Describing Elementary Students' Interactions in K-5 Puzzle-based Computer Science Environments using the Collaborative Computing Observation Instrument (C-COI)

Maya Israel
University of Illinois-Urbana Champaign
misrael@illinois.edu

Quentin M. Wherfel
University of Illinois-Urbana Champaign
wherfel2@illinois.edu

Saadeddine Shehab
University of Illinois-Urbana Champaign
shehab2@illinois.edu

Oliver Melvin
University of Illinois-Urbana Champaign
omelvin2@illinois.edu

Todd Lash
University of Illinois-Urbana Champaign
toddlash@illinois.edu

ABSTRACT

Despite efforts to integrate computer science (CS) into K-12 education, there are numerous unanswered questions about how students learn CS, how to provide positive computing experiences, and how students interact with each other during CS instruction. To begin to deconstruct these complexities for a diverse range of students, it is important to not only study the outcomes and products of students' computational experiences, but also the processes they take in creating those products. In recognizing the necessity for targeted, narrow research questions, this paper focused on how elementary students interacted with each other during puzzle-based CS instruction. Future work will focus on comparing these findings to students' collaborative interactions in more open-ended computing situations. Data analysis made use of the Collaborative Computing Observation Instrument (C-COI) [6] to analyze video screen captures of nine students as they engaged in CS activities within Code.org's Code Studio. Findings confirmed three predominant types of collaborative interactions: Collaborative problem solving, excitement and accomplishment related to CS activities, and general socialization.

CCS CONCEPTS

• **Social and professional topics → K-12 education**; • **Applied computing → Collaborative learning**;

KEYWORDS

K-12 Computer science education; collaborative computing; assessing computational behaviors; Collaborative Computing Observation Instrument (C-COI)

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICER'17, August 18-20 2017, Tacoma, WA, USA
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-4968-0/17/08…$15.00
https://doi.org/10.1145/3105726.3106167

1 INTRODUCTION

There is a growing body of literature regarding the importance and benefits of teaching computer science (CS) and computational thinking (CT) in K-12 settings to a broad range of learners. In fact, the newly reauthorized Every Student Succeeds Act, in its definition of a well-rounded education, includes CS alongside other subjects such as writing, mathematics, and science. This legislative mandate occurred at the same time as school districts such as Chicago Public Schools, New York City Schools, and San Francisco Unified School District are rapidly increasing the CS opportunities offered to students from early elementary school through high school.

Although computing at the elementary grades is not a new concept [e.g. 19], there is an increased focus on providing early CS experiences to both provide unique instructional opportunities and increase the diversity of the CS field [1, 16], students with disabilities, students from lower socioeconomic households, and students from culturally and linguistically diverse backgrounds in CS [17]. Leveraging research in science education that points to the importance of early and sustained exposure [2], the focus of this study is on CS instruction in elementary school.

Several options for computing software exist that can be used to teach CS to young learners at the elementary and middle school grades. These typically utilize visually intuitive block-based programming languages and fall into one of two broad categories: (a) Sequential, linear tutorial-focused experiences, and (b) open exploration experiences. Within the sequential experiences, examples include Code.org's Code Studio (https://studio.code.org/). These offer increasingly complex coding challenges, games, and maze completions [11]. Open-ended platforms such as Scratch (https://scratch.mit.edu/) provide a virtual playground where students explore computing concepts and practices in a more open-ended fashion [21].

1.1 Uncertainty in Ill-Defined CS Tasks

When evaluating students' interactions within computing environments, it is helpful to consider them along a continuum of experiences including scenarios where the students are familiar with the task and have the necessary strategies to successfully address the task as well as situations where the students are uncertain about

how to address a computing challenge or problem. Within computing instruction, students often experience uncertainty, wherein they do not know how to solve a problem that they face. Jordan and McDaniel (2014) defined uncertainty as an "individual's subjective experience of doubting, being unsure, or wondering about how the future will unfold, what the present means, or how to interpret the past" (p. 492) and stated that these times of uncertainty can facilitate social interactions among the students as they attempt to unpack or solve the problem [7].

When observing students during computing experiences, it quickly becomes apparent that students often experience uncertainty. It is, therefore, important to consider how students navigate uncertainty, how they interact with each other, and the outcomes that occur as a result of the students' problem solving.

1.2 Collaborative Computing

Several models of collaborative computing exist from student-driven to teacher-facilitated. Student-driven collaborations often occur because programming environments for novices typically encourage young learners to create computational artifacts that they and their peers will value [4]. These environments rely on social processes in which students are encouraged to share their work with peers both during the creation process and once they have products that they have completed. More generally and within CS education, collaboration, in fact, has been studied in the context of helping students increase persistence and engage in productive struggle [3, 18, 24]. Because students often naturally share successes and seek each other during problem solving, these types of collaborations occur without facilitation by teachers. There are also models of teacher-facilitated collaborations including peer tutoring [5] and pair programming [13].

Despite the social aspects of these programming environments, we have yet to fully understand the types of collaboration that exist between learners and the benefits that students gain through these collaborative computing experiences [4]. In fact, Good (2011) stated that the current computing environments are ideal "playgrounds for studying collaboration in the wild" (p. 21). We should, thus, begin to examine how to leverage collaboration to support student learning within K-12 CS activities.

1.3 Theoretical underpinnings of dialogicality

The focus of this study is on naturally-occurring conversations that students engage in while working on computing activities. The emphasis on naturally-occurring discourse has theoretical underpinnings within dialogicality, which posits that conversations between partners result in joint problem solving spaces that contribute to group cognition; thus, learning often occurs through social interactions [9], this perspective is important when considering technology-mediated discourse that occurs as students discuss their computing experiences. Stahl and colleagues (2014) explained that collaborative discourse in computer supported collaborative learning (CSCL) contexts constitutes new forms of discussion and provides innovative ways of exploring that discourse [22].

1.4 Purpose of this Study

To understand the computing interactions that students engage in during CS instruction, it is important to examine both puzzle-based and open inquiry CS approaches. It is also important to examine both naturally-occurring and teacher-facilitated interactions. However, it is imperative to not confound these different approaches. Therefore, this study focuses on naturally-occurring student collaborations within puzzle-based environments with the intent of examining the various interactive behaviors students exhibit while engaged in such activities. Future studies will focus on examining other environments as well. In this way, we can have a focused research question that will lead to increased understanding in future studies. Our research question was: What kinds of collaborative interactions occur during puzzle-based computing environments?

2 METHOD

This qualitative study made use of the Collaborative Computing Observation Instrument (C-COI) [6], a validated observation instrument for evaluating students' individual and collaborative interactions within computer-mediated learning, to code data from nine students in third and fourth grades across multiple observations within Code Studio Play Lab.

2.1 Participants

This study took place in one Midwestern elementary school that included computing and computational thinking as part of its K-5 curriculum. The classrooms of two teachers (3rd and 4th grades) were purposefully selected to participate in this study based on the level of academic and demographic diversity in those classes as well as the integration of computing into their classes. Both teachers were in their 3rd year of implementing computing instruction in their classrooms and during the current study, they scheduled 45-minute computing sessions at least once per week. During these computing times, students typically worked through computing activities in Code.org Code Studio or on unplugged activities that taught computational concepts in hands-on non-computing activities.

The research team created a matrix that included the following information for each student whose parents provided informed consent: Gender, socioeconomic status, disability status, students' level of computing expertise, and students' level of collaboration during classroom instruction. Based on this matrix, nine students were purposefully selected to attain a diverse sample (see Table 1). The research team classified socioeconomic status based on whether the students received free or reduced lunch. Disability status was coded based on whether the students received special education services and had an individualized education program (IEP). Lastly, the teachers provided information about students' level of computing ability as well as whether they generally collaborated with their peers. Because these variables were based on teachers' perceptions, they were not used as variables for analysis. Rather, they provided another means of ensuring diversity within our sample. The rationale for purposeful selection of a small sample of students is consistent with video data analysis methodology. Video data analysis examines interactions of students with their peers, teachers, and the computing software and, therefore, is more conducive to small

Table 1: Student Demographic Information

Pseudonym	Grade	Gender	Free/Reduced Lunch	Special Education
Adam	3	M	No	No
Jacob	3	M	Yes	No
Denny	3	M	Yes	No
Liza	4	F	No	No
Tonya	4	F	No	No
Kevin	4	M	No	No
Diana	4	F	Yes	No
Allison	4	F	No	Yes
Jason	4	M	Yes	No

sample size studies due to the qualitative nature of the analysis. The nine students in this study were observed multiple times over the course of the study to ascertain whether observed interactions remained consistent across observations or differed from one observation to another. For example, students may work independently during one computing session, but then display more collaborative behaviors during other times. It was important, therefore, to observe students at least three times in order to understand trends in the data more fully.

2.2 Video Screen Capture Collection

Students each had access to a laptop computer and could either work independently or collaboratively with peers. Teachers encouraged students to collaborate and seek help from their peers before asking the teacher whenever they encountered a problem. Student data was gathered using video screen capture through Screencastify, an open-source, Google Chrome application (https://www.screencastify.com/). This software recorded students' computer screens as well as their voices as they engaged in computing tasks. It was necessary to have access to both the computer screens and voices as it helped to fully capture the collaborative interactions around computing tasks. At the start of data collection sessions, the research team started the screen recording for the purposefully selected students. At the end of each computing session, the recordings were given a label based on predetermined student ID number and the date of the recording. To gain a sufficient control for data analysis, three or four observations were collected for each student in this study. In total, 29 individual student video sessions were collected across the nine students with an approximate total of 13.4 hours of coded videos.

2.3 Data Analysis

The primary data analysis tool in this study was the Collaborative Computing Observation Instrument (C-COI), a validated instrument developed to analyze video screen capture data of students as they engaged in computing activities [6]. It was developed and validated through a two-year process that involved multiple student observations as well as content and construct validity checks with experts in the fields of computer science education, collaborative learning, instrument development, and assessment of student behaviors.

The C-COI was used to analyze the on-screen behaviors of the students as well as the conversations that they had while they completed their computing tasks. The C-COI measures (1) time on task/persistence, (2) help seeking behaviors, and (3) individual and collaborative problem-solving behaviors. This instrument includes codes specifically related to how students engaged in computing activities as independent and collaborative behaviors by choosing among 16 nodes and the associated sub-nodes that describe those behaviors. These nodes indicate (a) the student's actions, (b) if the student encounters a problem during the computing activity, (c) if and to what extent the student is socializing with their peers or adults, and (d) how peers or adults respond to the student [6]. Additionally, field notes and transcriptions of student conversations were taken to add context to the accompanying codes.

Once the videos were analyzed with the C-COI codes, the data was used to generate directed graphs that visually displayed the data [23]. In these directed graphs, the nodes are states represented as questions (e.g. How did the adult or student begin the interaction?) and the edges (referred to as subnodes in the CCOI) are the actions taken by a student, peer, or instructor. These directed graphs allow for visual understanding of the students' interaction patterns.

2.3.1 Interrater Agreement. As with any study that analyzes student behaviors, it was important to achieve a high rate of internal consistency between researchers who coded the video data. Given the number of subnodes within each node in the C-COI, the research team made use of a rigorous procedure for establishing percent agreement [14]. This procedure was established because there would be too many subnode calculations per video to efficiently calculate Cohen's kappa (a measure of interrater reliability) per subnode. The following interrater agreement procedure was validated with the support of two research methodologists who were not connected to this research study. A matrix was developed in which the columns represented different researchers and the rows represented the nodes and subnodes within the C-COI. Each cell in the matrix was populated with the codes each researcher noted from the video analysis. Each time the researchers did not note the same code, a zero was placed in the matrix and each time the researchers noted the same code, a one was placed in the matrix. This matrix allowed the researchers to establish the percent agreement for each subnode and then calculate overall agreement for each video.

Across the twenty-nine videos analyzed, six (approximately 20%) were coded to establish interrater agreement by four members of the research team. Agreement procedures involved two phases. Phase 1 consisted of agreement on initial path: Collaborative or independent path. This is denoted with Nodes 0-A (collaboration path) or 0-B (independent path) in the C-COI as well as a time stamp of when the path began. To move onto Phase 2 of the interrater agreement procedure, 100% agreement in Phase 1 was needed to be established. In Phase 2, agreement was established within each subnode. The research team reached 100% agreement in Phase 1 across all six videos. For Phase 2, interrater agreement ranged between 80% and 89% across the 6 videos. These percentages of interrater agreement are within the acceptable range for "strong agreement" [12].

2.3.2 Limitations. This analysis was not intended to be the only way to evaluate students' collaborative computing processes. Rather, it provided one lens to further explain how students engaged in computing instruction. Consequently, the C-COI should ideally be used alongside other methodological processes such as the use of qualitative observations with field notes. In this study, therefore, the C-COI was used along with extensive field notes gathered from the transcripts of students' conversations as well as descriptions of what occurred on students' computer screens. Despite efforts to triangulate findings between codes and field notes, we acknowledge that the C-COI analyzes complex constructs that are still not fully understood in the CS education research community. Additionally, there were situations wherein the students' conversations were inaudible. For example, it can be difficult to track student conversations as they move around the classroom. In these cases, the C-COI includes codes for when the rater cannot decipher the conversations. Future work is underway to address this limitation through the use of more powerful mics and voice detection procedures.

3 RESULTS

Across the nine students in this study, 29 observations were collected and analyzed for collaborative interactions. When observing the video screen recordings, the research team began coding interactions each time a student transitioned from working independently to interacting with a peer or teacher. If, for example, a student was working independently, and he or she asked a peer a question, the team would begin coding that interaction. There were instances, however, in which a student appeared to talk with a peer, but this talk served as "self talk." For example, when a student was excited, she might state, "Oh, awesome!" or she might engage in a think-aloud such as, "Ok, I need to go left" while problem solving. These verbalizations, however, were not directed specifically towards another student or teacher. In these cases, the researchers coded those verbalizations as independent work with self talk.

Data analysis confirmed three recurring and distinct types of collaborative interactions: (1) computing-specific collaborative problem solving, (2) conversations that express excitement/curiosity/accomplishment related to the computing activities, and (3) general, off-topic socialization. Interestingly, although the C-COI includes numerous student behaviors including independent computing and various interactions, these three types of interactions proved to be most prevalent. Additionally, although these interactions are described as distinct types of interactions, it is important to note that the students in this study engaged in all three types of interactions and transitioned between these interaction types fluidly as they interacted with each other and with their teachers. Lastly, it is important to note that the students in this study also worked independently. However, our research question related to the kinds of collaborative interactions that students had while computing so the independent behaviors were outside the scope of this study. The C-COI, however, does allow researchers to analyze independent time on task as well as capture field notes of that independent work.

When examining the conversations initiated by students, 38% (n=78) of the interactions were related to solving computational problems or difficulties that the students encountered, 31% (n=63) of the interactions were socializations wherein students were engaged in non-computing discussions, and 16% (n=33) of the interactions involved conversations related to computing, but not for the purpose of problem solving, such as students describing an accomplishment related to their computing activity or excitement over a peer's computational work. There was an additional 15% (n=30) of the instances that either did not result in a full interaction (e.g., the student addressed a peer and the peer did not respond) or the coder was unable to hear the peer's voice. Each collaborative category is described below with illustrative case examples from the data.

3.1 Collaborative Problem Solving

When examining the C-COI data, the majority of interactions (n=78; 38%) that occurred were related to attempting to solve a computational problem. This was true for both student-initiated and teacher-initiated conversations. Most of these problems related to issues such as lack of understanding about the different blocks that the students encountered and ways of debugging code that was not working as intended. The students that exhibited collaborative problem solving varied greatly and included both students who sought help from peers due to a challenge they faced with the computing activity, or they were giving support to their peers.

For example, during the four observations of Diana, a 4th grade student from a low socioeconomic household, she usually entered collaborative conversations to problem solve computing difficulties that she encountered. The C-COI revealed that Diana interacted with a peer or adult 32 times, and most of her interactions revolved around problem solving (n=21; 65%). Of these 32 interactions, 23 were initiated by Diana wherein she sought help related to a computing task (n=14), socialized with peers (n=2), and expressed excitement about her work or the work of others (n=3). The data also included three instances wherein she addressed a peer and the peer did not respond.

The most prevalent C-COI path in Diana's data was:
0-A to node 1: Beginning of collaborative interaction
1-A to node 2: Student verbally addressed a peer
2-B to node 5: Student expressed a need for help, but not explicit to the problem
5-A to node 6: The problem related to computing/programming
6-A to node 7: Peer and student interacted
7-A to node 9: Peer and student discussed the problem/difficulty
9-A/9-B to node 15: Problem solved/problem not solved

Figure 1 provides a representation of Diana's typical C-COI codes. It is not a directed graph; instead, it is a simplified representation of Diana's progression from beginning a collaborative interaction through the end of that interaction.

For Diana, after only few seconds of attempting to work independently to solve a problem, which typically involved moving a single block, she immediately sought help from a peer (code 1-A) or adult (code 1-B). For example, one interaction began when Diana expressed, "Someone wanna help me? Anyone? I need help. Hey, one of you guys, help me." In these instances, a peer either solved the computational problem for Diana or ignored her request.

Additionally, when examining how Diana asked for help, the C-COI analysis indicated that she frequently sought help without

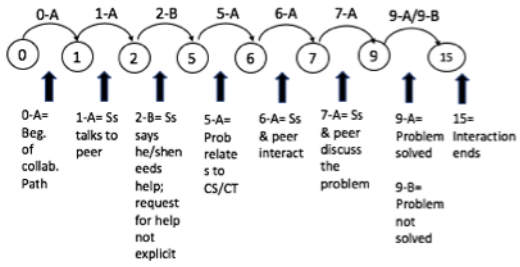

Figure 1: Diana's Most Common Interaction Type

verbalizing what challenges she faced (code 2-B). Rather than stating what she attempted to do or what did not work in her program, she simply stated that she needed help.

3.1.1 Help Seeking. As provided in Diana's example, many collaborative interactions occurred when a student expressed the need for help. Although students often did not explicitly describe the types of problems they faced with their peers initially (code 2-B=Student says he or she needs help, but the request does not provide explicit information about the problem), many interactions involved the peer and student engaging in a rich problem-solving discussion about the computational problem. An example of this type of interaction occurred between Tonya and two peers. Tonya worked on manipulating her code and iterated twice before leaving her computer to bring a friend over to help her.

Tonya brought Maria to her computer and asked for help by explicitly stating what she tried to do:

Tonya: See, I did it. I had forty [degrees]
Jacob: I had this. I will try to help you. Did you try this in here?
Tonya: I think it should be right there and this should be twenty [Tonya adds a turn right by 90 degrees block and
changes the 40 times in the repeat block to 20; then she clicks the reset and run button, the problem is not solved].
Jacob: Get rid of the right [Tonya removes the turn right by 90 degrees block]
Tonya: Okay, I got it [Tonya tries another combination of codes, clicks the reset + run button, the problem is still not solved.]

This problem remained unsolved, so Tonya asked another peer for help and began another conversation.

Allison: So again, what are you trying to do?
Tonya: I am trying to do that square (see Figure 2)
Allison: What could you do ifâĂę
Tonya: Now I am doing 150 [Tonya changes the 120 degrees to 150 degrees in the turn left by 120 degrees block, clicks the Reset + Run button, the problem is not solved; then the student changes the 150 degrees to 180 degrees, clicks the Reset + Run button, the problem is solved].

The codes for Tonya and her two peers is presented in Figure 2. Like Figure 1, this figure provides a simplified representation of Tonya's progression from encountering a problem to solving that problem.

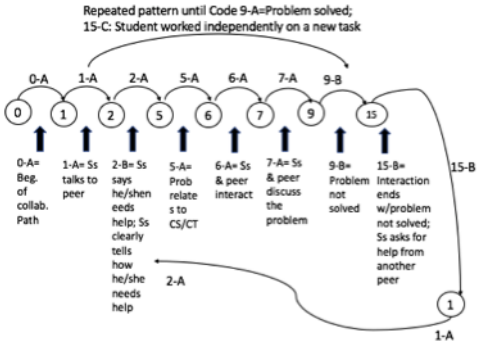

Figure 2: Tonya's Most Common Interaction Type

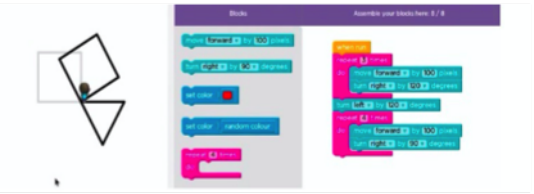

Figure 3: Tonya's Code Studio Problem Solving Example

Students often received help from their teachers rather than peers. In these situations, unsurprisingly, the conversations were much more focused on problem solving steps and isolating where the students were stuck. The teachers often explicitly problem solved with the students to help them understand the difficulty they were facing and why their codes were not working. An example interaction between Jason, a 4th grader, and his teacher showcased this type of interaction:

Jason: I put the blocks there but they would not run.
Teacher: Ok, so let's look and see what you got. So you said, when run, repeat until you get to the flower, do this. Turn right. Move forward. Turn right. Move forward. Turn right. Okay. So the first thing you are asking it to do is turn right and move forward.
Jason: You said turn right move forward. Turn right. Move forward.
Teacher: I was just reading what you had.
Jason: Ohhh!
Teacher: That's all I was doing. I was just decomposing the code you had written.
Jason: I have no idea how to do it.
Teacher: Okay, soâĂę.Look at your zombie and think about what you want him to do first.
Jason: Right. Turn right. Move forward.
Teacher: Ok, so he has to turn right and move forward. Then, what is he going to do?
Jason: Turn left.
Teacher: Turn left and move forward. Okay. Try that. You took it a part and decomposed it.

In this way, the teacher made the problem-solving steps explicit of first reading the code and then reflecting on what Jason wanted

to do. Interestingly, she did not provide Jason with the "correct" answer. Rather, she read his code out loud as a way of helping him think through his steps. She also used the term decompose, a computational thinking vocabulary that she had previously introduced. By using the term decomposition within the context of problem solving, she was both modeling the thinking process and providing an explicit connection with previous instruction on computational thinking practices.

3.2 Excitement, Curiosity, and Accomplishment

Approximately 16% of all conversations (n=33) students engaged in during computing instruction involved students discussing their computing experiences outside the context of problem solving. These conversations typically involved expressing excitement about their computing activities, curiosity about the work of their peers, or accomplishment for completing a computational task. If, for example, a student found something peculiar or surprising in their program, the student would often express curiosity, which would generate a conversation with a peer or adult. If a student persisted through a difficult problem that resulted in success, the student would often express a sense of accomplishment to their peers or teachers.

A typical excitement/accomplishment conversation can be seen with another student, Kevin, a 4th grade student and his peers:

Kevin: Mr. Connor, check this out. Woah, its darkly shaded. It's a black hole. I made a black hole. See? [Mr. Connor did not respond because he was working with other students]
Liza: Try to make it bigger.
Kevin: Oh look at that. Doesn't that look like space in the middle?
Liza: Oh yeah! It does. That is so cool!
Kevin: Doesn't that look like you're going into space?
Liza: Yeah. It's so cool, man.
Denny: It's like the end in Minecraft.
Kevin: It looks like Star Trek. [Sings: de-de-da-deeeeeee]

As compared with the collaborative problem-solving conversations, these conversations did not involve any degree of frustration or a request for help. Rather, these conversations occurred while students either worked independently on their own computing projects or were watching other students work.

Although explaining an entire directed graph with all the C-COI nodes and subnodes is outside the scope of this study, figure 4 illustrates how Adam, a 3rd grader who was observed three times during computing instruction, had both independent work and collaborative interactions. The collaborative interactions involved general talk about his project in which he showed excitement about his work or the work of his peers.

Adam's C-COI analysis revealed 25 observed collaborative instances. When examining these collaborative interactions in more detail, it was noted that there was a fairly even split between the interactions that he initiated (n=14) and those that a peer initiated (n=11). Out of 14 instances wherein Adam addressed a peer, 9 of those instances (82%) were related to expressing curiosity or excitement.

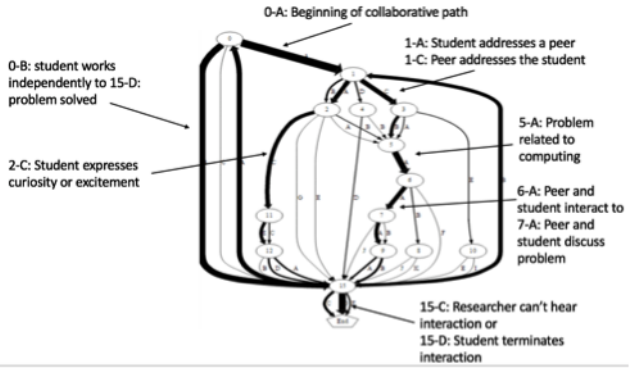

Figure 4: Adam's directed graph of excitement/socialization/accomplishment

Adam both worked independently (0-B: Student works independently to 15-D: Student worked independently on the same problem or topic until solved), and interacted with others (0-A: Beginning of a collaborative path). Most of Adam's interactions were with peers (1-A: Student verbally addresses a peer; 1-C: Student is initiated by a peer). In fact, Adam only had three interactions with an adult (1-D: Student is initiated by an adult) during the computing times. In examining the peer interactions, most of these involved Adam talking with a peer to express curiosity and excitement (2-C: Student expresses curiosity, excitement, or accomplishment). This excitement or sharing of an accomplishment was typically related to his own work rather than to the work of his peers (11-C: Student is excited about something associated with his/her own work; 11-E: Student is showing or expressing accomplishment on his/her own work).

3.3 Socialization

A final type of interaction occurred when students interacted with peers or adults on topics unrelated to computing. This type of interaction accounted for approximately 31% (n=63) of all student-initiated conversations. The typical path within the CCOI that captures these socialization instances was:

1-A to node 2: Student verbally addresses peer
2-D: Student socializes
13-A: Student socializes around an off-task topic
14-A: Peer verbally responds to the student's socialization or
14-B: Adult verbally responds to the student's socialization

As anticipated, these conversations varied greatly based on student interests, classroom activities, and social dynamics between the students. Socialization conversations could be related to what the students would do after school, their families, discussions of popular music or videos, etc. Interestingly, even though the students knew they were being recorded, it did not appear to stop them from talking freely with each other. For example, during the three sessions in which Kevin was observed, he had 28 interactions. He initiated 17 of these interactions and of those 17 interactions, 10 were related to socialization, 6 were conversations wherein he was

expressing excitement/curiosity/ accomplishment, and 1 interaction was related to problem solving.

One of Kevin's socialization conversations occurred after viewing a video hint from Mark Zuckerberg within Code.org Code Studio. The conversation was as follows:

Kevin: Mr. Connor. Mark made Facebook. He made Facebook.
Mr. Connor: That's why he is the richest man in America.
Kevin: Wait, he's rich?
Devin: Yes.
Macy: Every person who buys it, it like costs money to make it. There is probably over a million people that have it.
Kevin: Yeah, even in China. Da zing!

4 DISCUSSION

This is the first of a series of K-12 studies examining students' independent and collaborative experiences during computational activities. The purpose of this study was to describe elementary students' collaborative behaviors that occurred within a computing context. We began to attend to what it looks like from students' perspectives to be "stuck" on a computing problem, and how they maneuver around being stuck. Do they give up? Do they persist by thinking outside-the-box and using strategies to find the solution? Do they interact and ask a friend? Does the extent to which using CS/CT language relate to their computational knowledge?

4.1 Answering our research question: What kinds of collaborative interactions occur during puzzle-based computing environments?

As mentioned above, this is the first of several studies that will use the C-COI as a validated instrument to capture the "process" of students' individual and collaborative behaviors when presented with a computing task. This study revealed that there were three prevalent types of collaborative interactions and all students engaged in all three types of interactions: (a) Collaborative problem solving, (b) conversations about the computing activities that typically expressed curiosity, accomplishment, and excitement, and (c) conversations in which students socialized with each other. Additionally, the students tended to interact more with their peers than with the teachers. Instead, the teachers offered support, monitored learning, and encouraged the students to interact and collaborate with each other.

Although, this is a small sample and generalizations cannot be made to larger samples, we have begun to see trends that are encouraging for the K-5 computing education community. First, given the opportunity to collaborate, students are interacting around computational themes and helping each other solve problems. In the case examples of Diana and Tonya, she would purposely seek out peers and ask for their help. These types of collaborations produced rich conversations as students negotiated ideas, tried different strategies, and worked together to solve the problem. Secondly, in many instances, even when students encountered challenging computing activities, they would persist. For example, Tonya worked through her problem independently and engaged with two of her peers to understand the computing challenge she faced.

We do not, however, have answers yet regarding why some students persisted and others did not. Interestingly, although outside the scope of our research question, in analyzing the video screen captures of students as they struggled with their computing tasks, it appeared that many students struggled because they were working in puzzle levels that were too difficult. Several students, for example, skipped to harder levels because they wanted to be at the same level as their peers. Consequently, as the level of complexity increased in these levels, students became frustrated and required additional support.

4.2 Implications for Future Research

To begin to build instructional practices that support students' persistence, collaborative problem solving, and computational thinking, it is important to methodically study the processes that students undertake as they engage in computing activities. This study showcased how the C-COI could be used to obtain such deep-level data to begin to examine the processes students undertake when involved in computing tasks. Additionally, this study highlighted three types of collaborative conversations that were observed across the nine students in this study. Given the paucity of research into how students interact during computing instruction, this study left many unanswered questions. Future research should extend into the following areas:

(1) Examination of students' computational processes during open-ended computing tasks: This study made use of Code.org Code Studio wherein students worked through increasingly sophisticated computing puzzles. The types of interactions that were observed during this study where, therefore, mediated by the curricular organization of these computing tasks. One would anticipate that students' interactions may be different when they are engaged in more ill-defined and open-ended computing activities. For example, although Jordan and McDaniel (2014) described uncertainty as an individual's experience of doubting and being unsure [7], that experience may be different in more ill-defined and open ended computing experiences as compared to the ones within this study. Future research should, therefore, investigate differences and similarities of students' collaborative interactions within these different computing environments.

(2) Examination of students over time after introduction of pedagogical strategies: This study highlighted the need to more explicitly teach students strategies such as how to debug their projects when stuck, how to ask for help in more adaptive ways, and how to actively participate with peers during collaborative problem solving. For example, when observing Diana's data, it appeared that she demonstrated help seeking behaviors in a manner that may be consistent with learned helplessness theory, which posits that students exhibit passive behaviors if they have experienced failure or believe that they cannot succeed in a task [15, 20]. She did not attempt to individually problem solve before requesting assistance. Rather, she asked for help in a non-explicit manner as a request, rather than articulating her difficulty. Some studies have highlighted that help seeking, on its own, is not a negative trait. Instead, how students

ask for help can either promote learning or impede it [8]. Additionally, learned helplessness may be considered domain specific rather than general [10], so the way students behave in computing environments may not be representative of their behavior in other content areas. Research questions that should be studied include: (a) When students ask for help by fully articulating the issue rather than by saying things such as "I don't get it" or "help", are they more likely to develop stronger connections with the CS content? (2) What is the relationship between learned helplessness and help seeking behaviors within computing instruction as compared to other content areas? and (3) Do instructional practices that encourage collaboration lead students to seek help rather than attempt to individually problem solve? Addressing these questions requires research methodologies beyond the C-COI such as use of student interviews to obtain their perceptions of challenges as well as strategies to overcome those challenges.

This study began to describe the interactive behaviors that students engage in during computing activities. The C-COI provided a new lens for gaining perspective on how students problem solved and the kinds of conversations they had around computing tasks. The methodology presented can be used by CS education researchers to examine multiple research questions about computing instruction including to what extent students are engaging in collaborative problem solving and whether they are persisting through difficult, ill-defined computing tasks. Our hope is that the C-COI and findings from similar studies will help teachers create lessons that are accessible, engaging, and appropriately challenging to a broader range of learners.

ACKNOWLEDGMENTS

This work was partially supported by the National Science Foundation STEM+C Program (Award No. 1639837)

REFERENCES

[1] S. Cooper, S. Grover, M. Guzdial, and B Simon. 2014. A future for computing education research. *Commun. ACM* 57, 11 (2014), 34–36.

[2] N. DeJarnette. 2012. America's children: Providing early exposure to STEM (science, technology, engineering and math) initiatives. *Education* 133, 1 (2012), 77–84.

[3] J. Denner, L. Werner, S. Campe, and E. Ortiz. 2014. Pair programming: Under what conditions is it advantageous for middle school students? *Journal of Research on Technology in Education* 46, 3 (2014), 277–296.

[4] J. Good. 2011. Learners at the wheel: Novice programming environments come of age. *International Journal of People-Oriented Programming (IJPOP)* 1, 1 (2011), 1–24.

[5] M. Israel, J. Pearson, T. Tapia, Q. M. Wherfel, and G. Reese. 2015. Supporting all learners in school-wide computational thinking: A cross case analysis. *Computers & Education* 82 (2015), 263–279.

[6] M. Israel, Q. Wherfel, S. Shehab, E. Ramos, A. Metzger, and G. Reese. 2016. Assessing collaborative computing: Development of the Collaborative-Computing Observation Instrument (C-COI). *Computer Science Education* 26, 2-3 (2016), 208–233.

[7] M. E. Jordan and R. R. McDaniel Jr. 2014. Managing uncertainty during collaborative problem solving in elementary school teams: The role of peer influence in robotics engineering activity. *Journal of the Learning Sciences* 23, 4 (2014), 490–536.

[8] S. A. Karabenick and R. S. Newman. 2009. Seeking help: Generalizable self-regulatory process and social-cultural barometer. *Contemporary motivation research: From global to local perspectives* (2009), 25–48.

[9] Timothy Koschmann. 1999. Toward a dialogic theory of learning: Bakhtin's contribution to understanding learning in settings of collaboration. In *Proceedings of the 1999 conference on Computer support for collaborative learning*. International Society of the Learning Sciences, 308–313.

[10] I. Krejtz and J. B. Nezlek. 2016. It's Greek to me: Domain specific relationships between intellectual helplessness and academic performance. *The Journal of social psychology* 156, 6 (2016), 664–668.

[11] D. Kumar. 2014. Digital playgrounds for early computing education. *ACM Inroads* 5, 1 (2014), 20–21.

[12] James M. LeBreton and Jenell L. Senter. 2008. Answers to 20 questions about interrater reliability and interrater agreement. *Organizational research methods* 11, 4 (2008), 815–852.

[13] C. M. Lewis. 2011. Is pair programming more effective than other forms of collaboration for young students? *Computer Science Education* 21, 2 (2011), 105–134.

[14] M. L. McHugh. 2012. Interrater reliability: the kappa statistic. *Biochemia medica* 22, 3 (2012), 276–282.

[15] K. J. McKean. 1994. Academic helplessness: Applying learned helplessness theory to undergraduates who give up when faced with academic setbacks. *College Student Journal* 28, 4 (1994), 456–462.

[16] Jesús Moreno-León and Gregorio Robles. 2016. Code to learn with Scratch? A systematic literature review. In *Global Engineering Education Conference (EDUCON)*. IEEE.

[17] National Center for Science National Science Foundation and Engineering Statistics. 2017. *Women, Minorities, and Persons with Disabilities in Science and Engineering: 2017.* Technical Report. Special Report NSF 17-310. www.nsf.gov/statistics/wmpd/

[18] R. S. Newman. 1994. Adaptive help-seeking: A strategy of self-regulated learning. In *Self-regulation of learning and performance: Issues and educational applications*, D. H. Schunk and B. T. Zimmerman (Eds.). Erlbaum, Hillsdale, NJ, 283–301.

[19] S. Papert. 1980. *Mindstorms: Children, computers, and powerful ideas.* Basic Books, Inc.

[20] C. Peterson, S. F. Maier, and M. E. P. Seligman. 1993. *Learned helplessness: A theory for the age of personal control.* Oxford University Press, New York.

[21] M. Resnick and D Siegel. 2015. A different approach to coding: How kids are making and remaking themselves from Scratch. (2015). https://brightreads.com/a-different-approach-to-coding-d679b06d83a

[22] G. Stahl, N. Law, U. Cress, and S. Ludvigsen. 2014. Analyzing roles of individuals in small-group collaboration processes. *International Journal of Computer-Supported Collaborative Learning* 9, 4 (2014), 365–370.

[23] M. M. Tatsuoka. 1986. Graph theory and its applications in educational research: A review and integration. *Review of Educational Research* 56, 3 (1986), 291–329.

[24] J. D. Wilson, N. Hosking, and J. T. Nosek. 1993. The benefits of collaboration for student programmers. *ACM SIGCSE Bulletin* 25, 1 (1993), 160–164.