Whiteboarding: A Tool to Improve CS1 Student Self-Efficacy

John Chapin
Virginia Tech University
Blacksburg, VA, USA
chapin@vt.edu

Bradley Bowen
Virginia Tech University
Blacksburg, VA, USA
bowenb@vt.edu

ABSTRACT

Many students struggle in Introductory Computer Science (CS1) and fail or drop out of the class. A lack of CS self-efficacy - the belief that the individual can complete a task - is frequently the cause of this failure to succeed in CS1. Solutions have been proposed to improve student self-efficacy in CS1. Unfortunately, a lack of self-efficacy in CS1 classes is still a problem. This study examines a pedagogical tool, whiteboarding, and its effect on student perception of self-efficacy during the programming problem-solving process for novice programmers. Whiteboarding refers to students using whiteboards during the CS problem solving process. Focus group sessions, researcher notes, and memos were used to collect qualitative data. The whiteboarding intervention was conducted in two AP CS A classes during the first four weeks of the year. Seventeen 10th-grade students participated in the focus groups. Three focus groups of four students and one focus group of five students were conducted at the end of the intervention. These findings indicate that whiteboarding can be a vital tool that increases student self-efficacy by improving their success at programming activities, increasing collaboration and feedback, and providing an active, positive learning environment that holds students accountable for their work. The themes that emerged from the focus group sessions were: Engagement with the Problem, Engagement with Others, and Engagement with the Environment. Teaching success in the CS1 classroom requires student self-efficacy. This study highlights a teaching pedagogy that CS1 educators can implement to increase student self-efficacy.

CCS CONCEPTS

• Social and professional topics → Computer science education; K-12 education; CS1; • Software and its engineering → Collaboration in software development

KEYWORDS

programming, self-efficacy, whiteboarding, computer science pedagogy

CompEd 2023, December 5–9, 2023, Hyderabad, India.

ACM ISBN 979-8-4007-0048-4/23/12.
https://doi.org/10.1145/3576882.3617925

ACM Reference format:

John Chapin and Bradley Bowen. 2023. Whiteboarding: A Tool to Improve CS1 Student Self-Efficacy. In *Proceedings of 2nd ACM Global Computing Education Conference (CompEd 2023). December 5-9, 2023, Hyderabad, India.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3576882.3617925

1 INTRODUCTION

The majority of jobs for STEM students are CS-related, yet CS students quit or fail introductory programming classes at high rates. Self-efficacy is the primary predictor of success for novice programmers [11,31]. Computer science pedagogy is still in its infancy, and new tools are needed to help teachers support student success [9]. Whiteboarding is an interactive teaching and learning method where students use a whiteboard to visually represent, explore, and discuss concepts, algorithms, data structures, and problem-solving techniques. This hands-on, collaborative approach encourages engagement, facilitates clear communication, and often aids in understanding complex topics by providing a visual and tangible way to organize and connect information. As a CS classroom pedagogical tool, whiteboarding positively impacts multiple CS self-efficacy factors.

This study's purpose was to understand how whiteboarding during the programming problem-solving process impacted novice student perceptions of CS self-efficacy. Literature exists on the factors that influence general self-efficacy [2]. Research has also highlighted the factors that impact self-efficacy in the CS domain, such as programming experience, gender, goal orientation and metacognitive strategies, understanding of programming concepts, mental models, and a sense of belonging [11,12,19,20]. Literature also exists on different pedagogical tools that can impact CS domain self-efficacy, such as self-assessment, scaffolded activities, and goal setting [7,11,28]. No literature, however, explicitly analyzes whiteboard use as a pedagogical tool to positively impact a novice CS student's self-efficacy.

This study contributes to the literature on the self-efficacy of novice CS1 students by analyzing how whiteboarding impacts student self-efficacy during the problem-solving process. This study aims to add a practical instrument to the CS teacher's toolkit to increase novice programmer success and retention by focusing on a pedagogical tool, whiteboarding, which can positively impact self-efficacy. The author analyzed the results from 17 students in four focus group sessions to address:

RQ How does whiteboarding in CS during the programming problem-solving process affect student self-efficacy?

2 RELATED WORK

2.1 Self-Efficacy

Self-efficacy is an "estimation of one's ability to perform target behaviors successfully. Individuals who judge themselves capable of performing certain tasks are found to attempt and successfully execute them" [4]. Students' CS self-efficacy is the best predictor of student success in the first year of computer science [11,31]. In other words, if students think they can do it, they probably will do it. There are causes for a lack of self-efficacy, and there are ways to improve self-efficacy.

2.1.1 Negative Feedback Loop. Self-efficacy and performance form a feedback loop that can be positive or negative. Students who believe they can accomplish a task will be more engaged and work harder. This increased effort usually results in a successful outcome [18]. A negative feedback loop can also occur. Low self-efficacy can lead to giving up on a problem. This failure reinforces lower self-efficacy [24] and reduces perseverance, which increases the risk of failure. In CS1, students write their code in an integrative development environment (IDE). This software provides constant error feedback to the student. The programmer receives feedback on many types of errors (e.g., syntax, compiler, logic; [14]).

2.1.2 Improving Self-Efficacy – Success Breeds Success. There are general factors that can influence self-efficacy in almost any classroom. Self-efficacy cannot be instilled via notes in class or a PowerPoint presentation by the teacher. Students build self-efficacy through their experiences [14]. Teachers can positively influence student self-efficacy by giving encouragement, placing them in positive collaborative environments, and modeling positive behavior. The most crucial factor, however, is that the student has positive experiences [2]; success breeds success.

Success and student self-efficacy are not built in one class session but over time, and early failures can be challenging to overcome [14]. Self-regulated learning (SRL) theory is a framework based on the belief that students should become active participants in their learning [17]. According to the SRL framework, teachers help students understand the learning process, develop metacognitive strategies, and set intermediate achievable goals, increasing student self-efficacy [18]. Such SRL activities include students accomplishing a small task, observing others accomplishing a task, encouragement from others, and monitoring one's physiological state to increase student self-efficacy [2].

Most importantly, self-regulated learners respond to feedback, allowing the patterns in long-term memory to be altered and corrected [18]. Self-efficacy enables students to realize that they can succeed if they practice and learn from that practice and are resilient as they work through setbacks [18].

Research on self-efficacy in the CS domain has identified CS-specific factors that impact CS self-efficacy. One method of improving self-efficacy in CS is to help students create accurate mental models. Developing these models directly affects a student's success in CS and self-efficacy [20]. Assignments that require students to comprehend and create (e.g., understanding an existing program and adding a module) can help students develop the proper mental models [24].

Breaking assignments into smaller, achievable tasks with immediate and appropriate feedback is better than assigning larger projects [24]. Worked examples, peer programming, and collaboration allow students to observe another person complete a task [24]. Other methods to improve self-efficacy include social persuasion during collaboration, decreasing competition, creating a sense of belonging, and creating evaluation methods that focus on learning and student improvement [19,24].

2.2 Whiteboards in the STEM Classroom

Whiteboards can be a useful pedagogical tool in the STEM classroom. Liljedahl [16], in his research on student use of whiteboards to problem solve in the Math classroom, found that whiteboards increased engagement. According to a study by Ju et al. [13], the whiteboard allows the problem-solver to work through the iterative process of creative and critical behavior because of its temporary surface. The study notes that whiteboards allow quick sketching to scaffold ideas and ground the conversation.

Whiteboards are also excellent tools for encouraging student collaboration. Grounding communication (finding common ground) is vital during collaboration. Collaboration also requires copresence, visibility, audibility, and simultaneity to minimize effort and decrease communication costs [5], all of which are accomplished using whiteboards. Other researchers have also found whiteboards help the collaborative design process because participants [26]: (a) pay more attention to each other; (b) maintain a shared focus; (c) sketch out ideas; (d) are open to each other's ideas; (e) seek agreement and acknowledge disagreement; (f) maintain a sense of humor.

Studies in non-CS STEM classrooms have revealed that whiteboarding helps teachers identify student misconceptions, provide specific feedback on an individual's misperceptions, identify group misconceptions, and offer feedback to the whole class [10]. Whiteboards also help teachers keep students engaged in the classroom [10].

3 METHOD

This study used a grounded theory approach to identify how whiteboarding in the CS1 classroom affects student self-efficacy. Focus groups were used to identify the themes and categories of activities the students believed helped them during the whiteboarding part of the programming process. This is one of the first studies to analyze the effect of the whiteboard pedagogical tool on student self-efficacy during the programming process.

3.1 Intervention

The research team identified and recruited potential subjects from two AP CS A classes offered at a public high school. The students were high school sophomores. A total of 17 students participated in the focus groups. The study was conducted during the beginning of the first semester in the fall. The students had never

used a whiteboard to solve a CS problem in a classroom setting. The IRB at Virginia Tech approved this research.

The teacher used vertical whiteboards as part of the programming solution process to increase the self-efficacy and engagement of the students. The whiteboard intervention was used during the first unit of the AP CS A classes. The first unit included four weeks of instruction, consisting of 12 classes of 90 minutes, following a lecture by the teacher that either introduced a new topic or reviewed an old topic. An example problem solution was discussed or live-coded, demonstrating proper problem-solving techniques. The teacher then explained the whiteboard intervention as listed below.

3.1.1 Components of the Whiteboarding Intervention. Students:

1. were introduced to a problem to be solved using the Java programming language.
2. solved the problem simultaneously with other students on the whiteboards in the room.
3. worked independently and/or collaboratively with each other and received assistance from the teacher.
4. typed their code in replit.com (an online auto-grader and IDE) after the teacher reviewed and approved their solution on the whiteboard.

Twelve assignments were completed during this period using the whiteboard intervention. The students had never used a whiteboard to solve a CS problem in a classroom setting.

3.2 Data Collection

3.2.1 Focus Group Protocol Seven factors and drivers have been identified as critical to student self-efficacy in programming [25] and are listed in Table 1. These factors were used to generate questions in the focus group protocol listed in Table 1.

3.2.2 Focus Group Composition The qualitative data were gathered through semi-structured focus groups after the intervention to obtain insights into what the students felt were the significant aspects of the intervention. The qualitative phase of the research was conducted to generate categories or themes of activity during the CS-programming problem-solving process using whiteboards to increase student self-efficacy. Seventeen students participated in the focus group sessions. Three focus groups of four students and one focus group of five students were conducted at the end of the intervention.

3.2 Data Analysis

The transcribed data from the focus groups were coded and analyzed using the two-cycle coding method [27]. Descriptive coding was used in the first cycle, and pattern coding was used in the second cycle [27].

Table 1: Self-efficacy factors and the aligned focus group protocol questions.

Factors	Questions
Independence and Persistence	Did you find the whiteboard a valuable tool when you were frustrated or stuck, and how did you become unstuck?
Simple Programming Tasks	What was the biggest challenge with the small programming tasks when you were whiteboarding? How did you overcome them?
Self-regulation	What was your planning process, and how did you use the whiteboard? How did you use your time? How much time did you spend thinking about the problem, and how much time did you spend writing or typing?
Complex Teacher Interaction	How did you interact with the teacher during whiteboarding?
Peer interaction (observing others accomplishing a task, encouragement)	How did using the whiteboard affect communication with your peers?
Mental models	How did you use the whiteboard other than to write code? Did you make any drawings?
Complex programming task	What was the most complex program you had to write?
General Experience	How did you use the whiteboard? What programming activity did you learn the most from?

3.3 Validity Discussion

We enhanced the quality of our study by taking measures before and during data collection. We aligned our focus group protocol questions to known factors and drivers of self-efficacy. We used field notes, memos, and peer debriefing to enhance validity [27]. Although the students in the focus groups were from the same class, the students were chosen at random, and the facilitator allowed each student time to answer each question. Each recording was analyzed to ensure one student did not dominate the discussion.

One weakness of the study lies in the fact that we only used the focus group data for this analysis. Adding observations of the actual whiteboarding experiences would have allowed comparison to the experiences shared by the students during the focus group sessions. Another weakness is the uniqueness of the student population. This study was conducted with advanced computer science students taking AP Computer Science A as high school sophomores. This is a class typically taken by high school seniors. However, we believe that these results are typical of students' experiences who use the whiteboard in a CS1 class.

4 RESULTS

A thematic analysis of the comments from the four focus group sessions was conducted, and three themes were developed to represent the sources of self-efficacy during the process of programming problem solving using the whiteboard. These themes were engagement with the problem, engagement with others, and engagement with the environment. The theme engagement with the problem consisted of three subthemes: planning, writing, and verifying. The theme engagement with others consisted of three subthemes: sharing ideas, feedback, and collaboration. The theme engagement with the environment consisted of three subthemes: writing, active learning, and positive accountability.

5 DISCUSSION

The following discussion relates the themes and subthemes found during the qualitative analysis to self-efficacy. The results reveal that whiteboarding is a pedagogical tool that positively impacts multiple factors that increase student self-efficacy in the CS domain.

5.1 Engagement with the Problem

Successfully solving and coding a problem, a novice programmer's mastery experience is the critical component that contributes to self-efficacy [1]. Problem solving involves understanding the problem, planning a solution, writing a solution, and verifying the solution [23]. Whiteboarding creates a space that helps students master the CS problem-solving process by allowing them to slow down and engage with the last three steps of the problem-solving process: planning, writing, and verifying.

5.1.1 Planning a Solution. Many novice programmers are under the false impression that good programmers do not need to plan [11]. However, expert programmers spend significant time planning [11]. Novice programmers' lack of planning during problem solving contributes to their lack of success [32]. The students using the whiteboard noted how it helped them plan the logical flow and made them think about it more. Specifically, it helped "to visualize it more… when I'm really writing it, I have to really think about exactly what I'm writing." Although it seemed like the planning took longer and was more tedious, the students found it easier and quicker.

Creating subgoals is important for improving self-efficacy during the CS problem-solving process [14]. During the planning phase on the whiteboard, many students found they could "chunk" the problem into smaller subproblems and that the "smaller chunks were so much easier to do on a whiteboard."

The participants contrasted their experience planning on the whiteboard with the usual experience of problem solving by typing directly in an Integrated Development Environment (IDE). Many novice students in a typical CS1 classroom (that does not utilize whiteboards for problem solving) receive a programming problem, open up the IDE on their computer, and start typing, skipping the planning stage of problem solving. One participant related how (before using a whiteboard) they would "mindlessly jump into the code and sometimes be typing stuff I didn't even know I typed." Problem solving was just a trial-and-error process that was not repeatable and reduced self-efficacy. But, when students started planning a solution on the whiteboard, one student found that they had "to think about it, and that does seem more tedious, but in the end, it is way easier."

5.1.2 Writing a Solution. Two components of written code are required to solve the problem. The first component is the correct logical sequence, and the second is the correct coding language syntax [8]. Novice programmers are new to much of the programming language syntax, and self-efficacy decreases when they forget the syntax but feel they should remember it [11]. This lack of knowledge is highlighted when participants program directly in an IDE because the program underlines every syntax error in red. The participants mentioned they became distracted by these syntax errors (which are many for a novice programmer).

On the other hand, the students found that since the whiteboard offered no feedback on syntax errors, they could focus on making sure the logic of their code was correct. One participant shared that using a whiteboard allowed her to slow down and focus on their code's logic and problematic portions. Seeing their plan and code in the same space also allowed for an iterative process of writing and "looking back" at the plan and then writing more code. Using the whiteboard to help students focus on the logic first (and reducing the focus on perfect syntax) and then allowing them to focus on the syntax (when they typed their code in an IDE) after the logic was correct helped the participants "solve problems and code faster."

5.1.3 Verifying the Solution. Verifying a solution is the final step in the problem-solving process [23]. Good mental models positively affect the success of novice programmers because mental models play a significant role in debugging and modifying (verifying) code [24]. The participants shared that the whiteboard was an excellent surface to draw their models and verify their code. The students used the whiteboard to verify their code by stepping through their logic, drawing models of their code, and hand-tracing the values of the variables. The unstructured surface allowed students to draw models of the data structure values represented by their code. For example, they could "write out the string and the indices and hand trace it a few times to find some errors." One student found that when he only used an IDE, the IDE only "gives you an output error message or the output of a wrong answer," but that tracing code on the whiteboard "helps you see where your code went wrong."

The students contrasted the whiteboarding experience with only using an IDE to verify their code. The "IDE-only" experience was frustrating for many participants because cryptic error messages did not help them find and fix logic errors. On the other hand, several of the advanced students proficient in using the debugger in the IDE found that, for complex problems, the IDE's debugger made it easier to find and fix logic errors.

Using mental models and hand-tracing on the whiteboard to verify code led to students reporting reduced anxiety and an

increase in their confidence. As one student explained, "…when you are writing it and hand-tracing it… the fear of making mistakes goes away; it's almost none."

5.2 Engagement with Others

The second theme, engagement with others, consisted of three subthemes: sharing ideas, instructor feedback, and peer collaboration. Presenting a student with a problem they cannot solve without a more-knowledgeable other (student or teacher) and providing the opportunity for the student to interact with the more-knowledgeable other is when learning occurs in the Zone of Proximal Development (ZPD; [30]). In addition to learning, self-efficacy increases when students engage with others. Bandura [2] notes that engagement with others through vicarious experiences and social persuasion from a peer or teacher can increase self-efficacy. Helping students connect with their peers in a cooperative learning environment can reinforce students' sense of self-efficacy in the CS classroom [29].

The participants whiteboarding in a public space could see their peers' work, their peers could see the participants' work, and the teacher could easily and instantly see everyone's work. This public space allowed for an environment in which participants could engage with each other in their ZPD. Collaborative learning allowed the students to gain mastery experiences and increase their self-efficacy.

5.2.1 Sharing Ideas. Sharing ideas during problem solving with others increases a student's mastery orientation and self-efficacy [6]. A student's sense of belonging in the CS classroom is also closely linked to self-efficacy [29]. The participants found that working on the same problem at the same time (simultaneity), being physically next to each other (copresence), and seeing each other's work in an ample workspace made it easier to have a shared focus and to share ideas. This situation created an environment where students felt "in it together" (belonging) as they planned their work, viewed each other's different approaches to problem-solving, and identified and fixed errors.

Programming self-efficacy increases when students observe more talented peers [19]. Whiteboarding provides ample opportunity for students to learn from each other. One student would finish their code and then look at their neighbor's solution to discover, "what did they do, and how does this work compare to mine?" Some students needed help and found it was easy to "just look over at your peer and be like, 'Hey, I'm struggling here. Can you help me with my for loop?'" Other students liked working through their problems aloud and explaining them to the person next to them.

Learning environments in which there appears only to be one solution can lead to a lack of a sense of belonging and decreased self-efficacy [14]. Several of the participants noted that whiteboarding allowed them to see multiple solutions. As the students shared their ideas and solutions, they realized there may be more than one solution and that whiteboarding helps "you to think differently." One student mentioned that the public workspace and copresence of peers allowed their peers to help them because they could "physically show you exactly where the error is. Not just the line, but where on the line it is, too." The classroom participants became focused on learning, seeing multiple solutions, and how to improve rather than simply making the code work.

5.2.2 Instructor Feedback. Instructor feedback is essential to the whiteboarding experience and can significantly impact student self-efficacy [33]. Verbal persuasion (telling a student they can do it) and giving the student the fundamental tools to accomplish the task can also positively impact self-efficacy [1]. Whiteboarding allows a teacher to quickly identify students who need help, communicate with them, and offer support. The students related how they could quickly get the teacher's help when needed. The teacher could "easily see the code of everyone in the room," and another student mentioned that when "the teacher saw that I was stuck in the same place… the teacher came over and asked me to think through the process, which would help me to figure it out." Several students also mentioned that the feedback from the teacher was immediate because they could see all of the student's work.

This approach differed significantly from the participants' experiences of problem solving directly in the IDE. When they coded in the IDE, "it was not easy to ask for help. Sometimes, people do not ask for help until they absolutely cannot figure it out." The participants felt isolated when they only worked in an IDE because "we're in our own little world."

5.2.3 Collaboration. Providing students with opportunities to work with peers can improve their self-efficacy in the CS classroom [22]. By working with others, students can participate in vicarious experiences, verbally persuade each other, and increase their self-efficacy [1]. During whiteboarding, the participants collaborated with their peers to generate ideas, identify and fix problems, and develop solutions.

One student shared that they could "give each other ideas on how to code something. Instead of just like copying someone's code, you can actually understand how they did it. That helps you in the future." Collaboration was much easier if one student was stuck because the code and the mental model were visible to both students. Visualized thinking on the whiteboard allowed one student to draw a line under a specific piece of code that might be incorrect and walk the other through the code to understand the error and the correct solution. One student summed up collaboration during whiteboarding as "probably the most important thing in whiteboarding because the goal in CS is to find and develop new methods that are more and more efficient, and by working together and learning from each other's failures, we can achieve that a lot faster and a lot better."

5.3 Engagement with the Environment

The final theme from the focus group sessions was engagement with the environment. The subthemes were the act of writing, active learning, and positive accountability. Integrating the mind and the hands in "hands-on" (problem-based and collaborative)

education has been proven to increase self-efficacy [3]. When students stand at the whiteboard as a classroom group and work on the same problem, "hands-on" active learning creates a positive accountability environment in which students learn and the learning "sticks."

5.3.1 Act of Writing. The physical act of handwriting on the whiteboard forced the participants to slow down and think about what they were writing, allowing for better visualization and expression and leading to understanding and remembering the solution (the learning "sticks"). There were some drawbacks to handwriting on the whiteboard: some found it tedious compared to typing on a computer.

However, most participants found that handwriting code forced them to think about what they were writing and reduced their cognitive load. Novice programmers do not have the automaticity of syntax writing and basic algorithmic patterns [8]; therefore, they need to spend more time thinking about what they are writing. Slowing down allowed them to find "shorter, faster, more efficient solutions" because they were "actively thinking" about what they were writing.

The physical act of handwriting code also helped the students remember what they wrote, although sometimes, handwriting all their code on a whiteboard became unwieldy. One student shared that handwriting helped give them a "mental imprint of the syntax" and that writing things out forced them to learn it. However, several students mentioned that the whiteboard could seem crowded and messy when writing more extended code.

5.3.2 Active Learning. Instead of just sitting and listening, active learning engages students in thinking about and solving problems [21]. The physical nature of standing and handwriting code as they solved a CS problem helped the participants focus on the problem, forced them to work, and created a fun, positive environment.

Active learning helps students foster a master orientation and increases their self-efficacy [14]. The participants expressed how whiteboarding's active learning aspect helped them focus and "get into" what one student called "the programming zone." When the students sat and solved CS problems at their computers using the IDE, they found it difficult to concentrate and focus because their minds would wander, and it was easy to procrastinate. However, when the students were standing, writing, and moving around on the whiteboard, they were "thinking a lot more actively." The students mentioned that whiteboarding created an environment where they wanted to work and felt a sense of urgency.

Self-efficacy is increased when a student is in a calm physiological state and not when anxious or apprehensive [24]. The students explained that the whiteboarding environment was "fun," even though it was stressful for the first few days. After this short adjustment period, the students "got used to the process." Several students specifically mentioned whiteboarding created a "calm, positive learning environment."

5.3.3 Positive Accountability. Whiteboarding created a positive environment that encouraged accountability. One reason for this atmosphere was that the whiteboard does not offer negative feedback (in contrast to the red underlines for syntax errors in the IDE). In addition, since everyone made mistakes that could be seen by everyone else, this only increased the positive environment because the students realized that perfection was not expected. The public and personal aspects of writing their code increased student accountability and confidence.

High-ability expert programmers make errors [11]. However, novice programmers mistakenly assume that making errors is a sign of low ability, decreasing their self-efficacy [11]. Many students contrasted working in an IDE, which provides many error messages, with working on the whiteboard, with no error messages. One of the most surprising results from the focus groups was the negative experience that novice programmers have programming in an IDE. When students type their code in an IDE, it underlines any syntax errors in red as they type. The participants described the IDE as "screaming at you" and that it was really "in your face," saying, "YOU'RE WRONG." One student called the IDE "very judgmental." Another participant described the experience as "me failing and failing and failing, hitting run, no, run, no."

The participants contrasted the experience of problem solving in the IDE with the whiteboard, which was not judgmental and gave students greater self-confidence because they could write their plan and code down and not immediately receive negative feedback. Students could then solve the problems more quickly.

A sense of belonging correlates with CS self-efficacy [11]. Whiteboarding led to the participants feeling like they belonged in the CS class. In the first few days of the intervention, most participants thought the teacher checked their code on the whiteboard and expected it to be perfect. However, after a few days, they saw "a bunch of other people also making a mistake. I realized that I wasn't the only person." Another student expressed that "after you realize everyone else makes mistakes too, and you realize you're basically all in the same boat and there is nothing to be negative or nervous about." The students also mentioned that once they realized everyone makes mistakes and their work was public, giving and receiving help was easy, reducing student stress and increasing their sense of belonging.

Social pressure is among the few extrinsic factors positively correlated with CS self-efficacy [15]. Social pressure in the whiteboarding environment results from the public display of the student's work. The students explained that since their code was public, they had to "know how it works" if anyone asked about it. This situation increased the accountability and pressure on the students, but it did not result in negative stress; it resulted in the participants becoming "more confident."

6 CONCLUSION

Learning in the CS1 classroom requires student self-efficacy. This study indicates that whiteboarding can be a vital tool that increases student self-efficacy by improving their success at programming activities, increasing collaboration and feedback, and providing an active, positive learning environment that holds students accountable for their work.

REFERENCES

[1] Albert Bandura. 1977. Self-efficacy: Toward a unifying theory of behavioral change. *Psychological Review* 84, 2 (March 1977), 191–215. DOI: https://doi.org/10.1037/0033-295X.84.2.191

[2] Albert Bandura. 1986. *Social foundations of thought and action : a social cognitive theory.* Prentice-Hall, Englewood Cliffs, N.J. Retrieved from http://www.gbv.de/dms/bowker/toc/9780138156145.pdf

[3] Eric Brubaker, Vikas Maturi, Barbara Karanian, Sheri Sheppard, and David Beach. 2019. Integrating Mind, Hand, and Heart: How Students Are Transformed by Hands-On Designing and Making. In *2019 ASEE Annual Conference & Exposition Proceedings*, ASEE Conferences, Tampa, Florida, 32988. DOI: https://doi.org/10.18260/1-2--32988

[4] Pat Byrne and Gerry Lyons. 2001. The effect of student attributes on success in programming. In *Proceedings of the 6th annual conference on Innovation and Technology in Computer Science Education* (ITiCSE' 01), Association for Computing Machinery, Canterbury, United Kingdom, 49–52. DOI: https://doi.org/10.1145/377435.377467

[5] Herbert H. Clark and Susan E. Brennan. 1991. Grounding in communication. (1991).

[6] Catherine H. Crouch and Eric Mazur. 2001. Peer Instruction: Ten years of experience and results. *American Journal of Physics* 69, 9 (September 2001), 970–977. DOI: https://doi.org/10.1119/1.1374249

[7] Mohsen Dorodchi, Aileen Benedict, and Erfan Al-Hossami. 2019. CS1 Scaffolded Activities: The Rise of Students' Engagement. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*, ACM, Toronto ON Canada, 299–299. DOI: https://doi.org/10.1145/3291279.3341209

[8] Benedict Du Boulay. 1986. Some Difficulties of Learning to Program. *Journal of Educational Computing Research* 2, 1 (February 1986), 57–73. DOI: https://doi.org/10.2190/3LFX-9RRF-67T8-UVK9

[9] Sally Fincher and Anthony W Robins. 2019. *The Cambridge handbook of computing education research.* Retrieved August 16, 2022 from https://doi.org/10.1017/9781108654555

[10] Tricia Forrester, Carolyn E. Sandison, and Sue Denny. 2017. Vertical Whiteboarding: Riding the Wave of Student Activity in a Mathematics Classroom. *Australian Mathematics Teacher* 73, 4 (2017), 3–8.

[11] 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*, ACM, Virtual Event New Zealand, 170–181. DOI: https://doi.org/10.1145/3372782.3406273

[12] Nathan Higgins, Sarah Frankland, and Joseph Rathner. 2021. Self-Regulated Learning in Undergraduate Science. *IJISME* 29, 1 (April 2021). DOI: https://doi.org/10.30722/IJISME.29.01.005

[13] Ju, Wendy, Lawrence Neeley, Terry Winograd, and Larry Leifer. Thinking with Erasable Ink: Ad-hoc Whiteboard Use in Collaborative Design. 9.

[14] 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, Providence, Rhode Island, USA, 19–26. DOI: https://doi.org/10.1145/2016911.2016917

[15] Kris M.Y. Law, Victor C.S. Lee, and Y.T. Yu. 2010. Learning motivation in e-learning facilitated computer programming courses. *Computers & Education* 55, 1 (August 2010), 218–228. DOI: https://doi.org/10.1016/j.compedu.2010.01.007

[16] Peter Liljedahl. 2016. Building Thinking Classrooms: Conditions for Problem-Solving. In *Posing and Solving Mathematical Problems*, Patricio Felmer, Erkki Pehkonen and Jeremy Kilpatrick (eds.). Springer International Publishing, Cham, 361–386. DOI: https://doi.org/10.1007/978-3-319-28023-3_21

[17] Alex Lishinski, Aman Yadav, Richard Enbody, and Jon Good. 2016. The Influence of Problem Solving Abilities on Students' Performance on Different Assessment Tasks in CS1. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (SIGCSE' 16), Association for Computing Machinery, Memphis, Tennessee, USA, 329–334. DOI: https://doi.org/10.1145/2839509.2844596

[18] 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, Melbourne, VIC, Australia, 211–220. DOI: https://doi.org/10.1145/2960310.2960329

[19] Pruthikrai Mahatanankoon. 2018. Exploring the Antecedents to Computer Programming Self-Efficacy. In *Proceedings of the 10th International Conference on Advances in Information Technology - IAIT 2018*, ACM Press, Bangkok, Thailand, 1–6. DOI: https://doi.org/10.1145/3291280.3291791

[20] Lauri Malmi, Judy Sheard, Päivi Kinnunen, Simon, and Jane Sinclair. 2020. Theories and Models of Emotions, Attitudes, and Self-Efficacy in the Context of Programming Education. In *Proceedings of the 2020 ACM Conference on International Computing Education Research*, ACM, Virtual Event New Zealand, 36–47. DOI: https://doi.org/10.1145/3372782.3406279

[21] Nancy Kober (Ed.). 2015. *Reaching Students: What Research Says About Effective Instruction in Undergraduate Science and Engineering.* National Academies Press, Washington, D.C. DOI: https://doi.org/10.17226/18687

[22] Marina Papastergiou. 2010. Enhancing Physical Education and Sport Science students' self-efficacy and attitudes regarding Information and Communication Technologies through a computer literacy course. *Computers & Education* 54, 1 (January 2010), 298–308. DOI: https://doi.org/10.1016/j.compedu.2009.08.015

[23] George Polya. 2004. *How to solve it: A new aspect of mathematical method.* Princeton university press.

[24] Vennila Ramalingam, Deborah LaBelle, and Susan Wiedenbeck. 2004. Self-efficacy and mental models in learning to program. In *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education* (ITiCSE' 04), Association for Computing Machinery, Leeds, United Kingdom, 171–175. DOI: https://doi.org/10.1145/1007996.1008042

[25] Vennila Ramalingam and Susan Wiedenbeck. 1998. Development and Validation of Scores on a Computer Programming Self-Efficacy Scale and Group Analyses of Novice Programmer Self-Efficacy. *Journal of Educational Computing Research* 19, 4 (December 1998), 367–381. DOI: https://doi.org/10.2190/C670-Y3C8-LTJ1-CT3P

[26] John Rooksby and Nozomi Ikeya. 2012. Collaboration in Formative Design: Working Together at a Whiteboard. *IEEE Softw.* 29, 1 (January 2012), 56–60. DOI: https://doi.org/10.1109/MS.2011.123

[27] Johnny Saldaña. 2009. *The coding manual for qualitative researchers.* Sage, Los Angeles, Calif.

[28] Duane F. Shell, Leen-Kiat Soh, Abraham E. Flanigan, and Markeya S. Peteranetz. 2016. Students' Initial Course Motivation and Their Achievement and Retention in College CS1 Courses. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education - SIGCSE '16*, ACM Press, Memphis, Tennessee, USA, 639–644. DOI: https://doi.org/10.1145/2839509.2844606

[29] 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 - SIGCSE '13*, ACM Press, Denver, Colorado, USA, 65. DOI: https://doi.org/10.1145/2445196.2445220

[30] Lev Semenovich Vygotsky. 1980. *Mind in society: The development of higher psychological processes.* Harvard university press.

[31] Christopher Watson, Frederick W.B. Li, and Jamie L. Godwin. 2014. No tests required: comparing traditional and dynamic predictors of programming success. In *Proceedings of the 45th ACM technical symposium on Computer science education* (SIGCSE' 14), Association for Computing Machinery, Atlanta, Georgia, USA, 469–474. DOI: https://doi.org/10.1145/2538862.2538930

[32] Leon E. Winslow. 1996. Programming pedagogy—a psychological overview. *SIGCSE Bull.* 28, 3 (September 1996), 17–22. DOI: https://doi.org/10.1145/234867.234872

[33] Barry J. Zimmerman and Anastasia Kitsantas. 2002. Acquiring writing revision and self-regulatory skill through observation and emulation. *Journal of Educational Psychology* 94, 4 (2002), 660–668. DOI: https://doi.org/10.1037/0022-0663.94.4.660