Behaviors of Higher and Lower Performing Students in CS1

Soohyun Nam Liao[1], Sander Valstar[1], Kevin Thai[1], Christine Alvarado[1],
Daniel Zingaro[2], William G. Griswold[1], Leo Porter[1]
[1]University of California, San Diego
[2]University of Toronto, Mississauga

ABSTRACT

Although recent work in computing has discovered multiple techniques to identify low-performing students in a course, it is unclear what factors contribute to those students' difficulties. If we were able to better understand the characteristics of such students, we may be better able to help those students. This work examines the characteristics of low- and high-performing students through interviews with students from an introductory computing class. We identify a number of relevant areas of student behavior including how they approach their exam studies, how they approach completing programming assignments, whether they sought help after identifying misunderstandings, how and from whom they sought help, and how they reflected on assignments after submitting them. Particular behaviors within each area are coded and differences between groups of students are identified.

CCS CONCEPTS

• **Social and professional topics → Computing education**; **CS1**;

KEYWORDS

student performance, student behaviors, student interviews, CS1

ACM Reference Format:
Soohyun Nam Liao, Sander Valstar, Kevin Thai, Christine Alvarado, Daniel Zingaro, William G. Griswold, Leo Porter. 2019. Behaviors of Higher and Lower Performing Students in CS1. In *Innovation and Technology in Computer Science Education (ITiCSE '19), July 15–17, 2019, Aberdeen, Scotland, UK*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3304221.3319740

1 INTRODUCTION

Pass rates in introductory computing courses have been historically low with an average of 67.7% worldwide, and they have not improved significantly over the years [33]. At the same time, skyrocketing CS enrollments [7] make it more difficult for instructors to identify and support struggling students, as per-student resources are spread thinner.

Recent work has focused on building more accurate automated models for early identification of students who are likely to struggle to pass introductory courses. These models use a variety of naturally-occurring data sources, including programming behaviors and clicker data, and are capable of identifying potentially struggling students at increasing levels of accuracy [2, 5, 18, 24, 31, 34].

Yet, identifying which students are likely to struggle is only half the battle: we must also endeavor to help these students. Unfortunately, designing effective interventions is challenging, and the literature on interventions has yielded few successes [6, 9]. Interventions that require extra work, such as supplemental help sessions or the opportunity to redo prior work, usually fail to reach struggling students, who have too much on their plate already [16, 23]; more general "learning how to learn" interventions are too far removed from the specific course material to be effective [9]. Moreover, one-size-fits-all interventions may fail in CS because we have reason to suspect that students in CS classes may succeed for different reasons than those in other disciplines [36].

A deeper understanding of the behaviors employed by successful and unsuccessful students in introductory computing would aid in developing effective interventions. To gain a broad understanding of the gamut of student behaviors, we conducted a series of semi-structured interviews with students in an introductory CS course at a large public university and qualitatively analyzed interview transcripts. Our choice of qualitative research methods is informed by our goal of exploring students' individual contexts and identifying new variables of interest [8].

In these interviews, we inquired about student behaviors in the course, including how they prepare for exams and solve programming assignments. To analyze these interviews, we used an open-coding approach [12, 22] to identify and categorize student behaviors. A natural division appeared between behaviors associated with preparing for exams and behaviors associated with completing programming assignments. We believe some of the behaviors within these two categories are specific to learning computer science, whereas others fall under general study skills.

At the end of the course, with the knowledge of which students ultimately performed poorly or well, we analyzed the codes from interview transcripts and found that for some categories of behavior, the behaviors of unsuccessful and successful students varied in distinctive ways. This distinction lays the foundation for future work to study the prevalence of these behaviours. Moreover, researchers may be able to experiment with interventions geared at encouraging positive behaviors common to higher-performing students and/or discouraging behaviors common to lower-performing students.

2 BACKGROUND

Researchers have qualitatively studied the factors that motivate students to persist or drop out of introductory computing courses. Those studies found that time constraints, low prioritization of CS courses, ineffective study skills, difficulty, and motivation were characteristics often associated with students who drop CS1 [26, 32].

ITiCSE '19, July 15–17, 2019, Aberdeen, Scotland, UK

ACM ISBN 978-1-4503-6301-3/19/07…$15.00
https://doi.org/10.1145/3304221.3319740

Table 1: Components of the Studied Course

Component	Description
Lecture	Class with clickers and Peer Instruction. [27]
Podcast	Recording of class, posted online.
Discussion	Extra weekly lecture run by graduate student TAs. Attendance optional.
Office Hours	Held by undergraduate tutors, graduate TAs, or the instructor.
Piazza	Online question board and discussion forum.
Reading Assignment	Weekly textbook-reading assignment for lecture preparation.
Lab	Weekly programming practice session led by tutors.
Lab Quiz	Online quiz given at the end of each lab.
Review Quiz	Weekly online quiz for the week's material.
PSA	Weekly programming assignment.
Star Points	Optional extra credit assignments for each PSA.
Practice Midterm	Midterm exam questions from a previous term, provided by the instructor.
Midterm	50 minute In-class exams; held in week 4 and 7.
Final	Summative Assessment in Week 11.

These prior studies uncover some of the same behaviors that we identify, but they focus only on those students who drop CS1 or leave computer science. In addition, there is a strong quantitative tradition that investigates student persistence and progress in introductory CS courses, focusing on factors such as motivation [29], achievement goals [36], measures of programming behavior [4, 10, 15, 35], and measures of students' social behaviors [3]. While these studies have informed techniques for identifying at-risk students, they do not give insight into the particular antecedents of these behaviours.

Finally, there are numerous qualitative studies that report on student behaviours and attitudes, such as study strategies of students when they are stuck [21], debugging or problem-solving strategies during programming [14, 25], students' perceptions of their ability [17], and students' perceptions of their behaviors [30]. That work, however, focuses on class-level data, not the particular experiences of individual students.

3 METHOD

Our goal was to discover behaviors that might be associated with higher or lower student performance in introductory computing courses. We chose an open-coding technique, which iteratively groups and regroups related fragments of interview transcripts in order to identify themes and uncover patterns in the data [12, 22]. Below, we describe our data collection procedure, interview structure, and analysis and coding process.

3.1 Course Context

We selected an introductory CS course ("CS1") at a large North American research-focused university as the context for our study. Students from two terms (fall and winter) participated in the study. This 10-week CS1 course is designed for CS majors, intended CS majors, and non-majors alike. It uses a media computation curriculum to teach students basic programming in Java [13]. Table 1 lists the main components of the course; we refer to these components throughout the paper.

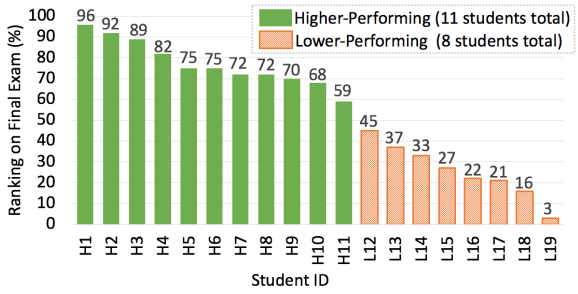

Figure 1: Interviewee Performance on Final Exam

3.2 Interview Participants

Students were recruited to participate in interviews, as approved by our institution's Human Subjects Review Board. Students were compensated for their time with gift cards. During the first term, we interviewed 8 students. We then used the data from these interviews to refine the interview process, and interviewed 19 students in the second term.

3.2.1 ***First Term.*** The first round of our study began in the middle of the term. The students had by that time completed their first midterm and received grades and feedback on it. The course instructor provided the midterm exam rankings of the 441 consenting students to a research colleague, who then helped us invite students (124) to participate. These 124 students were selected such that at least some portion were in the bottom 40% of midterm grades. Eight of these students agreed to participate in the interviews. To avoid unintentionally influencing the interviews, the interviewers were unaware of which students did poorly on the midterm.

3.2.2 ***Second Term.*** In the second round of the study, we began recruiting for the interviews before students took the first midterm, so we needed a different way of selecting a representative sample group. When recruiting students to participate in the study, we asked those interested to apply via an online survey. Part of that survey asked students to report the grade they were aiming to get in the course. 45 out of approximately 400 students responded to the survey. Because most students reported aiming for an "A" in the course, we first selected all students that aimed for a grade less than "A" (5 out of 45). One of those students did not respond to our request to schedule an interview. Students were then selected randomly until our participant pool reached 20 confirmed students (though one student dropped the class prior to the interview). Again, the interviewers did not inquire about actual student grades directly (though students sometimes volunteered the information) until after interviews were completed.

Although the grade a student is aiming for does not necessarily reflect their performance in the course, we felt that it would help us diversify the achievement levels of students in our pool. Indeed, as shown in Figure 1, the approach seemed successful.

3.3 Categorizing Performance

We used final exam grades to divide students from the second term into two categories after the interviews: Higher-Performing (HP) and Lower-Performing (LP). Students who received final exam

scores in the top 50% of the class were considered high performers, while those in the bottom 50% of the class were considered lower performers. This threshold reflects the way that Ahadi et al. defined at-risk and not-at-risk students [1]. We recognize that performance in a course is on a continuum; though we have used a binary classification in our analysis, we also include the rank of each student (among our interviewees) to help provide readers with a more nuanced perspective of performance.

3.4 Interviews

3.4.1 ***First Term.*** During the first term, we conducted interviews shortly after the first midterm and follow-up interviews after the second midterm. Starting one week before the second midterm, students were asked to keep a timetable. In this table they recorded the date, start time, finish time, what they studied, and the difficulty of the studied material (1-5, where 1 is very easy and 5 is very hard). Additionally, students were asked to take a picture of their work environment every time they filled in the table. During interviews, we used the timetable and pictures to help the students recall what exactly they were studying at a specific point in time, similar to Fincher et al.'s My Programming Week [11]. Finally, students were asked to bring all their study materials (e.g., laptop, notes, textbook) to the interview. Two authors interviewed the students; these authors were not associated with the course in any way.

3.4.2 ***Second Term.*** Three interviews were conducted for each student: after the first midterm, after the second midterm, and shortly before final exams. The main focus of our analysis is the first interview, as this interview captures student behaviors early in the term. To prepare for the first interview, students were asked to keep a timetable during the week prior to the first midterm. Students were again asked to take pictures of their study area and bring all their study materials to the interview. The interviews were semi-structured, with some prompts originating from our interview experience in the first term. Examples of these prepared questions include *"How is your term going so far?"*, *"How did you prepare for the midterm?"*, *"Do you usually begin your programming assignments on this day?"*, *"How did you fix this bug here in your code?"*, and *"Can you explain this method?"*. The same two authors from the first term interviewed the students and again were not associated with the course.

3.5 Interview Analysis

Analysis was performed by two of the authors. We first transcribed the interviews. Next, we chose two student interviews from the first term, segmenting their transcripts into topical components. We printed these snippets and worked together to assign codes to them. After this session we began coding the rest of the interviews. We iteratively improved our coding rubric and validated each iteration by having the two authors independently code at least 10% of our data (two interviews) and subsequently measuring the inter-coder reliability. After 3 total iterations of independently coding interviews and further refining the coding rubric, we had reduced our initial rubric from 111 codes to 41 codes and achieved an inter-coder reliability of 87% agreement.

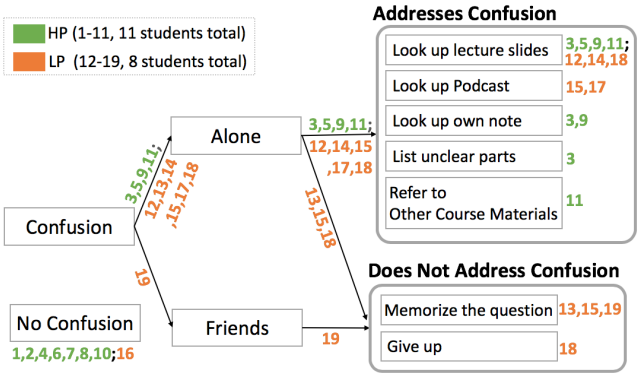

Figure 2: How Students Address Confusions Encountered on Practice Midterm (numbers represent student identifiers)

4 RESULTS

In this section, we describe our observations of how students prepare for the first midterm and work on programming assignments, including their particular study practices, help-seeking behaviors, willingness to correct identified gaps and confusion, and whether they continue to seek out answers to questions even after deadlines. We refer to students as "H" (Higher-Performing) or "L" (Lower-Performing) along with their final exam rank relative to other interviewees (e.g., 11 out of 19).

4.1 Midterm Preparation

All but four students started midterm preparation by working on the practice midterm. Two of those four students did not recall how they started their preparation. The third of the four, H11, used the practice midterm after reviewing other materials to assess whether they were prepared for the actual midterm. However, H11 identified some difficulties from the practice midterm, so they followed the same study pattern as the rest of the participants after taking the practice test: *"...I just got mistakes on, referencing [from the practice midterm], which showed I needed to study more on referencing" [H11].*

Figure 2 shows our codes for how students addressed concerns on the practice midterm while studying for the actual midterm. Seven HPs and one LP (L16) identified no significant difficulty on the practice midterm. Although they did not necessarily get every question entirely correct, they said they made only simple mistakes. For example: *"But they were just like ... dumb in my mind. I think one of them was uh, it asks 'What file is the, turtle class in?' And then I was like, 'Oh, isn't that dumb? It's in the turtle class.' But no, it's in the turtle.java, so I was like ...'Oh, of course' " [H6].*

Of the remaining 11 students, seven students reported resolving all of the difficulties that they identified (H3,5,9,11; L12,14,17), while the other four (L13,15,18,19) did not. All HPs who identified challenges succeeded in overcoming those challenges, while less than half of the LPs succeeded.

The following provides details on the four categories we identified in which HP and LP behavior differed.

4.1.1 ***Memorize or Give Up.*** The 11 students who identified confusion using the practice midterm worked to resolve it in different ways. As seen in Figure 2, the LPs who did not address their confusion either memorized the solution to relevant practice midterm questions (L13, 15, 19) or gave up, hoping it would not

be on the exam (L18): *"I just redid the practice exam properly to try and see how much of it I actually did remember how to do. That's kinda just standard procedure with me when I'm given a practice test" [L13]. "...the eight different kinds of primitives was actually on the midterm but...I didn't memorize them because I thought we didn't have to...[when studying them]" [L18].* Moreover, L15 mentioned making note cards for memorization: *"'Okay, I understand this problem,' and then just save that note card for in the future, for hypothetical case if I forgot how to do a certain problem..." [L15].*

L19 also exhibited a behavior of focusing excessively on getting the right answer, as opposed to gaining understanding (L19 was in the bottom 3% on the final exam): *"I would just sit there for a while trying to figure it out. And whenever I thought I finished, I'll get my friend to check it, and then he would tell me yes or no. And if he said no, then I would go erase it and do it again." [L19].*

4.1.2 ***Resources Used.*** As shown in the Addresses Confusion box in Figure 2, students used different resources when studying for the midterm and attempting to address their confusion. It was common for students to refer to lecture materials, with most examining the lecture slides. Only two students (L15,17), both LPs, watched the video podcasts of the lectures. One of those (L17) went so far as to watch multiple lectures back-to-back: *"I just sat on my bed, and put on the TV and then I sped them up a little and just listened to them...I skipped the first beginning ones because they were pretty straightforward, and then I picked some of the middle ones...then went up to whichever one before the midterm" [L17].*

Beyond lecture material, HPs availed themselves of additional resources such as their own notes, the textbook, quiz questions, and the Internet: *"...I look at how some functions the textbook asks us to do, or...in the slides, [but] some idea that I want to understand is not covered in the slides, I'm trying to refer to my notes to get the extra information" [H9].* H3 also took the time to list the things they were not sure about and reviewed it right before the midterm exam: *"If I missed something, like I didn't know this, sometimes I write it down...It was all just like, random notes...[and I read it in] like the morning before the midterm" [H3].* H11 generated and practiced their own questions to strengthen their understanding: *"So after finishing the test in, like, 30 minutes, I think I did, or 20 minutes. I just went back and studied references again...I went onto the Internet, and I learned it a little bit...And then I used the textbook. [Also] I just keep on trying to make new questions and see if it was right or not" [H11].*

4.1.3 ***Friends Involved.*** While all HPs and most LPs worked on the practice midterm on their own, two LPs (L14, 19) worked together with a friend (Friends box in Figure 2). However, both reported that their friend did not end up being helpful. L14 reviewed alone after the meeting with the friend to resolve confusion, and L19 did not resolve their confusion. L19 acknowledged that tutors would explain an answer in more detail, but ended up not going to tutor hours for the practice midterm because they were working on it at the last minute: *"one of my suite mates is also a CSE major. And I will just ask him. I mean, but the thing is that...he will just tell me how to do it. He doesn't teach me, just says...'This is how you do it.' I'm like, 'Okay.' But I don't know what this means. And that's why I'll go to the tutoring ... to be like, 'Uh, what does this mean? I don't get it"' [L19].*

4.1.4 ***After-exam Behavior.*** H1, whose final exam score was in the 96th percentile, worked to resolve a confusion after the midterm exam. No other students described such behavior: *"...I believe it's a problem about reference. I asked different tutors and they give me different answers. I just feel confused so I double-check it with my professor yesterday, even after the midterm" [H1].*

4.2 Programming Assignments

Figure 3 illustrates student behaviors when overcoming challenges during programming assignments. Every student except for H9 reported encountering challenges while working on programming assignments. The following subsections discuss each of the constitutive boxes of the figure, specifically the challenge they faced and whether they were successful in addressing the challenge.

4.2.1 ***Planning.*** Two HPs and three LPs started programming assignments quite late, a day or less before the deadlines. However, the HPs who reported starting late were the two strongest-performing HPs (H1,2), did not show any concern about starting late, and were able to complete the assignments successfully. On the other hand, two of the three LPs were the bottom-ranked LPs and bemoaned the lack of help available at the last minute. One of those two, L19, started assignments on the due date or the day before the due date. As such, the student had no choice but to work on the assignment until late at night: *"when I'm programming and there's an error, um, usually...it's late at night, so I have no one really to contact. So I just sit there for a long period sometimes trying to figure it out" [L19]. "Yes because I couldn't get my program to work properly, so...and there was no tutors, any help, because it was like almost last minute..." [L18].*

These two LPs also mentioned that they thought of posting questions on the Piazza discussion board, but did not feel that they had enough time to wait for a tutor's response. Therefore, they felt they had to either go back to campus to ask questions to a random classmate or just give up on the problem: *"... once I asked the question but didn't respond till the next day. But...I can't wait that long. So I just went to the lab...asked someone, sitting there, 'Do you know how to do this?"' [L19]. "considering like the time that I was working on it, I didn't ask on Piazza because I didn't expect a response by then... I worked on it too late so that was kind of the setback on being able to finish this" [L18].*

However, contrary to the rest of the LP students, L12 did not seem to worry about starting programming assignments late. Later, we noticed that L12 had a classmate who worked on the programming assignments with them, and L12 thought the classmate was much smarter than them. Based on L12's comments, it seemed likely that they were overdependent on their programming partner: *"... a lot of the times, I start on Saturday [the day before the due date] 'cause if I don't understand it the first day, I'll take a step back and come back the due date. And try and just do it because the assignments usually don't take that long."* After a long pause they continued, *"He's a lot smarter than me, 'Cause he has prior experience, but... I didn't really know what he was saying. So then, he wasn't that far into it, so we just started working together. He explained to me this part, and then, we just kind of did the rest of it together..." [L12].*

4.2.2 ***Debugging Challenges.*** All except H4, 6, and 9 brought up debugging experiences (i.e., fixing errors in the code) that were

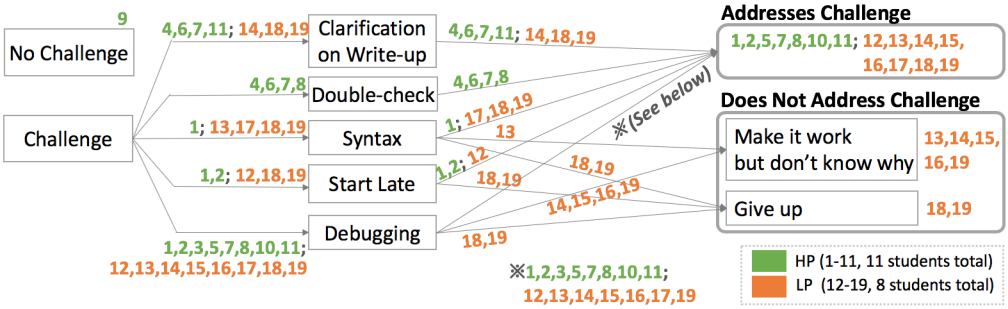

Figure 3: How Students Respond to Challenges during Programming Assignments (numbers represent student identifiers)

challenging to them. The majority of the LPs (L13,14, 15,16,18,19) were unable to resolve all of their debugging challenges. Once they were able to get their code to work, sometimes with help from tutors, they reported moving on without understanding why it worked. In some cases, the code was deemed to work because it appeared to have the desired effect or it passed test cases provided by the instructor: *"I don't know why it's going negative but my tests are passing. The whole time I was just trying to make my test pass. And when I finally got them passed, I was like, 'I shouldn't change anything.' 'Cause I feel like it's all gonna go down...the drain if I do. So I just left it as is...I turned it in" [L19].* When L14 was asked whether the friend's advice helped, they answered *"Yes. I think it did...We were able to somehow change it to make it actually display these values" [L14].*

Moreover, when we asked the LPs to explain a part of the code that they debugged with help from tutors or friends, some of them were not able to explain it at all. When L15 was asked about a return statement in their code, they replied *"Why I need that line? Um... It's similar to using the System.out.println()... is like the way I remember it... Okay, this was kind of confusing..." [L15].*

4.2.3 ***Syntax Challenges.*** H1 and four LPs (L13,17,18,19) revealed challenges on the correct usage of the Java language and methods. H1 had some initial difficulty with boolean operators, because the instructor had not covered that material yet. However, H1 figured it out by reading a web page: *"...at that time, we didn't learn this operator like || ...So I just look it up online, so I feel like, we need to use some concepts we never learned before...That's a challenge, I think" [H1].*

On the other hand, most LP students struggled with what they had already been taught in class. They either went to tutor hours to ask for help, or did not try to understand the problematic material: *"It was mostly questions that I was confused about. Like...arrays at the time. I knew what an array was... but I didn't understand how to code it" [L19]. "I wasn't sure why and how it [array indexing] is formatted that way...I was just a little lazy looking through the book so I didn't really understand it...it was probably part of the starter code, or I just looked at the book and just copied off it" [L13].*

4.2.4 ***Double-check Answers.*** Only HP students (H4,6,7,8) confirmed their answers with others—knowledgeable classmates or tutors—before they submitted their code: *"I talk with them...they're pretty knowledgeable in CS, too...It's more of just like, 'Hey, what did you do? Okay, I did this, too'... Like we both did it, we're just going over like, how we did it. Or if we're even doing it right...to make sure" [H6]. "I asked the TA, 'Oh, is this good?' But she told me, 'You gotta change it,'...so I had to re-change it...when the deadline comes up, I go to the lab. So, you know, just to make sure" [H7].*

4.2.5 ***Post-deadline Behavior.*** H2 did the Star Point assignment on their own: *"For PSA 2, um, it [the Star Point assignment] was a research into other object-oriented programming. That one, um, I didn't have enough time to actually start it, so I didn't do that one, either. Though, I did, like, afterwards [after the PSA deadline], when I had more time...I wouldn't be able to, like, uh, submit anything like that, but it was interesting to read on like, Python with how they do object-orientating" [H2].* On the other hand, L16 knew that they lost points from a submitted programming assignment and remembered the tutor's comments, but they did not understand why and did not try to address it: *"they said 'Even though you called the Math.abs(), you should put it in the while condition.' ... Eh, um, can I just say 'I don't know'?" [L16].*

5 DISCUSSION

5.1 Implications

Table 2 serves to summarize our findings from Section 4, focusing on the trends that emerged from our analysis. For each trend in the table, we refer back to the appropriate subsection in Section 4 for more details. Some of the observed trends may relate to more general (non-CS specific) study skills (e.g., HP students tend to seek out extra resources to help clarify material and resolve challenges). Similarly, some of these trends are tied to general student metacognition [28] (e.g., LP students may perform untargeted reviews of course materials before exams because they are unclear about where their conceptual weaknesses lie). Lastly, there are behaviors that clearly relate to student self-regulation [19] (e.g., HP continuing to work on assignments even after deadlines to ensure they fully understand course concepts, and the role of procrastination on each group). These trends may be encouraging to the computing education community insofar as we may be able to leverage literature outside of computer science to improve the ways that students study.

Some concerning behaviors do appear to be more computing-specific: assembling a programming assignment solution from bits contributed by friends and instructional staff, stopping work when the right answer is in hand before understanding why it is correct, or emphasizing memorization of code before an exam. These behaviors may be due to students' unfamiliarity with the field of computer science. They may simply be unaware that some behaviors, such as memorization of prior code solutions, may not be effective study strategies. Here, the computing education community can seek

Table 2: Summary of Differences Encountered Between Student Groups

	Higher-Performing	Lower-Performing
Exam study strategy	• execute targeted review of areas of perceived weakness (4.1.2) • create new questions to solve (4.1.2) • review course notes	• execute untargeted review (reviewing everything) (4.1.2) • memorize existing code (4.1.1)
Getting help	• solicit help, successfully addressing their challenges (4.2.3, 4.2.4)	• solicit help, only sometimes addressing challenges (ask friends for help, 4.1.3)
Addressing confusion	• refer back to their own course notes for answers (4.1.2) • seek out extra resources (4.1.2) • double-check solutions with others (4.2.4) • keep notes on areas of confusion (4.1.2)	• give up if cannot find answer (4.1.1, 4.2.1) • get correct answers from others, but do not seek to understand why the answers are correct (4.1.1) • through debugging, can get the code to work but may not understand why (4.2.2, 4.2.3)
Post-deadline behaviors	• will continue working on assignments (4.1.4, 4.2.5)	–
Procrastination	• when they do, it does not impact them significantly as they can still complete assignments (4.2.1)	• when they do, they are unable to find help when they need it to overcome problems (4.2.1)

ways to better inform students of productive and unproductive behaviors specific to computer science.

We also note that although some of these behaviors may result from lack of prior experience, we believe that many of the student struggles are tied to CS-specific and general study strategies. As such, we are encouraged that possible interventions might be successful without necessarily relying on background knowledge.

5.2 Next Steps and Possible Interventions

An ongoing next step to this work is to determine how common these behaviors are among students. This would allow us to craft interventions that target the most common (or most problematic) behaviors.

In the meantime, based on the results from Section 4, we argue for two features of interventions designed to help LPs that we believe may be most effective when applied in concert. First, help students to focus on achieving conceptual understanding rather than understanding particular solutions. While HPs tend to invest effort in clarifying and confirming their understanding (H3,9,11 in Section 4.1.2), LPs did not do so (L13,15,18,19 in Section 4.1.1), simply memorizing solutions without understanding the underlying concepts (Section 4.1.1 and the last two quotes of Section 4.2.2). Thus, the instructional team should keep in mind that there may be a mismatch between perceived and actual student understanding. We therefore argue for the importance of confirming conceptual understanding with students, rather than relying on student self-report.

Second, and perhaps more obviously, help students to be aware of deadlines and plan accordingly. Prior literature suggests that procrastination leads to lower grades on programming assignments [20]. We similarly find that effects of poor study practices appear to be amplified when students procrastinate. For instance, L19 could not ask for help to understand the concepts behind the practice midterm because they started working on it at the last minute. Thus, the instructional team might consider an intermediate deadline to encourage students to start earlier, or use email reminders to encourage early participation [20]. Similarly, exam review sessions might be scheduled well before the exam, rather than the day before. Ultimately, we believe these recommendations will be most effective when combined, as starting earlier will likely be ineffective unless one also abandons poor study strategies like code memorization.

5.3 Threats to Validity

Our interview participants were all from a single CS1 course, and our sample size is small, so findings may not generalize to students in this or other CS1 courses. For instance, if another CS1 course does not offer practice midterm questions, the observations we found might not be applicable. In addition, we are relying here on self-reported data, though we did ask students to log their behavior and take pictures of their work environment to increase reliability. Furthermore, we moved on to the next topic if we noticed, or students themselves mentioned, that they could not accurately reply to our prompt. Lastly, even though we sought not to impact student behaviors, students might have behaved differently during the term as a result of any reflection hatched by participation in our study.

6 CONCLUSION

This paper presents an analysis of CS1 interviews in an effort to understand the range of behaviors demonstrated by students. We distinguish between the behaviors of higher- and lower-performing students to investigate whether there are any notable trends in behaviors within or between groups. Our interviews revealed differing behavioral characteristics of higher- and lower-performing students that may be valuable in crafting potential interventions. Refining and deploying the intervention strategies suggested in this paper in a real classroom setting is an important next step for this work. The CS education research community has invested considerable effort in high-accuracy identification of at-risk students [1, 5, 18]. While that work remains important, we argue for a corresponding focus on using those identifications to intervene and help these students. We hope that our qualitative investigation here, along with our initial suggestions stemming from that work, will further this important research agenda.

7 ACKNOWLEDGEMENTS

We greatly appreciate the reviewers for their helpful feedback. This work was supported in part by NSF award 1712508.

REFERENCES

[1] A. Ahadi, R. Lister, H. Haapala, and A. Vihavainen. Exploring machine learning methods to automatically identify students in need of assistance. In *Proceedings of the 11th International Conference on Computing Education Research*, pages 121–130, 2015.

[2] S. Bergin, A. Mooney, J. Ghent, and K. Quille. Using machine learning techniques to predict introductory programming performance. *International Journal of Computer Science and Software*, 4(12):323–328, 2015.

[3] A. S. Carter and C. D. Hundhausen. With a little help from my friends: An empirical study of the interplay of students' social activities, programming activities, and course success. In *Proceedings of the 12th International Conference on Computing Education Research*, pages 201–209, 2016.

[4] A. S. Carter, C. D. Hundhausen, and O. Adesope. The normalized programming state model: Predicting student performance in computing courses based on programming behavior. In *Proceedings of the eleventh International Conference on Computing Education Research*, pages 141–150, 2015.

[5] K. Castro-Wunsch, A. Ahadi, and A. Petersen. Evaluating neural networks as a method for identifying students in need of assistance. In *Proceedings of the 48th Technical Symposium on Computer Science Education*, pages 111–116, 2017.

[6] S. P. M. Choi, S. Lam, K. C. Li, and B. T. M. Wong. Learning analytics at low cost: At-risk student prediction with clicker data and systematic proactive interventions. *Journal of Educational Technology & Society*, 21(2):273–290, 2018.

[7] CRA Enrollment Committee Institution Subgroup. Generation CS: Computer science undergraduate enrollments surge since 2006. Computing Research Association, 2017.

[8] J. W. Creswell and C. N. Poth. *Qualitative Inquiry and Research Design, Fourth Edition*. Sage Publications, Inc, 2017.

[9] L. Deslauriers, S. E. Harris, E. Lane, and C. E. Wieman. Transforming the lowest-performing students: an intervention that worked. *Journal of College Science Teaching*, 41:80–88, 2012.

[10] S. H. Edwards, J. Snyder, M. A. Pérez-Quiñones, A. Allevato, D. Kim, and B. Tretola. Comparing effective and ineffective behaviors of student programmers. In *Proceedings of the fifth international Conference on Computing education research*, pages 3–14, 2009.

[11] S. Fincher, J. Tenenberg, and A. Robins. Research design: necessary bricolage. In *Proceedings of the seventh international conference on Computing education research*, pages 27–32, 2011.

[12] B. G. Glaser and A. L. Strauss. The discovery of grounded theory: Strategies for qualitative research. *Transaction Publishers*, 1967.

[13] M. Guzdial. A media computation course for non-majors. *SIGCSE Bulletin*, 35(3):104–108, 2003.

[14] B. Hanks and M. Brandt. Successful and unsuccessful problem solving approaches of novice programmers. In *Proceedings of the 40th Technical Symposium on Computer Science Education*, pages 24–28, 2009.

[15] M. Jadud. Methods and tools for exploring novice compilation behaviour. In *Proceedings of the 10th International Conference on Computing Education Research*, pages 73–84, 2006.

[16] P. Jensen and R. Moore. What do help sessions accomplish in introductory science courses? *Journal of College Science Teaching*, 38(5):60, 2009.

[17] C. M. Lewis, K. Yasuhara, and R. E. Anderson. Deciding to major in computer science: A grounded theory of students' self-assessment of ability. In *Proceedings of the Seventh International Conference on Computing Education Research*, pages 3–10, 2011.

[18] S. N. Liao, D. Zingaro, M. A. Laurenzano, W. G. Griswold, and L. Porter. Lightweight, early identification of at-risk CS1 students. In *Proceedings of the 12th International Conference on Computing Education Research*, pages 123–131, 2016.

[19] D. Loksa and A. J. Ko. The role of self-regulation in programming problem solving process and success. In *Proceedings of the 12th Conference on International Computing Education Research*, pages 83–91, 2016.

[20] J. Martin, S. H. Edwards, and C. A. Shaffer. The effects of procrastination interventions on programming project success. In *Proceedings of the Eleventh International Conference on Computing Education Research*, pages 3–11, 2015.

[21] R. McCartney, A. Eckerdal, J. E. Mostrom, K. Sanders, and C. Zander. Successful students' strategies for getting unstuck. In *Proceedings of the 12th Conference on Innovation and Technology in Computer Science Education*, pages 156–160, 2007.

[22] A. J. Mills, G. Durepos, and E. Wiebe. Coding: Open coding. In *Encyclopedia of case study research*, 2010.

[23] R. Moore. Who does extra-credit work in introductory science courses? *Journal of College Science Teaching*, pages 12–15, 2005.

[24] J. P. Munson and J. P. Zitovsky. Models for early identification of struggling novice programmers. In *Proceedings of the 49th Technical Symposium on Computer Science Education*, pages 699–704, 2018.

[25] L. Murphy, G. Lewandowski, R. McCauley, B. Simon, L. Thomas, and C. Zander. Debugging: The good, the bad, and the quirky – a qualitative analysis of novices' strategies. In *Proceedings of the 39th Technical Symposium on Computer Science Education*, pages 163–167, 2008.

[26] A. Petersen, M. Craig, J. Campbell, and A. Tafliovich. Revisiting why students drop CS1. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*, pages 71–80, 2016.

[27] L. Porter, D. Bouvier, Q. Cutts, S. Grissom, C. Lee, R. McCartney, D. Zingaro, and B. Simon. A multi-institutional study of peer instruction in introductory computing. *ACM Inroads*, 7(2):76–81, 2016.

[28] G. Schraw, K. J. Crippen, and K. Hartley. Promoting self-regulation in science education: Metacognition as part of a broader perspective on learning. *Research in Science Education*, 36(1):111–139, Mar 2006.

[29] D. F. Shell, L.-K. Soh, A. E. Flanigan, and M. S. Peteranetz. Students' initial course motivation and their achievement and retention in college CS1 courses. In *Proceedings of the 47th Technical Symposium on Computer Science Education*, pages 639–644, 2016.

[30] B. Simon, S. Esper, L. Porter, and Q. Cutts. Student experience in a student-centered peer instruction classroom. In *Proceedings of the Ninth International Conference on Computing Education Research*, pages 129–136, 2013.

[31] A. Vihavainen. Predicting students' performance in an introductory programming course using data from students' own programming process. In *IEEE 13th International Conference on Advanced Learning Technologies*, pages 498–499, 2013.

[32] R. Vivian, K. Falkner, and N. Falkner. Computer science students' causal attributions for successful and unsuccessful outcomes in programming assignments. In *Proceedings of the 13th Koli Calling International Conference on Computing Education Research*, pages 125–134, 2013.

[33] C. Watson and F. W. Li. Failure rates in introductory programming revisited. In *Proceedings of the 19th Conference on Innovation and Technology in Computer Science Education*, pages 39–44, 2014.

[34] C. Watson, F. W. Li, and J. L. Godwin. No tests required: Comparing traditional and dynamic predictors of programming success. In *Proceedings of the 45th Technical Symposium on Computer Science Education*, pages 469–474, 2014.

[35] C. Watson, F. W. B. Li, and J. L. Godwin. Predicting performance in an introductory programming course by logging and analyzing student programming behavior. In *International Conference on Advanced Learning Technologies*, pages 319–323, 2013.

[36] D. Zingaro, M. Craig, L. Porter, B. A. Becker, Y. Cao, P. Conrad, D. Cukierman, A. Hellas, D. Loksa, and N. Thota. Achievement goals in CS1: Replication and extension. In *Proceedings of the 49th Technical Symposium on Computer Science Education*, pages 687–692, 2018.