International Journal of Child-Computer Interaction 17 (2018) 61–71

Contents lists available at ScienceDirect

International Journal of Child-Computer Interaction

journal homepage: www.elsevier.com/locate/ijcci

"Alright, what do we need?": A study of young coders' collaborative dialogue

Jennifer Tsan [a,*], Collin F. Lynch [a], Kristy Elizabeth Boyer [b]

[a] North Carolina State University, Raleigh, NC, USA
[b] University of Florida, Gainesville, FL, USA

ARTICLE INFO

Article history:
Received 10 February 2017
Received in revised form 11 February 2018
Accepted 5 March 2018
Available online 9 March 2018

Keywords:
Computer science
Coding
K-12
Dialogue
Collaboration
Pair programming

ABSTRACT

Collaboration is a vital part of the discipline of computer science, yet very little is known about how young children collaborate to learn programming in the classroom. Consequently, we have much to understand about how we can most effectively support this learning experience. We have conducted a study of fifth grade students (ages 9–11) in the United States. Students in this study enrolled in an elective computer science course in which they completed a pair programming project spanning one week of class time (45 min per day). This article reports on a deep qualitative analysis of six collaborative student pairs. We examine the ways in which pair programming practices emerge organically within elementary school collaborations, including the ways in which students' roles arise, equity of contributions to the dialogue, and how students manage their responsibilities during the collaborative process. Our results show that for some student pairs, making suggestions in the dialogue is a natural mechanism for swapping control, whereas for other students, the transition from "driver" to "navigator" requires substantial scaffolding. The findings provide insights into the ways in which we can scaffold the collaborative process to support young students' computer science learning.

1. Introduction

Collaboration during the computer science learning process has been shown to benefit students in many ways. It can lead to higher quality code, increased confidence, and a higher course completion rate in undergraduate students [1–3]. One popular form of collaboration in computer science education is pair programming, in which each student in the pair takes on a role as either driver or navigator. The *driver* has control of the keyboard and mouse, actively constructing and editing code. The *navigator* observes and actively seeks to identify errors as they are made, plans ahead to offer structural suggestions, and asks clarification questions [4]. Programmers switch roles within pair programming sessions, usually after a specific amount of time or after completing a subtask. In productive pair programming sessions, both driver and navigator participate in active discussion and work as a team.

Pair programming has long been studied for its impact on undergraduates and professionals [5]. However, there is much we do not know about pair programming for young students[1] [6]. In order to better understand the needs of these young students, we must first understand how they interact with each other, whether they adhere to beneficial pair programming practices, and whether their collaboration leads to successful program development.

This research sheds light on these questions by examining interactions between elementary school-level programming partners working in pairs within their classroom. Our overarching research goal is to develop a deeper understanding of how young students interact with one another during the pair programming process in order to better support their collaboration in the future. This goal gives rise to two distinct research questions that we investigate through our analyses: first, *How do young students balance their dialogue, turn-taking and control while collaborating when learning to program*; and second, *How do young students coordinate their dialogue during collaboration for computer science?*. To answer these questions, we analyzed the students' dialogue, turn-taking, and input control as proxies to measure the students' relationship balance. We collected a corpus of data including Scratch programs, planning documents, and video of students' collaborative process. We transcribed the videos and annotated individual student actions to track important behaviors such as keyboard and mouse control, making and responding to suggestions, and help seeking. This article examines the results of a quantitative and qualitative analysis which reveals factors related to the balance of students' work, such as the total amount of time each student spoke, the amount of time each student "drove", and the types of dialogue

* Corresponding author.
E-mail addresses: jtsan@ncsu.edu (J. Tsan), cflynch@ncsu.edu (C.F. Lynch), keboyer@ufl.edu (K.E. Boyer).

[1] We refer to children under the age of 14 as young students.

https://doi.org/10.1016/j.ijcci.2018.03.001
2212-8689/

that the students exchanged. The results show that some student pairs take turns at the controls naturally and both partners contribute actively to the dialogue, while other pairs resist taking turns and display an imbalance in dialogue contributions.

2. Background

2.1. Collaboration in computer science learning

The computer science education research community is converging on an understanding that collaboration is central to the discipline, and that young students should begin developing these skills early. In the US, the new K-12 Computer Science Framework[2] [7] emphasizes seven core practices, one of which is focused on collaboration.

Pair programming holds the potential to provide students with rich opportunities to hone the core practice of "collaborate around computing" as well as a few others such as "foster an inclusive computing culture", and "communicate about computing". However, research suggests that successful inclusion, collaboration, and communication may require significant scaffolding.

For example, in a study of equity in pair programming, researchers analyzed how a pair of girls in a high school elective on the topic of digital-making positioned themselves via speech and computer usage [8]. The analysis leveraged positioning theory [9], in which the roles of agents interacting with one another are considered fluid and are changed according to the interaction. The study of girls' collaboration found that one student established herself in a more knowledgeable and authoritative position by speaking more, giving commands, and by maintaining control of the equipment which was a keyboard and mobile phone [8].

Notable work has emerged in recent years examining the equity of elementary students' pair programming relationships while they solved problems in a summer computer science course [6,10]. The authors analyzed the frequency of the students' communication with one another as well as the content of their discussion. The authors reached the conclusion that equity in a pair programming relationship may be contextualized based on the curriculum, and inequity may also be the result of some students' desire to complete projects quickly. Their curriculum was meant to be self-paced and may have contributed to students' focus on completing the projects quickly in order to move on to other work.

In addition to examining the equity of pair programming relationships, researchers are also interested in the kinds of collaborative interactions that occur during puzzle-based computing environments. Researchers used the validated C-COI (Collaborative Computing Observation Instrument) [11] to identify types of interaction that occurred often between elementary students solving computing problems. The authors found that when collaborating, elementary students often talked about the computing problems they are solving, their achievements when solving the problems, and other topics unrelated to the problem [12].

Another important factor in collaboration is students' cultural background. In a study of Latina and white middle school students engaging in pair programming, researchers found that the all-white, all-Latina, and mixed-culture pairs communicated differently. All-white pairs communicated primarily by speaking while the all-Latina pairs often used body language [13].

Researchers were curious about how a pair's friendship influenced their academic outcomes in middle school. They found that when friends are paired together and if one student is more confident than the other, then the more confident student's programming knowledge improves when working with a less confident partner with more prior knowledge [14].

Research on pair programming at the undergraduate level has been a topic of interest for several years. Evidence suggests that when one collaborator starts out more knowledgeable than another, the more knowledgeable peer tends to take control, leaving the other partner confused or disengaged [15]. Another study with pairs of undergraduate students conducted in an introductory computer science course revealed that active participation from students in both the driver and navigator role was essential to successful collaboration [16]. In addition, pairs of students who worked together to resolve their uncertainties collaborated more effectively than pairs of students who moved from one problem to another without reaching a consensus.

Our work adds to the emerging body of knowledge regarding pair programming for younger students. In addition to dialogue, we quantify the amount of time spent driving, and we annotate how students make suggestions, request to switch roles, and ask for help.

2.2. Learning theory and collaboration

The social constructivist perspective on learning suggests that learning is a social and interactive process [17]. That perspective guides much of the current research on collaborative learning. In groundbreaking work on collaboration, Chi and her colleagues [18,19] introduced and expanded on the ICAP (Interactive–Constructive–Active–Passive) framework, which characterizes students' engagement when completing a learning activity. For example, students who merely listen to a lecture are *passive* recipients of information. Students who take verbatim notes or copy a solution are *active*. *Constructive* activities include drawing concept maps and asking questions. *Interactive* activities involve dialogue that builds on a previous contribution, such as defending and arguing a position. The authors hypothesized that these four types of activities lie on a spectrum, with passive being the least effective and interactive being the most effective for learning because students use critical thinking to form new ideas and build on their partners' ideas during interaction. A series of analyses have provided support for that hypothesis [19].

The benefit of collaboration has been highlighted in many empirical studies in various domains. For example, Tudge [20] conducted a study with children ranging from age 5 to age 9 collaboratively working to predict a mathematical balance beam's movement. He concluded that the success of the students' interactions was dependent upon whether they could reach a shared understanding of the problem, and each student's level of knowledge. Kruger [21] studied Girl Scout troops ranging from ages 7 to 10 who were asked to form moral arguments. Some of the participants collaborated with adults while others collaborated with their peers. The pairs were given dilemmas and were instructed to discuss possible solutions until they came to an agreement. Kruger's results showed that the students who worked with their peers formed more sophisticated arguments than those who worked with adults. Kruger concluded that this was due to the difference between active and passive listening. Students who worked with adults were likely to have listened to the adult's explanations without discussing their own ideas as much as they might have with a peer. Finally, in a study of 4th grade students in a science class, students were asked to complete a computer-based scientific reasoning task either individually or in pairs. The pairs developed higher quality hypotheses than the students who worked individually [22].

On the whole, prior work has established that the amount of time students spend interacting and talking, rather than being a

[2] This effort was led by the Association for Computing Machinery, Code.org, Computer Science Teachers Association, Cyber Innovation Center, and National Math and Science Initiative.

Table 1
Identifiers, pseudonyms, and genders of collaborating pairs, along with minutes of video collected and transcribed/annotated for each pair.

Pair ID	Student pseudonyms	Gender composition	Minutes of video
1	Charlie and Quinn	Male, Male	61
2	Ian and Aaron	Male, Male	54
3	Adalyn and Catarina	Female, Female	62
4	Alonzo and Gigi	Male, Female	54
5	John and Mia	Male, Female	30[a]
6	Greg and Harry	Male, Male	64

[a] John and Mia's second video was lost in the video transfer process, so we have only one class period instead of two.

passive use of time, is influential in the success of their collaborative learning [20,21]. It also suggests that taking active part in the construction of solutions is important [18,19]. Based upon this prior work, our analysis is centered around the percentage of time student pairs speak and drive, and on the nature of their dialogue contributions. The literature that we presented in the prior subsection highlighted the importance of talk in collaboration. In this study we use the amount of talk, turn-taking, and dialogue acts to understand the balance of the students' relationships.

3. Method

3.1. Participants

We conducted a study of fifth grade students enrolled in a computer science elective within an urban elementary school in the southeastern United States. The school's student body is 53.1% African American, 32.6% Caucasian, and 14.3% Hispanic, Latino, Native American, Asian, or mixed race. In the US, students whose family earnings fall below a set threshold receive free and reduced-cost lunches in school, and at our partner school, 47.4% of students received free or reduced-cost lunch. The total number of students who took the elective was 55 (39 males, 16 females). Of these students, 26 (16 males, 10 females) voluntarily consented to participate in data collection for this research. No compensation or course credit was provided for consenting to data collection. Of the consenting students, we obtained complete videos of 6 pairs (12 students) working on the programming project described below. Eight of these students were male and four were female. As is often the case with computer science electives, we saw a disproportionately high number of male students enrolled in this course.

Table 1 enumerates the six pairs by gender and provides the pseudonyms by which we will refer to them throughout this analysis.

3.2. Classroom context: Computer science elective

The computer science elective course consisted of thirty class periods, each of which was forty-five minutes long. The class periods we analyze in this article are referred to as collaboration "sessions". The class was offered to students four times each school year, and students enrolled at most once per school year. The course introduced topics including algorithmic thinking and programming, robotics, artificial intelligence, and computers in society, all in a way that had been developed and refined to be age appropriate. The first author of this article co-developed the course in partnership with the elementary school teacher who led it. In the class, students designed code in Scratch[3] and completed two programming projects in pairs. Our analysis was conducted on video data of the students pair programming during their first project.

The first project, and the basis of this study, began about halfway through the class, at which point students had received instruction on how to use Scratch to construct conditionals and loops, broadcast and receive signals, and move sprites (visual entities) in a scene. Students worked on the project for five to seven days, with the teacher adjusting the duration as needed based upon the students' progress. The project was designed to emphasize cause and effect. The students were assigned to pairs and were tasked with selecting a fairy tale and writing a program that simulated two scenes from the fairy tale demonstrating cause and effect by taking in and adapting to user input. The programs written also had to be user-friendly and to run consistently.

Before the students could begin programming, they were required to complete a scaffolded, paper-based design process in pairs to ensure that they planned out their approach to the problem and decomposed the task before they began to code. The scaffolded design activity was handed out on printed sheets and prompted the students to choose their sprites and backgrounds, identify the scenes they would program, identify the instances of cause and effect, draw out decision trees, and write pseudocode in order to plan the logic of their program.

During project implementation, the students worked together through pair programming. The class teacher made the pair assignments based upon his own discretion. The teacher generally asked students to switch roles every ten minutes; however, our research field notes indicate that there were days in which the teacher only reminded the students to switch once and did not enforce the roles. Some pairs did not switch when they were told to do so while others switched roles regularly. Although it is not necessarily important for the students to switch when directed, this is one way to teach good turn-taking habits. In addition, having programmers switch roles after a specified time is a common approach in pair programming.

3.3. Data collection

Over the course of the study we collected handwritten and code artifacts completed by the students, videos of students pair programming, screen capture videos[4] of students programming, field notes, and responses during pre- and post-interviews. We attempted to collect videos of all pairs programming, however, due to technical and logistical difficulties, we were not able to obtain videos of all pairs in all activities. The six pairs of students in this study were the only pairs with videos while they were working on this particular project. Fig. 1 shows an example of one pair's completed fairy tale project, submitted by Pair 1, Charlie and Quinn. They chose to work with the story of Snow White. In this scene, the witch is asking Snow White whether or not she would like to eat the apple. The witch's reaction is based on the user's response.

4. Data annotation

In order to address our research questions, we transcribed and annotated the videos and Scratch programs and brainstorming documents. We manually transcribed eleven videos of six pairs of students working on the project. Each video includes both of the students pair programming for one session. For pairs 1–4 and pair 6 we had videos of two consecutive sessions, and for pair 5 we had a video of only one session due to technical difficulties.

[3] https://scratch.mit.edu/.

[4] Screen capture videos are video records of everything that occurs on the screen as the students see it. We captured the videos using screen recording software called SMART Recorder.

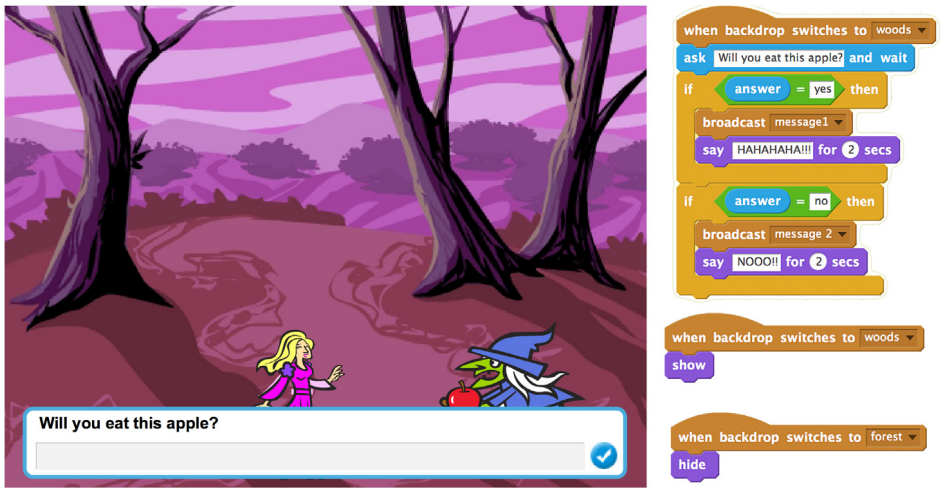

Fig. 1. Screenshots of the output and code of Charlie and Quinn's fairy tale project.

Table 2
The dialogue move annotation scheme created by the authors, ordered from most frequent to least frequent per session.

Dialogue move	Description	Example(s)	Avg. count[a]
Make Suggestion (MS)	Student verbally makes a suggestion or contributes an idea	*"Click that"* *"Here, how bout the fox".*	62
Ask Partner for Help (AP)	Student verbally asks partner for help	*"How do you do it?",* *"How did you get that?"*	6
Accept (AC)	Student verbally accepts or acknowledges partner's idea or suggestion	*"Okay"*	6
Reject (RJ)	Student verbally rejects partner's idea or suggestion	*"No it isn't"* *"No it shouldn't"*	2
Ask Other for Help (AO)	Student verbally asks someone other than their partner for help	*"Mr. Smith!"*	2
Ask to Drive (D)	Student verbally asks to switch roles	*"It's my turn"* *"Let me try"*	2

[a] The count was the average count per student per session.

During the transcription process, we also annotated non-verbal events in the video, which student had control of the keyboard and mouse (was in the role of driver), and we annotated the person to whom students directed their request for help (each other, another student, or the teacher). Fig. 2 shows a screenshot of the annotation software in which we conducted the transcription and annotations of nonverbal activity. The software allows users to add multiple tiers of transcriptions and annotations. In this case, each tier is one stream of data (*e.g.*, the "keyboard" tier is for annotating when each student is driving).[5]

Then, in a second round of annotation, we annotated different categories of *dialogue move*.[6] As shown in Table 2, these tags capture when a student verbally contributes an idea or makes a suggestion[7] ; when a student verbally accepts their partner's idea or suggestion; when a student verbally rejects a partner's idea or suggestion; when a student makes a verbal request to drive (we did not tag for non-verbal requests to drive); when a student asks their partner to help; and when a student asks someone other than their partner for help. For dialogue moves that did not fit any of these categories, we marked them as "Other". Examples of these moves include, "I don't know". and "Look! Mr. Smith, watch it!"; these moves do not offer new information, which is important for a suggestion, nor are they questions about the problem or a request to switch roles.

The first author annotated the entire dataset. To calculate inter-rater reliability, we randomly selected 20% of the sessions (3 sessions) for a second researcher to annotate. The agreement was $\kappa = 0.62$, showing *substantial* reliability [23]. After discussions of the disagreements, the first author refined her annotations of the full dataset before proceeding with analyses. (The kappa was not recomputed.)

5. Results: Talk time, driving time, and consecutive dialogue moves

First, this section examines the balance of talk time and driving time in each of the six pairs. Next, it examines consecutive dialogue moves by the same student. Consecutive dialogue is complementary to talk time because it indicates one student repeatedly making dialogue moves while the other remains silent. Finally, it examines the frequency of each dialogue move type across the pairs. The presentation of these results is followed by a section that delves more deeply into the collaborations one pair at a time.

5.1. Talk and driving time

Based upon the video annotation, we computed the number of seconds that each partner spoke (talk time) and the number of seconds each partner was at the controls (driving time). Fig. 3 displays the percentage of time each partner spoke and drove in

5 ELAN: https://tla.mpi.nl/tools/tla-tools/elan/.

6 The term "dialogue move" refers to a chunk of speech, and uses the word "move" in the same way as making a "move" on a chess board. Dialogue moves are sometimes similarly referred to as dialogue *turns* in the dialogue analysis literature and as "turns at talk" in conversation analysis literature. We adopt the term "move" to avoid any ambiguity with students taking *turns* at the controls while driving.

7 We refer to "contributing an idea" as offering any new information that might help solve the problem, while "making a suggestion" is providing information that is actionable.

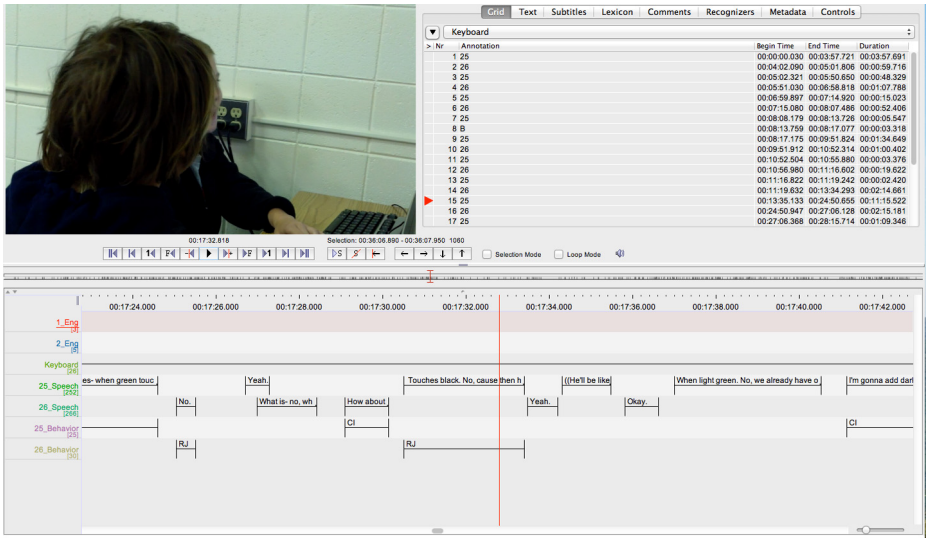

Fig. 2. Screenshot of the tagging software.

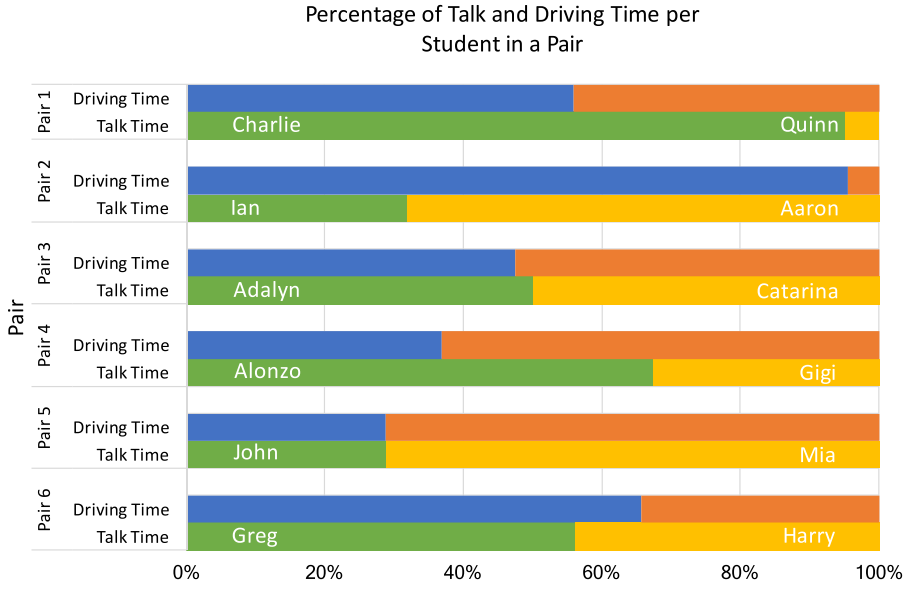

Fig. 3. The percentage of time each partner spoke and drove in each pair.

each pair throughout the sessions, ordered from least balanced to most balanced overall. The results revealed that some pairs were balanced in their percentages of driving and talk time, while others were highly imbalanced. The figure also shows that increased talk time and increased driving time sometimes went hand in hand, but often did not. As we will soon show, some students often chose to talk to themselves as though "thinking aloud" while in the driving role, significantly increasing their talk time. In contrast, other drivers spoke infrequently and responded to the navigator's dialogue through writing or editing code. Similarly, some students were highly engaged while serving as navigator, and others contributed little to the dialogue while navigating. We explore these phenomena in further detail below.

5.2. Consecutive talk

Talk time indicates the duration of time a student spent speaking, but it cannot capture whether that speech was primarily in a few long moves or in many short moves. Student collaboration is driven by dialogue and in order to understand their collaboration process and success, we must understand their dialogue behavior. To better understand students' dialogue behavior, we calculated the average and maximum number of consecutive moves each student made in their sessions (Table 3). We aggregated the data of the pairs that had two sessions. Consecutive dialogue moves occurred when one student made repeated dialogue moves without a partner dialogue move.

In some forms of dialogue, where speech is the only communication medium, one person taking two consecutive dialogue moves is relatively uncommon [24]. However, in collaborative dialogue for problem solving, consecutive moves occur regularly because one or both partners are actively engaged in the task at hand, which contributes to the collaborative space even if no speech is contributed [25]. When we see a very high number of consecutive dialogue moves, however, this can often indicate that either one partner is disengaged, or the talkative partner is offering speech that is not intended for dialogue, but as a sort of think-aloud or narration. We see that phenomenon in our data: Charlie at

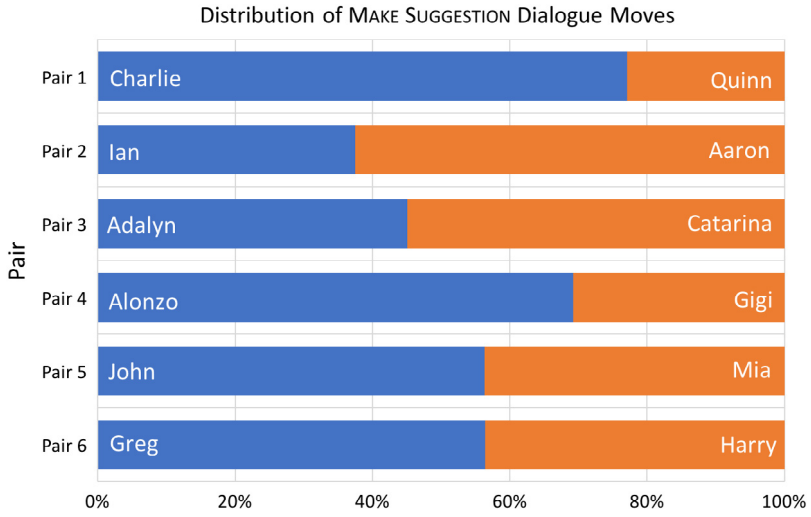

Fig. 4. Distribution of Make Suggestion dialogue moves for each pair.

Table 3
The average and maximum number of consecutive dialogue moves by each partner.

Pair	Student	Average	Max
1	Charlie	3.87	30
1	Quinn	1.16	4
2	Ian	1.64	9
2	Aaron	2.53	19
3	Adalyn	1.68	17
3	Catarina	1.93	13
4	Alonzo	2.24	14
4	Gigi	1.32	8
5	John	1.47	6
5	Mia	1.75	7
6	Greg	1.75	12
6	Harry	1.67	9

one point made 30 consecutive dialogue moves with no input from Quinn. Aaron made 19 while paired with Ian, who made at most 9 consecutive dialogue moves. At the low end, John and Mia each made at most 6 and 7 consecutive dialogue moves. We will examine how these consecutive dialogue move behaviors manifest in the case studies.

5.3. Dialogue move types

As described in Section 4, we annotated each transcribed dialogue move with a label indicating its purpose or intent within the conversation. The most common dialogue move by far was Make Suggestion (MS), which accounted for 77% of our labeled moves. (Dialogue moves that were tagged Other were excluded prior to the analyses presented here. While they contain dialogue worthy of future exploration, they are beyond the scope of this article.) Fig. 4 displays the relative frequency of Make Suggestion for each pair. Generally, comparing this frequency to the talk times indicated in Figs. 5 and 6 shows a correlation between the two, which is intuitive. However, there is not such a strong correlation between the percent of drive time and the percent of suggestions made. In pair programming, both driver and navigator should make suggestions actively, and we did observe this. For example, in Pair 2, Aaron only drove approximately 5% of the time, but he provided slightly more than 60% of the Make Suggestion dialogue moves.

We examine the other dialogue moves of Ask to Drive (D), Reject (RJ), Accept (AC), Ask Partner for Help (AP), and Ask Other for Help (AO) — in Fig. 7. We omit Make Suggestion (MS) from Fig. 7 because its frequency was so high that it obscured the differences between the other less frequently occurring moves.

6. Case studies

This section presents excerpts from three of the pairs in order to gain a deeper understanding of their interactions. We selected the three case studies to represent imbalanced talk time, imbalanced driving time, and balanced talk and driving time. The case studies are based on the transcribed dialogues, the videos, and a researcher's observations in the classroom.

6.1. Imbalanced talk time: Charlie and Quinn (Pair 1)

Charlie and Quinn had fairly balanced driving time, with Charlie driving approximately 56% of the time, but heavily imbalanced talking time, with Charlie taking more than 95% of the talk time. Charlie also had the highest number of consecutive dialogue moves of all 12 students, a surprisingly high 30. We found that Charlie was mostly talking to himself in the video, or at best, talking without expecting any response from his partner. Quinn did not reciprocate this behavior, with a maximum of four consecutive dialogue moves.

In terms of Make Suggestion (MS) dialogue moves, Charlie gave more than three times as many as Quinn gave, with 245 MSs from Charlie and 73 MSs from Quinn. They had a comparable number of Accept moves, with 16 and 17 respectively, and each a small number of Rejects, at 3 and 1. However, this means that Charlie rejected 3 out of Quinn's 73 ideas, while Quinn rejected only 1 out of Charlie's 245.

In our first video recording, we observed Charlie explaining the code to Quinn. It appears that Quinn was either absent the

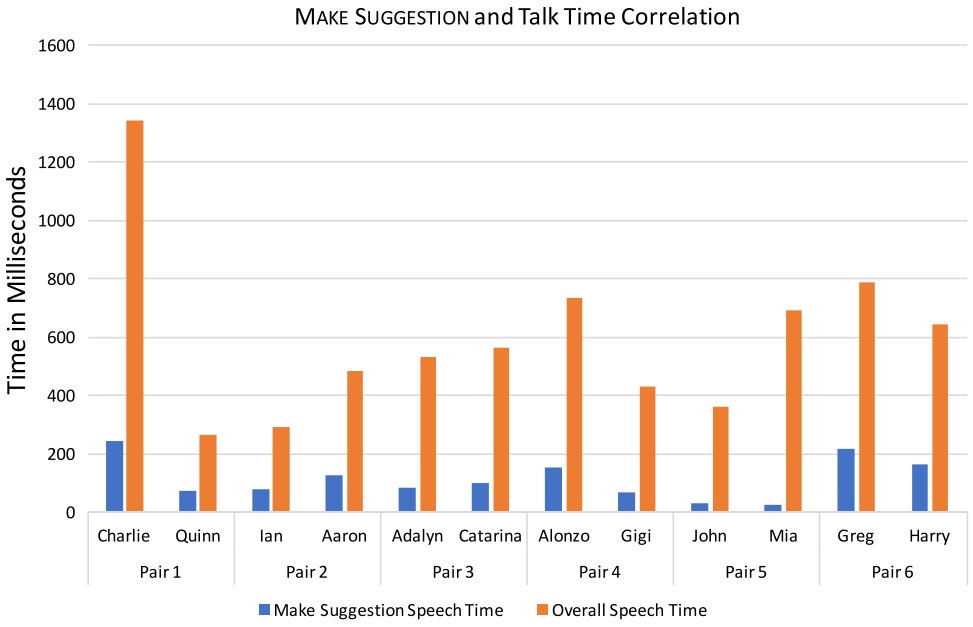

Fig. 5. Correspondence between overall speech time and MAKE SUGGESTION speech time for each pair.

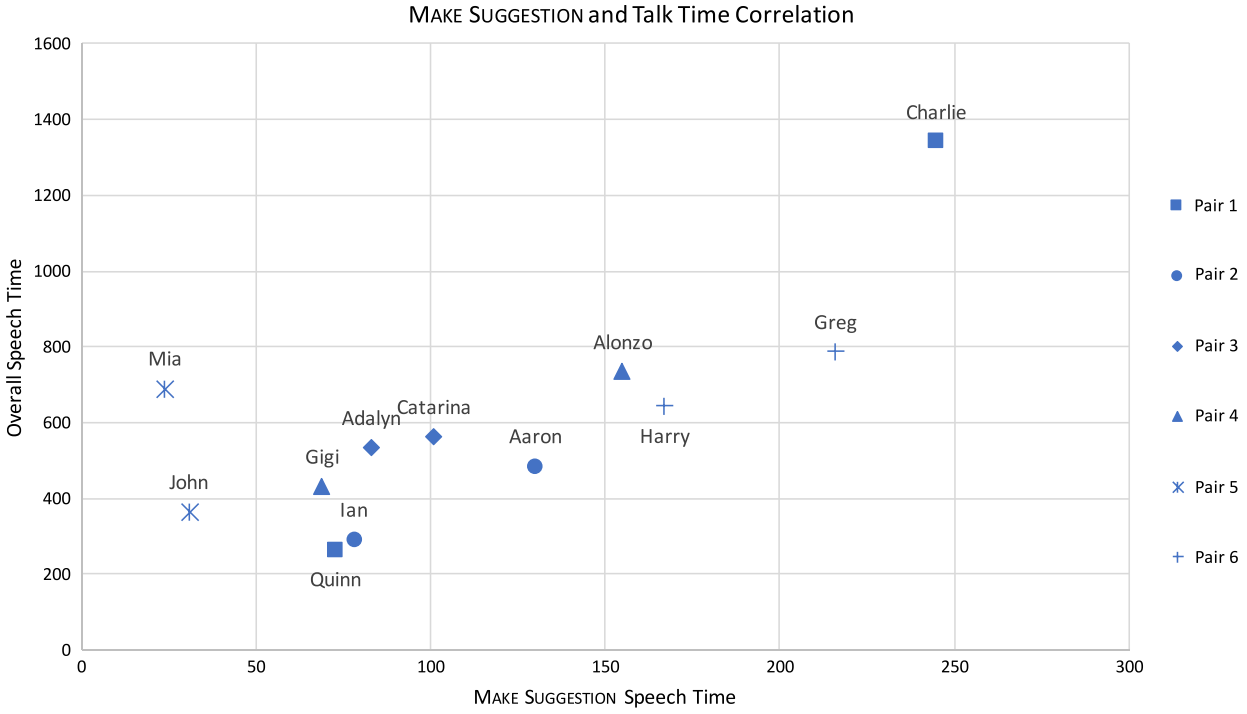

Fig. 6. Correspondence between overall speech time and MAKE SUGGESTION speech time for each pair.

day before, or that Charlie had chosen to work on the code at home. In addition to speaking more, Charlie was the first to drive during both sessions and drove more than Quinn over the span of both sessions. During the first session Charlie and Quinn seemed to work fairly well together despite the imbalance in dialogue: Charlie spoke much more, but Quinn was able to drive about 63% of the time during that first session and they both contributed to the project. However, during the second session, Charlie drove much more (about 74% of the time) and continued to dominate the conversation. In addition, Charlie seemed to be less receptive to Quinn's suggestions when compared to the prior day.[8] Overall, Quinn attempted to offer suggestions, but Charlie was focused on his trajectory and was reluctant to deviate from his plan.

We selected two excerpts to illustrate the typical interactions that occurred between the students. In both of the excerpts below, Charlie was driving. In Excerpt 1 Quinn was watching Charlie drive when he noticed Charlie making a mistake. Quinn suggested to Charlie that what he was doing was incorrect [04:38].

[8] The recordings were collected on contiguous days.

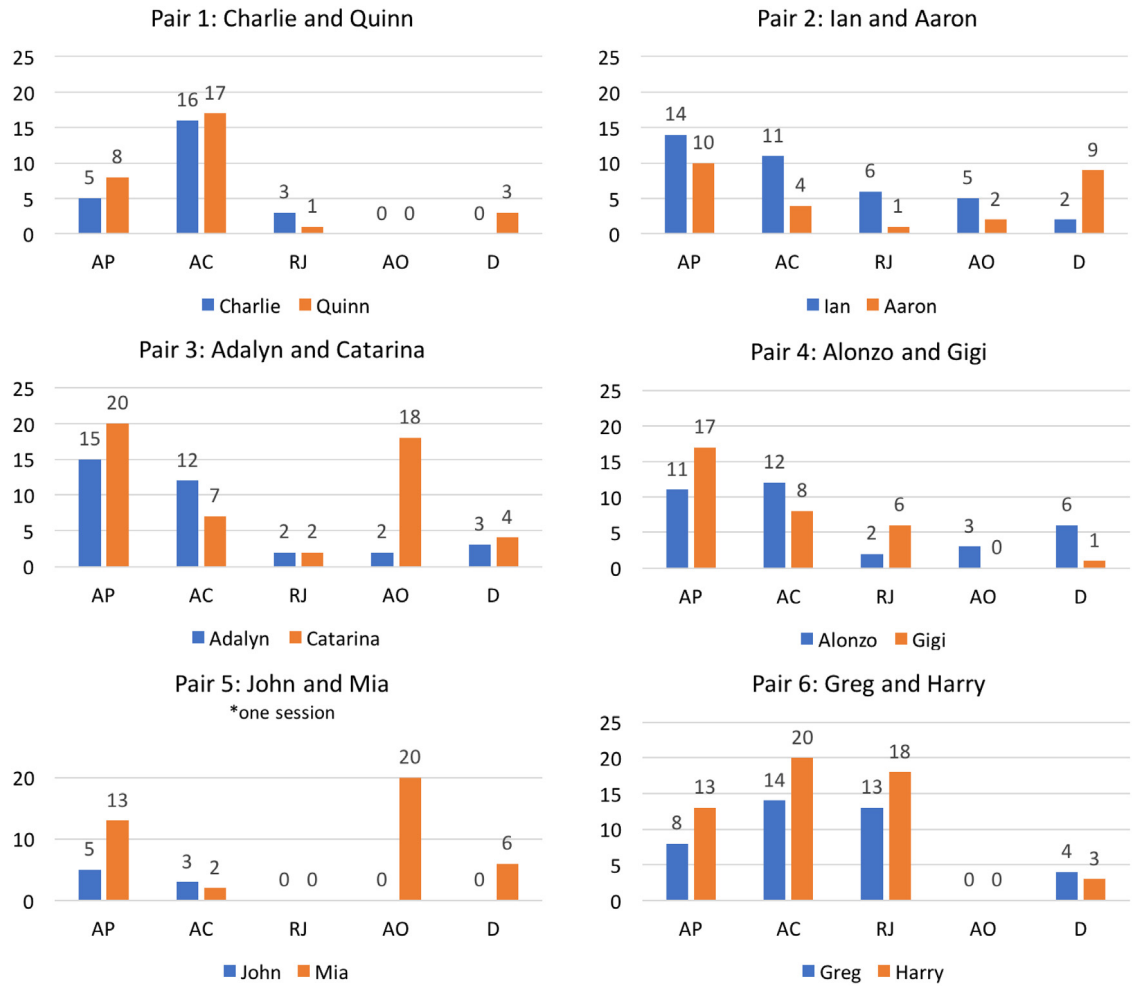

Fig. 7. Dialogue move frequencies from student pair programming sessions. AP — Ask Partner for Help; AC — Accept Suggestion; RJ — Reject Suggestion; AO — Ask Other for Help; D — Request to Drive.

Charlie rejected the suggestion [04:40] and Quinn reiterated his protest [04:41].[9] Then, Charlie vehemently argued, "…yes they do!"[04:42]. A few seconds after the argument subsided, Quinn became quiet and Charlie realized his partner was correct [04:50]. Charlie decided to amend the project, narrating it as he did so [04:52].

Excerpt 1 (Driver: Charlie, Navigator: Quinn):

[04:38] Quinn: They don't need to be down there.	[MS]
[04:40] Charlie: Yeah, they do.	[RJ]
[04:41] Quinn: I don't (()).	[MS]
[04:42] Charlie: (()) yes, they do!	[RJ]
[04:45] Quinn: It's the other one.	[MS]
[04:47] Charlie: They need to be down here for this backdrop.	[MS]
Charlie looks at the code, realizing he was incorrect.	
[04:50] Charlie: No they don't.	[MS]
[04:52] Charlie: That's why I'ma switch it.	[MS]

Excerpt 2 occurred closer to the end of the session. Quinn was fairly quiet throughout the session but he had an idea and wanted to show it to Charlie. Quinn requested to drive [23:28] and Charlie handed him the keyboard and mouse after completing his current task [23:50]. Immediately upon handing over the controls, Charlie asked Quinn what he was doing and then after only a few seconds, told him that the solution wouldn't work [24:08]. Quinn gave back the keyboard and mouse, and Charlie narrated his unwinding of Quinn's suggestion, "Fix. That doesn't work"[24:11]. Both excerpts illustrated Charlie's dominance and unwillingness to discuss Quinn's ideas before rejecting them.

These excerpts also illustrate the contrast between the students' navigation behaviors. Quinn filled the navigation role well, watching his partner build the project while pointing out when there was a problem. He showed that he was thinking ahead in Excerpt 2 when he requested to switch roles in order to implement an idea. When filling the navigator role, Charlie continued to control the direction of the project by directing his partner. He also showed signs of impatience after working as a navigator for a very short amount of time, asking multiple times whether or not the teacher wanted the students to switch again.

Excerpt 2 (Driver: Charlie, Navigator: Quinn):

[23:28] Quinn: Hold on, let me just.	[D]
[23:31] Charlie: There, let me just show you. (())	[O]
[23:50] Charlie: Okay, what're you gonna do?	[O]
Charlie hands over the controls.	
[24:03] Charlie: What're you doing?	[O]
[24:08] Charlie: Ugh, that doesn't work.	[O]
Quinn hands back the controls.	
[24:11] Charlie: Fix. That doesn't work.	[O]

6.2. Imbalanced driving time: Ian and Aaron (Pair 2)

Ian and Aaron had the most highly imbalanced driving time, with Ian driving more than 95% of the time. They were also the least

[9] (()) is used in place of inaudible speech. http://fave.ling.upenn.edu/downloads/Transcription_guidelines_FAAV.pdf.

balanced in talk time, but in the opposite direction: Ian talked 32% of the time to Aaron's 68%. Although Aaron was not given many opportunities to drive, he fulfilled the role of an active navigator well, offering suggestions and asking questions, which led to his higher proportion of talk time. Aaron made up to 19 consecutive dialogue moves, and made more than 60% of the MAKE SUGGESTION dialogue moves in that session.

In the first session we recorded, neither Ian nor Aaron seemed to notice the teacher giving the cue to switch, and they seemed content with Ian driving the entire time. However, in the second session, both the teacher and Aaron were more insistent about switching roles. Although Aaron did drive for a small portion of the second session, he did not drive for long, making just a couple of changes each time. This pair saw the highest count of REQUEST TO DRIVE moves among all pairs, with Aaron asking nine times to drive. It appears that most of the requests to switch were ignored, as Aaron drove less than 5% of the time.

Aaron was clearly the more talkative of the two, with close to double the talk time of Ian. Ian occasionally asked Aaron for help, 14 times over the two sessions, and otherwise worked independently with determination. Ian usually did not verbally acknowledge Aaron's suggestions; however, he did verbally accept 11 of them, and through the videos we can see that Ian generally accepted his partner's ideas through his work in the code.

Excerpt 3 is an example of how Ian showed a preference towards working individually. This excerpt was from the first session. Ian was the driver and Aaron was the navigator. Aaron had fulfilled the role of the navigator well, by giving suggestions and asking questions. Prior to [26:47] Ian had responded to Aaron's suggestions with "I know" three times. Although he acknowledged Aaron's ideas, Ian's responses indicated that he knew what he was doing and he didn't need help. Aaron also stated that he knew what to do [26:57], which Ian indirectly rejected by stating that he too knew what to do [26:58]. Aaron expressed confusion at Ian's response [27:01], then said "Let me help you" [27:04]. After this insistence, Ian at least engaged in one contentful exchange, answering a question [27:15].

Excerpt 3 (Driver: Ian, Navigator: Aaron):	
[0:26:47] Aaron: Move to two hundred.	[MS]
[0:26:53] Aaron: Go to the house button.	[MS]
[0:26:55] Ian: One second. No (()) I know.	[AC]
[0:26:57] Aaron: Wait, I know what to do.	[O]
[0:26:58] Ian: I know what to do.	[O]
[0:27:01] Aaron: What?	[O]
[0:27:04] Aaron: Let me help you.	[O]
[0:27:05] Aaron: Click the home button.	[MS]
[0:27:09] Aaron: It in. Is- it does not say. Okay, why do you have the when clicked there?	[AP]
[0:27:15] Ian: So it will turn back after fox touches.	[MS]
[0:27:18] Aaron: Don't you think (()) that you should put those three together?	[MS]
[0:27:20] Ian: No, (()).	[RJ]

During the second session, Aaron was more assertive and made requests to drive. Excerpt 4 illustrates Ian's unwillingness to relinquish control of the keyboard and mouse. While Ian was driving, Aaron had an idea and asked to use the mouse to try out the idea. Ian stated that he would like to try one thing first. Ian continued to work for another minute despite Aaron's persistent requests to drive.

Excerpt 4 (Driver: Ian, Navigator: Aaron):	
[07:45] Aaron: Wait, can I try something?	[D]
[07:47] Ian: I'm just gonna try one thing.	[O]
[07:52] Ian: Where did he go?	[AP]
[07:53] Aaron: He's under the house.	[MS]
[07:55] Ian: Right here?	[AP]
[07:58] Aaron: I wanna try something. Where'd the pig go?	[D]

6.3. Balanced talk and driving time: Adalyn and Catarina (Pair 3)

Adalyn and Catarina were balanced in terms of drive and talk time, with 49.9% and 50.1% talk time per partner and 47.6% and 52.4% drive time per partner. Although they had a balanced talk and driving distribution, they often were not speaking to each other. Catarina often spoke to the teacher to ask him for help, and Adalyn spoke to herself at times. This distinction highlights the importance of considering several metrics of collaboration success, equal talk time still may not indicate that the students were functioning in a healthy partnership.

We see further evidence of this dynamic in Adalyn's consecutive dialogue moves, there were as many as 17, and Catarina's, as many as 13. This metric indicates a lower degree of give-and-take than we would hope to see in a balanced collaboration. Catarina contributed slightly more through MAKE SUGGESTION dialogue moves, with 101 from Catarina and 83 from Adalyn.

When she needed help, Catarina frequently turned to the teac her. Catarina asked Adalyn a question 20 times, but asked others a question 18 times. This nearly 1:1 ratio is much higher than the 3 to 1 ratio of partner vs. other that we usually observed.

Excerpt 5 took place during the second session that we record ed. After testing out her code, Catarina realized it was not working as expected. She stopped driving and called out to the teacher for help [05:31, 05:45]. Adalyn took control of the mouse and keyboard while Catarina remained focused on getting the teacher's attention [06:00]. While Adalyn drove, Catarina watched the screen and after a while, Catarina seemed to realize something [06:14]. She then said, "No. No, no, no". [06:24] and took the keyboard and mouse back from her partner saying, "Gimmie this". [06:29]. In this case, Catarina had given up after encountering a problem, and began waiting for the teacher. Her partner resumed active work in the meantime, but when Catarina noticed a bug, rather than discuss it from her navigator role, Catarina seized control. The pair did not discuss the problems or possible solutions; instead, they both focused on the screen and their own ideas, trying to work through their confusion independently.[10]

Excerpt 5 (Driver: Catarina, Navigator: Adalyn):	
[05:31] Catarina: See, it's not moving Mr. Smith.	[AO]
[05:45] Catarina: Mr. Smith.	[AO]
Catarina stops driving.	
[05:59] Adalyn: (())	[O]
[05:59] Catarina: Mr. Smith.	[AO]
Adalyn begins driving by grabbing the mouse.	
[06:00] Adalyn: [singing](())[\singing]	[O]
[06:14] Catarina: Oh, that's because I'm on the um.	[MS]
[06:24] Catarina: No. No, no, no.	[O]
[06:29] Catarina: Gimmie this.	[D]
Catarina takes the keyboard and mouse.	

7. Discussion

Our overarching goal in this work was to develop a better understanding of how students work together during the pair programming process in order to better support their collaboration in the future. To work towards this goal we investigated and answered two research questions, *How do young students balance their dialogue, turn-taking and control while collaborating when completing programming activities?*, and *How do young students coordinate their dialogue during collaboration for computer science?* We answered the first research question by using talk time and driving time as proxies for relationship balance. We reviewed the students' talk and driving time to determine whether they were balanced or imbalanced in each pair and we found that most of the pairs in our

10 [singing][\singing] indicates the student was singing.

dataset were not balanced according to these measures. In order to answer the second research question, we annotated the students' utterances from transcriptions of their pair programming patterns and we found that the most common type of utterance was MAKE SUGGESTION. Additionally, we found that some pairs of students accepted more suggestions than others and certain pairs asked their peers and teachers for help more than they asked each other.

These analyses of elementary school students' collaborative pair programming dialogues revealed several patterns that provide useful insights as we work towards designing effective supports for collaborative activities for young students.

First, we observed that a discrepancy in preparation influences the tone of the collaboration. Our case study of Charlie and Quinn revealed that Charlie had already worked ahead, either because Quinn had been absent or because Charlie had the resources to work on the project at home. This inequality only grew as Charlie and Quinn worked together. Alonzo and Gigi (Pair 4) had a similar dynamic: Alonzo appeared to have completed some of the work outside of class, and although Gigi drove more (63% of the recorded time), their collaboration was characterized by Alonzo dictating to Gigi what she should do, and by asking to drive six times while Gigi asked once. Despite this imbalance, Alonzo and Gigi actively asked each other questions, both with more than twice the frequency of Charlie and Quinn. Gigi also retained agency over the collaborative process, rejecting six of Alonzo's suggestions and accepting eight while he rejected only two of hers and accepted twelve. From the interactions between students in our dataset, it appears that when two students in a partnership have contrasting levels of preparation, they cope with the discrepancy in different ways and with varying degrees of success. While pairing students by similar skill or motivation level may mitigate this issue, in a classroom it is not possible to guarantee equal partnerships all of the time. Fostering effective strategies for collaboration between students with different levels of coding experience or skill is a crucial area for future research.

Next, there are specific dialogue strategies which may bear fruit for young computer science students. First, the case of Ian and Aaron in Pair 2 highlights a strategy with particularly strong potential: when faced with a partner who was resistant to involve him, Aaron adopted a strategy of self-advocacy, finally stating bluntly, "Let me help you". In fact, Aaron did not have a solution to offer, but rather, questions that helped to further the goals of the pair. His insistence succeeded (albeit briefly) in engaging Ian in more substantive dialogue. Another example of self-advocacy, coupled with awareness of what he needed during the collaboration process, occurred with Alonzo and Gigi (Pair 4). Gigi was impatient to finish a subtask and said, "What-whatever, (()), just pick one". A few moments later Alonzo replied, "Alright, wait. I'm tryin' to think of what we need bruh". He advocated for time that he needed, when his partner was trying to rush. Alonzo immediately followed that up with another dialogue move, a portion of which is the namesake for this paper: "Alright, what do we need? An animal noise, right?"

In contrast to Ian and Aaron's successful strategies, we can infer that a strategy ***not*** used by Quinn in Pair 1 might have led to a different outcome than the argument that actually ensued. When faced with a partner who was not inclined to listen to him, Quinn stated his ideas as facts, which may have heightened the confrontational tone of the discussion. If Quinn had instead asked Charlie a question that required Charlie to reason through the bug he had introduced into the program, for example, "What would happen if you move it down there?", the outcome of that exchange may have been different. The cases of self-advocacy and strategic question-asking are just two examples of the dialogue strategies that young students need to develop in order to foster thriving, productive collaboration when programming. Strategies like these are essential in many forms of collaboration. In computer science, which in many parts of the world is still severely lacking in gender and racial/ethnic diversity [26], good collaboration strategies have the particularly important potential to increase equity and broaden participation.

While these case studies have focused primarily on the content of the dialogues and not on the overall quality of the code produced, it is worth noting the outcome of their collaborations in terms of producing a project that met the teacher's expectations and used the computer science concepts that were targeted. Several of these student pairs produced programs that largely worked, turning well-known fairy tales into a program that adapts to user input. For example, Ian and Aaron (Pair 2) scored a 100% on their fairy tale project: their program prompted the user correctly and used the necessary selection logic to adapt based on the user's input. Pair 1, despite Charlie's dominance of the conversation and his heated exchange with Quinn, scored 80% on their project. Alonzo and Gigi (Pair 4) and Mia and John (Pair 5) both achieved a 60% project score. Two pairs struggled noticeably to ultimately meet the project specifications: Adalyn and Catarina (Pair 3), who displayed the most balanced drive and talk time, but relied heavily on the teacher to help them rather than working together, scored only 40%, and the pair with the lowest quality project was Greg and Harry (Pair 6) who scored 20%. While they were very engaged and excited while working on their project, they did not meet many of the project requirements: they only had one instance of cause and effect, their interface did not instruct the user on how to use the program, and their program did not run consistently.

7.1. Implications for additional research

Research in coding and computer science learning for children is in its early stages. Our research highlights two open research questions for the community to explore in the future.

How can we foster good dialogue practices for young students collaborating when learning computer science? Years of research in various domains have contributed to an emerging set of best practices for collaborating when learning computer science: collaborators benefit from self-explanation [27], question asking [28], discussing the challenges of the task rather than criticizing each other [29], and building upon each others' ideas to establish common ground [30]. Without scaffolding or instruction, young students do not necessarily follow good dialogue practices for collaboration, which may hinder the success of their collaborative efforts. To promote improved collaboration in young computer science students, researchers should investigate strategies for teaching and facilitating students to engage in collaborative dialogue. It is still an open question how we can support young students in developing these and other beneficial collaborative dialogue practices. ***How can we foster an appreciation for inclusion and engagement in collaborative learning for computer science?*** Pair programming requires sharing: students must take turns with the controls, exchange ideas, and mutually hold responsibility. With young students, the resistance to sharing when faced with such an inherently collaborative activity can lead to conflict, and it is a nontrivial proposition to impart to students an appreciation for each other's contributions. The collaborative process could potentially be improved based on traits of individuals in the pairs (i.e. personality, experience, friendship). In fact, this finding is not unique to children: it has been observed in undergraduates as well [5]. The research community should tackle the tremendous challenge of identifying innovative approaches that develop childrens' appreciation for inclusion and engagement in collaboration. From our results and from previously published work, we suggest that when conducting research on pair programming, researchers should ensure that students are trained in their roles. While this may not necessarily force them to be inclusive and engaged, it will help with sharing, which may lead to students understanding the benefits of collaboration.

8. Conclusion

Many decades of research have established that collaboration is an essential component of learning. For computer science in particular, collaborative practices are central to the practice and learning of the discipline. In a study of fifth grade students pair programming in a computer science elective, we have examined talk time, "driving" time, consecutive dialogue moves, contributions of ideas, and other types of dialogue moves. We have seen that pairs vary tremendously in their balance of talk and control, and that depending upon their approaches to the challenges they face, they often achieve varying degrees of success. Strategies for self-advocacy and question-asking may be particularly important for the success of collaborative dialogues for computer science, as are strategies for succeeding in the face of contrasting levels of prior experience among partners.

These findings shed light on the pair programming process with young students and highlight important questions for the research community to address. The collaborative process for young students is highly complex, and this article has only examined a few of its dimensions. In addition to examining students' speech and driving patterns, it is important for future work to conduct detailed task analysis of concepts and problems used in code, as well as how students decompose tasks. Additionally, rich nonverbal communication occurs in these collaborations, and we must examine it through modalities such as physical gestures, eye gaze, and facial expression.

By gaining insight into the collaborative practices of young students, we can identify strategies that are most beneficial to them, and work towards scaffolding that helps children achieve their full potential in collaborative learning.

Acknowledgments

This work is supported in part by the Wake County Public School System, the National Science Foundation through Grant CNS-1453520, Grant DRL-1721160, and Google through a CS Capacity Research Award.

References

[1] G. Braught, T. Wahls, L.M. Eby, The case for pair programming in the computer science classroom, ACM Trans. Comput. Educ. 11 (1) (2011) 2.
[2] A. Cockburn, L. Williams, The costs and benefits of pair programming, in: G. Succi, M. Marchesi (Eds.), Extreme Programming Examined, Addison–Wesley Publishing Co., Reading, MA, USA, 2000, pp. 223–247.
[3] C. McDowell, L. Werner, H. Bullock, J. Fernald, The effects of pair-programming on performance in an introductory programming course, ACM SIGCSE Bull. 34 (1) (2002) 38–42.
[4] L. Williams, R.L. Upchurch, In support of student pair-programming, ACM SIGCSE Bull. 33 (1) (2001) 327–331.
[5] L. Williams, D.S. McCrickard, L. Layman, K. Hussein, Eleven guidelines for implementing pair programming in the classroom, in: Proceedings of the Agile 2008 Conference, IEEE, 2008, pp. 445–452.
[6] C.M. Lewis, N. Shah, How equity and inequity can emerge in pair programming, in: Proceedings of the 11th International Computing Education Research (ICER) Conference, ACM, 2015, pp. 41–50.
[7] K-12 computer science framework, 2016. URL http://k12cs.org/wp-content/uploads/2016/09/K%E2%80%9312-Computer-Science-Framework.pdf.
[8] E. Deitrick, R.B. Shapiro, B. Gravel, How do we assess equity in programming pairs? in: Proceedings of the 12th International Conference of the Learning Sciences, ICLS, 2016, pp. 370–377.
[9] L.V. Langenhove, R. Harré, Introducing Positioning Theory, in: L.V. Langenhove, R. Harré (Eds.), Blackwell Publishing, Oxford, 1999, pp. 14–31.
[10] N. Shah, C. Lewis, R. Caires, Analyzing equity in collaborative learning situations: A comparative case study in elementary computer science, in: Proceedings for the 11th International Conferences of the Learning Sciences, ICLS, 2014, pp. 495–502.
[11] M. Israel, Q.M. Wherfel, S. Shehab, E.A. Ramos, A. Metzger, G.C. Reese, Assessing collaborative computing: development of the Collaborative-Computing Observation Instrument (C-COI), Comput. Sci. Educ. 26 (2–3) (2016) 208–233.
[12] M. Israel, Q.M. Wherfel, S. Shehab, O. Melvin, T. Lash, Describing elementary students' interactions in K-5 puzzle-based computer science environments using the Collaborative Computing Observation Instrument (C-COI), in: Proceedings of the 13th International Computing Education Research (ICER) Conference, ACM, 2017, pp. 110–117.
[13] O. Ruvalcaba, L. Werner, J. Denner, Observations of pair programming: variations in collaboration across demographic groups, in: Proceedings of the 47th Technical Symposium on Computing Science Education, SIGCSE, ACM, 2016, pp. 90–95.
[14] L. Werner, J. Denner, S. Campe, E. Ortiz, D. DeLay, A.C. Hartl, B. Laursen, Pair programming for middle school students: does friendship influence academic outcomes? in: Proceeding of the 44th Technical Symposium on Computer Science Education, SIGCSE, ACM, 2013, pp. 421–426.
[15] G. Braught, J. MacCormick, T. Wahls, The benefits of pairing by ability, in: Proceedings of the 41st Technical Symposium on Computer Science Education, SIGCSE, ACM, 2010, pp. 249–253.
[16] F.J. Rodríguez, K.M. Price, K.E. Boyer, Exploring the pair programming process: Characteristics of effective collaboration, in: Proceedings of the 48th ACM Technical Symposium on Computer Science Education, SIGCSE, 2017, pp. 507–512.
[17] A.S. Palincsar, Social constructivist perspectives on teaching and learning, Annu. Rev. Psychol. 49 (1) (1998) 345–375.
[18] M.T.H. Chi, Active-constructive-interactive: A conceptual framework for differentiating learning activities, Top. Cogn. Sci. 1 (1) (2009) 73–105.
[19] M.T. Chi, R. Wylie, The ICAP framework: linking cognitive engagement to active learning outcomes, Educ. Psychol. 49 (4) (2014) 219–243.
[20] J.R.H. Tudge, Processes and consequences of peer collaboration: a Vygotskian analysis, Child Dev. 63 (6) (1992) 1364–1379.
[21] A.C. Kruger, The effect of peer and adult-child transactive discussions on moral reasoning, Merrill-Palmer Q. 38 (2) (1992) 191–211.
[22] S.D. Teasley, The role of talk in children's peer collaborations, Dev. Psychol. 31 (2) (1995) 207–220.
[23] J.R. Landis, G.G. Koch, The measurement of observer agreement for categorical data, Biometrics 33 (1) (1977) 159–174.
[24] A. Stolcke, K. Ries, N. Coccaro, E. Shriberg, R. Bates, D. Jurafsky, P. Taylor, R. Martin, C. Van Ess-Dykema, M. Meteer, Dialogue act modeling for automatic tagging and recognition of conversational speech, Comput. Linguist. 26 (3) (2000) 339–373.
[25] B. Barron, When smart groups fail, J. Learn. Sci. 12 (3) (2003) 307–359.
[26] S. Zweben, B. Bizot, The 2015 Taulbee survey, Comput. Res. News 28 (5) (2016) 2–60.
[27] M.T. Chi, N. Leeuw, M.-H. Chiu, C. LaVancher, Eliciting self-explanations improves understanding, Cogn. Sci. 18 (3) (1994) 439–477.
[28] C.P. Rosé, J.D. Moore, K. VanLehn, D. Allbritton, A comparative evaluation of socratic versus didactic tutoring, Proc. Cogn. Sci. Soc. (2001) 869–874.
[29] B. Weiner, The role of affect in rational (attributional) approaches to human motivation, Educ. Res. 9 (7) (1980) 4–11.
[30] C.B. Cazden, S.W. Beck, Classroom discourse, in: A.C. Graesser, M.A. Gernsbacher, S.R. Goldman (Eds.), Handbook of Discourse Processes, Lawrence Erlbaum Associates, Mahwah, New Jersey, 2003, pp. 165–197.