Why do CS1 Students Think They're Bad at Programming? Investigating Self-Efficacy and Self-Assessments at Three Universities

Jamie Gorson
Northwestern University
Evanston, IL
jgorson@u.northwestern.edu

Eleanor O'Rourke
Northwestern University
Evanston, IL
eorourke@northwestern.edu

ABSTRACT

Undergraduate computer science (CS) programs often suffer from high dropout rates. Recent research suggests that self-efficacy – an individual's belief in their ability to complete a task – can influence whether students decide to persist in CS. Studies show that students' self-assessments affect their self-efficacy in many domains, and in CS, researchers have found that students frequently assess their programming ability based on their expectations about the programming process. However, we know little about the specific programming experiences that prompt the negative self-assessments that lead to lower self-efficacy. In this paper, we present findings from a survey study with 214 CS1 students from three universities. We used vignette-style questions to describe thirteen programming moments which may prompt negative self-assessments, such as getting syntax errors and spending time planning. We found that many students across all three universities reported that they negatively self-assess at each of the thirteen moments, despite the differences in curriculum and population. Furthermore, those who report more frequent negative self-assessments tend to have lower self-efficacy. Finally, our findings suggest that students' perceptions of professional programming practice may influence their expectations and negative self-assessments. By reducing the frequency that students self-assess negatively while programming, we may be able to improve self-efficacy and decrease dropout rates in CS.

CCS CONCEPTS

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

KEYWORDS

Self-efficacy; Self-assessments; CS1; Persistence

ACM Reference Format:
Jamie Gorson and Eleanor O'Rourke. 2020. Why do CS1 Students Think They're Bad at Programming? Investigating Self-Efficacy and Self-Assessments at Three Universities. In *Proceedings of the 2020 International Computing Education Research Conference (ICER '20), August 10–12, 2020, Virtual Event, New Zealand*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3372782.3406273

ICER '20, August 10–12, 2020, Virtual Event, New Zealand

ACM ISBN 978-1-4503-7092-9/20/08…$15.00
https://doi.org/10.1145/3372782.3406273

1 INTRODUCTION

While failure rates in CS1 courses have improved over the past decade, computer science (CS) programs still struggle to retain students in the major [9, 11, 12, 45, 54]. Furthermore, women and underrepresented minorities drop out at a higher rate than their counterparts, amplifying the lack of diversity in CS programs [4, 19, 23]. This has led researchers to investigate why students drop out of CS [1, 8, 29, 42, 55]. In many domains, studies show that low self-efficacy is associated with higher dropout rates from college majors and can impact career choice [13, 33, 41, 49]. Self-efficacy is defined as a person's perception of her ability to succeed in a particular domain [7]. Recently, researchers have found that self-efficacy and negative beliefs about programming abilities may contribute to the high dropout rates in computer science [34, 51].

Foundational research in psychology has shown that students' self-assessments of their ability have a strong impact on self-efficacy [6]. While evaluating knowledge and monitoring progress are an important part of self-regulation [3, 15, 46], students who frequently engage in negative self-assessments are likely to have lower self-efficacy [7]. Researchers have found that CS1 students assess their programming ability often and may negatively self-assess during unexpected moments, such as when stopping to think or plan [24] or after a positive programming performance [30]. These negative evaluations often occur when students' problem-solving process does not match their performance expectations [30]. For example, a student might think she has performed poorly on a programming problem if it takes longer to complete than she expected. Since self-assessments strongly influence self-efficacy, the frequency of negative self-evaluations in CS could contribute to student dropout in CS.

Recent studies have identified a number of criteria that students use to evaluate their programming ability. For example, Lewis et al. found that CS1 students assess their ability based on prior experience, speed, and grades [34], and Kinnunen and Simon found that they use speed, social comparisons, and whether their program works [30]. We identified an even broader set of criteria in our prior work, for example finding that some students think looking up syntax and getting errors are signs of low ability [24]. Surprisingly, many of these criteria contradict the practices that instructors think are important to novice success [21, 27, 37], or the practices of professional programmers [27, 31, 40, 44, 50, 53]. This suggests that students may have inaccurate expectations of the programming process, which could impact how they self-assess. While this prior work provides important insights into the criteria that students use to evaluate ability, we do not know what programming moments prompt negative self-assessments.

To overcome these challenges, we conducted a survey study with CS1 students at three universities. The goals of this study were to (1) identify the programming moments that cause students to negatively self-assess, (2) identify any differences in these moments across universities, (3) identify whether there is a relationship between negative self-assessments and student self-efficacy, and (4) explore whether students with inaccurate perceptions of professional programming practice tend to negatively self-assess during those natural moments in the programming process. We designed vignette-style survey questions to measure whether students negatively self-assess at thirteen programming moments that occur naturally during professional practice, such as struggling with errors or spending time planning. We found that students at all three universities reported that they negatively self-assess at each of the moments, with some moments eliciting more consistently negative reactions than others. Interestingly, we found few differences across the universities, showing that these self-assessment moments generalize across contexts. Additionally, we found that students who negatively respond to the self-assessment vignette questions more frequently and strongly tend to have a significantly lower self-efficacy in their programming course. Finally, we found that students' perceptions of professional programming practice may help explain some of the self-assessment moments, but other factors may contribute as well. These findings highlight the prevalence of self-assessments across university contexts, and suggest that we may be able to improve student self-efficacy by reducing self-assessments in response to moments that arise as a natural part of the programming process.

2 BACKGROUND

A number of studies have explored aspects of self-efficacy in the domain of CS. We first present a background on self-efficacy theory, discussing both the sources that inform self-efficacy and how self-efficacy influences student persistence. We then describe how self-efficacy theory has been specifically applied in the domain of CS. Finally, as self-assessments inform self-efficacy, we discuss recent literature on the criteria that CS1 students use when making self-assessments about their programming ability.

2.1 Self-efficacy Theory

Bandura defined self-efficacy as an individual's judgment of their ability to execute tasks or achieve mastery in a particular domain [7]. The theory states that self-efficacy is based on four principal sources of information: (1) enactive attainments, or the results of performing tasks related to mastering the subject, (2) vicarious experiences observing others performing subject-related tasks, (3) verbal persuasion from others, like words of encouragement, and (4) physiological states, like stress [6]. Self-assessments are one way that people interpret their performance on a task, which influences their enactive attainments. Enactive attainments are the most influential of these sources because the information comes from the performance of tasks that contribute to mastering the domain. Therefore, when students evaluate their performances as successful, their self-efficacy increases, but when students view their performances as failures, their self-efficacy decreases.

Researchers in psychology have established a strong relationship between self-efficacy and persistence. Studies show that students with higher self-efficacy are more likely to persist through challenges and be more resilient through their long-term goals [7, 49]. Researchers have directly explored how self-efficacy influences career choice, and found that students with lower self-efficacy are less likely to persist through their college majors [13, 26, 32, 33, 41]. For example, Lent et al. surveyed undergraduates in technical and science majors and found that students with higher self-efficacy were more likely to persist in their field [32]. Since higher self-efficacy has been found to increase students' persistence in their college major, self-efficacy may contribute to the dropout issues in CS.

2.2 Self-efficacy in CS

Researchers in computer science education have, in fact, identified a relationship between self-efficacy and student persistence in the CS major [34, 39, 51]. In a semester long study, Lewis et al. found that one factor students consider in their decision to major in CS is their perception of their CS ability. Other studies used formal self-efficacy surveys to examine the relationship between self-efficacy and persistence [39, 51]. For example, Miura found that students with higher self-efficacy were more likely to take a CS course in college, which may help to explain the gender diversity issues in CS since male students have higher self-efficacy on average [39]. These studies highlight the importance of understanding the factors that impact self-efficacy for improving retention in CS programs.

Researchers have identified many factors that correlate with self-efficacy, including: previous programming experiences [28, 47], gender [5, 35, 39], computer literacy [5, 28], goal orientation and metacognitive strategies [35], understanding of programming concepts [25, 47], and sense of belonging in CS [52]. For example, Rammaligan used a survey to measure students' comprehension of software programs and found that students with better mental models of programming concepts were more likely to report a higher programming self-efficacy [47]. Similarly, Askar and Davenport found that students with more years of computer experience had higher programming self-efficacy, and that males on average had higher programming self-efficacy than females [5]. These studies show a number of different and diverse factors that contribute to students' self-efficacy in computer science. In this paper, we focus on one particularly influential factor, namely students' self-assessments of their programming ability.

2.3 Self-assessments in CS

Self-assessments, or students' evaluations of their performance on tasks, are a particularly important source of information for self-efficacy [6]. Recently, researchers have begun to explore self-assessments in CS1, finding that students assess their ability frequently [24, 30]. This may be because many CS1 students are new to the field [48] and feel pressure to choose a college major [34]. While evaluating knowledge and monitoring progress are a necessary part of the self-regulated learning process [3, 15, 46], studies show that CS1 students sometimes negatively self-assess in response to natural parts of the programming process and after positive programming episodes [24, 30]. Kinnunen and Simon suggest that these negative evaluations may occur when a programming experience does not match the student's expectations. For example, students may negatively self-assess if they take longer to solve a problem than expected, even if they complete it successfully. These findings

suggest that CS1 students may have unnecessarily high or inaccurate expectations of the programming process, which could lead to more frequent negative self-assessments. Researchers have not established a relationship between self-assessments and self-efficacy in CS, a gap that we address through this paper. However, self-efficacy theory in other domains suggests that frequent negative evaluations will result in lowered self-efficacy.

To better understand these self-assessments, researchers have started to explore the criteria that students use to evaluate themselves. Kinnunen and Simon conducted a series of interviews and identified that CS1 students use these criteria in their self-assessments: speed, social comparisons, and whether their program works [30]. Lewis et al. also conducted an interview study and identified three similar criteria: perceived prior experience, speed, and grades [34]. In our previous work, we built on these findings by creating a more thorough list of criteria, including the ability to: remember syntax, solve a problem without stopping to think, and solve a problem without asking for help [24]. Through a survey with CS1 students, we found that some students agreed with each of the criteria. Interestingly, we also found significant variation in agreement with the self-assessment criteria, showing that CS1 students define programming intelligence in different ways [24].

Surprisingly, many of the self-assessment criteria that we identified in that study are an expected part of professional programming practice [24]. For example, studies of expert programmers have shown that they frequently plan [21, 50, 53], get errors [17, 44], and ask for help [27, 31, 40]. However, students negatively self-assess using these criteria. For example, some students consider *spending time planning* as indicative of lower programming intelligence [24], even though planning is widely documented as part of the professional process [50, 53] and as a good practice for novices [21]. Given these misalignments, we suspect that students may not have accurate perceptions of professional programming practices.

While many of the criteria from the previous study relate to moments in the programming process, we only asked about criteria broadly rather than situated in the context of a programming session. Specifically, our previous questions were framed around overall programming intelligence and not in-the-moment, task-specific self-assessments. Therefore, we do not know if students self-assess in response to specific programming moments. Additionally, our previous questions asked about the criteria broadly, and did not distinguish potentially important nuances. For example, we found that some students consider *getting errors* as a sign of low ability, but we did not distinguish between complex semantic errors and simple syntax errors [24]. In this paper, we dive deeper into the student programming experience by studying the specific programming moments that prompt students to negatively self-assess and how these self-assessments interact with their overall self-efficacy. We also explore students' perceptions of professional programmers to understand how these relate to their self-assessments.

3 STUDY DESIGN AND METHODS

In this paper, we ask four research questions about the programming moments that cause students to negatively assess their ability, to help us better understand the student programming experience:

RQ1: *When presented with scenarios of programming moments, which do students say cause them to negatively self-assess?*

RQ2: *Are there any differences in the moments that students say cause them to negatively self-assess across university contexts?*

RQ3: *Do students who report negatively self-assessing in response to more moments have a lower self-efficacy in their CS course?*

RQ4: *Do students' perceptions of professional programming practices correlate with the moments that cause them to negatively self-assess?*

Through these questions, we aim to identify the programming moments that prompt students to negatively self-assess, and uncover any differences across university contexts. We also aim to measure the relationship between self-assessments and student self-efficacy, and determine whether students' inaccurate perceptions of professional programming practice contribute to their self-assessments.

To answer these questions, we conducted a survey study. We chose a survey methodology because it ensures that we can measure each student's reactions to the same set of moments, in comparison to an observational approach where similar events might not naturally occur for each student. We were interested in learning how responses might vary across different populations of students, so we conducted our study at three different universities. Additionally, we conducted follow-up interviews with a small portion of the survey participants to learn more about their thought process when answering the survey questions.

3.1 Survey Design

We designed a survey with a number of sections, each measuring a different construct. We measured student self-efficacy in their programming course first, to ensure that responses would be unbiased by later sections of the survey which describe potentially challenging programming moments. We adapted the five-question general academic efficacy survey from [38] by changing their references to "coursework" to "in my CS1 class" (where "CS1" is replaced with the name of the student's course). For example, we asked students to rate how much they agree with the statement, *"I'm certain I can master the skills taught in my CS1 class this term"* on a 6-point forced-choice Likert scale. We chose to use a survey that measured self-efficacy in their CS course, instead of a survey about programming self-efficacy in general, because course expectations are well-defined and consistent for all of the students. In comparison, students may interpret the definition of "good" in a programming self-efficacy survey differently. For example, some students might report low self-efficacy if they believe they are not good at programming relative to experts, even if they have high self-efficacy about their ability to learn CS and succeed in the course.

Next, we measured whether specific programming moments elicit negative self-assessments. We designed vignette-style survey questions that provide short descriptions of specific programming moments to convey vivid descriptions and then asked students to report how they feel when they experience similar moments. To design the vignette survey questions, we curated a list of self-assessment moments. We define a self-assessment moment as a point in the programming process that might elicit student evaluations about how they are doing on a task. We started with the list of moments that are associated with the programming intelligence self-assessment criteria from our previous study [24] and added additional moments based on needfinding interviews with students.

Self-assessment moment	Vignette
Getting a simple error	Jen is working on her programming assignment. She runs her code. An error pops up. She immediately realizes that she left out a parenthesis. She adds the parentheses and her code runs successfully. Jen thinks: "That was a stupid mistake. A good programmer wouldn't make small mistakes like this."
Starting over	Nadia is working on a hard homework problem. She plans out a solution. She writes a few lines of code. She realizes that her approach to the problem will not work. She decides to start over. Nadia feels frustrated that she wasted time. She erases all her code and starts again.
Not understanding an error message	Frank is working on a programming problem. He runs his code. An error pops up. Frank has no idea what the error message means. He is not sure what to try next. He thinks: "I'm doing so badly, I don't even know what this message means".
Stopping programming to plan	Diego starts working on a programming problem. He writes a few lines of code. He realizes that he is confused about what to do next. He pauses and plans his next steps. Diego wishes that he did not have to stop writing code to plan.
Getting help from others	Julie is working on her homework assignment. She gets stuck. Julie meets with an instructor to get help in order to finish the assignment.
Spending a long time on a problem	Tamyra is working really hard on a programming problem. She solves the problem. She is proud of herself. Tamyra looks at the clock and realizes how many hours she spent on the problem. She feels upset because it took her so long to finish it.
Not knowing how to start	Miguel reads his programming homework assignment. He opens up the editor but has no idea where to start. Miguel feels disappointed in himself because he doesn't even know how to approach the problem.
Using resources to look up syntax	Arjun is working on a programming problem. He can't remember the syntax. He uses Google to look up the syntax. He is disappointed that he could not remember the syntax on his own.
Spending time planning at the beginning	Jake is unsure how to begin his programming assignment. He spends time planning how to solve the problem. Eventually, Jake comes up with a plan and begins to write code. Jake wishes that he did not need to spend as much time planning before writing code.
Spending a long time looking for a simple error	Isabella is working on a challenging problem. She runs into an error. She looks through the code but can't find it. After a long time, she realizes that it was a small typo. She thinks to herself: "Wow. I am so bad at programming. A good programmer would not take so long to find a simple error."
Struggling to fix errors	Daniel is working on his programming homework. He runs his code and gets an error. He struggles to fix the error for a long time. When he runs the code, another error comes up. He struggles again. Eventually, he fixes it. Then, a different error comes up.
Not able to finish in time expected	Sirena is working on her programming assignment. She expects to finish it in one night. After a while, she decides to stop working because it got late. She feels upset that she was not able to finish it in one night.
Does not understand the problem statement	Fatima reads her programming homework assignment. She does not understand what the problem statement is asking her to do. She feels upset and frustrated because she can't even understand the question.

Table 1: The thirteen self-assessment moments and the vignettes that we included on the survey.

Then, we designed vignettes that describe a character encountering each of the programming self-assessment moments. To refine the vignettes and ensure that they represented the moments accurately, we conducted preliminary user studies that asked students to talk out loud while reading the questions to reveal their interpretations. The moments and the associated vignettes can be seen in Table 1.

After each vignette, we asked participants to rate if they negatively evaluate themselves when they experience a similar moment while programming. For example, the statement following the *using resources to look up syntax* vignette is: *I feel like I'm not doing well on a problem when I can't remember the syntax and have to look it up. (6-point Likert scale)*. In this paper, we call these questions *self-assessment vignette questions*. Since we wanted to understand when students were negatively evaluating themselves, potentially causing feelings of low self-efficacy, we only asked about negative reactions to the vignettes. We also included a check question after two of the vignettes that instructed participants to enter a specific response to ensure that they were reading the survey questions carefully. To control for biases in students' reactions to the questions based on gender or ethnicity of the character, we randomized the names of the characters for the survey participants. We also randomized the order that the vignettes appeared to control for earlier vignettes affecting responses to later vignettes.

We chose to use vignette-style survey questions because it allows for detailed descriptions of moments in a programming process to elicit students' memories of similar experiences. This style of survey has been used extensively in the field of psychology as an approach to reduce self-report biases, particularly with survey questions in cases where participants might be concerned about social approval from the researcher [2, 18]. Vignette questions are also useful for questions about decisions, judgements or situations that the participants may not have previously considered, like when students self-assess while programming. In those cases, asking a direct question might lead to inaccurate results compared to a vignette. However, since vignette surveys are hypothetical, they can not generate the same emotional experience as a laboratory or field experiment [16, 20, 43].

The last section of the survey measures whether students believe that professionals encounter the moments described in the self-assessment vignette questions. We hypothesize that students strive to be like professionals in their field and thus their perceptions of professional programmers might correlate with the moments that prompt them to negatively self-assess. Studies show that professional programmers encounter many of the moments described in the vignette questions [27, 44, 50], however, the results from our previous study indicated that students do not believe that "good

programmers" encounter the self-assessment moments [24]. We decided to ask about professional programmers rather than more advanced students or "good programmers" because we wanted to capture students' perceptions of undeniable experts and not of people who are better but may still be learning.

For each of the thirteen self-assessment moments, we asked students to finish a sentence about professional programmers. For example, the question associated with the *using resources to look up syntax* self-assessment moment states:

> Professional programmers:
> - often forget the exact syntax and use Google and other resources to help them remember.
> - remember the syntax they need and rarely have to look it up.

In each question, one option states that the self-assessment moment occurs in professional programming practice, aligning with research on professional practice. The other option states that the self-assessment moment rarely or never occurs in professional practice, which contradicts research on professional practice. The questions are presented in a randomized order to control for earlier questions affecting students' responses to later questions. The options following the questions are also presented in a randomized order to control for potential bias caused by the order in which students read them.

We included a few additional sections in the survey, which we will not discuss in this paper because they are out of the scope of these research questions. These sections included an open-ended response question, a programming intelligence mindset survey, and a sense of belonging in CS survey. Each vignette was also followed by a second question that asked students how they would evaluate the performance of the character in the vignette. We do not report on this question because our paper focuses on understanding students' evaluations of themselves rather than of others.

The full survey can be found at https://bit.ly/2B7irzC.

3.2 Participants

We recruited participants from CS1 courses at three universities that serve different populations of students, all located within the same metropolitan area in the midwestern United States. University 1 is a highly selective, private, research-focused university (R1) that is mid-sized (8,200 undergraduates) and primarily residential. University 2 is a selective, private research university (R2) that is larger (14,500 undergraduates) and primarily nonresidential. University 3 is a less selective, public university with masters programs (M1) that is mid-sized (6,400 undergraduates) and primarily nonresidential. University 3 has been recognized as the most diverse university in the midwest. See Table 2 for the demographics of our participants at each university.

A total of 283 students took our survey over a three week period in the middle of their programming courses. We removed data from participants who did not answer the check questions correctly. This left us with a total of 214 participants, with 78 from University 1, 57 from University 2, and 79 from University 3. We recruited participants from two introductory CS courses at University 1, one that is required for all CS majors and another that is not part of the major sequence. We recruited students from the introductory programming course at University 2. We recruited students from the first two classes in the introductory computer science sequence for CS majors at University 3. Since we were most interested in studying self-evaluations across the different school contexts, we focus our analysis at the school level rather than the class level.

School	#	F	AA	A	LA	W	O	2+
University 1	78	58%	6%	35%	4%	42%	4%	8%
University 2	57	32%	7%	23%	10%	47%	5%	5%
University 3	78	17%	7%	24%	31%	24%	8%	6%

 # = Total responses, F = Female, AA = African American, A = Asian, LA = Latin American, W = White, O = Other, 2+ = Two or More Races.

Table 2: Demographic data from survey participants for each University in percentages.

3.3 Survey Procedure

All participants completed the survey in a proctored room with a researcher present. However, the recruitment and distribution of surveys varied based on the constraints of the university policies, course structure, and instructor preferences. For some of the classes, a researcher announced the survey in class and posted the announcement on the class discussion board, directing interested students to take the survey with the researcher at a designated time and place on campus, outside of class time. For other classes, a researcher announced the survey in class and interested students were given time to fill out the survey at the end of class. For the remainder of the classes, a researcher announced the survey in the course lab section and interested students could fill out the survey during or directly after that lab section. These different recruitment methods may have impacted the participation rates in the classes, so our findings may be subject to participation bias. However, given that this is the first study of self-assessments across multiple universities, we believe it still makes an important contribution despite this limitation. All survey participants provided informed consent for the survey and were compensated for their time with a $5 gift card of their choice.

3.4 Follow-up Interviews Procedure

We conducted semi-structured follow-up interviews to understand whether students interpreted the vignette-style survey questions in the ways we expected. We also explored whether the vignettes encouraged students to recall similar moments in their own programming experiences. Finally, we used the interviews to investigate students' rationales for their responses. We randomly selected survey respondents from each of the three universities to participate in the follow-up interviews from the participants who indicated interest in participating in future research studies. Of the students we contacted, we interviewed 6 participants from University 1, 4 from University 2 and 3 from University 3.

Students who agreed to participate in the interview met with the first author individually, either via video conference or in-person at their respective university. They first signed a consent form allowing for audio and video recording of the interview. Then, the researcher gave the participants their previously completed survey and asked them to re-read each of the vignette questions.

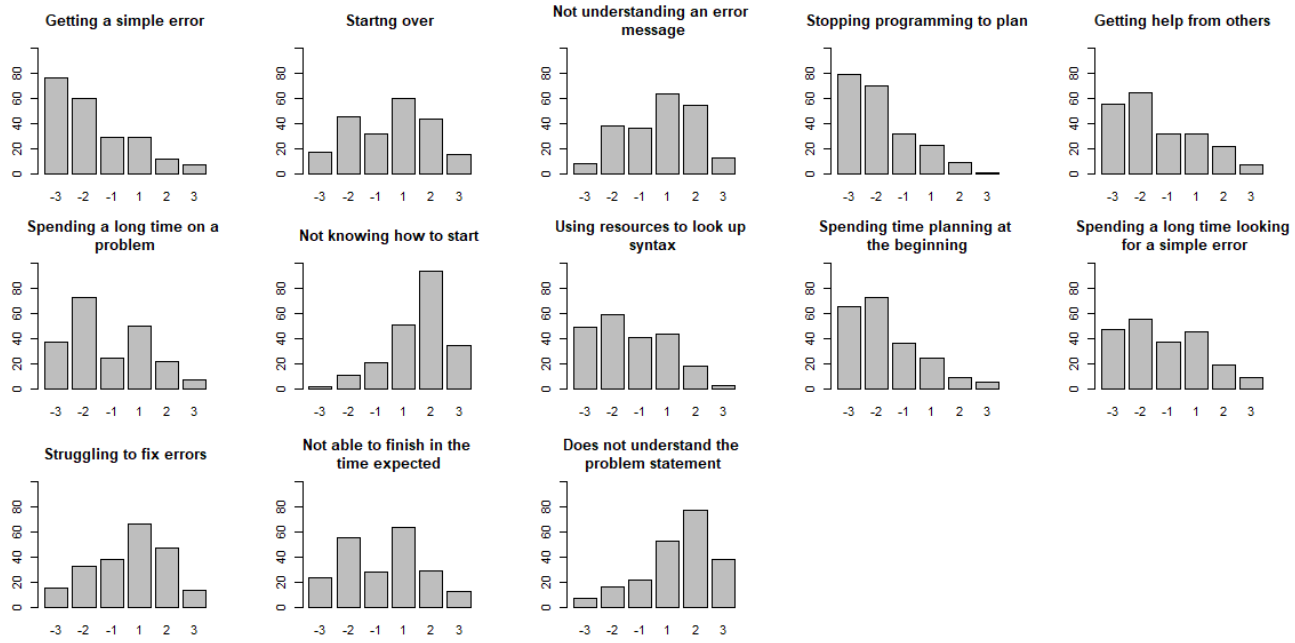

Figure 1: Histograms of the responses to each of the self-assessment vignette questions on a 6-point Likert scale ranging from strongly disagree (-3) to strongly agree (3). Students who answered on the agree side of the scale were reporting that they negatively self-assess at that programming moment.

Students were asked to think-aloud while reading the questions and say anything that came to mind, similar to a talk-aloud [22]. The researcher also asked students to explain why they answered each of the survey questions the way they did.

4 DATA ANALYSIS AND FINDINGS

To answer our four research questions, we analyzed a number of measures calculated from the survey responses. Before beginning the analysis, we evaluated the normality of our data using a Shapiro-Wilk test and found that it was statistically significant for all of our measures. We therefore use non-parametric statistical methods for the analysis presented in this paper.

4.1 Students from all three universities reported negative self-assessments

To answer RQ1, we analyzed students' responses to the self-assessment vignette questions to evaluate if these moments prompt negative self-assessments. First, we plotted a histogram of student responses to each self-assessment vignette question, shown in Figure 1. We expected that only some students would report that they negatively self-assess at each moment because in our previous study, we found high variation in students' agreement with the self-assessment criteria [24]. The histograms show that the responses to some of the self-assessment vignette questions were widely distributed across the scale, like *struggling to fix errors*, while others leaned heavily to one side, like *does not understand the problem statement*. A few of the questions even have a bimodal distribution, like *starting over*, which shows that students report two distinct views of these moments. Overall, we found that each question has at least some responses across the full scale, indicating that all of the moments prompt negative self-assessments for some students. These results are notable because the moments described in the vignettes are natural parts of expert programming practice, and should not necessarily prompt students to negatively self-assess.

To gain an overarching view of the directionality of student responses, we calculated the percentage of participants who agreed with each self-assessment vignette question by combining participants who responded with *slightly agree*, *agree*, and *strongly agree*. Agreeing with a question represents a negative self-assessment at that moment. A summary of the percent agreement for each self-assessment vignette question is shown in Table 3. The results ranged from 15% of students reporting that *stopping programming to plan* prompts negative self-assessments to 84% of students reporting that *not knowing how to start* prompts negative self-assessments. These results help us identify the programming moments that most students use to assess their ability.

To answer RQ2, we analyzed whether there were any differences in the moments that students say cause them to negatively self-assess across university contexts. We enumerated their responses to the 6-point Likert scale, ranging from negative three for *strongly disagree* up to positive three for *strongly agree*. We used a Kruskal-Wallis test to compare students' responses at each university for each of the self-assessment vignette questions, the results of which are shown in Table 3. Out of the thirteen self-assessment moments, only one showed a significant difference between the three universities: *finishing in the time expected* ($H(2) = 10.16$, p-value = .006). For this self-assessment vignette question, the mean responses were 0.077 for University 1, 0.053 for University 2 and -0.760 for

Self-assessment moment	Percentage that self-assesses at the moment	Comparison of responses to self-assessment vignette questions across universities	Percentage with inaccurate perception of professionals	Relationship between each self-assessment vignette and the associated professional programmers question
Getting a simple error	22.43%	(H(2) = 0.267, p = 0.875)	19.16%	U(214) = 3823, p = 0.421
Starting over	55.61%	(H(2) = 1.328, p = 0.515)	14.49%	U(214) = 3196.5, p = 0.249
Not understanding an error message	61.68%	(H(2) = 0.374, p = 0.829)	57.94%	U(214) = 6025.5, p = 0.306
Stopping programming to plan	15.42%	(H(2) = 3.859, p = 0.145)	10.28%	**U(214) = 2784.5, p = 0.010**
Getting help from others	28.50%	(H(2) = 4.828, p = 0.089)	26.64%	U(214) = 5171.5, p = 0.074
Spending a long time on a problem	36.92%	(H(2) = 1.034, p = 0.596)	14.95%	**U(214) = 3595.5, p = 0.029**
Not knowing how to start	84.11%	(H(2) = 2.495, p = 0.287)	50.93%	U(214) = 6061, p = 0.430
Using resources to look up syntax	30.37%	(H(2) = 1.227, p = 0.542)	42.06%	**U(214) = 6920, p = 0.002**
Spending time planning at the beginning	18.22%	(H(2) = 4.471, p = 0.1069)	15.42%	**U(214) = 4038, p < 0.001**
Spending a long time looking for a simple error	34.58%	(H(2) = 4.051, p = 0.1319)	35.98%	**U(214) = 6246, p = 0.022**
Struggling to fix errors	59.81%	(H(2) = 1.834, p = 0.3997)	44.39%	U(214) = 6445.5, p = 0.071
Not able to finish in time expected	49.53%	**(H(2) = 10.16, p = 0.006)**	17.29%	U(214) = 3760.5, p = 0.1457
Does not understand the problem statement	78.97%	(H(2) = 1.173, p = 0.556)	48.60%	**U(214) = 6688.5, p = 0.027**

Table 3: Results of our statistical analysis of the self-assessment moments. The second column displays the percentage of students who agree with each vignette question. The third column shows the Kruskal-Wallis tests evaluating the differences in student responses to the vignette questions across the three schools. All but one of the vignette questions showed no significant difference, suggesting that most of these moments can be generalized across contexts. The fourth column shows the percentage of students who report that professional programmers do not encounter the self-assessment moments. The last column displays the Mann-Whitney U tests evaluating the correlations between responses to the self-assessment vignette questions and associated professional programmers questions. Six of the moments showed significant results, suggesting that these perceptions may contribute to students' self-assessments at these moments. Significant results are bolded.

University 3, suggesting that fewer students at University 3 view completion speed as a sign of ability, compared to the students at Universities 1 and 2. It is surprising that only one of the self-assessment moments was significantly different across the three universities because these are different types of institutions serving different populations of students. This suggests that most of these moments can be generalized across contexts.

4.2 Students understood and related to the vignettes

We analyzed the twelve interviews to ensure that students interpreted the vignettes in the ways we intended, and to evaluate potential risks in the survey design. We did not conduct a formal analysis of the interviews because our sample size was small. Thus, instead of looking for themes, we extracted instances in the interviews when students described their interpretation of a vignette scenario, provided rationale for their response, or mentioned that a vignette sparked the recollection of a particular programming memory. Since the vignette questions were newly designed, they presented a number of risks that could impact the validity of our findings. One risk is that students might answer the self-assessment vignette questions based on hypothetical moments rather than memories of specific programming experiences. However, we found that students referenced their own related programming experiences when discussing their answers to the self-assessment vignette questions in the interviews. For example, P6 discussed a specific experience from the current week's homework when explaining his response to the *starting over* vignette:

> *"A lot of the times I have to erase my code, like many-a-times, like even for this current homework assignment I had to erase several lines of code, sometimes just starting from the beginning."*

We also saw participants relate events that happened to the vignette characters to their own programming experiences. For example, P10 noticed that both he and Diego get help from the TAs. He said:

> *"[The question was] kinda cool for me because Diego had to get help from the instructor and I often have to get help from the TAs a lot and I don't feel like I'm doing bad. Like I feel like I'm just improving myself."*

Another potential risk is that students might only answer the questions negatively because the questions are framed with a negative angle. We would expect students to respond in both directions because our previous findings show that they have varied reactions to these moments [24]. To evaluate whether this happened, we looked for instances in the interviews when students explained the rational for their responses to the questions in both positive and negative directions. In the interviews, we found that students agreed and disagreed with the questions and had a distinct rationale for their responses. For example, we heard students describe both opinions around the topic of planning. P3 described planning as an important part of the process:

> *"I actually [stop to plan in the middle of a problem] a lot but I don't feel bad about it because I feel that planning*

> *is a really huge part before you even start anything. So it is something that you shouldn't feel bad about. I actually think it is one of the things that everybody should do before just jumping into a problem."*

On the other hand, P1 said that when she spends time planning, she feels that it means she is not properly prepared for the problem:

> *"Yeah I definitely feel bad when I have to spend time planning and can't start programming right away, because at that time I am in a situation where I feel that oh I did a lot of practice and still I was stuck and I did not know where to start from and where to end . . . as a computer science student, we don't have to think after reading the question, that ok this is the plan going on in my mind. Rather, we should implement it and we should write it whatever way we feel and by running the program, we can get to know the error instead of wasting time in the beginning."*

Both participants personally related to the vignette whether they agreed or disagreed with the character's negative assessment. This suggests that the vignettes elicited reactions and captured the differences in from students who felt both similarly and differently from the character.

Another potential risk is that students might not consider the nuances in the survey question when providing their answer, which could discount the distinctions we incorporated into the vignettes. However, we found that participants mentioned that they weighed these specific details in the vignettes when deciding how to answer the questions. For example, P7 discussed the nuances in the different types of errors:

> *"You are not doing poorly if you are getting a bunch of syntax errors. But sometimes you do get errors that you completely don't understand and don't make any sense and then that is a loss of concept . . . I think it would feel more like a setback if it was a concept understanding because I feel like that is more part of the understanding process. . . So, I would feel like I'm doing less well in comparison to where I get a syntax error."*

This shows how the nuances in different types of errors impacted the way P7 answered the question, suggesting that students consider these details when reacting to the scenarios. This indicates that incorporating nuanced details into survey questions is important for gaining an accurate representation of the moments that elicit negative self-assessments.

Overall, we found that students resonated with the vignettes and reflected on their own experiences while answering the questions. These interviews help establish that the self-assessment vignette questions capture students' interpretations of programming moments with reasonable accuracy.

4.3 Students who self-assess more frequently have lower self-efficacy

To answer RQ3, we analyzed the relationship between students' responses to the self-assessment vignette questions and the self-efficacy survey. Enactive attainments, or the results of performing tasks related to subject mastery, are the most influential information source for students' self-efficacy [6], so we expected that students who negatively self-assess more strongly or more frequently would have lower self-efficacy in their programming course.

To test this hypothesis, we created a measure that represents the degree to which each student negatively self-assesses, which we call the *self-assessment compound score*. We computed this measure by averaging students' responses to all of the self-assessment vignette questions after converting the responses to numerical values ranging from negative three to positive three. In averaging the questions, we are not suggesting that students' responses should be internally consistent, but rather that they each represent an instance of a negative self-assessment. For example, two students could have the same self-assessment compound score, yet one student might report making negative self-assessments when he gets errors while the other student reports making negative self-assessments when she spends time planning. We therefore use this measure to get an overall sense of how strongly and frequently each student negatively self-assesses. We also averaged students' responses to the five self-efficacy questions to create a *self-efficacy compound score*, after confirming that the responses have high internal consistency (Cronbach's alpha of 0.91).

Then, to test our hypothesis, we used a Spearman's rank correlation coefficient to measure whether there was a correlation between the two compound scores. The results show a significant negative association between students' self-assessment compound score and their self-efficacy ($r_s(212) = -0.418$, $p < 0.001$). This finding shows that students who negatively self-assess more frequently and strongly tend to have lower self-efficacy in their CS1 course. From this analysis, we cannot determine whether this is a causal relationship. However, since self-efficacy theory has previously established that negative self-assessments influence students' overall self-efficacy [6], it is possible that there is a causal relationship between these negative self-assessments and self-efficacy. Future research should therefore try to establish this causal relationship.

We were also interested in measuring whether there were any differences in the relationship between students' negative self-assessments and self-efficacy across the three universities. First, we tested whether there was a difference in the compound self-efficacy scores between the three schools. A Kruskal-Wallis test revealed a significant difference ($H(2) = 13.96$, p-value < 0.001). Students at University 1 reported an average self-efficacy compound score of 1.808, compared to an average of 1.375 at University 2 and an average of 2.071 at University 3 (higher scores indicate higher self-efficacy). Given there was a difference in students' self-efficacy across the universities, but not in their responses to the self-assessment vignette questions, we hypothesized that there might be a difference in the correlation between the self-assessment compound score and self-efficacy. To test this, we ran a non-parametric ANCOVA (using the sm.ancova subroutine in R), which analyzes differences in correlations between a set of non-parametric regression curves [14]. We chose a smoothing parameter of h=1 because this provided an accurate representation of the data without overfitting. The test returned a p-value < .001, showing that there is a significant difference in the strength of the correlation between the self-assessment compound score and self-efficacy across the three schools. The fit lines in Figure 2 show that the self-assessment moments have a stronger correlation with self-efficacy for students at University 2, and a weaker correlation for students at University 3. This could

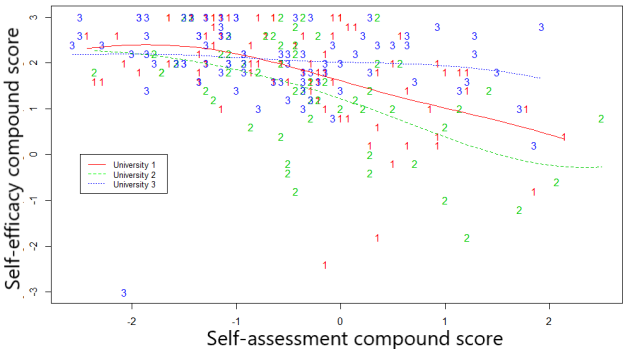

Figure 2: Graph showing the self-assessment compound score and the self-efficacy compound score for each survey participant, grouped by university. The results of the non-parametric ANCOVA show that the fit lines for each of the universities are significantly different from each other.

be because a number of other factors beyond self-assessments correlate with self-efficacy in computer science [5, 25, 35, 47, 52]. For example, if students already have high self-efficacy due to factors like encouragement from the instructor, high grades, or an expectation that the course is easy, self-assessments may not have as strong of an impact on students' beliefs in their ability to succeed.

4.4 Perceptions of professional programmers may influence self-assessment moments

We expect that multiple factors may impact the set of moments students use to negatively self-assess while programming, and findings from our previous study suggested that students' perceptions of more experienced programmers may be one of those factors [24]. Therefore, to answer RQ4, we explored students' perceptions of professional programming practice to evaluate whether these perceptions correlate with the moments that cause them to negatively self-assess. First, we evaluated how frequently students reported that each of the self-assessment moments rarely occur in professional practice. The results ranged from 10% of students believing that professionals do not stop programming to plan to 58% of students believing that professionals always understand their error messages, shown in Table 3. This indicates that that there is a wide variation in students' perceptions of professional programmers and shows that many students' perceptions do not align with studies of professional practices [27, 44, 50].

Next, we wanted to measure whether students' responses to the self-assessment vignette questions correlated with their responses to the professional programming survey questions. We hypothesized that students who think that professional programmers do not encounter the self-assessment moments would be more likely to negatively self-assess during these moments. For example, if a student believes that professional programmers do not spend time planning, she may be more likely to negatively self-assess when she has to spend time planning. We used a Mann-Whitney U test to evaluate if there was a significant difference in students' responses to the self-assessment vignette questions based on their responses to the corresponding professional programming questions, as shown in Table 3. We found a significant effect for six of the moments, for example *using resources to look up syntax* and *spending time planning in the beginning*. Additionally, two of the other moments were trending towards significance (p-value between .05 and .1). We found no significant effect for five of the moments. These results suggest that in some cases, students' perceptions of the professional programming process may influence the moments when they negatively self-assess.

Finally, we analyzed the interviews to understand whether students' perceptions of professional practice factored into their responses to the self-assessment vignette questions. We also analyzed the interviews with a more exploratory lens to identify other rationales that students provided for their responses. We noticed that multiple students, unprompted, brought up professional programming practice when explaining their response to the self-assessment vignette questions. For example, when P10 explained why she does not negatively self-assess after getting a syntax error, she said:

> *"This is with simple errors ... I strongly disagree that you are doing badly on it. I don't even feel like I'm doing badly on it because I get a small error. Again, I'm pretty sure professionals make mistakes."*

Similarly, P9 explained her response about using resources saying:

> *"I said slightly agree because I've gone to a lot of tutoring centers so I know that professionals or people with more experience than me do use websites to help them figure it out. So I feel like I am not doing well [when I need to use resources] because I wish that I could figure it out on my own and I would be doing better if I could figure it out on my own but it's not a major way to see that I'm not doing well."*

Because P9 knows that professional programmers use resources, she responded with *slightly agree* instead of *agree*. While her perception of professional programmers may not have fully informed her reactions to this moment, she still factored it into her response. Additionally, when P8 explained her response to the vignette about memorizing syntax, she said:

> *"I [think] that programmers need to know every single little piece of syntax and every code and how to at least start doing everything... so the fact that I don't have it memorized just made me feel bad."*

Note that we cannot tell from this statement whether P8 is referring to professionals or simply more experienced programmers. In either case, it is clear that she is considering her perception of expert practice in making her self-assessment. Overall, these quotes show that students use their perceptions of more experienced programmers to form their assessments of particular programming moments.

Through our analysis, we identified additional factors beyond perceptions of professionals that students considered when reporting on the self-assessment moments. For example, many participants used social comparisons. P13 said:

> *"Yeah I definitely feel badly on a problem if I don't know where to start. Just like I said before, just seeing everyone else around me being able to solve it. I don't know if I'm just struggling and everyone else is breezing*

through it, then I feel badly about myself and I feel like I'm not as smart as everyone else."

Self-efficacy theory states that the vicarious experience of watching peers informs students' enactive attainments [6], which could explain why social comparisons arose as a rationale for students' responses. Students also rationalized their responses by citing recommendations given by their professors. For example P4 said:

"I put disagree, mainly because of our professors. They always tell us that planning should be first and once you have your plan then you start coding."

These quotes suggest that peer comparisons and recommendations from professors are additional factors that may contribute to the particular moments that prompt students to negatively self-assess, which should be explored further in future studies. These other factors may help explain why we did not find a correlation between students' responses to the perceptions of professional programmer questions and the self-assessment vignette questions for every moment. Students' inaccurate perceptions of professional programming practice partially explain why students make negative self-assessments at natural parts of the programming process, but other factors also play an important role in determining students' self-assessments.

5 CONCLUSION

In this paper, we contribute the results of a survey study with 214 CS1 students from three universities. We found that some students reported that each of the programming moments prompt them to negatively self-assess, even though these moments occur in professional practice. Interestingly, there was no significant difference in students' responses between the three universities for twelve of the thirteen self-assessment vignette questions despite large differences in the populations of students that these universities serve. This suggests that many self-assessment moments generalize across different university populations. We also found that the frequency with which students negatively self-assess correlates with their overall self-efficacy in their programming course. While there was little difference in the self-assessment moments across the schools, the degree to which self-assessment moments correlated with students' overall self-efficacy significantly differed between the universities. Finally, we found that students' perceptions of professional programmers correlated with their responses to some of the self-assessment vignette questions, suggesting that these perceptions may influence when students negatively self-assess.

Our findings demonstrate that students are negatively self-assessing often and, importantly, in response to moments that occur in professional programming practice. This suggests that students may not have a good understanding of professional practice and the experiences they should expect to encounter while programming. This gap in knowledge may exist because CS1 courses typically do not teach about the cognitive aspects of programming, including problem-solving strategies and programming practices [36]. By explicitly teaching about programming practices in CS1, for example with direct instruction [36] or by showing recordings of professional programmers [10], we may be able to help students build accurate expectations of the programming process and reduce negative self-assessments. This type of intervention might also improve students' self-efficacy in their programming course, since we found that students who negatively self-assess more frequently and strongly tend to have lower self-efficacy. However, while helping students develop more accurate representations of professional practice may be one promising intervention strategy for changing student expectations, other messaging around peer comparisons and instructor expectations may also be needed to reduce negative self-assessments. Since studies show that students factor their perceived ability into their decision to major in CS [34, 39, 51], these types of interventions may help to lower the dropout rates in CS programs.

While these results provide valuable insight into CS1 student experiences, our study has a few important limitations. First, even though we chose the self-assessment moments included in our survey based on previous research and preliminary user studies, it is likely that our set of moments is not comprehensive. There may be other moments in the programming process that prompt students to make negative self-assessments. In particular, cultural differences both within and outside of the US may strongly influence the moments that prompt students to negatively self-assess. Additionally, while our interviews with a small sample of students provide promising initial evidence that our survey accurately captures student self-assessments, we need to conduct a more formal validation of the survey. Finally, our results rely on student self-reports based on remembered experiences triggered by the vignettes. While retrospective assessments are still relevant for understanding students' perceptions of ability, we do not know whether these responses accurately reflect the thoughts that arise during programming episodes. However, we chose this methodology because the survey allowed us to collect a larger sample of data with consistent experiences between students.

We believe there are many opportunities to extend and apply these findings through future work. First, studying the self-assessment moments across a wider variety of contexts and countries would help generalize these findings and allow for interesting cross-cultural comparisons. We are also interested in exploring factors beyond student perceptions of professional programmers, such as class format and social comparisons, to understand why students negatively self-assess in these moments. Finally, since our study was designed to measure correlations, future work could identify the factors that cause students to self-assess and confirm that negative self-assessments have a causal effect on students' self-efficacy. Our findings show that many students negatively self-assess at moments that are natural parts of the programming process, and that these self-assessments negatively correlate with self-efficacy. This research lays a theoretical foundation for designing interventions that reduce unnecessary negative self-assessments for novice programmers.

6 ACKNOWLEDGEMENTS

We thank Rachel Trana for collaborating on the survey design and our community partners for providing access to students. We thank our Delta Lab colleagues for valuable discussions and feedback. This work was supported by the National Science Foundation under Grant IIS-1755628 and by the National Science Foundation Graduate Research Fellowship Program.

REFERENCES

[1] Tuukka Ahoniemi, Essi Lahtinen, and Teemu Erkkola. 2007. Fighting the student dropout rate with an incremental programming assignment. In *Proceedings of the Seventh Baltic Sea Conference on Computing Education Research-Volume 88*. 163–166.

[2] Cheryl S. Alexander and Henry Jay Becker. 1978. The Use of Vignettes in Survey Research. *Public Opinion Quarterly* 42, 1 (1978), 93. https://doi.org/10.1086/268432

[3] Susan A. Ambrose (Ed.). 2010. *How learning works: seven research-based principles for smart teaching* (1st ed ed.). Jossey-Bass, San Francisco, CA. OCLC: ocn468969206.

[4] Eugene Anderson and Dongbin Kim. 2006. Increasing the success of minority students in science and technology. (2006).

[5] Petek Askar and David Davenport. 2009. An investigation of factors related to self-efficacy for Java programming among engineering students. *TOJET: The Turkish Online Journal of Educational Technology* 8, 1 (2009).

[6] Albert Bandura. 1982. Self-efficacy mechanism in human agency. *American psychologist* 37, 2 (1982), 122.

[7] Albert Bandura. 1997. *Self-efficacy: The exercise of control*. Macmillan.

[8] Lecia Barker, Christopher Lynnly Hovey, and Leisa D. Thompson. 2014. Results of a large-scale, multi-institutional study of undergraduate retention in computing. In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*. IEEE, Madrid, Spain, 1–8. https://doi.org/10.1109/FIE.2014.7044267

[9] Theresa Beaubouef and John Mason. 2005. Why the high attrition rate for computer science students: some thoughts and observations. *ACM SIGCSE Bulletin* 37, 2 (June 2005), 103. https://doi.org/10.1145/1083431.1083474

[10] Jens Bennedsen and Michael E. Caspersen. 2005. Revealing the programming process. In *ACM SIGCSE Bulletin*, Vol. 37. ACM, 186–190.

[11] Jens Bennedsen and Michael E. Caspersen. 2007. Failure rates in introductory programming. *ACM SIGcSE Bulletin* 39, 2 (2007), 32–36.

[12] Jens Bennedsen and Michael E. Caspersen. 2019. Failure rates in introductory programming - 12 Years Later. *ACM Inroads* 10, 2 (2019), 6.

[13] Nancy E Betz and Gail Hackett. 1983. The relationship of mathematics self-efficacy expectations to the selection of science-based college majors. *Journal of Vocational Behavior* 23, 3 (Dec. 1983), 329–345. https://doi.org/10.1016/0001-8791(83)90046-5

[14] Adrian W Bowman and Adelchi Azzalini. 1997. *Applied smoothing techniques for data analysis: the kernel approach with S-Plus illustrations*. Vol. 18. OUP Oxford.

[15] 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. http://journals.sagepub.com/doi/abs/10.3102/00346543065003245

[16] Spencer E. Carlson, Daniel G. Rees Lewis, Elizabeth M. Gerber, and Matthew W. Easterday. 2018. Challenges of peer instruction in an undergraduate student-led learning community: bi-directional diffusion as a crucial instructional process. *Instructional Science* 46, 3 (June 2018), 405–433. https://doi.org/10.1007/s11251-017-9442-0

[17] Ryan Chmiel and Michael C. Loui. 2004. Debugging: from novice to expert. *ACM SIGCSE Bulletin* 36, 1 (2004), 17–21. http://dl.acm.org/citation.cfm?id=971310

[18] Jonathan Cohen, D, Lauren Margulieux, Maggie Renken, and W. Monty Jones. 2020. Conclusions from the Validation of a Vignette-Based Instrument to Measure Maker MIndsets. In *Proceedings of the Fifteenth International Conference for the Learning Sciences (ICLS) 2020*. International Society of the Learning Sciences, Inc.[ISLS]., Nashville, TN, USA.

[19] J. M. Cohoon. 2006. Just get over it or just get on with it. *Retaining women in undergraduate computing. In J. Cohoon & W. Aspray (Eds.), Women and information technology: Research on underrepresentation* (2006), 205–238.

[20] Jessica L. Collett and Ellen Childs. 2011. Minding the gap: Meaning, affect, and the potential shortcomings of vignettes. *Social Science Research* 40, 2 (March 2011), 513–522. https://doi.org/10.1016/j.ssresearch.2010.08.008

[21] Alireza Ebrahimi. 1994. Novice programmer errors: Language constructs and plan composition. *International Journal of Human-Computer Studies* 41, 4 (1994), 457–480.

[22] K Anders Ericsson and Herbert A Simon. 1984. *Protocol analysis: Verbal reports as Data*. MIT Press, Cambridge, MA.

[23] Allan Fisher and Jane Margolis. 2002. Unlocking the clubhouse: the Carnegie Mellon experience. *ACM SIGCSE Bulletin* 34, 2 (2002), 79–83. http://dl.acm.org/citation.cfm?id=543836

[24] Jamie Gorson and Eleanor O'Rourke. 2019. How Do Students Talk About Intelligence? An Investigation of Motivation, Self-efficacy, and Mindsets in Computer Science. In *Proceedings of the 2019 ACM Conference on International Computing Education Research - ICER '19*. ACM Press, Toronto ON, Canada, 21–29. https://doi.org/10.1145/3291279.3339413

[25] Desmond Wesley Govender and Sujit Kumar Basak. 2015. An investigation of factors related to self-efficacy for Java programming among computer science education students. *Journal of Governance and Regulation* 4, 4 (2015), 612–619. https://doi.org/10.22495/jgr_v4_i4_c5_p6

[26] Gail Hackett and Nancy E Betz. 1981. A self-efficacy approach to the career development of women. *Journal of Vocational Behavior* 18, 3 (June 1981), 326–339. https://doi.org/10.1016/0001-8791(81)90019-1

[27] Brian Hanks and Matt Brandt. 2009. Successful and unsuccessful problem solving approaches of novice programmers. In *ACM SIGCSE Bulletin*, Vol. 41. ACM, 24–28.

[28] Bassam Hasan. 2003. The influence of specific computer experiences on computer self-efficacy beliefs. *Computers in Human Behavior* 19, 4 (July 2003), 443–450. https://doi.org/10.1016/S0747-5632(02)00079-1

[29] Amanpreet Kapoor and Christina Gardner-McCune. 2018. Considerations for switching: exploring factors behind CS students' desire to leave a CS major. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education - ITiCSE 2018*. ACM Press, Larnaca, Cyprus, 290–295. https://doi.org/10.1145/3197091.3197113

[30] Päivi Kinnunen and Beth Simon. 2012. My program is ok – am I? Computing freshmen's experiences of doing programming assignments. *Computer Science Education* 22, 1 (March 2012), 1–28. https://doi.org/10.1080/08993408.2012.655091

[31] Thomas D. LaToza, Gina Venolia, and Robert DeLine. 2006. Maintaining mental models: a study of developer work habits. In *Proceeding of the 28th international conference on Software engineering - ICSE '06*. ACM Press, Shanghai, China, 492. https://doi.org/10.1145/1134285.1134355

[32] Robert W Lent, Steven D Brown, and Kevin C Larkin. 1984. Relation of self-efficacy expectations to academic achievement and persistence. *Journal of counseling psychology* 31, 3 (1984), 356. Publisher: American Psychological Association.

[33] Robert W Lent and Gail Hackett. 1987. Career self-efficacy: Empirical status and future directions. *Journal of Vocational Behavior* 30, 3 (June 1987), 347–382. https://doi.org/10.1016/0001-8791(87)90010-8

[34] Colleen M. Lewis, Ken Yasuhara, and Ruth E. Anderson. 2011. Deciding to major in computer science: a grounded theory of students' self-assessment of ability. In *Proceedings of the seventh international workshop on Computing education research*. ACM, 3–10.

[35] 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 *In Proceedings of the 2016 ACM Conference on International Computing Education Research*. ACM Press, 211–220. https://doi.org/10.1145/2960310.2960329

[36] Dastyni Loksa, Amy J. Ko, Will Jernigan, Alannah Oleson, Christopher J. Mendez, and Margaret M. Burnett. 2016. Programming, Problem Solving, and Self-Awareness: Effects of Explicit Guidance. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM Press, 1449–1461. https://doi.org/10.1145/2858036.2858252

[37] Robert McCartney, Anna Eckerdal, Jan Erik Mostrom, Kate Sanders, and Carol Zander. 2007. Successful students' strategies for getting unstuck. In *ACM SIGCSE Bulletin*, Vol. 39. ACM, 156–160.

[38] Carol Midgley, Avi Kaplan, Michael Middleton, Martin L. Maehr, Tim Urdan, Lynley Hicks Anderman, Eric Anderman, and Robert Roeser. 2013. *Patterns of Adaptive Learning Scales*. Technical Report. American Psychological Association. https://doi.org/10.1037/t19870-000 type: dataset.

[39] Irene T. Miura. 1987. The relationship of computer self-efficacy expectations to computer interest and course enrollment in college. *Sex Roles* 16, 5-6 (March 1987), 303–311. https://doi.org/10.1007/BF00289956

[40] Laurie Murphy, Gary Lewandowski, Renée McCauley, Beth Simon, Lynda Thomas, and Carol Zander. 2008. Debugging: the good, the bad, and the quirky–a qualitative analysis of novices' strategies. In *ACM SIGCSE Bulletin*, Vol. 40. ACM, 163–167. http://dl.acm.org/citation.cfm?id=1352191

[41] Frank Pajares and M David Miller. 1994. Role of self-efficacy and self-concept beliefs in mathematical problem solving: A path analysis. *Journal of educational psychology* 86, 2 (1994), 193. Publisher: American Psychological Association.

[42] Ilias O. Pappas, Michail N. Giannakos, and Letizia Jaccheri. 2016. Investigating Factors Influencing Students' Intention to Dropout Computer Science Studies. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*. ACM Press, 198–203. https://doi.org/10.1145/2899415.2899455

[43] Brian Parkinson and A. S. R. Manstead. 1993. Making sense of emotion in stories and social life. *Cognition & Emotion* 7, 3-4 (May 1993), 295–323. https://doi.org/10.1080/02699939308409191

[44] Michael Perscheid, Benjamin Siegmund, Marcel Taeumel, and Robert Hirschfeld. 2017. Studying the advancement in debugging practice of professional software developers. *Software Quality Journal* 25, 1 (March 2017), 83–110. https://doi.org/10.1007/s11219-015-9294-2

[45] Leo Porter and Beth Simon. 2013. Retaining nearly one-third more majors with a trio of instructional best practices in CS1. In *Proceeding of the 44th ACM technical symposium on Computer science education - SIGCSE '13*. ACM Press, Denver, Colorado, USA, 165. https://doi.org/10.1145/2445196.2445248

[46] Chris Quintana, Meilan Zhang, and Joseph Krajcik. 2005. A Framework for Supporting Metacognitive Aspects of Online Inquiry Through Software-Based Scaffolding. *Educational Psychologist* 40, 4 (Dec. 2005), 235–244. https://doi.org/10.1207/s15326985ep4004_5

[47] Vennila Ramalingam, Deborah LaBelle, and Susan Wiedenbeck. 2004. Self-efficacy and mental models in learning to program. In *ACM SIGCSE Bulletin*, Vol. 36. ACM, 171–175.

[48] Dale H. Schunk. 1989. Attributions and Perceptions of Efficacy during Self-Regulated Learning by Remedial Readers. (1989).

[49] Dale H. Schunk. 1995. Self-efficacy, motivation, and performance. *Journal of Applied Sport Psychology* 7, 2 (Sept. 1995), 112–137. https://doi.org/10.1080/10413209508406961

[50] Sabine Sonnentag. 1998. Expertise in professional software design: A process study. *Journal of applied psychology* 83, 5 (1998), 703. Publisher: American Psychological Association.

[51] F. Boray Tek, Kristin S. Benli, and Ezgi Deveci. 2018. Implicit Theories and Self-Efficacy in an Introductory Programming Course. *IEEE Transactions on Education* 61, 3 (Aug. 2018), 218–225. https://doi.org/10.1109/TE.2017.2789183

[52] Nanette Veilleux, Rebecca Bates, Cheryl Allendoerfer, Diane Jones, Joyous Crawford, and Tamara Floyd Smith. 2013. The relationship between belonging and ability in computer science. In *Proceeding of the 44th ACM technical symposium on Computer science education*. ACM, 65–70.

[53] Willemien Visser. 1987. Strategies in programming programmable controllers: A field study on a professional programmer. In *Empirical Studies of Programmers: Second workshop (ESP2)*. Ablex, 217–230.

[54] Christopher Watson and Frederick W.B. Li. 2014. Failure rates in introductory programming revisited. In *Proceedings of the 2014 conference on Innovation & technology in computer science education - ITiCSE '14*. ACM Press, Uppsala, Sweden, 39–44. https://doi.org/10.1145/2591708.2591749

[55] Aharon Yadin. 2011. Reducing the dropout rate in an introductory programming course. *ACM Inroads* 2, 4 (Dec. 2011), 71–76. https://doi.org/10.1145/2038876.2038894