular interventions to improve self-regulation include reflective activities and visualization of progress, which in turn improve students' self-efficacy and performance [37]. A recent systematic review suggests that fostering self-regulation skills among students can lead to improved learning outcomes for programming tasks [37].

2.2.1 GenAI, Metacognition, and Self-Regulation. Metacognitive knowledge is difficult to achieve in domains about which the learner has little content knowledge [17]. GenAI can provide context-aware explanations and scaffolding to bridge this gap early in the learning process. Furthermore, monitoring learning outcomes (i.e., a component of self-regulation) is complex [46], and learners can use GenAI to support the monitoring by tracking progress, highlighting areas for improvement, and creating personalized feedback. Immediate and personalized feedback is important in the development of metacognition and self-regulation strategies [8, 10].

While the opportunities and challenges of GenAI in terms of metacognition are being discussed, there is little empirical evidence to date. Denny et al. note that the developers of Codex named over-reliance as a key risk of GenAI and caution that relying too heavily on GenAI tools could hinder the development of crucial metacognitive skills [14], something that Becker et al. name as a critical competency in programming [3]. Prather et al. interviewed CS1 educators who mentioned that GenAI may push metacognitive demands upon learners early, perhaps before they are ready [38].

2.3 Self-Efficacy in Programming Education

Self-efficacy is an individual's belief in themselves to achieve specific goals across different situations [1]. According to Bandura's self-efficacy theory, it affects individuals' resilience in different situations; people with lower self-efficacy tend to avoid and quit challenging situations more frequently than those with higher self-efficacy [1]. Students with higher self-efficacy set higher goals and persist in achieving their goals while students with lower self-efficacy view setbacks and challenges as proof of a lack of ability, causing them to give up on the task, influencing their perseverance [21, 30]. In CS education, self-efficacy has been identified as a strong predictor of student success and achievement [25, 29, 30]. Correlational research has found that self-efficacy is the root cause of other student success predictors, such as gender [7, 30]. Experimental research in introductory CS courses shows that self-efficacy

interventions, which increase CS-specific self-efficacy, improve performance in those courses [21, 25, 29].

GenAI might improve self-efficacy in programming courses by providing students with support, feedback, and resources while learning. According to Bandura, feedback and performance are two of the four main influences on self-efficacy [1]. Specifically, Schunk (1991) discovered that self-efficacy improved when students received feedback that they were making progress and provided information about how to continue [39]. Further, successfully completing tasks, especially early in the learning process, increases self-efficacy, even when students are given ample support [1, 39]. AI tools can help students receive this additional support. They can provide timely, positive feedback that helps students identify and correct their errors. Additionally, they can provide resources, such as starter code or debugging support, that help students complete tasks, building their self-efficacy.

2.4 Fear of Failure

Overcoming failure is part of the learning process [23]. However, failure can damage learners' self-efficacy, which is especially fragile in novices with little experience in a field [2]. Further, fear of failure, which predicts how strongly one will try to avoid failure, varies significantly from person to person [11]. Though there is little work on fear of failure in CS education, research on academic fear of failure shows that a high fear of failure can result in task avoidance, either through procrastination or quitting [9]. Thus, fear of failure, like self-efficacy, is an important predictor of student persistence [9]. GenAI might help students work through a fear of failure by providing more support. Two aspects of programming in which students typically struggle are interpreting error messages and debugging [5, 24]. In these, students are facing failure and unsure of how to proceed, but GenAI may help in both cases.

3 METHODOLOGY

Data were collected from students in an introductory programming course at Towson University, a mid-size, public US university, that ran from January to May 2023. Students were encouraged to use GenAI tools to support their learning and problem-solving as they wished. Full IRB approval was received to run this study.

3.1 Measurements & Procedures

Students in the course were asked to describe their use of AI tools in addition to submitting their assignments six times throughout the semester. Responses to AI use questions were optional. Collecting data regularly afforded detecting changes in AI use over time. Two types of data were collected quantitatively with checkboxes:

- Timing of AI use: 1) before attempting solution, 2) before completing solution, 3) after completing solution, or 4) not used.
- Why AI was used: 1) problem statement understanding/explanation, 2) assistance to speed up writing lines of code, 3) to add a feature of the solution, 4) to add multiple features of the solution, 5) helped with debugging, 6) other.

To complement these data, researchers collected qualitative data by asking students to complete a short reflection in which they compared their code to AI-generated code and described their process for completing the assignment with AI.

To examine the effect of these patterns of AI use, several dependent variables were measured at the beginning and end of the semester to track how they changed over time.

Self-Regulation. Self-regulation was measured with the Motivated Strategies for Learning Questionnaire (MSLQ) self-regulation subscale [33], which is the most commonly used scale of self-regulation in computing and general education research [32, 37]. It uses a 7-point Likert-type scale, ranging from 7="very true of me" to 1="not very true of me".

Self-Efficacy. Self-efficacy was measured with the self-efficacy scale developed explicitly for programming students designed by Steinhorst et al. [40]. The scale has been validated with data collected from multiple institutions and compared with similar self-efficacy measures to establish validity [40]. It uses a 7-point Likert scale, ranging from 7="Strongly Agree" to 1="Strongly Disagree" with an additional choice of "No Answer".

Fear of Failure. Academic fear of failure was measured with the Fear of Failure in Learning Scale [9]. The scale has four subscales–feelings of shame, performance avoidance, learned helplessness, and self-handicapping. It uses a 5-point Likert scale, ranging from 5="Strongly Agree" to 1="Strongly Disagree".

Performance. To determine whether students' behaviors, self-regulation, self-efficacy, and fear of failure affected their performance, final grades were collected.

3.2 Participants

In total, 54 students participated in the study. For the quantitative analysis, students missing either the pre-test or post-test for the self-regulation, self-efficacy, or fear of failure measures (n = 11) and students missing more than one of the six measurements for AI use and reflection (n = 3) were excluded. As a result, the final quantitative dataset included 40 participants (i.e., 74% inclusion rate, which is reasonable for a semester-long repeated measures design). No systematic differences were found for those excluded based on final grade, demographic characteristics, or other measures. For the qualitative analysis, the data were used to explore how students used AI, which does not require complete datasets. Thus, all data available were used in the qualitative analysis. Of the 40 participants with complete datasets, 6 were missing demographic data. Learner characteristics are described below.

- Gender: 20 men, 14 women, 0 other
- Age: M = 20.85, SD = 4.5
- Employment: 76% full-time students, 24% employed
- Race: 10% Asian, 37% Black, 3% Hispanic/Latino, 31% White, 19% Mixed
- Major: 74% CS, 26% IT or Information Systems
- High school grades/GPA: M = 3.54, SD = .45
- College grades/GPA: M = 3.13, SD = .58
- College year: 53% 1st-year, 24% 2nd-year, 23% 3rd-year+
- Expected grade: 65% A, 32% B, and 3% C

The researchers examined correlations between learner characteristics and other data collected to determine whether any characteristics should be considered as covariates. Because this analysis was not primary to the research questions and because some statistical artifacts are expected when running dozens of correlations, we will report only those relationships with a medium-to-large

Table 1: Correlation trends between learner characteristics and AI use. Negative relationships indicate that a high score in one variable corresponds to a low score in the other.

Use of AI	College GPA		Expected Grade	
	r	*p*	*r*	*p*
Speed up code writing	-0.42	.02	-0.13	.47
Solve one part of solution	-0.41	.02	-.35	.04
Solve 2+ parts of solution	-0.45	.01	-0.38	.02
Help understand problem	-0.55	<.01	-0.43	.01
Explain code to me	-0.61	<.01	-0.44	.01

effect (i.e., $r > 0.4$). In this analysis, we found that women tended to use AI later in the problem-solving process, $r = 0.41$, $p = .02$. In addition, students with higher high school grades/GPA tended to use AI later, $r = 0.50$, $p = .01$. We also found a consistent trend that students with higher college grades/GPA and those who expected to earn a higher grade in the CS1 course used AI less (see Table 1).

3.3 Limitations

While the researchers tried to capture various aspects of the learning experience and environment related to the effect of using GenAI tools on self-regulation strategies, it was not practical to collect other data that might provide additional relevant information. First, because students were completing assignments outside of class, we do not know what other tools they were using to support their problem-solving, such as IDEs. Similarly, we do not know how much time they spent on assignments. While both of these variables might affect how students use AI tools, we did not want to overburden students with data collection. For the same reason, we also did not ask students about their attitudes or philosophy about using AI as a tool. While these factors might affect their behavior, we assumed that they were likely to evolve over the semester, given the novelty of GenAI. Similarly, we did not ask students about their prior experience with AI tools.

3.4 Data Analysis Procedures

Because data from the same instruments were collected from students multiple times throughout the semester, repeated measures ANOVA was the primary statistical test used. This analysis links all data from one participant together to improve the statistical power and account for non-independent data points. Thus, it affords comparisons between students and how a student changes over time. The Huynh-Feldt adjustment was used when the assumption of sphericity was violated, which is common, to make the results more conservative. Because the study used non-experimental methods, the results of these analyses should be interpreted as relational, rather than causal, effects. When correlations were used to analyze data, Spearman's correlation coefficient was used when both variables were continuous, and the point biserial correlation coefficient was used when one was continuous and the other was dichotomous.

To check assumptions for inferential statistics, the distribution, kurtosis, and skewness of each quantitative measure were examined. The kurtosis and skewness for all measures were within the -2 to +2 acceptable range. The measures of self-regulation, self-efficacy, fear of failure, and performance were all normally distributed. However, the measures of how students used AI all followed a bi-modal distribution with a peak for low AI use (used AI on 1-2 out of 6 assignments), a peak for high AI use (used AI on 5-6 out of 6 assignments), and a smaller number in the middle (used AI on 3-4 out of 6 assignments). Thus, the measurements for how students used AI were reclassified into bins for low, medium, and high AI use for analysis. The timing of AI use followed the same pattern, and students were reclassified into bins for early AI users (i.e., used AI before attempting to write code or before creating a working solution), late AI users (i.e., used AI after creating a working solution or not at all), or mid AI users (i.e., mix of both). Before students were reclassified, data were visually inspected to determine whether any trends could better describe their patterns of use (e.g., increasing use over time), but no clear trends were found for over 90% of students (i.e., about four students), leaving too little statistical power to account for potential other use cases.

To analyze qualitative data, four authors applied content analysis to review the free-form question "Please describe the process you went through to complete this assignment. If you can, include when you decided to use AI, why you decided to use the AI each time you did, and if you were successful using the AI for what you wanted. How was the AI helpful to you? Was it ever not helpful? If not, how was it not helpful?" For one of the labs, they coded one or two words for each of the 'sub-questions' and then met to compare and agree on words before coding the remaining labs. After all labs had been coded, two authors met to review that questions had been correctly coded. At that point, there were 74 unique words describing answers across all labs. Those words were then grouped according to the current categories described below in Section 4.1.

4 RESULTS

4.1 Novices' Use of AI Tools

Our first research question explored how novices used GenAI tools to solve programming problems. To address this exploratory question, we used a mixed methods approach with quantitative data related to when and for what purpose students used AI tools and qualitative data related to strategies students used for solving problems with AI. We also collected quantitative data about how useful students found AI. These data were collected six times throughout the semester to examine how timing, use, and strategies changed.

In the quantitative data, we found a consistent correlation between the timing of AI use and different types of AI use (see Table 2). These data show that students who used AI earlier in the problem-solving process also tended to use AI more consistently over the six assignments, except to speed up code writing. We also found that the perceived usefulness of AI linearly decreased over time, $F = 7.20$, $p = .01$. In addition, students classified into the high AI use group found it more useful than those classified into the low AI use group, $F = 39.31$, $p < .01$, partial eta$^2 = .72$, unsurprisingly.

To examine patterns of AI timing and use, we used a repeated measures analysis to determine whether AI timing and use were consistent within participants across time. We found no strong linear effect over time, $F = 2.12$, $p = .08$. However, the data did show a U-shaped curve, $F = 5.01$, $p = .03$. In this pattern, students tended to use AI earlier in the problem-solving process during the middle of the semester compared to the beginning or end.

Table 2: Correlation between timing of AI use and type of AI use over six assignments. Negative relationships indicate that earlier use relates to more consistent use over time

	Timing of AI Use	
	r	*p*
Speed up code writing	-0.22	.19
Solve one part of solution	-0.46	<.01
Solve 2+ parts of solution	-0.41	.01
Help understand problem	-0.57	<.01
Explain code to me	-0.52	<.01

Related to how students used AI, the patterns vary. AI was used to speed up code writing by an average of 46% of students with no change over time, $F = 1.78$, $p = .12$. Similarly, AI was used to explain code by a larger percentage of students, 66%, with no significant change over time, $F = 2.14$, $p = .07$. AI was intermittently used to solve one part of a problem by an average of 63% of students, $F = 2.69$, $p = .03$, but only the 4th order equation was significant, $F = 7.17$, $p = .01$, meaning that the direction of the slope changed three times across six time points. This result likely means that whether students used AI in this way depended on the assignment rather than following a pattern over time.

The last two types of AI use followed consistent patterns. AI was used to solve multiple parts of the problem by an average of 50% of students, $F = 4.99$, $p < .01$. This pattern followed a strong U-shaped curve, $F = 13.56$, $p < .01$, with more students using AI for this purpose mid-semester, as the complexity of problems increased, than earlier or later. In contrast, while many students used AI to understand the problem on average, 67%, this type of use decreased linearly throughout the semester, $F = 8.17$, $p = .01$.

The qualitative data provide more information about these patterns (see Figure 1). When asked why they used AI, the responses were most commonly classified as “Problem-Solving and Solution” with an average of 20 mentions across all labs. Some of the 22 items in this theme were: explain, solution, starting, example, compare, and advice. This fits well with the noted trends of students using AI to explain code and help solve pieces of the problem. The second most popular theme was “Process and Improvement” with an average of 12 mentions per lab. Some of the 17 items in this theme were: speed, save time, accelerate, explore, and unstuck. This also fits well with the trend that students used AI to speed up code writing. The theme “Understanding and Clarification” contained 9 items such as: didn’t understand, could not explain, unfamiliar solutions, need context, and confusion. This theme shows a low but consistent trend in students’ confusion at responses from the AI tools they were using.

When asked how AI was helpful, the most popular answers across all labs (except for labs 1 (15 vs 13) and lab 4 (11 vs 9)) was also “Problem-Solving and Solutions.” Students wrote about using AI to check their own complete solutions, helping them start a solution, or helping them get over the syntax barrier for a new programming language. Interestingly, this pattern changed drastically over time, peaking in labs 2 and 5. This supports our quantitative findings on students using AI to solve one part of a problem.

The most frequent category for "How Was AI Not Helpful" was "Coding Challenges and Errors" (18). The top two concepts included

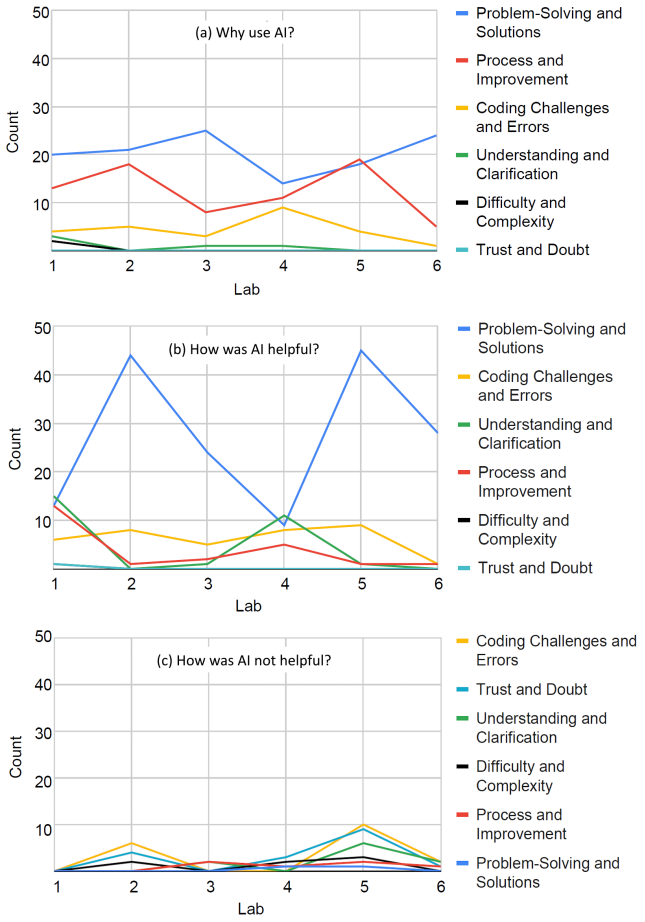

Figure 1: Occurrences of qualitative themes in student lab responses from week to week based on three questions we asked: (a) Why use AI? (b) How was AI helpful? (c) How was AI not helpful?

errors and foreign syntax. The second most identified category was "Trust and Doubt" (17), which included concepts like code looked odd, wrong language, incorrect, and inaccurate. It is not surprising that students did not trust a system that sometimes gave answers in a different programming language.

4.2 Effects of AI on Self-Regulation, Self-Efficacy, & Performance

Our second research question examined how students’ use of AI tools related to their self-regulation and self-efficacy through their introductory programming course. In these repeated measures analyses, the between-subjects factors were timing and use of AI (i.e., early, mid, and late timing and low, medium, and high use), and the within-subjects factors were self-regulation and self-efficacy, which were collected at the beginning and end of the semester. As described in the previous section (4.1), individual AI use behaviors were highly correlated (i.e., students who used AI earlier also tended to use it more consistently and for more types of tasks).

Table 3: Means and standard deviations for self-efficacy (S-E) scores (scale of 1-5) differentiated by AI use behaviors.

	Pre-Test S-E		Post-Test S-E	
	M	SD	*M*	SD
Low AI Use	3.31	.54	4.25	.73
Medium AI Use	3.03	.70	3.77	.38
High AI Use	2.77	.72	3.35	.67

Table 4: Means and standard deviations for fear of failure (FoF) (scale of 1-5) differentiated by AI use behaviors.

	Pre-Test FoF		Post-Test FoF	
	M	SD	*M*	SD
Low AI Use	3.17	.76	3.07	.97
Medium AI Use	3.23	.87	3.38	.78
High AI Use	3.84	.61	3.84	.70

While we explored all individual behaviors, the analyses followed the same pattern, so we also created an overall AI use variable as the between-subjects variable to simplify the results: low AI use n = 12, mid AI use n = 8, high AI use n = 19.

Between AI use and self-regulation, the results show no relationship. Self-regulation behaviors did not change from the pre-test, M = 3.27, SD = .40, to the post-test, M = 3.22, SD = .47, F = 0.46, p = .50. There were also no differences between different types of AI use, F = 0.52, p = .60, nor an interaction, F = 0.13, p = .88, suggesting students' self-regulation behaviors and AI use were unrelated.

Between AI use and self-efficacy, however, the results show large changes. There was a large increase across time, F = 40.60, p < .01, partial eta^2 = .53, and a difference between different levels of AI use, F = 6.10, p = .01, partial eta^2 = .25 (see Table 3). Independent subscales of Steinhorst's [40] self-efficacy in programming instrument were considered separately but followed the same pattern. Thus, the overall score was used. In this pattern, students who used AI more and earlier had lower self-efficacy, and students who used AI less and later had higher self-efficacy. There was no interaction between time and AI use, F = 1.01, p = .37, suggesting that all students' self-efficacy increased at about the same rate.

To complement these analyses, we also explored the relationship between AI use behaviors and students' final grades in the course. Consistently, we found that students who use AI later or less also had higher grades: timing, r = 0.33, p = .04; speed up code writing, r = -0.45, p < .01; solve one part of solution, r = -0.37, p = .02; solve 2+ parts of solution, r = -0.54, p < .01; help understand problem, r = -0.57, p < .01; and explaining code, r = -0.59, p < .01.

4.3 Effects of Fear of Failure on AI Use

Our last research question explored whether students' academic fear of failure interacted with their use of AI, self-regulation, self-efficacy, or performance. The latter three were analyzed with Spearman's correlation, given the continuous nature of each variable. We found that there were no correlations between fear of failure and self-regulation, self-efficacy, or performance for either pre- or post-tests. These results suggest that fear of failure is a unique characteristic of students that is unrelated to these other factors.

That fear of failure is a unique characteristic is important because, like self-efficacy and performance, it was related to how students used AI. To explore this relationship, we again used a repeated measures analysis to account for measures of fear of failure collected at the beginning and end of the semester and classified participants with the low, medium, and high AI use categories. We found no change across time, F = 0.02, p = .89, but a substantial difference between different types of AI use, F = 4.62, p = .02, partial eta^2 = .22, with no interaction effect, F = 0.32, p = .73. Students with a higher fear of failure tended to have high and earlier AI use rather than the low or medium use (see Table 4).

5 DISCUSSION & CONCLUSION

Students in our study entered the class with a baseline level of AI use and all but a few students maintained that baseline level of use relative to other students. This study provides the first empirical evidence that instructor fears of over-reliance [4, 26] might be overblown as our findings indicate at least some students use GenAI to support, not replace, their own problem-solving. Furthermore, the perceived usefulness of AI linearly decreased over time. Our qualitative findings indicate students utilize AI to help them when needed but are still interested in learning and that some do not inherently trust the answers that it gives and recognize when the AI does not provide useful or correct responses.

It has recently been postulated that LLMs have the potential to support less prepared students by providing rich, on-demand scaffolding [3, 38]. We found that student factors such as prior grades, self-efficacy, and fear of failure correlated with students' use of AI, and lower-performing students tended to use AI more. This finding could be an empirical indication that LLMs can, with proper guidance, provide scaffolding to help less prepared and less confident students. While we found that AI use is related to performance, the same factors that correlate with AI use are also predictors of performance, so these data do not suggest that AI use directly affects performance. Further experimental work is needed.

Our qualitative data also suggest that students might need help productively using GenAI (e.g., a good prompt should not give answers in the wrong language). One way to support students would be to integrate metacognitive support strategies in GenAI tools and their design [36, 41]. Emerging pedagogical techniques that integrate LLMs, such as "Prompt Problems" are attempting to do this by helping students to learn how to use AI while still learning to code [13]. This approach organizes the process of problem specification and solution evaluation in an iterative form, which fits well into emerging self-regulation frameworks that promote classic programming metacognition and self-regulation strategies and skills while using LLMs [36]. Explicitly teaching students to use GenAI serves much the same purpose that teaching them self-regulation strategies does—to help them help themselves.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under grant #1941642 and the Research Council of Finland under grant #356114.

REFERENCES

[1] Albert Bandura. 1977. Self-efficacy: Toward a Unifying Theory of Behavioral Change. *Psychological Review* 84, 2 (1977), 191.

[2] Albert Bandura. 1986. Self-regulation of Motivation and Action Through Internal Standards and Goal System. *Goal Concepts in Personality and Social Social Psychology* (1986), 19–85.

[3] Brett A. Becker, Michelle Craig, Paul Denny, Hieke Keuning, et al. 2023. Generative AI in Introductory Programming. https://csed.acm.org/large-language-models-in-introductory-programming

[4] Brett A. Becker, Paul Denny, James Finnie-Ansley, Andrew Luxton-Reilly, et al. 2023. Programming Is Hard - Or at Least It Used to Be: Educational Opportunities and Challenges of AI Code Generation. In *Proc. 54th ACM Technical Symposium on Computer Science Education V. 1* (Toronto ON, Canada) *(SIGCSE 2023)*. ACM, NY, NY, USA, 500–506.

[5] Brett A. Becker, Paul Denny, Raymond Pettit, Durell Bouchard, et al. 2019. Compiler Error Messages Considered Unhelpful: The Landscape of Text-Based Programming Error Message Research. In *Proc. Working Group Reports on Innovation and Technology in Computer Science Education* (Aberdeen, Scotland) *(ITiCSE-WGR '19)*. ACM, NY, NY, USA, 177–210. https://doi.org/10.1145/3344429.3372508

[6] Susan Bergin, Ronan Reilly, and Desmond Traynor. 2005. Examining the role of self-regulated learning on introductory programming performance. In *Proc. 1st Intl. Workshop on Comp. Ed. Research*. 81–86.

[7] Sylvia Beyer. 2014. Why are women underrepresented in Computer Science? Gender differences in stereotypes, self-efficacy, values, and interests and predictors of future CS course-taking and grades. *Computer Science Education* 24, 2-3 (2014), 153–192.

[8] Deborah L Butler and Philip H Winne. 1995. Feedback and self-regulated learning: A theoretical synthesis. *Review of Educational Research* 65, 3 (1995), 245–281.

[9] Beomkyu Choi. 2021. I'm afraid of not succeeding in learning: Introducing an instrument to measure higher education students' fear of failure in learning. *Studies in Higher Education* 46, 11 (2021), 2107–2121.

[10] Yiu Bun Chung and Mantak Yuen. 2011. The role of feedback in enhancing students' self-regulation in inviting schools. *J. of Invitational Theory and Practice* 17 (2011), 22–27.

[11] Marco Estêvão Coreia, António Rosado, Sidónio Serpa, and Vítor Ferreira. 2017. Fear of failure in athletes: Gender, age and type of sport differences. *Revista Iberoamericana de Psicología del Ejercicio y el Deporte* 12, 2 (2017), 185–193.

[12] Paul Denny, Viraj Kumar, and Nasser Giacaman. 2023. Conversing with Copilot: Exploring Prompt Engineering for Solving CS1 Problems Using Natural Language. In *Proc. of the 54th ACM Tech. Symp. on Comp. Sci. Ed. V. 1* (Toronto ON, Canada) *(SIGCSE 2023)*. ACM, NY, USA, 1136–1142. https://doi.org/10.1145/3545945.3569823

[13] Paul Denny, Juho Leinonen, James Prather, Andrew Luxton-Reilly, et al. 2024. Prompt Problems: A New Programming Exercise for the Generative AI Era. In *Proc. 55th ACM Tech. Symp. on Comp. Sci. Ed. V. 1* (Portland, OR, USA) *(SIGCSE 2024)*. ACM, NY, USA, 7 pages. https://doi.org/10.1145/3626252.3630909

[14] Paul Denny, James Prather, Brett A. Becker, James Finnie-Ansley, et al. 2024. Computing Education in the Era of Generative AI. *Commun. ACM* 67, 2 (jan 2024), 56–67. https://doi.org/10.1145/3624720

[15] James Finnie-Ansley, Paul Denny, Brett A. Becker, Andrew Luxton-Reilly, et al. 2022. The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming. In *Proc. 24th Australasian Computing Education Conf.* (Virtual Event, Australia) *(ACE '22)*. ACM, NY, NY, USA, 10–19.

[16] James Finnie-Ansley, Paul Denny, Andrew Luxton-Reilly, Eddie Antonio Santos, et al. 2023. My AI Wants to Know If This Will Be on the Exam: Testing OpenAI's Codex on CS2 Programming Exercises. In *Proc. 25th Australasian Computing Education Conf.* (Melbourne, Australia) *(ACE '23)*. ACM, NY, NY, USA, 97–104.

[17] John H Flavell. 1979. Metacognition and cognitive monitoring: A new area of cognitive–developmental inquiry. *American Psychologist* 34, 10 (1979), 906.

[18] Marietjie Havenga. 2015. The role of metacognitive skills in solving object-oriented programming problems: a case study. *TD: The Journal for Transdisciplinary Research in Southern Africa* 11, 1 (2015), 133–147.

[19] Arto Hellas, Juho Leinonen, Sami Sarsa, Charles Koutcheme, et al. 2023. Exploring the Responses of Large Language Models to Beginner Programmers' Help Requests. In *Proc. 2023 ACM Conf. on International Computing Education Research - Volume 1* (Chicago, IL, USA) *(ICER '23)*. ACM, NY, NY, USA, 93–105.

[20] Irene Hou, Sophia Mettille, Owen Man, Zhuo Li, et al. 2024. The Effects of Generative AI on Computing Students' Help-Seeking Preferences. In *Proc. 26th Australasian Computing Education Conf.* (Sydney, NSW, Australia) *(ACE '24)*. ACM, NY, NY, USA, 39–48. https://doi.org/10.1145/3636243.3636248

[21] Christopher D Hundhausen, Anukrati Agrawal, and Pawan Agarwal. 2013. Talking about code: Integrating pedagogical code reviews into early computing courses. *ACM Transactions on Computing Education (TOCE)* 13, 3 (2013), 1–28.

[22] Breanna Jury, Angela Lorusso, Juho Leinonen, Paul Denny, et al. 2024. Evaluating LLM-Generated Worked Examples in an Introductory Programming Course. In *Proc. 26th Australasian Computing Education Conf.* (Sydney, NSW, Australia) *(ACE '24)*. ACM, NY, NY, USA, 77–86. https://doi.org/10.1145/3636243.3636252

[23] Manu Kapur. 2015. Learning from productive failure. *Learning: Research and Practice* 1, 1 (2015), 51–65.

[24] Ioannis Karvelas, Annie Li, and Brett A. Becker. 2020. The Effects of Compilation Mechanisms and Error Message Presentation on Novice Programmer Behavior *(SIGCSE '20)*. ACM, NY, NY, USA, 759–765.

[25] Päivi Kinnunen and Beth Simon. 2011. CS majors' self-efficacy perceptions in CS1: results in light of social cognitive theory. In *Proc. 7th International Workshop on Computing Education Research*. 19–26.

[26] Sam Lau and Philip Guo. 2023. From "Ban It Till We Understand It" to "Resistance is Futile": How University Programming Instructors Plan to Adapt as More Students Use AI Code Generation and Explanation Tools Such as ChatGPT and GitHub Copilot. In *Proc. 2023 ACM Conf. on International Computing Education Research - Volume 1* (Chicago, IL, USA) *(ICER '23)*. ACM, NY, NY, USA, 106–121.

[27] Juho Leinonen, Arto Hellas, Sami Sarsa, Brent Reeves, et al. 2023. Using Large Language Models to Enhance Programming Error Messages *(SIGCSE 2023)*. ACM, NY, NY, USA, 563–569. https://doi.org/10.1145/3545945.3569770

[28] Mark Liffiton, Brad E Sheese, Jaromir Savelka, and Paul Denny. 2024. CodeHelp: Using Large Language Models with Guardrails for Scalable Support in Programming Classes. In *Proc 23rd Koli Calling Int. Conf. on Comp. Ed. Research* (Koli, Finland). ACM, NY, USA, 11 pages. https://doi.org/10.1145/3631802.3631830

[29] Alex Lishinski and Aman Yadav. 2021. Self-evaluation interventions: Impact on self-efficacy and performance in introductory programming. *ACM Transactions on Computing Education (TOCE)* 21, 3 (2021), 1–28.

[30] Alex Lishinski, Aman Yadav, Jon Good, and Richard Enbody. 2016. Learning to program: Gender differences and interactive effects of students' motivation, goals, and self-efficacy on performance. In *Proc. 2016 ACM Conf. on International Computing Education Research*. 211–220.

[31] Dastyni Loksa, Amy J Ko, Will Jernigan, Alannah Oleson, et al. 2016. Programming, problem solving, and self-awareness: Effects of explicit guidance. In *Proc. 2016 CHI Conf. on Human Factors in Comp. Sys.* 1449–1461.

[32] Dastyni Loksa, Lauren Margulieux, Brett A. Becker, Michelle Craig, et al. 2022. Metacognition and Self-Regulation in Programming Education: Theories and Exemplars of Use. *ACM Trans. Comput. Educ.* 22, 4, Article 39 (Sept 2022), 31 pages. https://doi.org/10.1145/3487050

[33] Paul R Pintrich and Elisabeth V De Groot. 1990. Motivational and self-regulated learning components of classroom academic performance. *J. of Educational Psychology* 82, 1 (1990), 33.

[34] Paul R Pintrich and Dale H Schunk. 2002. *Motivation in education: Theory, research, and applications*. Prentice Hall.

[35] Leo Porter and Daniel Zingaro. 2023. *Learn AI-Assisted Python Programming with GitHub Copilot and ChatGPT*. Manning, Shelter Island, NY, USA.

[36] Prajish Prasad and Aamod Sane. 2024. A Self-Regulated Learning Framework using Generative AI and its Application in CS Educational Intervention Design. In *Proc. 55th ACM Tech. Symp. on Comp. Sci. Ed.* (Portland, OR, USA) *(SIGCSE 2024)*. ACM, NY, USA, 7 pages.

[37] James Prather, Brett A. Becker, Michelle Craig, Paul Denny, et al. 2020. What do we think we think we are doing? Metacognition and self-regulation in programming. In *Proc. 2020 ACM Conf. on Int. Comp. Ed. Research*. 2–13.

[38] James Prather, Paul Denny, Juho Leinonen, Brett A. Becker, et al. 2023. The Robots Are Here: Navigating the Generative AI Revolution in Computing Education. In *Proc. 2023 Working Group Reports on Innovation and Technology in Computer Science Education* (Turku, Finland) *(ITiCSE-WGR '23)*. ACM, NY, NY, USA, 108–159. https://doi.org/10.1145/3623762.3633499

[39] Dale H Schunk. 1991. Self-efficacy and academic motivation. *Educational Psychologist* 26, 3-4 (1991), 207–231.

[40] Phil Steinhorst, Andrew Petersen, and Jan Vahrenhold. 2020. Revisiting self-efficacy in introductory programming. In *Proc. 2020 ACM Conf. on International Computing Education Research*. 158–169.

[41] Lev Tankelevitch, Viktor Kewenig, Auste Simkute, Ava Elizabeth Scott, et al. 2023. The Metacognitive Demands and Opportunities of Generative AI. arXiv:2312.10893 [cs.HC]

[42] Andrew Taylor, Alexadra Vassar, Jake Renzella, and Hammond Pearce. 2024. dcc –help: Transforming the Role of the Compiler by Generating Context-Aware Error Explanations with Large Language Models. In *Proc. 55th ACM Technical Symposium on Computer Science Education V. 1* (Portland, OR, USA) *(SIGCSE 2024)*. ACM, NY, USA, 7 pages.

[43] Sierra Wang, John C. Mitchell, and Chris Piech. 2024. A Large Scale RCT on Effective Error Messages in CS1. In *Proc. 55th ACM Tech. Symp. on Comp. Sci. Ed. V. 1* (Portland, OR, USA) *(SIGCSE 2024)*. ACM, NY, USA, 7 pages.

[44] Matt Welsh. 2022. The End of Programming. *Commun. ACM* 66, 1 (2022), 34–35.

[45] Benjamin Xie, Dastyni Loksa, Greg L Nelson, Matthew J Davidson, et al. 2019. A theory of instruction for introductory programming skills. *Computer Science Education* 29, 2-3 (2019), 205–253.

[46] Barry J Zimmerman. 1990. Self-regulated learning and academic achievement: An overview. *Educational Psychologist* 25, 1 (1990), 3–17.