Understanding the Reasoning Behind Students' Self-Assessments of Ability in Introductory Computer Science Courses

Melissa Chen
melissac@u.northwestern.edu
Northwestern University
Evanston, Illinois, USA

Yinmiao Li
yinmiaoli@u.northwestern.edu
Northwestern University
Evanston, Illinois, USA

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

ABSTRACT

Although enrollments in introductory computing courses are rising, many students still struggle to learn programming. Previous research has found that students' perceptions of the programming process may be one factor that contributes to this problem. Students often assess their own programming abilities overly harshly when experiencing low-level programming moments that are considered normal and expected parts of learning to program. For example, many students think they are doing poorly if they need to stop coding to plan. Research has also shown that students who self-assess negatively in these moments tend to have lower self-efficacy, defined as one's belief in their ability to achieve a particular outcome. In turn, students with lower self-efficacy tend not to persist in their computing studies. While the criteria that students use to assess their ability have been studied extensively, we have a limited understanding of the origins of these criteria and students' reasons for adopting them. To address this gap, we conducted a total of 36 interviews with seven introductory computer science students throughout an academic quarter. In each interview, we asked students to think aloud and explain their reasoning while filling out a self-assessment survey. Through a qualitative analysis of the data, we identified the most common reasons students gave for negatively assessing their performance, including having high expectations for their abilities and feeling like they cannot overcome a struggle. We also identified common reasons why students do not negatively assess their ability in these moments, including believing an experience is "normal" or feeling like they can learn from or overcome a struggle. These findings contribute valuable new knowledge about the underpinnings of students' self-assessments of ability, and suggest that interventions that explicitly emphasize best practices and normalize struggles in the programming learning process are needed to increase student self-efficacy and persistence in computing.

CCS CONCEPTS

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

KEYWORDS

CS0, CS1, self-assessments

ACM Reference Format:
Melissa Chen, Yinmiao Li, and Eleanor O'Rourke. 2024. Understanding the Reasoning Behind Students' Self-Assessments of Ability in Introductory Computer Science Courses. In *ACM Conference on International Computing Education Research V.1 (ICER '24 Vol. 1), August 13–15, 2024, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3632620.3671094

ICER '24 Vol. 1, August 13–15, 2024, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-0475-8/24/08
https://doi.org/10.1145/3632620.3671094

1 INTRODUCTION

Enrollments in introductory computer science courses at the undergraduate level are rising, but retention rates for these courses are still low [42–44] and failure and drop rates are high [6]. Recent research in computing education has shown that students' perceptions of the programming process may contribute to their decisions around whether or not to persist [17, 18, 22, 30]. Students frequently self-assess their programming abilities negatively in low-level programming moments that are considered common or best practices by their instructors [18]. For example, a student might think that they are doing poorly if they need to use resources to look up syntax [18], even though this is considered good practice [16, 20, 31, 38].

Prior research has shown that students who tend to self-assess more negatively in these programming moments also tend to have lower self-efficacy [18], defined as one's belief in their ability to achieve a particular goal [3–5]. Students who have lower self-efficacy are also less likely to persist in undergraduate computing programs [30]. Furthermore, women tend to be more critical of their own programming abilities [19, 22] and exhibit lower self-efficacy than men in computing [2, 8, 33], which may exacerbate the gender gap in undergraduate programs.

Given the prevalence of negative self-assessments and their relationship to student self-efficacy, researchers have argued that we need to support students in developing more accurate beliefs and expectations about themselves and the process of learning to program [18, 22, 24, 25, 30]. However, while we have a good understanding of the low-level programming moments that may lead students to negatively assess their programming abilities, we do not yet know why students use these particular moments to evaluate themselves, or how these are impacted by students' experiences. Gorson and O'Rourke found that inaccurate perceptions of professional practice may explain some negative self-assessments, but these perceptions did not fully explain their results, so they call for more research into the reasoning behind students' overly negative self-assessments [18].

To address this gap, we conducted a longitudinal study to understand the reasoning behind students' self-assessments. We conducted biweekly interviews with seven students during one academic quarter, with the goal of capturing how their evolving experiences motivated and grounded their self-assessments. During each of the 36 interviews, students talked aloud while completing a survey adapted from [18] that captured whether they negatively assess their ability in response to common programming moments, and were then asked to explain their responses. Through a qualitative analysis, we found that students often negatively self-assess when they think they should not encounter a particular situation or cannot recover from a struggle. In contrast, students opt not to negatively self-assess when they believe the moment is a normal experience, is expected for novices, or is a struggle they can recover and learn from. Finally, we found that some students ground their reasoning in specific sources, including observations of instructors and peers, course policies, and conversations with professionals.

Our findings deepen our understanding of the underlying reasons behind students' self-assessments, and show that perceptions of professional practice are only a small piece of the puzzle. In our discussion, we argue that researchers and practitioners can help students develop more accurate expectations about the learning process by making evaluation criteria explicit, supporting students' reflections on their self-assessment criteria, and helping students develop the skills they need to recover from struggles. Implementing these suggestions could lead to increased self-efficacy and retention in undergraduate computing courses.

2 RELATED WORK

2.1 Self-efficacy

Self-efficacy, or one's belief in their own ability to achieve a goal [3–5], has a large impact on student behavior. Studies have shown that having high academic self-efficacy predicts higher performance, resilience, and persistence [11, 21, 28, 33, 37, 39]. Self-efficacy has also been shown to impact major [7] and career [29] choices. Increasing student self-efficacy has the potential to help improve retention in computing majors, which is important as computing-related skills are increasingly needed in everyday life and many careers [9].

Self-efficacy is well-studied in computing education. Researchers have found that students make frequent judgements about their ability to succeed in computing [18, 24, 25, 30], and that they interpret their present context (e.g., classroom environment), measurements of ability (e.g., experience, speed, and grades), and mindset (e.g., fixed or growth [14]), as factors when determining whether they should persist in computing [30]. Furthermore, students do not always make positive self-efficacy judgements after successful programming sessions, nor negative judgements after failed sessions [24, 25], which may be explained by students' expectations for themselves, particularly in comparison to peers [25].

Self-efficacy is also a factor that exacerbates the gender gap in computing, with studies reporting that women exhibit lower computing self-efficacy than men [2, 8, 33]. This gap can be attributed to differences in prior experience and computer self-efficacy [2]. Especially as the participation gap for women and Black, Latine, and Indigenous students widens [27], we must understand how to support students' self-efficacy equitably.

2.2 Self-assessments

Previous work has identified a connection between students' self-efficacy and self-assessments. Bandura stated that people make momentary self-efficacy appraisals, which influence how they behave, particularly whether they will attempt and how long they will persist at a task [3]. For example, students may avoid programming assignments if they feel they cannot complete them successfully. To make self-efficacy appraisals, people call upon four sources of self-efficacy information: enactive attainments, or past successes and failures; vicarious experiences, or observations of others; verbal persuasion, such as encouragement or discouragement; and physiological state, such as feeling stressed or anxious [3–5].

Self-assessments are an important part of self-regulated learning [10], and the accuracy of a self-assessment is important for helping students draw accurate conclusions about their progress [40]. However, not all student self-assessments are accurate. For example, novices tend to be overly optimistic in their self-assessments of ability [40]. Computing education research shows that novices tend to overestimate their performance on harder tasks and underestimate their performance on easier tasks [12]. Furthermore, cognition is often implicit, so students are left to regulate their learning based on potentially inaccurate information [45]. Researchers have warned that when students rely on internal feedback [10], they cannot assess the accuracy of their self-assessments or understand the negative impact of inaccurate self-assessments [40].

2.3 Self-assessments in Computing Education

Previous work in computing education has drawn connections between self-assessments and self-efficacy [18, 24, 30], specifically a correlation between the frequency of negative self-assessments and lower self-efficacy [18]. Self-assessments can also predict persistence [22] and interest in computing [32]. Studies show that women tend to be more self-critical than men when making self-assessments [19, 22] because they are more critical when comparing themselves to peers, have higher standards for their performance, and are more likely to be disrespected when speaking with a peer or instructor [22]. This gender gap makes it important for researchers and practitioners to study the underlying causes of self-assessments so that we may equitably support students.

Gorson and O'Rourke studied students' self-assessment criteria, and found that students often assess their programming abilities critically in response to low-level programming moments that are typical parts of the programming process (see Table 1) [18]. For example, a student may believe they are doing poorly because they need to plan before writing code [18], even though planning is good practice [26]. While this study identified the moments when students may be overly critical of their programming abilities, we do not yet know why students adopt these criteria. To address this gap, this paper explores the reasoning behind student self-assessments with the goal of understanding why students think they are doing poorly in response to natural and expected programming experiences.

3 METHODS

In this study, we aim to understand the reasoning behind students' self-assessments in response to a set of programming moments

that have been shown to elicit overly critical judgements of programming ability. To achieve this goal, we conducted biweekly semi-structured interviews with students, during which we asked them to fill out a survey about their self-assessments while thinking-aloud about their answers to capture their reasoning.

3.1 Data Collection

To elicit students' reasoning for their responses to the surveys, the first author conducted semi-structured, think-aloud interviews. These interviews were conducted in weeks 3, 5, 7, 9, and 11 of the fall 2023 quarter. Two students (P3 and P6) opted to participate in a sixth interview after grades were published, and P1 missed one interview, resulting in 36 interviews. We choose a longitudinal design with repeated interviews to understand how students' evolving experiences with computing influenced their self-assessments. This design also allowed us to build rapport with students and design individually-tailored follow-up questions based on our analysis of early interviews. The students' instructors were not involved in the research, and were therefore not aware of which students participated nor the content of the interviews. As a result, it is unlikely that instructors made adjustments to their teaching practices in ways that would have affected student responses during the quarter.

We video, audio, and screen-recorded all interviews. We asked students to fill out the survey and explain their reasoning as they responded. If a student's response was unclear or required more context to understand, the interviewer would follow up with questions to clarify their reasoning. In the weeks between interviews, the first author open-coded the interview transcripts to generate hypotheses and formulate new questions for the participants based on those hypotheses (see Section 3.4).

This protocol was approved by our university's Institutional Review Board. Students signed consent forms for their participation and were compensated with gift cards for their time.

3.2 Survey Design

We adapted Gorson and O'Rourke's self-assessment survey for this study [18]. The survey captures self-assessments in response to fourteen moments when students are often overly critical of their programming abilities. We describe these moments and present evidence that each is considered typical practice in Table 1. We grouped moments that are topically similar (e.g., both related to simple errors), especially if participants tended to confound the questions in interviews.

Each survey question presents a vignette that tells a story about a hypothetical student (e.g., "*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.*"). The vignette is followed by two questions that use a six-point Likert-scale (strongly disagree to strongly agree), which ask if the character is doing badly in that situation and if the student thinks they would be doing badly if they encountered that situation. The names and pronouns of the characters and the ordering of the questions were randomized. While questions are all framed from a negative perspective, we found that students felt comfortable disagreeing with the characters' negative self-assessments. More details on the survey design are available in [18].

While we adopted the vignette questions from [18] without modifications, we designed new questions about students' prior programming experiences to capture more details, including timing (i.e., did the experience occur in elementary, middle, high school, or college?) and modality (e.g., was this a course in school, an extracurricular experience, an online resource, or a job?). The goal of these new questions was to better understand how different types of prior experiences provide context for students' self-assessments. While we did not formally validate these modified questions, we often confirmed students' past experiences by comparing with the interview data. Our survey also included questions that are not included in the analysis for this paper.

3.3 Participants

Since we were interested in studying the self-assessments of students in introductory courses, we recruited students who were enrolled in CS0, CS1, and CS1.5 at a highly-selective private university in the United States. Students who took either CS0 or CS1 can take subsequently take CS1.5, which introduces object-oriented programming. CS0 is a course designed for non-majors and CS1 is designed for computing majors. CS0 and CS1.5 are taught in Python, while CS1 is taught in Racket.

First, we invited students to complete the survey described above via email and an in-class verbal announcement during week 2 of the quarter. This served as a screening survey. Since we were most interested in understanding why students are overly critical of themselves, we invited the students who had the most negative self-assessments to participate in our interviews. Concretely, we invited students who assessed themselves as doing very poorly (i.e., answered "strongly agree" on the six-point Likert scale) in any of the fourteen vignettes, or assessed themselves as doing poorly to any degree (i.e., answered "strongly agree," "agree," or "slightly agree") in nine or more of the moments.

Twenty-three students filled out the filtering survey, eleven met our criteria, and seven opted to participate in the study. Table 2 outlines students' background information. We use their self-reported pronouns throughout the paper.

3.4 Data Analysis

We sought to build a grounded understanding of students' reasons for their self-assessments. For this reason, we formed hypotheses as we iteratively engaged with the data, then connected the concepts that emerged to existing theories and literature to better understand how they are situated within prior research. We followed the qualitative data analysis process as outlined by Miles et al. [36].

The first author conducted bi-weekly interviews and, between interviews, performed inductive, descriptive coding on the interview transcripts to understand students' responses. The first author also wrote memos, consulted with the other authors, and generated hypotheses and questions for the students, such as:

- *What are your post-college plans, and how does taking this class help?* We sought to contextualize students' claims that engaging with certain strategies would help with their goals and understand concerns about peer competition.

Moment type	Description of moment	Typical practice
Simple errors	Getting a simple error	Novices often make simple syntax errors and take a long time to fix them [13]
	Taking a long time to fix a simple error	
Complex errors	Not understanding an error	Programmers will look up error messages that they do not understand [31]
	Struggling with an error	Programmers can to take a long time to fix errors [15], and it is a significant portion of professionals' workload [41]
Restarting	Restarting on a problem	Practices like commenting out code are a common debugging strategy for novices [16]
Planning	Stopping to plan and think	This is a common debugging strategy for novices [16]
	Planning at the beginning	Professionals take time to understand problems and plan high-level solutions [26]
Understanding the problem	Not knowing how to start	Students can struggle to understand an assignment specification or how to start [23–25]
	Not understanding the problem statement	
Time	Taking a long time to finish a problem	Students often make negative judgements on their performance when it takes longer than they or their instructors intended [25]
	Taking more time than expected to finish a problem	
Help	Getting help from an instructor or TA	Seeking help is a common problem solving strategy for students [34]
Resources	Using resources to look up syntax	Students frequently use resources [16], which is good practice when stuck [20, 31, 38]
	Using resources to research an approach	

Table 1: List of moments when students commonly negatively assess their programming ability.

	Gender	Started in	Other experience	Major	Year	Class
P1	Man	College	Other classes	Biology	2	CS1.5
P2	Woman	High school	AP Computer Science Principles	Cognitive science	1	CS1
P3	Man	College	Other classes, internship	Computer science	2	CS1
P4	Woman	College	Other classes	Journalism	2	CS1.5
P5	Woman	College	Second time in CS0	Theatre	3	CS0
P6	Man	College	None	Computer science	1	CS1
P7	Man	High school	Coursera	Computer science	1	CS1

Table 2: Each participant's demographics and computing experience.

- *How is class going?* We asked this weekly starting in the second interview to contextualize moments students were recalling, such as saying that planning was helpful for a recent topic.
- *What does "doing well" mean?* Often, students gave answers that simply cited criteria for doing well, so we used their definitions of the term to contextualize their responses.

We also generated questions for specific participants. For example, we asked P6, "what is considered a simple problem, versus a complex one?" to clarify what he meant when he made this distinction in his responses.

After completing data collection, the first and second authors conducted two rounds of qualitative coding on the transcripts. All coding was conducted collaboratively, with discussion until consensus. We wrote memos throughout to document our process and changes in our understanding of the phenomena. We adopted this

analysis approach with the goal of producing a robust understanding or our data through deep discussion and continual testing of our emergent theories against the data.

First, we conducted a round of inductive, descriptive coding. This generated codes that reflected the reasons, sources, and criteria that students used when completing the survey. Then, we conducted a second round of coding to identify similarities and differences among the reason and source codes generated during the first round with the goal of refining our definitions of the concepts. During this process, we grouped codes that arose from the data to create topical categories. For example, we grouped the codes "I am a novice" and "this is new" to capture the expectation that students should not know something because the content is new. Then, we merged that group with the code "expectation of human capability," or a students' expectation that people make mistakes, to define the category "I do not expect to know this". During and after the second round of coding, the first author generated possible theories to explain patterns in the data, and then discussed and refined these in collaboration with the second and third authors.

While we originally planned to use our longitudinal data to analyze changes to student self-assessments over time, a preliminary analysis showed that there were often no stable patterns in the changes in students' responses. That is, students' self-assessments often fluctuated from week to week, or they used different reasons and sources to justify the same self-assessment over time. We did not have additional data to identify possible causes for these changes. As a result, we focused on identifying and characterizing the reasoning behind individual self-assessments.

4 RESULTS

Through our analysis of students' think-aloud interviews, we developed a model for describing students' self-assessments and the reasoning behind them. In reaction to a low-level programming *moment* (the vignette questions), a student will make a *self-assessment* of their programming ability (either "I think I am doing poorly" or "I do not think I am doing poorly" in this moment) on the Likert-scale. Then, students justify the usage of this particular moment as evidence of their ability with a *reason*. These reasons, such as believing that a particular behavior is normal, are identified through students' think-alouds. At times, students attribute their reasons to *sources* during the think-alouds, for example hearing about best practices from a professor. Students can share one or more reasons and sources.

We consider the following think-aloud by P3 as an example of how these four components appear and interact. In response to a vignette about not understanding an assignment description (the moment), P3 responded with "agree" on the survey (his self-assessment), indicating that he would self-assess his programming ability negatively. When asked to explain his reasoning, P3 says:

> I do get very stressed when I don't understand anything. I feel like we have such an expectation, especially with the job market right now [...]. It's not very good. So it feels like we're competing with some geniuses and [...] people who have been coding their whole life and there's a lot of imposter syndrome. So I feel like those people sometimes in my head I'm thinking like, "oh, those people who try this assignment, they would understand it right away. Why am I not doing that?"

P3 explains that he would self-assess negatively because he would expect himself to understand the problem statement (his reason). Unprompted, he cites imposter syndrome due to peer comparison when explaining why he has this expectation (his source).

While there is a source in this example, in many cases, students do not attribute sources to justify their reasons, even when asked. For example, consider P1's response to a vignette about struggling to fix an error. P1 responded "agree" to the survey, indicating that he would self-assess his programming ability negatively. He says, "for the same reason from the previous question [where] I don't master the contents, I always feel that I'm not doing well when I'm not able to go through a code and go deeper, etc." He refers to his answer to another question where he was disappointed that he had to restart, which indicated to him that he did not understand the content as expected (his reason). When asked why errors indicate a lack of understanding, P1's explanation was, "I dunno. I just feel like [the character] tries to run [the program] and he doesn't know how to fix it. I actually dunno." When pushed, he expresses that he does not know why he uses this criteria.

There are also cases where students refer to a similar past experience. For example, when reacting to a story about a student not being able to finish solving a problem in the time they had expected, P5 self-assesses her performance negatively due to how she reacted in a past experience.

> I thought I could get it done the night before it was due and then it randomly started having a lot of errors [...] so that made me feel like I wasn't doing well. [...] I still had made good headway into the assignment but since it wasn't working that made me feel like I was doing bad.

In the past, P5 encountered a bug and could not produce a working program in the time she had expected. Because she self-assessed negatively in this past experience, P5 responded to the survey similarly. This is neither a source nor reason because it does not explain why she used the criteria nor where it comes from. However, answers like these suggest that students connect the stories in the vignettes to their own experiences, which supports the claim that the vignettes accurately elicit students' self-assessments [18].

In Sections 4.1 and 4.2, we illustrate the most common reasons that students cite when explaining their responses to the self-assessment survey. Students' reasons for self-assessing negatively were distinct from their reasons for not self-assessing negatively. We do not consider these to be positive self-assessments; since the survey questions are posed negatively (do you agree that you are doing badly in this moment?), we call these non-negative self-assessments. In Table 3, we present a summary of the most common reasons students give, as well as a list of each participant who uses that reason at least once for each category of moment. In Section 4.3, we discuss the common sources that students use to justify their reasoning. We outline the sources and list the participants who use them at least once in each moment in Table 4.

Tends to be used when self-assessing	Reasons	Help seeking	Planning	Using Resources	Restart	Simple Errors	Complex Errors	Time	Not understanding the problem
Negatively	I should know this		P5	P2, P3, P4, P5, P6, P7	P1, P2, P6	P1, P3, P4	P1, P2, P3, P5, P6	P1, P3, P6	P3, P4, P6, P7
	I have to be doing badly to do this	P1, P5	P1	P1, P2				P5	
	I cannot recover						P2, P6	P4	P2, P4, P5
Non-negatively	I do not expect to know this	P4, P7		P2, P4, P5, P6, P7		P1, P4, P5	P4, P5, P6	P1, P4, P6, P7	P2, P6
	This is normal or expected	P3, P5	P1, P3, P4, P5, P6, P7	P2, P3, P4, P5, P6	P2, P4, P6, P7	P1, P2, P3, P4, P6, P7	P1, P2, P3, P6	P1, P2, P3, P6, P7	P3, P7
	I can learn or recover	P1, P4, P6, P7	P1, P2	P2, P3, P4, P7	P1, P2, P3, P5, P6	P1, P3, P4, P5, P7	P1, P2, P3, P4, P5, P6, P7	P1, P2, P3, P7	P1, P2, P3, P5, P6, P7

Table 3: In this table, we list the participants who used each reason at least once across all interviews to justify their self-assessment in each category of moment. This table does not represent the number of times a student used a particular reason, nor how many times any reason was used across all participants, but rather demonstrates usage patterns across moments. We see that many reasons are commonly used across all moments, but not all students use the same reasons. Students also do not necessarily adopt the same reasoning for all moments. We observe a direct contrast between reasons on the negative and non-negative sides, such as students' expectations of "I should know this" contrasting with "I do not expect to know this" and "this is how others behave", and students' beliefs that "I cannot recover" contrasting with "I can learn or recover".

4.1 Reasons for Self-Assessing Negatively

After reading the vignettes presented on the survey, students often agree that they would be doing poorly if they experienced that moment. The most common negative reasons that students cited were that they believe they should not encounter this problem, that they have to be struggling to be in this situation in the first place, and that they do not have the skills to overcome this setback.

4.1.1 I should know this. Students often negatively self-assess their performance if they do not meet their expectations for themselves. Many students express that they ought to know something or should not encounter a particular setback. This reason appeared for at least one participant in response to every category of moment except for asking for help. Below, we describe the ways this expectation appears in students' explanations.

One way that students' expectations manifest is through believing that they should be able to overcome the setback easily. For example, when reacting to a vignette where a character did not understand the error message they encountered, P3 self-assessed negatively. His reasoned that he should know how to fix the error, especially since he perceives his peers would be able to:

> You think to yourself, "oh, I should be understanding how to fix this. My friend would just do this right away." Especially in computer science, when you're surrounded by so many intelligent people. [...] I end up comparing myself a lot to them as well. I think, "oh, my friend here, he wouldn't get this error."

P3 views not understanding the error as a hindrance to resolving the bug. He expects that he should be able to understand how to fix the bug, especially when he thinks his peers are able to fix the bug quickly or perhaps not encounter the bug at all. This peer comparison is the source for P3's reason.

Another type of expectation manifests as students' desire to not encounter the issue. For example, P3 self-assesses negatively in response to a story about having a simple error, saying that, "It can be discouraging [...] I think, 'that shouldn't have happened. I shouldn't do that.'" P3 expresses how he should not have caused this bug, implying that he should be skilled enough to avoid this error. Echoing P3, P4's negative self-assessment in reaction to a story about taking a long time fix a simple error also emphasizes what "should" happen given her skill level. She says, "I feel like I should have detected the error more quickly. I feel like I'm stuck in one logic for a long time and that fixed perspective didn't allow me to see my simple error." Her expectation is that she should be a more efficient debugger.

Similarly, students express that, even if they struggle with other setbacks, they should at least be able to overcome this. For example, consider P6's negative self-assessment in reaction to a vignette about not understanding an assignment description. He says, "I would expect from myself [...] to at least know what the question is asking me [...] to do." P6 expects that he should at least be able to understand the question, even if the rest of the problem has other challenges.

Students also call upon their preferred practices and express disappointment when reality does not align with their plans. For example, when reacting to a vignette about not being able to finish within the time allotted, P3 self-assesses negatively. He describes his time-blocking strategy, saying:

> I'm like, I'll finish this until six today, and then I'll take a break for myself. And then I really look forward to taking a break, but then it's 6:30 and [...] I'm not where I want to be and it makes me frustrated because I really just wanna put this down, but at the same time, I told myself I was gonna finish it today and that really stresses me out.

P3 highlights how he would feel disappointed when he does not meet his goal for how long a task would take, leading to a negative self-assessment. He sets these expectations based on his practices and priorities.

Importantly, we found that students who do not have prior experience hold high expectations for their programming performance. P5 illustrates this through her negative self-assessment in reaction to a vignette about planning; she is the only student who used this reason in this category of moment. She reasons that "if I really was doing well, maybe I would just be able to figure it out as I'm coding and not have to stop." P5, who has no prior experiences, holds herself to this very high standard of being able to code without stopping.

Through our analysis of students' expectations as reasons for their self-assessments, we see that students often hold high expectations that lead to negative self-assessments when the outcome does not match. These are not in line with instructors' expectations for students, which suggests a need to help students better align their expectations with what is possible given their learning progress. Interestingly, students use this reason in all moments except help-seeking, suggesting that they may view help as an part of the learning process.

4.1.2 I have to be doing badly to do this. Another reason why students feel like they are doing poorly in response to a moment is because they feel like they have to be doing badly to experience it in the first place. This reason mostly appears in response to moments related to help seeking and using resources, but also appears for planning and time-related moments. P1 expresses this reason when agreeing that he would be doing poorly in reaction to a vignette about asking for help, saying, "Yes. He's not doing well. So he's asking for help." Here, P1 expresses the belief that you would never seek help unless you were doing poorly, so almost by definition, seeking help leads to a negative self-assessment.

For P5, this reason manifests as a barrier to seeking help, although she knows it would be beneficial for her progress. She says, "I have to feel a certain extent of like I'm doing badly to be able to muster up the energy and courage to go to a TA or professor. But it's not really bad. [...] Once I meet with them I'll feel much more confident." While she ultimately does not self-assess negatively in this moment because it would be helpful, she does qualify this self-assessment, saying that she needs to feel badly to seek help.

This reasoning is troubling as it suggests that some students view moments in the programming process, like seeking help from resources or others, as a sign of existent poor performance, which may discourage beneficial learning behaviors.

4.1.3 I cannot recover. Students also express concern that they do not have the skills or resources to overcome their setback. This appears in moments related to complex errors, not understanding the problem, and spending a long time on a problem. For example, consider P2's self-assessment in reaction to a vignette about not understanding an error message. She self-assesses negatively, reasoning, "If I'm not able to ask for help right away and I have an error message, then it [...] makes me get stuck in the problem. 'cause then [...] I won't be able to fix it without knowing what it is." Here, P2 demonstrates a reliance on help and feels she would not be able to overcome this error if she could not find assistance.

This reason also arises when students get derailed from their typical programming process and cannot recover. For example, P2's reaction to a vignette about not understanding the problem results in a negative self-assessment because "understanding the assignment to begin with is the most important, first step." Because she is unable to complete this step, she worries about her ability to finish the assignment. This is similar to how students cite their expectations of their performance, but differs because P2's primary reaction is that she feels unable to move forward.

When students feel that they cannot overcome a setback, it sometimes results in avoidance behaviors. For example, P5's negative self-assessment when not knowing how to start working on a problem leads to procrastination. She says, "I feel stressed when I don't know how to start on something and [...] I end up either procrastinating it or feeling I can't do it instead of taking the steps needed to start." She expresses that this moment would lead her to react in a way that deters her from taking productive actions towards resolving her issue and exacerbates her inability to recover.

Students self-assess negatively when they feel they cannot move past an issue in their programming processes, leading to feeling stuck or procrastinating. This suggests the need to equip students with debugging and self-regulation skills that they can use independently to resolve these situations.

4.2 Reasons for Not Self-Assessing Negatively

In this section, we explore cases where students disagree with the questions posed on the survey, indicating that they do not self-assess negatively in these moments. These responses are interesting because these beliefs about ability and the learning process are more aligned with instructors' expectations. By understanding why these students do not self-assess negatively in certain cases, we can learn about how current teaching practices support accurate expectations about learning and where improvements are still needed.

4.2.1 I do not expect to know this. While we saw that some students self-assess negatively due to high expectations for their performance, students can instead use their expectations to justify not self-assessing negatively when they do not expect to know something. In this case, students set more reasonable expectations of their abilities based on their perceptions of what is expected of them as learners and, more generally, people. At least one participant uses this reason in all moments except planning and restarting.

For example, when responding to a vignette about seeking help, P4 would not self-assess negatively. She explains, "that's the point of having a professor or TA [...] if you master the homework from the beginning, why are you taking that course?" Since she is a student, she expects to seek help during the learning process.

Similarly, students also refer to their perceived level of expertise as a reason for their self-assessments. For example, consider P5's decision not to self-assess negatively after reading a vignette about making a simple error. She says, "I would feel bad for not putting that parenthesis. And maybe I didn't know to put it because I have so little knowledge of coding, but I wouldn't feel like I'm not doing well." P5 expresses that because she has "so little knowledge of coding," she might not have known to put the parenthesis that she missed. Similarly, students also do not self-assess negatively when they encounter new content. For example, when self-assessing in reaction to not knowing how to start on a problem, P6 expresses that he does not necessarily feel badly in this moment "'cause it can be something completely new that I haven't encountered before." As a novice learning new content, he believes that not knowing where to start is expected.

Students also feel like they cannot make accurate estimates of how long it will take them to complete a task with new content. For example, P6 does not self-assess negatively when tasks take longer than he expects. He says,

> I would say it depends on the scenario. If it's something new [...], it's pretty tough to also estimate how long is something gonna take you if you haven't really worked with it yet. So I wouldn't say I or somebody else shouldn't feel bad, but of course in some cases if you know the type of an exercise and it takes you way longer than you imagine, then I would feel that I'm not doing that well.

He says that with newer content, it can be difficult to know how long it will take, but for more familiar content, he would expect himself to make those judgements accurately.

Students also do not self-assess their programming abilities negatively because of more generic expectations of themselves as people. For example, P4 does not self-assess negatively when reacting to a question about taking a long time to find and fix a simple error. She says, "People make mistakes. It's the same with when you are writing an email [...] and you just had a typo [...] That doesn't mean something [does or] doesn't relate to your writing ability."

In these cases, students' expectations for themselves are more aligned with instructors' expectations for novice behavior, suggesting that instructors may need to make their expectations for their students' processes more explicit.

4.2.2 This is normal or expected. Our analysis shows that students often try to determine if their behavior is normal within the computing discipline. Students who perceive that their behaviors align with the behaviors of others, or feel that there is no standard in the computing community, tend to not to self-assess negatively. Students call upon others' behaviors as reasons in all moments on the survey.

For example, students may perceive that "everyone" in the computing community behaves a particular way. Consider P5, who does not self-assess negatively in reaction to a question about using resources to research her approach. She says, "I think everyone uses resources. I've seen my professor use resources." P6 provides

another example when prompted to clarify why he thinks "everyone" encounters simple errors during the programming process and thus why he would not self-assess negatively during this moment. He explains, "You even see when the teacher is programming on the board and he is trying to show something and he runs the code, there is a parenthesis missing. He is like, 'oh, oops.' And he is a guy with a PhD." For P5 and P6, their professor's actions (the source) show that someone with a lot of experience and qualifications uses resources or makes simple mistakes, signalling that this is typical.

Students also consider what is a normal part of the programming process. For example, P7 does not self-assess negatively in response to a vignette about not understanding the problem. He reasons that this is typical, saying, "Part of the challenge is just thinking where to start. [...] So that's gonna be a hard part. And so you're obviously gonna struggle doing that. It's pretty natural." Alternatively, students also observe that there is no "standard" practice among programmers. P6 uses this reason to explain why he doesn't think that taking a long time on a problem means he is doing poorly on it. He says, "I would say if you take really long to solve something that shouldn't take that long, it kind of means you're not doing well. But, as long as you finish it, I think it's okay. Everybody has a different pace and everybody is good at something else." While he would potentially feel poorly if he were taking too long, he prioritizes finishing the problem and notes that everyone can take different amounts of time to do so.

However, it's important to note that believing that a particular behavior is normal does not always prevent a negative self-assessment. Consider P3's negative self-assessment in response to a question about struggling with an error:

> So the ability to fix the errors is something very important [...] for CS. I think there was a saying where 80% of the time it's just debugging as a software engineer. But I mean the scope of a CS class too, I think it applies. Once you have the fundamentals down, once you go to class, pay attention, pick up the material, you know how to write the problem, it's just the [...] way you translate your words ideas into code. That's where errors can come up and you need to know how to. Once you have the fundamental idea, it's not that hard to fix errors [...] 'cause it usually comes down to [small syntax errors]. It's not usually a fundamental thing where you know you have the completely wrong idea. [...] You should be able to look out for them and fix them pretty easily. So I would say, I would say errors are pretty important knowing how to fix it.

Despite knowing that debugging is typical and takes a long time, P3 believes that he should solve errors with ease, especially since most errors are simple mistakes, not conceptual misunderstandings. Although solving bugs takes a majority of professional programmers' time, P3 wants to be able to solve each error quickly.

While P3 poses an interesting case, most students do not self-assess negatively when calling upon typical behavior (or the lack thereof) as a reason for their self-assessments. They perceive themselves to be performing on par with others and thus think they are not doing poorly. This suggests that normality is an important metric for students, so it is important for instructors to help students build an awareness of what to expect in the programming process.

4.2.3 I can learn or recover. Finally, we found that some students also consider whether they feel like they have the skill to recover from a setback or learn from the situation. When asked if she would self-assess negatively when encountering a simple error, P4 stated, "No, I just fix that, it's not a big deal." In our data, at least one student mentioned this reason across all categories of moments.

Another example is P4's self-assessment regarding seeking help. She does not self-assess negatively, saying, "I find getting help from TA [or] professor really helpful. You not only fix the problem but also understand why you have this problem in the beginning. I feel like that's most effective way of learning." P4 finds help-seeking beneficial for her learning, and sees it as an opportunity to overcome the struggle she encountered. Similarly emphasizing learning, P7 explains why he does not self-assess negatively in reaction to a survey question about taking a long time to solve a problem, saying, "I don't think [...] you're not doing well in the assignment if you took a long time. You probably understand it better if you took a long time too." P7 reasons that taking longer would better support his understanding.

However, some students self-assess negatively even when they feel like they have the skills to recover. For example, consider P3's reaction to taking a long time to fix a simple error. Although he thinks these errors are "quite easy to fix," he self-assesses negatively, saying:

> It is very discouraging. A lot of errors can be very hard to fix because, there's some simple errors are very easy to overlook, like [several types of syntax errors]. These little things, can really, really mess up your code. It's pretty discouraging. But they're quite easy to fix, I think, because we get like the line number. If we're a little more careful, then we can figure it out.

Despite feeling that he can fix the error, he feels discouraged and like he is doing poorly. P3 presents a rare case of students feeling like they are doing poorly even when they can recover.

We found that students who think they can overcome a setback tend to feel less negative about their programming abilities, which highlights the importance of helping students build skills to unblock themselves either independently or by using resources.

4.3 Sources for Reasons

While the sources for students' self-assessments were not a focus of the interviews, students often attributed their reasons to sources of information, unprompted. Since we did not ask students for the sources behind their reasoning directly, we do not make claims about the absence of sources when interpreting our results. However, our analysis shows that there is value in understanding patterns in the sources that were attributed.

Four categories of sources arose from students' answers: instructors' actions, instructors' policies and designs, peer comparison, and hearing from professionals. The majority of references to sources occur when students explain why they did not self-assess negatively in response to a moment. Among the cases when sources were provided while explaining a negative self-assessment, most

are due to peer comparisons when taking a long time and more than half come from P3. In this section, we discuss students' sources in their answers to the survey questions (see Table 4).

4.3.1 Instructors' actions. Students' observations of their instructor's behavior is influential in two moments: using resources and encountering simple errors. For example, when responding to a question about taking a long time to fix a simple error, P2 cites her professor when explaining why she believes simple errors happen to everyone, saying, "Everyone makes typos. My professor makes typos and then he has to go back and change them." While this may not have been an intentional act by the professor, this demonstration informed P2's reason.

4.3.2 Instructor policy & course designs. In addition to observing their instructors, students also infer their professor's intentions by interpreting course policies. At least one student uses this reasoning in every category of moment except restarting and simple errors.

In some cases, students are told by instructors what behaviors are acceptable. For example, P1 cites his professor when explaining why he would not self-assess negatively in response to a question about using a resource to look up syntax. He says, "My [CS1.5] instructor encourages us to use ChatGPT to solve assignments. So it [is] a really important resource, especially with the rise of AI. So I think that yes, I really enjoyed this approach of using resources and yes, I don't feel I'm doing bad 'cause I'm looking to ChatGPT. I just feel like having some support besides the one that I can get in class." Because the CS1.5 professor encourages him to use ChatGPT as a resource (with attribution), P1 does not feel that doing so reflects poorly on his ability.

In other cases, we found that students infer their instructor's intentions from their designs. For example, P7 cites a course design choice when explaining why he is not doing poorly when using a resource to look up syntax.

> You're not gonna memorize [...] all the functions, orders, and stuff like that. [...] Especially 'cause they're not asking us to memorize all these functions and they're giving us glossaries with functions on the actual tests. I don't think they're expecting us to memorize all of the syntax.

P7 infers from the presence of a glossary that students are not expected to have all of the syntax memorized. So, he does not feel poorly about his programming abilities when he needs to use resources.

However, course policies can instead reinforce students' negative self-assessments. For example, consider P3's reaction to a vignette about struggling with an error:

> [You] need to fix these errors in order for code to work [...] I feel like a lot of my classes this quarter really emphasized, if your code doesn't run, even if it has all of this stuff right, it's not good code. [...] It's important to turn in a complete, running product over something that's partially complete and [...] full of errors. [CS1] was pretty unforgiving with errors. I feel like in quizzes, I would miss one parentheses and then I was wondering why I only got half credit because my code is fundamentally correct. I should have only [...] 2-3 points off. Instead, why did I get 5-7 points off? And the professor always tells me, your code wouldn't run if you [had ...] one parenthesis mismatch. And that's really important that it runs. So I kind of learned that errors are actually really critical.

P3 observes that he is getting half credit on his quiz when he makes small syntax mistakes because his code would not run and thus is incorrect. So, P3 places value on small errors and extrapolates that to include all errors. He thinks he is not doing well if he is struggling with an error because it would mean that he is unable to turn in fully-functional code. It is likely that this policy was an artifact of large class sizes and automated grading, and was not meant to insinuate that error-free code is essential. Regardless, P3 interpreted the policy to mean he is doing poorly whenever his code has errors.

The frequency with which students cited instructor policies suggests that this is a highly influential source of information for determining what behavior is expected in their courses.

4.3.3 Peer comparison. In addition to observing the behaviors of their instructors to determine what is normal and expected, we found that students also look at their peers' actions. This source arose in our interviews across all categories of moments except planning, using resources, and restarting.

In some cases, peer comparison results in students not self-assessing negatively. For example, when presented with a vignette about not understanding an error message, P2 says, "Even like [the undergraduate TAs] when I would show them an error message, they're like, 'oh I don't know what that is.'" Because her more experienced near-peers encounter this issue, P2 does not feel poorly about her performance.

In contrast, we see P3's self-criticality arise when he uses peer comparison in a negative self-assessment. When presented with a question about taking a long time to solve a problem, he says,

> It does frustrate me. [...] Especially if my friends finish it faster, they're telling me, "oh, I'm done with the homework. I'm gonna like relax. I'm gonna go hang out with my friends," and I'm stuck here just doing the homework until [the due date]. And it does make me frustrated and it feels like I'm a worse programmer than them.

P3 compares himself to his peers and perceives that he is not doing as well as they are. He feels like he is a "worse programmer" because he takes longer than his friends. These cases highlight how students observations of their peers influence their self-assessments and how students will make self-assessments based on their perceptions of their peers' behaviors, even if they do not have a complete picture of their peers' processes and experiences.

These examples show that observations of peers can lead to both negative and non-negative self-assessments, highlighting the importance of helping students develop accurate pictures of their peers' ability and identify assumptions they may be making about their peers' practices, so that they may, in turn, make accurate self-assessments in comparison.

4.3.4 A professional told me. Finally, only two students cite a professional as source. P3 and P5, who know professional programmers,

Sources	Help seeking	Planning	Using Resources	Restart	Simple Errors	Complex Errors	Time	Not understanding the problem
Instructors' actions			P3, P5		P2, P6			
Instructor policies & designs	P2, P6	P1, P2, P3, P4, P6, P7	P1, P2, P3, P4, P7			P1, P2, P3	P2, P3	P2, P3, P4
Peer comparison	P2				P1	P2, P3	P1, P2, P3, P4, P5	P3, P6
A professional told me		P3	P3		P5			

Table 4: In this table, we list the participants who used a source at least once across all interviews in their self-assessments in each category of moment. This table does not show how often sources are cited by students nor used overall, but shows patterns in source usage. We see that instructors' policies are the most cited source, but other sources are common in particular categories of moments.

cite professionals in moments related to planning, using resources, and simple errors. In reaction to a vignette about taking a long time to fix a simple error, P5 says, "I recently had a problem like this and it was a simple little mistake and I was getting help with it from someone who is a professional programmer and he said that he does that all the time. [...] So I think it's pretty common." While this is a rarer case in our dataset, we draw similarities to how students observe and listen to their instructors to see that the behaviors of professionals can be an influential source for students.

5 DISCUSSION

The most common reasons that students gave when explaining their responses to the self-assessment survey were based on their expectations of themselves, perceptions of their abilities, and beliefs about others' practices. When we compare the reasons students gave for negative and non-negative self-assessments, we notice that many are opposites of each other. For example, many students self-assess negatively because they feel like they should know something, and many do not self-assess negatively because they feel that they *shouldn't* be expected to know something. Similarly, we see that students negatively self-assess when they believe they cannot recover, and do not self-assess negatively when they think they can. This suggests that the underlying difference between students' responses are related to their knowledge of what to expect when learning to program and their confidence in their skills.

When we examine the programming moments when students apply certain reasons, we also notice interesting patterns that may have implications for instructional and intervention design. For example, we saw multiple instances of students thinking they should not need to restart because they should know how to solve the problem, but we did not see corresponding instances of students expecting that they may need to restart because they do not expect to know how to solve the problem. This suggests that while at least some students are aware of expected practices in many moments, this is not true for all moments.

Furthermore, when we examine the sources that students provided, we notice that students gave many more sources in cases where they did not self-assess negatively than where they did. This suggests that interactions with professionals, instructors, course policy, and peers can help students understand what experiences are expected while learning to program, while the absence of these interactions may lead to negative self-assessments due to a lack of accurate information about what to expect. This finding highlights an important opportunity for instructors to provide explicit sources that set student expectations about the programming process, especially since we found evidence that course materials may (unintentionally) embed implicit standards that reinforce negative self-assessments.

Research has shown that students' self-assessments can be implicit [45], meaning that students may make inaccurate assessments of their abilities without being aware of it. Our study reveals the potential value of making students' self-assessments explicit so they are more aware of their self-assessment criteria. In an unprompted comment about taking part in our study, P3 said it "was a good thing to reflect, like in my perspective as well," showing that he found it helpful to talk about his beliefs. This suggests that structured reflections around self-assessment criteria may help students develop more accurate expectations about the learning process.

Research on formal self-assessments has also shown that making the criteria that students should use explicit and providing feedback on self-assessments can be helpful for increasing assessment accuracy [1]. While in-moment self-assessments differ from formal self-assessments, providing students with explicit knowledge about what are (and are not) signs that they are doing poorly would likely be helpful. Our findings provide two suggestions for information that we could make more explicit. First, we could be clear about what we expect students to be able to do as learners. The most

common reason for negative self-assessment in our data was high expectations that did not align with the student's skill levels (e.g., feeling like they should not have bugs). Second, we could provide more explicit sources of information to ground students' expectations. Our data revealed the power that instructors have to model practices such as debugging and to set course policies that are in line with accurate expectations about the learning process. Being more intentional could go a long way towards reducing overly negative self-assessments.

In addition to making appropriate self-assessment criteria explicit, instructors can help students build skills to recover from the setbacks they encounter. We found that students often self-assess negatively when they do not have an immediate path to a solution. While help-seeking is a good strategy, help is not always available, as P2 noted in her self-assessment when not understanding an error message. Equipping students with more problem-solving strategies and techniques for referencing resources may help to resolve this issue. Research on debugging education suggests that explicitly teaching debugging strategies is necessary [35], and our data provides further evidence to support this claim.

Finally, we want to highlight our finding that, in the absence of explicit information about what to expect while learning programming, students are interpreting all of the information they can access, including the words, actions, and policies of their instructors, to formulate their expectations. This can be positive in some cases, like when students see their instructor make mistakes, but detrimental in others, like when grading policies reinforce negative self-assessment criteria. To avoid the latter case, instructors must be conscious that they are conveying messages through their words and actions, even those that are unintended.

5.1 Limitations

While our study provides valuable new insights into the reasoning behind students self-assessments, it has a number of limitations. First, we worked with a self-selected group of students from the same university, and our data only represents the perspectives of seven students. As a result, we do not make claims about the generality of our findings, but rather seek to illustrate the phenomena we observed. While we expect that our findings may translate to other students and contexts, future research will need to confirm this. Also, we chose to interview students who gave the most negative self-assessments on the filtering survey since we were most interested in understanding and supporting these students. While all of our participants responded with both negative and non-negative self-assessments each week, future work should study whether there are differences in the reasons and sources used by students who tend not to negatively self-assess.

Another limitation is that we did not observe students' self-assessments in-situ; rather, we interviewed students about their responses to hypothetical vignettes. We found that students often drew on their past experiences when responding, which gives us confidence that their statements reflect how they react to these situations, but we cannot be certain without observations. Finally, our results may be biased by the negative framing of the vignettes, which all describe cases of students feeling badly about their abilities. Our participants disagreed with the vignettes often, stating that they do not feel badly in similar moments, which suggests that the framing does not necessarily bias students towards negative self-assessments. However, further research is needed to understand how students would react to positively-framed vignettes.

6 CONCLUSION

In this paper, we explore the common reasons for students' self-assessments. We found that students often rely on their perceptions of their capabilities, their expectations, and their observations of "normal" behavior in the computing community to self-assess their performance. Furthermore, we identify unprompted sources for these reasons, such as the behaviors and policies of instructors and the actions of peers and professionals. By understanding the reasons and sources for students' self-assessments, we gain a better understanding of what interventions are needed to support accurate self-assessments. In future work, we look to further analyze the role of identity and motivation in students' self-assessments through case studies, explore the impact of students' interpretations of course policies, directly investigate the sources behind self-assessments, and design interventions to help students make more accurate self-assessments.

ACKNOWLEDGMENTS

This project is supported by the National Science Foundation under grant IIS-2045809 and a Design Cluster Research Fellowship from the Center for Human-Computer Interaction and Design at Northwestern University. We thank the Delta Lab, Bruce Sherin, and Reed Stevens for their feedback, and the participants and instructors of these courses for their time.

REFERENCES

[1] Heidi Andrade and Anna Valtcheva. 2009. Promoting Learning and Achievement Through Self-Assessment. *Theory Into Practice* 48, 1 (Jan. 2009), 12–19. https://doi.org/10.1080/00405840802577544

[2] Petek Askar and David Davenport. 2009. An Investigation of Factors Related to Self-Efficacy for Java Programming Among Engineering Students. *The Turkish Online Journal of Educational Technology* 8, 1 (2009).

[3] Albert Bandura. 1982. Self-efficacy mechanism in human agency. *American Psychologist* 37 (1982), 122–147. https://doi.org/10.1037/0003-066X.37.2.122

[4] Albert Bandura. 1997. *Self-efficacy: The exercise of control.* W H Freeman/Times Books/ Henry Holt & Co, New York, NY, US.

[5] Albert Bandura and Nancy E. Adams. 1977. Analysis of self-efficacy theory of behavioral change. *Cognitive Therapy and Research* 1, 4 (Dec. 1977), 287–310. https://doi.org/10.1007/BF01663995

[6] Jens Bennedsen and Michael E. Caspersen. 2019. Failure rates in introductory programming: 12 years later. *ACM Inroads* 10, 2 (April 2019), 30–36. https://doi.org/10.1145/3324888

[7] 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

[8] 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 (July 2014), 153–192. https://doi.org/10.1080/08993408.2014.963363

[9] Paulo Blikstein and Sepi Hejazi Moghadam. 2019. Computing Education: Literature Review and Voices from the Field. In *The Cambridge Handbook of Computing Education Research*, Anthony V. Robins and Sally A. Fincher (Eds.). Cambridge University Press, Cambridge, 56–78. https://doi.org/10.1017/9781108654555.004

[10] 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. https://doi.org/10.2307/1170684

[11] Simon Cassidy. 2015. Resilience Building in Students: The Role of Academic Self-Efficacy. *Frontiers in Psychology* 6 (Nov. 2015). https://doi.org/10.3389/fpsyg.2015.01781

[12] Elizabeth B. Cloude, Pranshu Kumar, Ryan S. Baker, and Eric Fouh. 2024. Novice programmers inaccurately monitor the quality of their work and their peers' work in an introductory computer science course. In *Proceedings of the 14th Learning Analytics and Knowledge Conference (LAK '24)*. Association for Computing Machinery, New York, NY, USA, 35–45. https://doi.org/10.1145/3636555.3636848

[13] Paul Denny, Andrew Luxton-Reilly, and Ewan Tempero. 2012. All syntax errors are not equal. In *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education (ITiCSE '12)*. Association for Computing Machinery, New York, NY, USA, 75–80. https://doi.org/10.1145/2325296.2325318

[14] Carol S. Dweck. 2006. *Mindset: The new psychology of success*. Random House, New York, NY, US.

[15] Alireza Ebrahimi. 1994. Novice programmer errors: language constructs and plan composition. *International Journal of Human-Computer Studies* 41, 4 (Oct. 1994), 457–480. https://doi.org/10.1006/ijhc.1994.1069

[16] Sue Fitzgerald, Renée McCauley, Brian Hanks, Laurie Murphy, Beth Simon, and Carol Zander. 2010. Debugging From the Student Perspective. *IEEE Transactions on Education* 53, 3 (Aug. 2010), 390–396. https://doi.org/10.1109/TE.2009.2025266

[17] 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*. ACM, Toronto ON Canada, 21–29. https://doi.org/10.1145/3291279.3339413

[18] 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 ACM Conference on International Computing Education Research (ICER '20)*. Association for Computing Machinery, New York, NY, USA, 170–181. https://doi.org/10.1145/3372782.3406273

[19] Jamie Gorson and Eleanor O'Rourke. 2021. CS1 Student Assessments of Themselves Relative to Others: The Role of Self-Critical Bias and Gender. *International Journal of the Learning Sciences* (2021), 597–600.

[20] Brian Hanks and Matt Brandt. 2009. Successful and unsuccessful problem solving approaches of novice programmers. In *Proceedings of the 40th ACM technical symposium on Computer science education (SIGCSE '09)*. Association for Computing Machinery, New York, NY, USA, 24–28. https://doi.org/10.1145/1508865.1508876

[21] Toni Honicke and Jaclyn Broadbent. 2016. The influence of academic self-efficacy on academic performance: A systematic review. *Educational Research Review* 17 (Feb. 2016), 63–84. https://doi.org/10.1016/j.edurev.2015.11.002

[22] Cynthia Hunt, Spencer Yoder, Taylor Comment, Thomas Price, Bita Akram, Lina Battestilli, Tiffany Barnes, and Susan Fisk. 2022. Gender, Self-Assessment, and Persistence in Computing: How gender differences in self-assessed ability reduce women's persistence in computer science. In *Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 1 (ICER '22)*. Association for Computing Machinery, New York, NY, USA, 73–83. https://doi.org/10.1145/3501385.3543963

[23] Paivi Kinnunen and Beth Simon. 2010. Experiencing programming assignments in CS1: the emotional toll. In *Proceedings of the Sixth international workshop on Computing education research (ICER '10)*. Association for Computing Machinery, New York, NY, USA, 77–86. https://doi.org/10.1145/1839594.1839609

[24] Päivi Kinnunen and Beth Simon. 2011. CS majors' self-efficacy perceptions in CS1: results in light of social cognitive theory. In *Proceedings of the seventh international workshop on Computing education research (ICER '11)*. Association for Computing Machinery, New York, NY, USA, 19–26. https://doi.org/10.1145/2016911.2016917

[25] 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

[26] Thomas D. LaToza, Gina Venolia, and Robert DeLine. 2006. Maintaining mental models: a study of developer work habits. In *Proceedings of the 28th international conference on Software engineering (ICSE '06)*. Association for Computing Machinery, New York, NY, USA, 492–501. https://doi.org/10.1145/1134285.1134355

[27] Kathleen J. Lehman, Julia Rose Karpicz, Veronika Rozhenkova, Jamelia Harris, and Tomoko M. Nakajima. 2021. Growing Enrollments Require Us to Do More: Perspectives on Broadening Participation During an Undergraduate Computing Enrollment Boom. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (SIGCSE '21)*. Association for Computing Machinery, New York, NY, USA, 809–815. https://doi.org/10.1145/3408877.3432370

[28] 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), 356–362. https://doi.org/10.1037/0022-0167.31.3.356

[29] 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

[30] 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 (ICER '11)*. Association for Computing Machinery, New York, NY, USA, 3–10. https://doi.org/10.1145/2016911.2016915

[31] Annie Li, Madeline Endres, and Westley Weimer. 2022. Debugging with stack overflow: web search behavior in novice and expert programmers. In *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Software Engineering Education and Training*. ACM, Pittsburgh Pennsylvania, 69–81. https://doi.org/10.1145/3510456.3514147

[32] Alex Lishinski and Aman Yadav. 2021. Self-evaluation Interventions: Impact on Self-efficacy and Performance in Introductory Programming. *ACM Transactions on Computing Education* 21, 3 (June 2021), 23:1–23:28. https://doi.org/10.1145/3447378

[33] 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 *Proceedings of the 2016 ACM Conference on International Computing Education Research (ICER '16)*. Association for Computing Machinery, New York, NY, USA, 211–220. https://doi.org/10.1145/2960310.2960329

[34] Robert McCartney, Anna Eckerdal, Jan Erik Mostrom, Kate Sanders, and Carol Zander. 2007. Successful students' strategies for getting unstuck. In *Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education*. ACM, Dundee Scotland, 156–160. https://doi.org/10.1145/1268784.1268831

[35] Renée McCauley, Sue Fitzgerald, Gary Lewandowski, Laurie Murphy, Beth Simon, Lynda Thomas, and Carol Zander. 2008. Debugging: a review of the literature from an educational perspective. *Computer Science Education* 18, 2 (June 2008), 67–92. https://doi.org/10.1080/08993400802114581

[36] Matthew B. Miles, A. M. Huberman, and Johnny Saldaña. 2014. Fundamentals of Qualitative Data Analysis. In *Qualitative data analysis: a methods sourcebook* (third edition ed.). SAGE Publications, Inc, Thousand Oaks, Californiia.

[37] Karen Multon, Steven Brown, and Robert Lent. 1991. Relation of self-efficacy beliefs to academic outcomes: A meta-analytic investigation.Journal of Counseling Psychology, 38, 30-38. *Journal of Counseling Psychology* 38 (Jan. 1991), 30–38. https://doi.org/10.1037/0022-0167.38.1.30

[38] 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. *ACM SIGCSE Bulletin* 40, 1 (March 2008), 163–167. https://doi.org/10.1145/1352322.1352191

[39] Frank Pajares. 2002. Gender and Perceived Self-Efficacy in Self-Regulated Learning. *Theory Into Practice* 41, 2 (2002), 116–125. https://www.jstor.org/stable/1477463

[40] Ernesto Panadero, Gavin T. L. Brown, and Jan-Willem Strijbos. 2016. The Future of Student Self-Assessment: a Review of Known Unknowns and Potential Directions. *Educational Psychology Review* 28, 4 (Dec. 2016), 803–830. https://doi.org/10.1007/s10648-015-9350-2

[41] 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

[42] Leo Porter and Beth Simon. 2013. Retaining nearly one-third more majors with a trio of instructional best practices in CS1. In *Proceedings of the 44th ACM technical symposium on Computer science education (SIGCSE '13)*. Association for Computing Machinery, New York, NY, USA, 165–170. https://doi.org/10.1145/2445196.2445248

[43] Chris Stephenson, Alison Derbenwick Miller, Christine Alvarado, Lecia Barker, Valerie Barr, Tracy Camp, Carol Frieze, Colleen Lewis, Erin Cannon Mindell, Lee Limbird, and Debra Richardson. 2020. Data analysis. In *Retention in Computer Science Undergraduate Programs in the U.S.: Data Challenges and Promising Interventions*. Association for Computing Machinery, New York, NY, USA.

[44] Jodi L. Tims, Cindy Tucker, Mark A. Weiss, and Stuart Zweben. 2023. Computing Enrollment and Retention: Results from the 2021–22 Undergraduate Enrollment Cohort. *ACM Inroads* 14, 4 (Nov. 2023), 24–43. https://doi.org/10.1145/3629980

[45] Philip H. Winne. 2011. A Cognitive and Metacognitive Analysis of Self-Regulated Learning. In *Handbook of Self-Regulation of Learning and Performance*. Routledge, Florence, United States.