ABSTRACT

TSAN, JENNIFER. Developing Adaptive Support for Collaborative Problem Solving Between Upper-Elementary Computer Science Learners. (Under the direction of Collin Lynch and Kristy Boyer.)

Collaborative learning has long been investigated for its benefits to learners. Prior research indicates that collaboration may help students develop more sophisticated arguments and learn more effectively. While a great deal of research has been completed on how elementary students collaborate, there has been little research on how they collaborate during computer science problem solving. Researchers in this area have started to analyze the equity of pair programming relationships and the content of the students’ collaborative conversations. Despite the demonstrated effectiveness of pair programming for undergraduate students, our observations and previous work have indicated that elementary learners need more support than their older counterparts when pair programming. Without scaffolding or instruction, young students do not necessarily follow good dialogue practices for collaboration, which may hinder the success of their collaborative efforts.

Conflict can occur in any collaborative interaction. Certain kinds of conflicts can play an important productive role during collaboration, such as by prompting students to reconcile differences in their perspectives which can help to improve rapport. Conflicts related to the task at hand are often beneficial, as students may benefit from being challenged by their partner and listening to each other’s perspectives. On the other hand, conflicts during the collaboration process and those related to interpersonal relationships can negatively impact members of a group. I have observed that younger students often encounter conflicts while pair programming.

This dissertation research aims to address these challenges by iteratively developing, evaluating, and refining two collaborative support features in a block-based programming environment. The first is called the talking-reminder and the second is the switching-reminder. These features are meant to support the students’ collaborative dialogue and reduce the number of conflicts they encounter, especially unproductive computer control conflicts.

This dissertation investigates the amount of productive dialogue strategies students use, how they relate to conflict, and how well the new interface features support upper elementary students in their pair programming process. High-conflict pairs antagonized their partner, whereas this behavior was not observed with low-conflict pairs. The results also show more praise and uptake in low-conflict pairs than high-conflict pairs. All pairs exhibited some conflicts about the task, but high-conflict pairs also engaged in conflicts about control of the computer and their partners contributions. Additionally, pairs who used the new features were more balanced in conversation than pairs who did not, and pairs who used time-based role switching often entered fewer conflicts than those that used task-based role switching. I also found that the students had a favorable view of the role-switching feature and they often immediately followed the reminder. On the other hand, they expressed frustration and annoyance with the talking-
reminder and explained that it interrupted them while they worked. The results presented here provide insights into the collaborative process of young learners in CS problem solving, and also hold implications for building learning environments that support students in this context.
Developing Adaptive Support for Collaborative Problem Solving Between Upper-Elementary Computer Science Learners

by
Jennifer Tsan

A dissertation submitted to the Graduate Faculty of North Carolina State University in partial fulfillment of the requirements for the Degree of Doctor of Philosophy

Computer Science

Raleigh, North Carolina
2020

APPROVED BY:

______________________________  ________________________________
Tiffany Barnes                Eric Wiebe

______________________________  ________________________________
Collin Lynch
Co-chair of Advisory Committee

Kristy Boyer
Co-chair of Advisory Committee
BIOGRAPHY

Jennifer (Jen) Tsan was born in Ontario and grew up in Red Lion, Pennsylvania. She learning about programming and web design as a high school student. This lead her to purse a degree in computer science. She earned her computer science degree from West Chester University of Pennsylvania. As an undergraduate, she became involved in outreach opportunities and learned about computing education. Those experiences inspired her to become a researcher in computing education. As a graduate student, Jen became an active member of STARS and continued her outreach efforts through SPARCS. Jen will continue to work towards broadening participation in computing for all students.
I would like to thank the following:

My advisors, Kristy Boyer and Collin Lynch for taking a chance in a student with very little research experience, especially Kristy, who brought me on to the project that led to my current research area. Thank you both for your guidance on the way to the completion of this dissertation.

My committee members Tiffany Barnes and Eric Wiebe for your feedback on my work. Your years of CS education and STEM education research experience have been valuable to me.

My coauthors, Jessica Vandenberg, Danielle Boulden, Zarifa Zakaria, and Fernando Rodriguez for your countless hours spent tagging data, writing, and editing with me. Without all of you, this dissertation truly would not be possible. Thank you all also for your endless emotional and mental support when I most needed it.

Fernando Rodriguez, Mehmet Celepkolu, and Jessica Vandenberg for all of your expertise in CS collaboration and Jessica Vandenberg for answering many of my questions about education research.

The past and current members of the LearnDialogue and ArgLab groups for your support.

Markus Eger, Sasha Azad, Sean Mealin, Thomas Price, Sheldon Abrams, Alicia, and many others for game nights, cooking, brewing, hiking, and other social activities that kept me sane.

My family for your support through all my years of schooling.

All of the teachers whom we worked with, especially the teachers who took their time and classes for my dissertation studies.

Drs. Bin Liu and Richard Burns for inspiring me to attend graduate school and informing me about computer science education as a field of research.

Special thanks to Nicholas Beers for your unwavering support. I would not have gotten through this without you.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>LIST OF TABLES</th>
<th>viii</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>x</td>
</tr>
</tbody>
</table>

## Chapter 1 INTRODUCTION

1.1 Overarching Context and Motivation ........................................ 1
1.2 Contributions ................................................................................. 4

## Chapter 2 BACKGROUND AND RELATED WORK

2.1 Collaborative Learning & Problem Solving .................................... 5
2.1.1 Collaborative Problem Solving in Computer Science ..................... 5
2.1.2 Collaborative Problem Solving with Children ............................ 9
2.1.3 Designing for Children and for Collaboration ............................ 10
2.1.4 Conflicts During Collaboration ............................................. 11
2.2 Block-based Programming Environments ...................................... 13
2.3 Computer Science in Elementary School ..................................... 15
2.4 Systems with Adaptive Support for Collaboration .......................... 18

## Chapter 3 AN INITIAL LOOK AT ELEMENTARY STUDENTS’ COMPUTER SCIENCE COLLABORATIVE DIALOGUE

3.1 Method ......................................................................................... 24
3.1.1 Participants .............................................................................. 24
3.1.2 Classroom Context: Computer Science Elective .......................... 25
3.1.3 Data Collection ......................................................................... 26
3.2 Data Annotation ............................................................................. 26
3.3 Results: Talk Time, Driving Time, and Consecutive Dialogue Moves 29
3.3.1 Talk and Driving Time ............................................................ 29
3.3.2 Consecutive Talk ..................................................................... 30
3.3.3 Dialogue Move Types ............................................................. 31
3.3.4 Answering RQ1a and RQ1b ...................................................... 34
3.4 Case Studies: Talk Time, Driving Time, and Consecutive Dialogue Moves 35
3.4.1 Imbalanced Talk Time: Charlie and Quinn (Pair 1) .................. 35
3.4.2 Imbalanced Driving Time: Ian and Aaron (Pair 2) .................... 37
3.4.3 Balanced Talk and Driving Time: Adalyn and Catarina (Pair 3) ... 39
3.5 Results: Giving and Taking Suggestions ....................................... 40
3.5.1 Answering RQ1ba and RQ1bb .................................................. 42
3.6 Case Studies: Giving and Taking Suggestions ............................... 42
3.6.1 Charlie and Quinn: “I think I can...” ....................................... 42
3.6.2 Ian and Aaron: “I know.” ......................................................... 43
3.6.3 Alonzo and Gigi: “Try it...” ...................................................... 44
3.6.4 John and Mia: “Oh, that’s stupid.” ........................................... 45
3.6.5 Harry and Greg: “No, ‘cause...” ............................................... 46
3.7 Discussion .................................................................................... 47
3.8 Conclusions and Future Work ...................................................... 50
3.8.1 Implications for My Dissertation Research ............................. 50
6.4.6 Focus group question 6: We programmed NetsBlox to tell you to switch after every task. We had other students switch every 12 minutes. Which do you think you prefer and why? ........................................... 88
6.4.7 Focus group question 7: (5th Grade Groups only) What was different between your pair programming experience last year and your pair programming experience this year? ........................................... 88
6.4.8 Focus group question 8: What went well with using the task list? .... 89
6.4.9 Focus group question 9: What did not go well with using the task list? .. 89
6.4.10 Focus group question 10: What would you improve about the task list? 90

6.5 Discussion and Implications .................................................. 91

Chapter 7 INVESTIGATING HOW ADAPTIVE SUPPORT AFFECTS UPPER ELEMENTARY STUDENTS’ CONFLICTS AND DIALOGUE 94
7.1 Collaborative Support Tool Refinement ...................................... 95
7.2 Methods ................................................................................. 96
7.2.1 Research Context and Data Collection .................................. 96
7.2.2 Sampling Data ..................................................................... 97
7.2.3 Annotating Data ................................................................... 98
7.2.4 Datasets Used ..................................................................... 99
7.3 Results ..................................................................................... 99
7.3.1 H2.1 .................................................................................. 100
7.3.2 H2.2 .................................................................................. 102
7.3.3 H2.3 .................................................................................. 103
7.3.4 Qualitative Analysis and Findings ....................................... 106
7.3.5 Focus Group Results ........................................................... 116
7.4 Discussion ............................................................................... 117
7.4.1 Reminder Usage .................................................................. 120
7.5 Study 2 Plan ........................................................................... 121
7.6 Conclusion .............................................................................. 121

Chapter 8 CONCLUSION .............................................................. 122
8.1 Summary ................................................................................. 122
8.2 Contributions .......................................................................... 123
8.3 Future Work ........................................................................... 125

BIBLIOGRAPHY ............................................................................. 127

APPENDICES .................................................................................. 141
Appendix A FOCUS GROUP INTERVIEW QUESTIONS AND PROTOCOL:
  VERSION 1 ............................................................................. 142
  A.1 Protocol .............................................................................. 142
  A.2 Task/Task+ Questions ......................................................... 143
  A.3 Time/Time+ Questions ......................................................... 144
Appendix B FOCUS GROUP INTERVIEW QUESTIONS AND PROTOCOL:
  VERSION 2 ............................................................................. 145
  B.1 Protocol .............................................................................. 145
  B.2 Task/Task+ Questions ......................................................... 146
  B.3 Time/Time+ Questions ......................................................... 147
Appendix C  CS CONCEPTS ASSESSMENT  .................. 148
Appendix D  COLLABORATION ATTITUDES SURVEY  ........... 164
Appendix E  COMPUTER SCIENCE ATTITUDES SURVEY  ........... 165
Table 3.1 Identifiers, pseudonyms, and genders of collaborating pairs, along with the duration of the videos collected and transcribed/annotated (in minutes) for each pair. *John and Mia’s second video was lost in the video transfer process, so we have only one class period instead of two. .......................... 25
Table 3.2 The dialogue move annotation scheme, ordered from most frequent to least frequent per session. ................................................................. 28
Table 3.3 The average and maximum number of consecutive dialogue moves by each partner. ................................................................. 31
Table 3.4 The list and examples of suggestion types. ................................. 41
Table 4.1 Descriptive statistics of the length and stages of the conflicts. ........ 55
Table 4.2 Descriptive statistics broken down by pair. ................................. 56
Table 4.3 An excerpt of Corey and Shawn’s conflict. ................................. 57
Table 4.4 An excerpt of Arif and Hamzah’s conflict. ................................. 58
Table 4.5 An excerpt of Chris and Matt’s conflict. .................................. 60
Table 4.6 An excerpt of Darla and Michael’s conflict. ................................. 61
Table 5.1 The dialogue move annotation scheme. .................................... 67
Table 5.2 Descriptive statistics of the conflicts and dialogue moves broken down by pair. Conflict category and balance category (see section 5.2.2), number of moves, number of conflicts, number of self-explanations, other explanations, directives, and questions, conflict types (task=T, control=C, partner=PC). Not shown are the percentage of antagonization, uptake, and praise. Antagonization was used 1% of the time overall. The maximum any pair used Uptake and Praise was 1% of the time. Finally this shows whether each pair used uptake (U), praise (P), or antagonization (A). The pairs are first ordered by conflict category and then balance category. 69
Table 5.3 Conflict excerpt of Rupert and Anthony (pair 8). ....................... 71
Table 5.4 Conflict excerpt of Sandy and Mitchell from pair 5. ................... 72
Table 5.5 Conflict excerpt of Sandy and Steve from Pair 7. ....................... 72
Table 6.1 Focus group information. *The data for group 5 did not save correctly and could not be analyzed. ................................. 84
Table 7.1 Focus group information. .......................................................... 97
Table 7.2 Focus group information. .......................................................... 98
Table 7.3 Pairs in each condition ............................................................. 98
Table 7.4 Datasets .......................................................... 99
Table 7.5 Descriptive statistics of the conflicts and dialogue tags broken down by pair. ............................................................... 100
Table 7.6 Overall descriptive statistics of the conflicts. ............................ 100
Table 7.7 Comparing conflicts between conditions. The top half is task-based role switching and the bottom half is time-based role switching. 103
Table 7.8 .......................................................... 104
Table 7.9 .......................................................... 104
Table 7.10 .......................................................... 107
| Table 7.11 | .......................................................... | 108 |
| Table 7.12 | .......................................................... | 108 |
| Table 7.13 | .......................................................... | 109 |
| Table 7.14 | .......................................................... | 109 |
| Table 7.15 | .......................................................... | 110 |
| Table 7.16 | .......................................................... | 111 |
| Table 7.17 | .......................................................... | 111 |
| Table 7.18 | .......................................................... | 112 |
| Table 7.19 | .......................................................... | 113 |
| Table 7.20 | .......................................................... | 113 |
| Table 7.21 | .......................................................... | 114 |
| Table 7.22 | .......................................................... | 114 |
| Table 7.23 | .......................................................... | 115 |
| Table 7.24 | .......................................................... | 115 |
| Table 7.25 | .......................................................... | 115 |
| Figure 2.1 | Scratch Interface. | 13 |
| Figure 2.2 | Multiple scripts bound to one event on one sprite [Wei18b]. | 17 |
| Figure 2.3 | The sequence learning trajectory one of three K-8 computational thinking learning trajectories developed and described by Rich et. al. [Ric18]. | 17 |
| Figure 3.1 | Screenshots of the output and code of Charlie and Quinn’s fairy tale project. | 27 |
| Figure 3.2 | Screenshot of the tagging software. | 28 |
| Figure 3.3 | The percentage of time each partner spoke and drove in each pair. | 30 |
| Figure 3.4 | Correspondence between overall speech time and MAKE SUGGESTION speech time for each pair. | 32 |
| Figure 3.5 | Correspondence between overall speech time and MAKE SUGGESTION speech time for each pair. | 32 |
| Figure 3.6 | Distribution of MAKE SUGGESTION dialogue moves for each pair. | 33 |
| Figure 3.7 | Dialogue move frequencies from student pair programming sessions. | 34 |
| Figure 4.1 | OBS recording. | 53 |
| Figure 4.2 | Bar chart of descriptive statistics broken down by pair. | 56 |
| Figure 5.1 | Talking balance of all pairs. | 70 |
| Figure 6.1 | The NetsBlox interface. The new task list feature (described below) is visible in the bottom right corner. The driver is displayed in the top right corner. | 79 |
| Figure 6.2 | The task List feature I added to NetsBlox. | 80 |
| Figure 6.3 | Switching-Reminder | 80 |
| Figure 6.4 | Talking-Reminder | 81 |
| Figure 6.5 | Posters of the features with their explanations. | 82 |
| Figure 7.1 | 1 minute warning | 96 |
| Figure 7.2 | Scaffolded datasets Total Talk Distribution | 105 |
| Figure 7.3 | Non-scaffold dataset Total Talk Distribution | 105 |
Chapter 1

INTRODUCTION

1.1 Overarching Context and Motivation

Collaborative learning has long been investigated for its benefits to learners [Tea95; Kru92; CW14]. Prior research indicates that those who collaborate may develop more sophisticated arguments [Kru92] and learn more effectively [CW14]. Additionally, collaboration is an important skill for learners to develop at an early age, as it takes years to develop good practices which will later be vital in the workplace [Par; AC09].

While a great deal of research has been completed on how elementary students collaborate, there has been little research on how they collaborate during computer science problem solving [LS15; Sha14; Isr17]. Researchers in this area have started to analyze the equity of pair programming relationships and the content of the students’ collaborative conversations. Based on empirical results, researchers have emphasized the importance of teaching students how to collaborate. In fact, “Collaborating around Computing” is now a core practice in the K-12 Computer Science Framework [Ala18]. Initial work in collaborative learning for computer science education has generally focused on using the pair programming paradigm. In pair programming, coders take work together on one computer and take on one of two roles, driver and navigator [WK02]. In productive pair programming sessions, both driver and navigator participate in active discussion and work as a team. In particular, researchers have found that pair programming improves students’ learning experiences [Wil02], increases student retention in computer science courses [O’D15], and promotes good programming practices such as planning a solution before implementing it [PS13].

Despite the demonstrated effectiveness of pair programming, our prior work [Tsa18a; Tsa18b] has indicated that pair programming may not be the ideal collaborative paradigm for students in elementary school. Elementary learners generally need more support than their older counterparts in general. For example, we observed that elementary students may encounter more and different types of conflicts than older learners; our observations and analyses have shown that students in our studies often had physical conflicts with each other, which is discussed in Chapter 4. This includes students reaching for the mouse while their partner is driving,
removing their partner’s hands from the mouse, and wrestling for control of the device.

Additionally, other researchers have shown that students may have negative perceptions of computer science. These include the perception that computer science is competitive [Lew16], asocial [Lew16; YB07], or only for smart students [YB07]. Collaboration may help students overcome some of these negative perceptions. One recent study suggests that collaborative programming helps students make friends [CB18b].

On the whole, prior work has established that the amount of time students spend interacting and talking, rather than sitting passively, is influential in the success of their collaborative learning [Tud92; Kru92; Bar03]. Prior work also suggests that taking active part in the construction of solutions is important [Chi09; CW14]. Using dialogue to build knowledge and understanding is vital to developing sophisticated argumentation skills [Kru92]. Years of research in various domains have contributed to an emerging set of best practices for collaborative learning: collaborators benefit from self-explanation [Chi94], question asking [Gra94], and building upon each others’ ideas to establish common ground [CB03]. Without scaffolding or instruction, young students do not necessarily follow good dialogue practices for collaboration, which may hinder the success of their collaborative efforts.

In our work with elementary students learning computer science, we have sought to understand learners’ general collaboration process when pair programming. We analyzed the students’ dialogue, patterns of switching controls and conflicts to find area where researchers and practitioners may provide support for these learners.

In this work I investigated how elementary students collaborate when solving computer science problems and how I can support them by adding features to their programming environment. I have analyzed the collaborative dialogue of upper elementary students to investigate their collaborative process using pair programming data that was collected as a part of a larger project. I have also implemented and investigated the efficacy of features in a collaborative block-based programming environment that support the students’ collaborative process. The dissertation answers the following research questions:

- **RQ1**: To what extent do upper elementary students use self-explanation, question generation, uptake, antagonization, and praise when collaboratively solving computer science problems?

- **RQ1a**: How do upper elementary students’ collaborative dialogue practices relate to the number and type of conflicts that emerge during pair programming?

- **RQ2**: To what extent do the following features in an adaptive programming environment support upper elementary students’ collaborative problem solving process?
  - A built in timer for switching control
  - A built in tasklist for switching control
  - A reminder to talk to their partner
My theory of action and hypotheses were:

My theory of action views collaborative problem solving within the context of the socio-cultural perspective while upper elementary students learn computer science. The features I implement will scaffold the 4th and 5th grade dyads throughout the collaborative problem solving process as they use a block-based programming language to create their projects. The features will increase the talk balance of the students’ relationships, decrease conflicts, and foster self-efficacy. The pairs’ balance will increase due to the system instructing them on when they should switch roles, and by encouraging them to talk and reinforce the need for the students to express their ideas and discuss them together. Through the guidance of the system, the students will argue less about role-switching, and spend more time explaining their thoughts and actions. This will lead to a decrease in overall conflicts. If the students relationships are more equitable and they spend more time discussing their ideas, their self-efficacy and attitudes in computer science will also increase.

**H1.1:** Upper elementary students use self-explanation, question generation, uptake, antagonization, and praise sparingly (≤ 20% of the overall utterances) when collaboratively solving computer science problems.

**H2.1:** An adaptive programming environment that offers talking reminders will lead to an increase in students’ reported enjoyment of the collaborative process when compared to students who use the programming environment without the scaffold.

**H2.2:** An adaptive programming environment that offers talking and switching reminders will reduce the number of conflicts that arise between students compared to students who use the programming environment without the scaffolds.

**H2.3:** An adaptive programming environment that offers talking reminders will foster increased use of collaborative dialogue strategies more frequently compared to students who use the programming environment without the scaffold.

To answer my research questions, I first annotated previously-collected transcripts of the students’ collaborative dialogue. After answering RQ1, I implemented the two features described above. I piloted the features and conducted focus group interviews to improve the features based on the students’ feedback. Then I conducted a full study with another round of focus group interviews. I then analyzed the dialogue, conflicts, and equity of the students in each condition to answer RQ2. This work falls in the intersection of computer science education, computer-supported collaborative learning, and artificial intelligence in education. The work this dissertation will benefit students around the world. First, it will contribute to the growing knowledge about how young students collaborate in general. Second, it will reveal how young students collaborate while learning computer science. Finally, this work will generate evidence about features that are built to support collaboration in the computer science context. All of
this work will inform future researchers and educators on how to support elementary students who are collaborating while solving computer science problems.

1.2 Contributions

The full list of expected contributions of this dissertation is as follows:

- A large and rich dataset of elementary students’ collaboration process while solving computer science problems.
- Dialogue move annotation scheme.
- Conflict annotation scheme.
- Dialogue and conflict analysis of multiple elementary classrooms of pair programming relationships.
- A set of curricula and activities that are appropriate for teaching computer science concepts to elementary students.
- Empirical data and analyses of the effectiveness of an adaptive system to support elementary students collaborating to solve computer science problems.
- Design recommendations for adaptive support for elementary students pair programming.
Chapter 2

BACKGROUND AND RELATED WORK

My work is informed by, and aims to contribute to, several areas of research: collaborative learning and problem-solving, computer science education for elementary students, the design and use of block-based programming environments, and environments with adaptive support for collaboration.

2.1 Collaborative Learning & Problem Solving

The social constructivist perspective on learning suggests that learning is a social and interactive process [Pal98]. That perspective guides much of the current research on collaborative learning. In groundbreaking work on collaboration, Chi and her colleagues [Chi09; CW14] introduced and expanded on the ICAP (Interactive-Constructive-Active-Passive) framework, which characterizes students’ engagement when completing a learning activity. Under this framework, students who merely listen to a lecture are passive recipients of information. Students who take verbatim notes, copy a solution, or follow a worked example by contrast, 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 [CW14].

2.1.1 Collaborative Problem Solving in Computer Science

2.1.1.1 Undergraduate Pair Programming Findings

A popular form of collaboration in computer science is pair programming, which is a collaborative paradigm in which two programmers work on the same code on one computer [WK02].
In this paradigm, one person writes the code and holds the controls such as the keyboard and mouse. This programmer is called the *driver*. The other person, the *navigator*, watches her partner, looks for mistakes, offers suggestions, and plans ahead. The programmers switch roles based on time or completion of a subtask. The programmers are instructed to communicate with each other throughout the process. Pair programming started out as a practice in industry and has been shown to improve programmers’ efficiency, satisfaction, and confidence [Wil00; Nos98]. Additionally, Nosek found that developers that used the pair programming paradigm developed code that was more readable and functional than individual programmers [Nos98]. An early survey completed to evaluate the effects of pair programming based on a survey of 108 developers suggests that pair programming seemed to increase programmers’ satisfaction, which supports previous results, and increase the communication between developers, the speed of communication about design changes, and the organization of meetings [Suc02].

Not long after pair programming was introduced as an industry practice, universities began implementing this practice in computer science classrooms. Studies on pair programming at the undergraduate level have revealed that students had a higher success rate in their classes (C or better), performed better on midterms, tests, and projects [Wil02; Wer04] than students that worked individually. Pair programmers also produced higher quality code than individual programmers [WU01]. Additionally, course retention and major retention have been improved with pair programming [Wer04]. In agreement with previous findings, Werner found that pair programming helped to improve undergraduate students’ confidence, especially female students and this may be a way to reduce the gender gap [Wer04]. While there are many benefits to pair programming, there is evidence to suggest that when one collaborator starts out more knowledgeable than another, the more knowledgeable peer tends to take control, leaving the other partner confused, disengaged, or unhappy [Bra10].

Early work on pair programming has focused on the outcomes and often used surveys, code, and grades as a way to evaluate the effectiveness of the collaboration paradigm. Recently, researchers have turned their attention to deep quantitative and qualitative analyses to understand the pair programming process and the programmers’ affective and cognitive states. Celepkolu and Boyer [CB18b], and Toma and Varenhold [TV18] used semi-structured interviews and qualitative analyses of reflection papers to understand how undergraduate students viewed pair programming. These analyses revealed that undergraduate students expressed positive sentiments towards collaboration and pair programming generally, as well as help-seeking [TV18]. Additionally, Ying et al. [Yin19] explored student responses about their pair programming experiences using thematic analysis. The authors found that the students generally had positive experiences and women reported feeling more engaged and confident and they also felt less frustrated due to pair programming. Rodríguez et al. investigated dialogue and programming actions between undergraduate pair programmers. They found that high performing pairs had more programming actions (e.g. programming edits, running the program, changing categories) and tended to run their programs more often than low performing pairs. Additionally, high
performing pairs gave positive feedback more often than low performing pairs [Rod17a]. In a follow up analysis, the authors found that high performing pairs resolved their moments of uncertainty before moving on to another problem while low performing pairs worked on multiple problems at once [Rod17c].

### 2.1.1.2 K-12 Pair Programming Findings

High school pair programming studies often focus on physical computing. 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 [Dei16]. The analysis leveraged positioning theory [LH99], 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 in this case was a keyboard and mobile phone [Dei16]. Lui et al., [Lui20] discovered that when high school students worked in pairs to collaborate with physical computing activities, they collaborated while planning. However, they worked individually while constructing their project. Additionally, the authors found that the students’ view of their partners as a socioemotional resource were associated with how high the quality of the pairs’ interactions were.

Middle school research has focused on the gender gap, culture, collaborative programming games, and students’ confidence and knowledge. Buffum et. al. also found promising results on reducing the gender gap in a game-based learning environment that teaches computational thinking by leveraging collaborative gameplay. The authors found that although girl middle school students started with less programming knowledge than their boy counterparts, there was no significant difference in their knowledge at the end of their game-play sessions [Buf15a; Buf16]. Denner et. al. found that middle school students who engaged in pair programming did better in computational thinking post-tests and had a greater increase in Alice knowledge than individual programmers [Den14]. Werner et. al. found that when middle school friends are paired together, 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 [Wer13]. Additionally, Campe et al. discovered that middle school students spent a third of their time not interacting with each other [Cam20]. Factors that significantly influenced interactions and behaviors included attitude toward collaboration and prior programming knowledge [Cam20]. On average, pairs of Latino/a students used more non-verbal behaviors than pairs of Caucasian students [Ruv16].

In an early study of an interdisciplinary activity with younger students using programming and knowledge of science topics, such as the human brain, Kafai and Ching found that when upper-elementary students were planning and designing a science website, they engaged in discussions of scientific concepts more when designing screens [KC01]. They also found that students who had more experience in software design also contributed more to discussions.
about the scientific topics within the context of design.

Lewis collected collaboration data from a sixth grade code camp to understand which collaborative paradigm is better for younger students solving computer science problems: pair programming and “intermittent collaboration” [Lew11]. In the pair programming condition, the students were required to switch roles every five minutes. In the “intermittent collaboration” condition, the students were required to discuss their progress and challenges every five minutes. In addition, they had to ask each other for help before they could ask an instructor for help. The authors found that students in the “intermittent collaboration” condition had more positive responses to continuing Scratch and programming in the future and completed activities in less time.

Other researchers that have taken a closer look at younger students’ collaborative process have found that the equity of a collaborative relationship depends in part on how the individual’s goals align [LS15], and the way the students position each other socially [Sha14]; students may view their more knowledgeable peers as as having more authority and decisions often lean towards the knowledgeable peers and pair programming seemed to amplify this inequity rather than help it [SL19]; and elementary and middle school students often engaged in both problem-specific discussions [BL11; Isr17; Cam20] and discussions of their achievements, as well as more general off-topic conversations [Isr17; Cam20]. Problem-specific questions involved talking through the problem [Isr17], asking for help [BL11], and exchanging ideas [BL11]. In a study with kindergarten children, Fessakis, et al. [Fes13] found that the students’ class interactions fell into one of five categories: competition; interference concerning command proposals and instructions; collaboration; moral support; or dialogue development among the rest of the children.

Additionally, Campe et al., [Cam19] developed a K-12 pair programming toolkit as a guide for teachers. This includes information on teaching students how to collaborate, options for teachers in terms of pairing students, and various pair programming activities they can use.

2.1.1.3 Scaffolded Collaboration Modes

As technology continues to advance, researchers continue to investigate collaboration methods that can are scaffolded by block-based programming environments. Using a modified version of MIT AppInventor, Deng [Den17] investigated how different collaboration models affect the collaboration process. Deng implemented three collaboration models for undergraduate students: project-level collaboration (only one user can edit the code at a time), component-level collaboration (only one user can edit each component or set of block at a time), and real-time collaboration (any user can edit anything at any time). Deng found that the real-time collaboration model yielded the lowest maximum and average length of turns, and the lowest mistake rate. The component-level collaboration model yielded the lowest communication-level and highest mistake rate. The project-level collaboration model yielded the greatest maximum and average length of turns and highest communication rate. Lytle et al. [Lyt20] investigated three different collaboration methods in NetsBlox. Those methods were called: Pair-Separate,
Pair-Together, and Partner-Puzzle. The Pair-Separate paradigm allowed the students to work in the same project with different tasks. The Pair-Together paradigm allowed the students to work in the same project with access to all of the code. The Pair-Puzzle paradigm allowed each student access to a specific set of code blocks. The researchers found that pairs preferred the Partner-Puzzle paradigm. These collaboration modes scaffolded by block-based programming environments are promising paths for future research.

Thus there has been extensive research on collaboration in computer science, especially in the area of pair programming. It also demonstrates the range of contexts, curricula, and participants that have been studied. Computer science collaboration research has evolved from mostly quantitative to greatly qualitative work. Much of the work reviewed above [Rod17a; Ruv16; LS15; Isr17] used tailored annotation schemes or collaborative assessment tools to complete their analyses. I have drawn upon these schemes to inform the development of my annotation scheme for this dissertation and I have used both quantitative and qualitative methods to answer my research questions.

2.1.2 Collaborative Problem Solving with Children

The study of collaboration with children is a large field. In this subsection, I review some foundational work as well as some recent work in collaborative problem solving with young children.

The benefits of collaboration have been highlighted in empirical studies in a number of domains. For example, Tudge [Tud92] studied children ranging from age 5 to age 9 collaboratively working to predict a balance beam’s movement. She concluded that the success of the students’ interactions was dependent upon whether they could reach a shared understanding of the problem, and upon each student’s level of knowledge. Kruger [Kru92] studied Girl Scout troops ranging from ages 7 to 10 who were asked to form moral arguments. The pairs were given dilemmas and were instructed to discuss possible solutions until they came to an agreement. Some of the participants collaborated with adults while others collaborated with their peers. Kruger’s concluded that the students who worked with their peers formed more sophisticated arguments than those who worked with adults. Kruger argued that this was due to the difference between active and passive listening. Students who worked with adults were more 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, the students were asked to complete a computer-based scientific reasoning task either individually or in pairs. The paired students developed higher quality hypotheses than those who worked individually [Tea95]. In a study with second grade students completing sorting activities, Fawcett and Garton found evidence that students who did not perform well on the pretest improved significantly if paired with students who had performed well on the pretest. Additionally, those who spoke more with their partners performed better than those who did not and those who worked individually [FG05].
In a highly influential work, Barron studied successful and unsuccessful collaborative interactions among groups of high-achieving sixth-grade students in mathematics using both qualitative and quantitative methods [Bar03]. One finding from that work is that groups which performed better collaboratively are more apt to accept one another’s proposals and/or engage in discussion around them, while the less successful groups are more prone to reject or simply ignore their partners’ suggestions. In a study of how roles form naturally in a collaborative group, Dornfeld and Puntambekar evaluated the discourse of a group of sixth grade students in a physics class [DP15]. The students’ utterances were classified as conceptual, procedural/negotiation, or social. In their analysis, the authors found that each student had a dominant discourse type, and that students who contributed more to conceptual discussions, containing utterances about the concepts the students were learning, had the highest learning gains. Downton studied the collaborative processes that unfold during group work on a fourth grade music production task [Dow15], investigating how students reflected on their actions in their dialogue. The excerpt presented by Downton illustrated how students reflected on aspects of the music production task and how these spoken reflections would be referenced and built upon by their partners, adding to the collective knowledge of the group. Fink [Fin20] studied turn-taking evolved to collaborative-play with 9th grade students during collaborative math learning. She found that turn-taking collaborative-play supported different types of productive collaborative problem solving.

Psychological theories play a role in how researchers interpret their data and greatly affect their findings. Sociocultural theory, for example, asserts that students’ experiences are based on their family and background and that background shapes their language and perspectives. The students’ perspectives and language can be reinforced or modified by their school [Wel94]. Some students are adept at using precise language to readily engage in argumentation about the task at hand [Bou86]. Therefore, to understand activities requires analyzing the individual, their social partners, and context [Rog95]. Shared understanding “form the ground for communication” and guided participation can increase children’s cognitive development [WT90].

2.1.3 Designing for Children and for Collaboration

Collaborative environments have been well studied in communities such as IDC. This work has included several different collaborative modes: full-body collaboration in which users interact with the digital environment using physical movements [DL17; MB15]; tabletops that support multiple users working together [Ant14; Fie15; Ric09]; and desktop interactions such as programming in block-based programming environments [GJ13; KV15]. My work focuses on desktop interactions because desktop or laptop computers are more often available in schools than other types of devices.

Novel research has emerged in the design of desktop interfaces to support collaboration. Giannakos et al. observed the collaborative behaviors of twelve year old children during a programming workshop in which the children were asked to design and implement a game for the
Arduino system using Scratch [GJ13]. Ginnakos et al. observed that there was a negative correlation between the size of each group and their effectiveness. They argued that smaller groups have greater cohesion and a lower risk of losing control thus they are more effective. Kafai et al. proposed a novel desktop-based interaction, where high school students were assigned to groups and tasked with crafting a low-tech board game and integrating it with high-tech components [KV15]. By expanding the traditional coding activities beyond the screen, they argued that it is possible to create more shared spaces and to further increase the quality of collaboration. Similar to my work, these researchers’ goals were to improve collaboration between children. However, these works did not include implementing features to support collaboration. Great strides have been made in research on collaboration and collaborative problem solving. A number of studies have been performed on activities in subjects ranging from math to science to ethics. In my work I draw from the methods used in the empirical studies above and relevant psychology theories, including sociocultural theory to inform my work.

2.1.4 Conflicts During Collaboration

Conflict is one factor that affects group members and their relationships. In this work, I define conflict broadly as one child attempting to influence the actions of another who resists or directly opposes that influence [Har88; Sha86]. According to Rubin, Pruitt, and Kim [1994], peer conflict typically cycles through four stages: initiation, escalation, de-escalation, and conclusion. I use similar stages in my work and analyzed whether the conflicts reached a resolution. The conclusion of the conflict can be characterized by a resolution when a substantive agreement is made among group members [Rub94], or a state of continued disagreement upon which “it may remain frozen at the status quo” [p. 99]jeong2008understanding.

2.1.4.1 Conflict During Collaborative Learning and Working

Peer conflicts during collaborative activities can, in fact, be beneficial to group dynamics and outcomes. Rubin et al. [1994] argued that conflicts can help group members reconcile differing perspectives and ideas and help to foster group solidarity. Church et al. [2012] asserted that verbal conflict can offer the opportunity to produce creative argument structures. This was supported in later work as Lee, Huh, and Reigeluth [2015] distinguished between three types of intra-group conflict, namely task-related, process-related, and relationship-related. They found that task-related conflicts can have a positive influence on peer collaboration, while process and relationship conflicts typically have negative impacts. In particular, task-related conflicts facilitate reflection, idea building, and coordination. These findings support those of earlier studies by Greer, Jeph, and Mannix [2008] and Martínez-Moreno, et al. [2012]. However, Huang [2010] found teams focused on performance-related tasks may encounter relationship conflicts triggered by task conflicts more than teams that place less emphasis on performance.

Some students worry about voicing their concerns for fear of backlash from their teammates [PN19]. These students may feel that it is safer to work on tasks alone in order to avoid fur-
ther conflicts. Näykki et al. [2014] found that students may use avoidance when they encounter uncomfortable collaborative learning situation. With scaffolding, such as reflections and discussions, students may be able to discuss their differences and work towards a norm that is mutually beneficial [BS15]. When students are able to negotiate their ideas and perspectives, they may also feel safer in voicing their opinions with their peers [RG17].

2.1.4.2 Children and Conflicts

Research on conflicts with children has varied in age and context. Early work on conflict and children focused on students’ emotions and aggression. Shantz [1986] found that 1st and 2nd grade students who entered conflicts frequently also showed a higher percentage of physically aggressive behavior. Arsenio and Lover’s [1997] research revealed that the students often expressed higher intensity of emotions during conflicts, and they were more likely to express anger during conflicts than they were outside of conflicts.

With respect to collaboration with technology, researchers have found evidence that control over the device they are using can be a point of contention with children [And13; Top16; Bra19]. Researchers have found that the amount of disputative talk was fewer for the students using tangibles than students sharing a single-mouse computer [And13]. Additionally, a lack of listening and changes made to students’ code is a point of contention that students encounter while collaborating [Bra19]. Barfurth [1995] found that 4th and 5th grade students in a design and construction environment were able to resolve those disagreements through discussion. Prata et al. [2009] investigated the interpersonal and cognitive conflicts between sixth grade students in a mathematics CSCL environment. They found that non-cooperative behaviors increased the non-cooperative student’s learning but not their partner’s. This was also true of students who insulted their partners. Wise et al. [2015] discovered that pairs of fifth grade students who were assigned roles while using tabletop software had fewer conflicts that were resolved unilaterally and fewer jointly resolved conflicts than pairs who did not have roles.

Given how common conflicts are and the fact that they can be beneficial or detrimental to learning, researchers have investigated the best ways to support children when conflicts arise. Vestal and Jones [2004] found that children who had teachers that were trained in conflict resolution were able to think of more relevant solutions to conflicts that do not involve using force. Cheong et al. [2015] developed a serious game for children to learn how to resolve conflicts. Shen, Slovak, and Jung [2018] found that preschool and kindergarten students who had the robot mediator were more likely to constructively resolve their conflicts than students who did not have a mediator. Finally, Frizzell [2015] found that even when fifth grade students had the skills to resolve a conflict, fear of repercussions from peers often prevented them from doing so.
2.2 Block-based Programming Environments

Block-based programming environments allow users to program by composing scripts of executable blocks using a point-and-click interface. These types of languages are often used to teach novices computer science and programming since users do not have to worry about syntax. This is especially beneficial to younger students who may have difficulty understanding syntax and typing. Popular block-based programming languages include Scratch, Snap!, Alice, and MIT App Inventor. Other languages and resources that have emerged are Blockly, BlockPy, and NetsBlox. Figure 2.1 displays the Scratch interface.

Scratch\(^1\) is one of the first block-based programming environments and is often used to teach younger students how to program. In addition, Scratch has an online community for users to share their creation and features that are helpful to teachers. These features include a way to create an online space for their classroom projects. Snap\(^2\) is similar to Scratch, however, it also includes additional language features such as first class procedures, lists, objects, and continuations from Scheme. NetsBlox is an environment originally built to allow learners to create distributed systems [Bro16; BL17]. NetsBlox was built on the Snap! codebase. An additional feature that was added to NetsBlox was the collaborative feature which allows multiple users to edit the same program [Bro18].

Alice\(^3\) is a block-based programming language based on Java. It is commonly used to teach younger students how to code animations, games, and interactive narratives. AliCe-ViLlagE

---

\(^1\)https://scratch.mit.edu/
\(^2\)http://snap.berkeley.edu/
\(^3\)http://www.alice.org/
[AJP14] is an environment that was built on the Alice code base and modified to allow for online collaboration. Features were added to support collaboration, which includes text and video chat, and real-time synchronization of code changes. Additionally, the researchers added role enforcement for pair programming and only allows the driver to make changes to the code.

An initial study with undergraduate students show promising results of using this system to collaborate while programming [AJP16].

Google’s Blockly\(^4\) is a code library that supports users in implementing their own block-based languages. A number of block environments built on Blockly have been released. Two of the most popular implementations are App Inventor and BlockPy. MIT AppInventor\(^5\) is a language that allows users to build apps for Android devices. The language abstracts away a number of the communication tasks associated with app development. In their master’s thesis, Deng [Den17] implemented a version of AppInventor to allow people to work on the same project from different accounts. BlockPy\(^6\) is a hybrid block and text-based environment that allows users to develop applications either in block form or with Python code, and it supports users in translating between the two languages.

Hill et. al. evaluated three existing block-based programming environments and tools for upper-elementary students: Scratch; Scratch Jr., and Blockly. They found that Scratch had many distractions, such as the ability to add sprites, and some blocks were too complicated; Scratch Jr. only used images and was not mature enough for their age group; and Blockly was not ready out of the box. Based on their evaluation, they developed an environment that was more appropriate for their target age group. This programming environment, LaPlaya, allows for prepopulated scripts and sprites, users can hide and lock scripts and sprites, blocks were simplified to remove math concepts that were too advanced, and distracting interface elements, such as the ability to add sprites, were removed [Hil15].

Block-based programming environments are great tools for elementary students to use to learn programming. Although most environments were created without specific collaboration scaffolds and features, developers have begun to implement such features in order to aid collaboration. It is also promising that there are researchers and developers working to ensure these environments can properly support younger students. In my work I used NetsBlox since it uses the Snap interface, which is similar to the programming language commonly used with younger students (Scratch), it is more general than MIT App Inventor and Alice, and it allows for synchronous collaborative editing. To answer RQ2, I modified the environment and NetsBlox also has the advantage of being open source.

\(^4\)https://developers.google.com/blockly/
\(^5\)http://appinventor.mit.edu/explore/
\(^6\)http://think.cs.vt.edu/blockpy/
2.3 Computer Science in Elementary School

There is a high demand for skilled computer scientists and not enough students graduating with computer science degrees to fill the necessary positions. In response to this, most US states have changed policies to support computer science education. Researchers have also been inspired to bring computer science and computational thinking knowledge to the elementary school level. Efforts began with summer camps and quickly moved into classroom settings. With organizations such as CSforAll and Code.org as well as the CS K12 Framework, the CS ed community is rapidly expanding its knowledge and resources for elementary learners. Early work with elementary students in computer science showed that games produced by young children demonstrated sequencing, event handling, and conditionals. Additionally, most of the games were functional, original, and had clear instructions.

Prior researchers have investigated how children program in general and how the community can design and improve programming environments for children. Dwyer et al. ran focus groups with fourth grade students to assess the affordances of a block-based programming environment that may benefit or hinder them. When asking the students to predict what a script will do, some students were able to use the blocks to make correct predictions. However, some students also made incorrect predictions based on the sprites’ locations and pose. For example, one student predicted that “the unicorn would take a step because its feet were in the air” [Dwy15].

In a study of children aged 9 to 12 programming in LaPlaya, a block-based programming environment based on Scratch, Hansen et al. observed many different programming behaviors among learners in different grade-levels [Han16]. Seraj et al. studied children’s interactions with an interface that allowed them to program a smart home [Ser19]. Weintrop and Wilensky investigated how block-based, text-based, and hybrid block/text environments affected novice high school students’ understanding of programming concepts [WW18]. Cabrera, Maloney, and Weintrop found differences in the way students interacted with interfaces when they were able to see immediate results to physical computing devices after changing their code [Cab19]. However, the research covered here did not focus on collaborative coding.

To better understand how upper-elementary students implemented ideas in block-based programming languages, Seiter developed a rating scheme called the SOLO (Structure of the Observed Learning Outcome) taxonomy to rate 4th grade student programs. She discovered that students from schools with low math and reading comprehension struggled with basic programming tasks. Other frameworks were developed to analyse fifth grade students’ computational thinking skills in the context of robotics and reasoning of everyday events [Che17].

Additionally, researchers automatically analyzed Scratch projects using Dr. Scratch and other tools. Franklin et. al. [Fra13] analyzed digital artifacts from a pro-

---

7https://advocacy.code.org/
8https://www.csforall.org/
9https://code.org/
gramming environment built for this age group. They found that concepts such as placing instructions in a sequence was visible in the work, which is consistent with Wilson, et. al [Wil12]. They also discovered that more difficult concepts, such as initialization, was difficult for 4th and 5th grade students to understand, but 6th grade students were able to grasp it. Initialization involves resetting the sprites and variables at the beginning of each run so the program would run consistently. Additionally, 6th grade students were more skilled at programming sprites to move on an x-y axis. The older students also used more block categories than their younger counterparts [Fra17]. In a similar context, Weintrop et. al. [Wei18b] analyzed student-authored, open-ended projects and found that students used wait blocks to slow down scripts for specific sprites and coordinate timing between sprites. Students also used many event blocks to initiate behaviors. In some cases students bound multiple scripts to the same event on one sprite without the intention of having them run in parallel. An example of this is shown in Figure 2.2. Bound to the “When left arrow key pressed” block are four separate actions and it seems that the student intended for them to run sequentially. This demonstrates that students are able to use blocks, but may not use them as intended by the developers of the environment.

As important as it is to understand how students implement code, it is also important to understand how they comprehend code. In terms of assessment, Salac et al. [Sal20b] discovered that personalizing student assessments using their own code led to the students answering questions differently than if the assessments had generic code examples. Salac et al. [SF20] explored how students understood the code they wrote and discovered that the students’ use of a code construct does not necessarily mean they understand it enough to answer the question correctly on an assessment. Franklin et al., [Fra20b] investigated how students understood Scratch code in an unplugged activity called Scratch Charades where they act out code and other students guess what code they are acting out. The researchers discovered misunderstandings, such as how loops work. For example, if some students received the code example repeat 2 times (move 10 steps, wait), they interpreted the code as move 10 steps 2 times, wait 2 times. Salac et al. [Sal20a] investigated how math and reading comprehension related to students’ CS learning and found results similar to [Che17].

Early work on computational thinking learning trajectories for elementary students was started by Seiter and Foreman [SF13]. Rich et. al. [Ric17] presented groundbreaking work on computational thinking learning trajectories for K-8 grade students. The learning trajectories were developed through the process of extracting, categorizing, clustering, and assembling 671 learning goals from 108 articles. The learning trajectories discussed and displayed in the article were sequence, repetition, and conditionals. One example learning trajectory from the publication is displayed in Figure 2.3. Rich et. al. also developed a learning trajectories for decomposition [Ric18] and debugging [Ric19]. These learning trajectories can be used inform K-8 computational thinking curriculum developers.

In addition to learning trajectories, researchers have worked on developing curricula, activities, and integrating CS into other subjects. Franklin et al. [Fra20e] developed a culturally
Figure 2.2 Multiple scripts bound to one event on one sprite [Wei18b].

Figure 2.3 The sequence learning trajectory one of three K-8 computational thinking learning trajectories developed and described by Rich et. al. [Ric18].
relevant curriculum. This curriculum has a modular and sequential structure and it was developed for students who have already been introduced to the basics of programming. Each module has three possible tracks: multicultural, youth culture, and gaming. Fofeng et al., [Fof20] developed activities for mathematics and computational thinking where the subjects complemented and mutually supported each other. The activities were developed for 4th grade students. Finally, researchers have also explored how teach quantum reversibility with 8-9 year old students [Fra20c].

Recent work has brought about the creation of new teaching strategies such as Use-Modify-Create, which has shown promising results with 4-8 grade students [Fra20a] as well as TIPP&SEE, which stands for Title, Instructions, Purpose, and Play, and Sprites, Events, and Explore [Sal20c; Fra20d]. TIPP&SEE is a metacognitive learning strategy that helps to scaffold students in Scratch from the Use to Modify steps.

Other work in computer science education for younger students include investigating the equality of computer science in K-8 classrooms between low- and high-performing schools [Sal19] and investigating how coding affects preschool children’s cognitive and problem-solving abilities [ÇB20].

Although computer science education research surrounding elementary students has only begun gaining traction recently, researchers have learned a great deal about how children in this age group learns CS concepts and develop problem solving skills. The community is also working together to develop resources and curricula and to encourage schools and school districts to implement computer science curricula. This work has continued to inform my work, particularly helping me in developing activities for studies.

2.4 Systems with Adaptive Support for Collaboration

Intelligent tutoring systems have traditionally focused on supporting individuals learning specific concepts or subjects. In recent years, researchers have started to investigate how adaptive systems can support students learning collaboratively.

Harrer et. al. [Har06] aimed to support collaboration using an intelligent tutoring system while students used collaborative software. To reach their goal, the authors proposed a process called bootstrapping novice data (BND), where data on learners collaborating is collected and used to form the basis for a tutor. When the data is collected, students’ actions are represented by a network graph that is a combination of all the groups’ solutions. The tutor can then intervene if it notices that a student is going down a path that leads to incorrect solutions. Harrer et. al. implemented BND and evaluated it using two studies. From their first study they discovered that students needed support in five areas of their collaboration process: conceptual understanding, visual organization, task selection, task coordination, and task coherence. In their second study, their goal was to validate the findings from the previous study and assess how they could support students in those dimensions. From their second study, the authors found
that groups that had a good solution to the problem had a less balanced work distribution, which is consistent with findings from the first study. They also found that results of task coherence and task selection differed between studies. In study 1, groups that had good or incomplete solutions had similar task coherence while groups with poor solutions seemed to have coherent problem-solving strategies. In the second study, there was little difference between the groups in terms of task coherence. In terms of task selection for the first study, groups with good or incomplete solutions focused on objects based on their level of abstraction but groups with poor solutions focused on objects based on their visual proximity. As with task coherence, the authors found little difference in task selection between groups in the second study.

Walker, Rummel, and Koedinger [Wal09] developed the Collaborative Tutoring Research Lab (CTRL) framework for research on adaptive collaborative learning support. This framework covers the data collection, the development of adaptive support features based on the data collected, and the implementation of studies to evaluate the support features. The CTRL framework consists of six types of components: tools, tutors, translators, learner management, research management, and control module. The integration of the different components is facilitated by the control module. The authors implemented this framework into Cognitive Tutor Algebra (CTA) and evaluated it in a peer tutoring study. The authors found that there was a positive correlation between the number of incorrect tutee problem solving attempts and tutor learning. They also found that there was a negative correlation between the number of incorrect attempts to solve a problem and tutee learning.

Diziol et. al. [Diz10] proposed using adaptive collaborative scripts to support students collaborating in an intelligent tutoring system. The system evaluates the students’ problem solving behavior to determine when and how to intervene. This technique had often been used in individual intelligent tutoring systems but has yet to be adapted for collaborative support. The authors ran two studies in two different contexts to evaluate this system; the first context was a peer collaboration context and the second was a peer tutoring context. In the first study, the intelligent tutoring system detected student behaviors such as trial and error and hint abuse and provided feedback based on those behaviors. The students in this study worked in a peer collaboration scenario where they worked together to solve an algebra problem. The authors discovered that their system reduced trial and error and hint abused behaviors and increased student collaboration in the peer collaboration scenario. In the second study, the students worked in a peer tutoring scenario; the tutor gave feedback to the tutee based on the tutor’s and tutee’s actions. From the results, there was evidence to suggest that peer tutors benefited from using the system but the tutees did not. Overall, the adaptive collaborative scripts seemed to improve student performance over fixed collaborative scripts, which are much more rigid.

In the intelligent tutoring system domain, Olsen and colleagues have implemented an ITS to support elementary students collaboratively learning mathematics. Following this implementation were studies and analyses that investigated collaboration and tutor responses. In their most
recent work, Olsen et al. [Ols18] used the gaze and dialogue data to investigate the students’ problem solving process. The authors found that the students’ gazes were more similar when the students were discussing more abstract ideas and when the students receive feedback that their work was correct. Olsen et. al. [Ols15a] also found that over time, students who interacted with conceptually-oriented tutors had a significant decrease in their amount of interactive talk and the number of errors the students made decreased significantly. There was also a strong positive correlation between the change in interactive talk and steps with errors. Students who worked collaboratively with the tutor were able to complete problems more quickly than students who worked individually [Ols16] but there were not any differences between the learning gains of individual versus collaborative learners [Ols15b]. An investigation of whether individual, collaborative, or both was more effective in the tutoring environment, Olsen et. al. [Ols17; Ols19] found that the students who learned both individually and collaboratively had higher learning gains than those that only learned individually or collaboratively.

In another study of collaboration in math, Olsen and Finkelstein [OF17] examined the conversations of fourth grade students collaborating on word problems within the intelligent tutoring system. Students were randomly assigned to one of three conditions where they viewed problem-solving example videos: an individual student thinking aloud solving a math problem, a pair collaborating on a problem from an unrelated topic, and a pair collaborating on a math problem. Researchers annotated collaborative session segments based on the quality of the reasoning (talking about the problem) and rapport (synergy between the partners). There were significant positive correlations between the rapport and reasoning state and between the reasoning state and post test scores which indicate that pairs with better relationships may perform better when problem solving [OF17].

Harsley et. al. modified an existing intelligent tutoring system, ChiQat, that supports computer science learning to also support collaboration. The new system was named Collab-ChiQat. The researchers investigated using collaboration feedback such as hints about successful collaboration, graphics about their equity, and a collaboration score. Initial student impressions were positive [Har16a]. In a comparison of a version of Collab-ChiQat that did not provide feedback (unstructured) with one that did provide feedback (semistructured), Harsley et. al. discovered that more students in the unstructured condition found the system more helpful but the students in the semistructured condition were able to describe three attributes of good pair programming [Har16b]. Further analyses revealed that the semistructured students may not have had balanced relationships which caused some students’ learning to suffer [Har17a]. Additionally, it appears that students in the semistructured condition did not take as much time to plan before beginning to work and they attributed more individual acts to good collaboration than the unstructured students did [Har17a]. In another study comparing students working in pairs versus students working individually in Collab-ChiQat, Harsley et. al. found that pairs requested fewer examples than individuals; individuals spent more time on most problems than pairs; individuals spent more time looking at their examples; and pair programmers coded more
efficiently [Har17b]. Similar to Harsley et al.’s work, Celepkolu [Cel20] developed a dialogue visualization tool to enable students to reflect on their pair programming relationship. Celepkolu developed this through iterative refinement with studies and focus group interviews. The final version of the tool displayed the number of words spoken over time for each partner through a line graph, the total number of words spoken by each partner in a pie chart, the total number of questions spoken by each partner in a pie chart, and the distribution of question types spoken by the pair. Additionally, the students could view and interact with their transcripts, view keywords of their transcripts at the time they select, and watch their screen during the time they select.

In addition to regular programming environments, researchers have also explored how collaborative programming games can support learners. Shi et al. developed a programming game that supported undergraduate novices in collaborating while solving problems [Shi19]. In the game, turn taking was enforced and the students were also given constraints on which programming constructs they could implement. They found that the students had more balanced time in terms of typing in this condition and the students planned more than the students using pair programming. Melcer and Isbister compared individual and collaborative problem solving using a desktop interface versus a tangible interface for a robot programming game [MI18]. They found that the undergraduate students had less programming anxiety when collaborating and those who collaborated with the tangible interfaces enjoyed the game more. These researchers explored new and exciting collaborative programming games, however, they were designed to work with undergraduates rather than younger children.

To improve collaboration and gender equity between middle school students solving CS problems in a game-based learning environment, Buffum et al. designed and implemented learning companions as an intervention [Buf15b]. The students interacted with the learning companions four times between their interactions with the game-based learning environment, Engage. The interactions were separate from Engage, but were designed to support the students’ frustration. The students interacted with the companions by choosing from a menu of conversation options and the companions would respond to the students’ choices. Buffum did not find evidence of the learning companions affecting the distribution of talk or ratio of suggestions to emotive utterances between students collaborating. Additionally, the learning companions did not affect the proportion of suggestion or question utterances. The learning companion condition had a higher number of verbal abuse utterances towards the agents. While he found that the learning companions did not support every aspect of collaborative dialogue, his research revealed just how complex collaborative interactions are and further highlights the importance of research on collaboration and how adaptive systems can mitigate the challenges [Buf17].

In very recent work, Stewart et al., [Ste19] developed a computer-supported collaborative system that automatically detected construction of shared knowledge, negotiation/coordination, and maintaining team function. They tested their models with undergraduate students in ten teams of three. The students collaboratively worked through the Minecraft-themed Hour of
Code. The authors found promising results in their spoken language models.

There are a number of other tools and environments for collaborative programming in text based-environments (e.g. [Res09; Wei18a]). However, unlike many block-based programming environments such as AliCe-ViLagE, these tools were designed primarily for professionals and do not support the younger learners that we focus on. To our knowledge, there has not been little research about how to create or modify programming environments to support elementary students while they pair program. Our work attempts to fill this gap.

The work presented in this section informed my work to answer RQ2. Some researchers’ work focused on the same age group as my work. Many researchers’ work is about supporting computer science learning, however, their support methods differ from mine or they worked with a different age group. Additionally, Buffum et al., and Celepkolu investigated students’ collaborative dialogue, which is involved in both of my research questions.
Chapter 3

AN INITIAL LOOK AT ELEMENTARY STUDENTS’ COMPUTER SCIENCE COLLABORATIVE DIALOGUE

My overarching research goal is to develop a deeper understanding of how elementary students interact with one another during pair programming in order to better support their collaboration in the future. My research goal gives rise to two distinct research questions that we investigate through our analyses: first, *RQ1a: How do young students balance their dialogue, turn-taking and control during collaborative computer science learning?*; and second, *RQ1b: 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. In addition, we have two secondary research questions that relate to the first research question: *RQ1ba. How do elementary students make suggestions while engaging in pair programming?* and *RQ1bb. How do elementary students respond to suggestions while engaging in pair programming?*. I answered this question by using a qualitative approach to review the students’ verbal suggestions, acceptances, and rejections. This research examines interactions between elementary school-level programming partners working in pairs within their classroom and was early work we completed to begin to understand elementary students’ collaborative process in computer science. The methods used in this work are similar to the methods that I have used to answer the research questions of this dissertation and the results of this work led us to RQ2 for this dissertation. This work was completed in the Spring and Fall of 2017 and were published in the International Journal of Child-Computer Interaction (IJCCI) and the Technical Symposium on Computer Science Education (SIGCSE).

I collected a corpus of data including code submissions, planning documents, and video
of the 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 chapter 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 type of dialogue moves that the students made. The results show that some student pairs take turns at the controls naturally and that both partners contribute actively to the dialogue, while other pairs resist taking turns and display an imbalance of dialogue contributions.

3.1 Method

3.1.1 Participants

This work was based on 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 were in this group. As is often the case with computer science electives, we saw a disproportionately high number of male students enrolled in this course. The total number of students who took the elective was 55 (39 boys, 16 girls). Of these students, 26 (16 boys, 10 girls) 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.

Table 3.1 lists the six pairs by gender and provides the pseudonyms by which we will refer to them throughout this analysis.
Table 3.1 Identifiers, pseudonyms, and genders of collaborating pairs, along with the duration of the videos collected and transcribed/annotated (in minutes) for each pair. *John and Mia’s second video was lost in the video transfer process, so we have only one class period instead of two.

<table>
<thead>
<tr>
<th>Pair ID</th>
<th>Student Pseudonyms</th>
<th>Gender Composition</th>
<th>Minutes of Video</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Charlie and Quinn</td>
<td>Male, Male</td>
<td>61</td>
</tr>
<tr>
<td>2</td>
<td>Ian and Aaron</td>
<td>Male, Male</td>
<td>54</td>
</tr>
<tr>
<td>3</td>
<td>Adalyn and Catarina</td>
<td>Female, Female</td>
<td>62</td>
</tr>
<tr>
<td>4</td>
<td>Alonzo and Gigi</td>
<td>Male, Female</td>
<td>54</td>
</tr>
<tr>
<td>5</td>
<td>John and Mia</td>
<td>Male, Female</td>
<td>30*</td>
</tr>
<tr>
<td>6</td>
<td>Greg and Harry</td>
<td>Male, Male</td>
<td>64</td>
</tr>
</tbody>
</table>

3.1.2 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 chapter are referred to as collaboration “sessions”. The class was offered to students four times each year, and students may enroll 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. In the summer and fall of 2014, I co-developed the course in partnership with the elementary school teacher who led it. In the class, students designed code in Scratch 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, how to broadcast and receive signals, and how to 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 responding 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 the students were prompted 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 the implementation process, the students worked together through pair programming. The class teacher made the pair assignments based upon his own criteria which were unknown to us. The teacher generally asked the students to switch roles every ten minutes; however, our research field notes show 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.

### 3.1.3 Data Collection

Over the course of the study we collected handwritten and code artifacts produced by the students, videos of the students pair programming, screen capture videos\(^1\) 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. Figure 3.1 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.

### 3.2 Data Annotation

In order to address our research questions, we transcribed and annotated the videos and problem-solving artifacts. We manually transcribed eleven videos covering six pairs of students working on the project. Each video covers one session in which both students engaged in pair programming. 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.

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 the person

---

\(^1\)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.
to whom students directed their request for help (each other, another student, or the teacher).

Figure 3.2 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).²

Then, in a second round of annotation, we annotated different categories of dialogue move³. As shown in Table 3.2, these tags capture when a student verbally contributes an idea or makes a suggestion⁴; 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; 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, “Whoa!” 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 data set. To calculate inter-rater reliability, we randomly selected 20% of the sessions (3 sessions) for a second researcher to annotate. The level of agreement was κ = 0.62, showing substantial reliability [LK77]. After discussion of the disagreements, the first author refined her annotations of the full data set before proceeding with analyses. (The kappa was not recomputed.)

²ELAN: https://tla.mpi.nl/tools/tla-tools/elan/
³The term “dialogue move” here 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.
⁴We define “contributing an idea” as offering any new information that might help solve the problem, while “making a suggestion” means providing information that is actionable
Figure 3.2  Screenshot of the tagging software.

Table 3.2  The dialogue move annotation scheme, ordered from most frequent to least frequent per session.

<table>
<thead>
<tr>
<th>Dialogue Move Category</th>
<th>Description</th>
<th>Example(s)</th>
<th>Average Count per Student per Session</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make Suggestion (MS)</td>
<td>Student verbally makes a suggestion or contributes an idea</td>
<td>“Click that”</td>
<td>62</td>
</tr>
<tr>
<td></td>
<td></td>
<td>“Here, how bout the fox.”</td>
<td></td>
</tr>
<tr>
<td>Ask Partner for Help (AP)</td>
<td>Student verbally asks partner for help</td>
<td>“How do you do it?”, “How did you get that?”</td>
<td>6</td>
</tr>
<tr>
<td>Accept (AC)</td>
<td>Student verbally accepts or acknowledges partner’s idea or suggestion</td>
<td>“Okay”</td>
<td>6</td>
</tr>
<tr>
<td>Reject (RJ)</td>
<td>Student verbally rejects partner’s idea or suggestion</td>
<td>“No it isn’t’</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>“No it shouldn’t’</td>
<td></td>
</tr>
<tr>
<td>Ask Other for Help (AO)</td>
<td>Student verbally asks someone other than their partner for help</td>
<td>“Mr. Smith!”</td>
<td>2</td>
</tr>
<tr>
<td>Ask to Drive (D)</td>
<td>Student verbally tries to switch roles</td>
<td>“It’s my turn”</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>“Let me try”</td>
<td></td>
</tr>
</tbody>
</table>
3.3 Results: Talk Time, Driving Time, and Consecutive Dialogue Moves

In this section we first examine the balance of talk time and driving time in each of the six pairs. Next, we examine 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, we examine 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.

3.3.1 Talk and Driving Time

We computed the number of seconds that each partner spoke (talk time) and the number of seconds that each partner was at the controls (driving time) based upon the video annotation. Figure 3.3 displays the percentage of time that each partner spoke and drove in 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 the navigator, and others contributed little to the dialogue while navigating. We explore these phenomena in further detail below.
3.3.2 Consecutive Talk

Talk time indicates the duration of time a student spoke, but it cannot capture whether that speech was primarily in a few moves that lasted longer or in many moves that were shorter. 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 3.3). 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 [Sto00]. 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 because the non-speaking partner is listening [Bar03]. When we see a very high number of consecutive dialogue moves, however, this can often indicate that either one partner is disengaged, or that 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 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
Table 3.3  The average and maximum number of consecutive dialogue moves by each partner.

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

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.

### 3.3.3 Dialogue Move Types

As described in Section 3.2, 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 chapter.) Figure 3.6 displays the relative frequency of MAKE SUGGESTION for each pair. Generally, comparing this frequency to the talk times indicated in Figures 3.4 and 3.5 shows a correlation between the two, which is intuitive. However, desirably, there is not such a strong correlation between the percent of drive time and the percent of ideas contributed. In pair programming, both driver and navigator should contribute ideas 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.
Figure 3.4 Correspondence between overall speech time and MAKE SUGGESTION speech time for each pair.

Figure 3.5 Correspondence between overall speech time and MAKE SUGGESTION speech time for each pair.
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 Figure 3.7. We omitted Make Suggestion (MS) from Figure 3.7 because its frequency was so high that it obscured the differences between the other less frequent moves.
3.3.4 Answering RQ1a and RQ1b

Sections 5.1 and 5.2 answered RQ1a, How do young students balance their dialogue, turn-taking and control during collaborative computer science learning? We found that in general, the students in our data set do not balance their dialogue, turn-taking, and control. Section 5.3 answered RQ1b, How do young students coordinate their dialogue during collaboration for computer science? In our dataset, the students’ utterances were more often MAKE SUGGESTION than any other type and the number of suggestions made correlated with the amount of time they talked. Additionally, some pairs had more ACCEPT utterances while others had more REJECT utterances.
3.4 Case Studies: Talk Time, Driving Time, and Consecutive Dialogue Moves

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.

3.4.1 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 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. 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]. Charlie rejected the suggestion [04:40] and Quinn reiterated his protest [04:41]. 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].

5The recordings were collected on contiguous days.
Excerpt 1 (Driver: Charlie, Navigator: Quinn)

<table>
<thead>
<tr>
<th>Quinn</th>
<th>They don’t need to be down there.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charlie</td>
<td>Yeah, they do.</td>
</tr>
<tr>
<td>Quinn</td>
<td>I don’t (()).</td>
</tr>
<tr>
<td>Charlie</td>
<td>(()) yes, they do!</td>
</tr>
<tr>
<td>Quinn</td>
<td>It’s the other one.</td>
</tr>
<tr>
<td>Charlie</td>
<td>They need to be down here for this backdrop.</td>
</tr>
</tbody>
</table>

[Charlie looks at the code, realizing he was incorrect.]

| Charlie     | No they don’t.                     |
| Charlie     | That’s why I’m switch it.           |

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 the teacher wanted the students to switch again.
3.4.2 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 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 substantive exchange, answering a question [27:15].

<table>
<thead>
<tr>
<th>Excerpt 3 (Driver: Ian, Navigator: Aaron)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aaron Move to two hundred.</td>
</tr>
<tr>
<td>Aaron Go to the house button.</td>
</tr>
<tr>
<td>Ian One second. No (()) I know.</td>
</tr>
<tr>
<td>Aaron Wait, I know what to do.</td>
</tr>
<tr>
<td>Ian I know what to do.</td>
</tr>
<tr>
<td>Aaron What?</td>
</tr>
<tr>
<td>Aaron Let me help you.</td>
</tr>
<tr>
<td>Aaron Click the home button.</td>
</tr>
<tr>
<td>It in. Is- it does not say. Okay,</td>
</tr>
<tr>
<td>Aaron why do you have the when clicked</td>
</tr>
<tr>
<td>there?</td>
</tr>
<tr>
<td>Ian So it will turn back after fox</td>
</tr>
<tr>
<td>touches.</td>
</tr>
<tr>
<td>Aaron Don’t you think (()) that you</td>
</tr>
<tr>
<td>should put those three together?</td>
</tr>
<tr>
<td>Ian No, (()).</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Excerpt 4 (Driver: Ian, Navigator: Aaron)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aaron Wait, can I try something?</td>
</tr>
<tr>
<td>Ian I’m just gonna try one thing.</td>
</tr>
<tr>
<td>Ian Where did he go?</td>
</tr>
<tr>
<td>Aaron He’s under the house.</td>
</tr>
<tr>
<td>Ian Right here?</td>
</tr>
<tr>
<td>Aaron I wanna try something. Where’d</td>
</tr>
<tr>
<td>the pig go?</td>
</tr>
</tbody>
</table>
3.4.3 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 went without speaking to one another. 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, her longest sequence was 17 consecutive moves, and Catarina’s was 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 teacher. 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 recorded. 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 was 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.
Excerpt 5 (Driver: Catarina, Navigator: Adalyn)

Catarina See, it’s not moving Mr. Smith. [AO]
Catarina Mr. Smith. [AO]
[Catarina stops driving.]
Adalyn (()) [O]
Catarina Mr. Smith. [AO]
[Adalyn begins driving by grabbing the mouse.]
Adalyn [singing](())\singing] [O]
Catarina Oh, that’s because I’m on the um. [MS]
Catarina No. No, no, no. [O]
Catarina Gimmie this. [D]
[Catarina takes the keyboard and mouse.]

3.5 Results: Giving and Taking Suggestions

We identified several types of suggestions and responses from the excerpts discussed here (Table 3.4). A command is a direct instruction to the partner. A proposal is a suggestion that is detailed enough to inform the receiver of what exactly needs to be done next but less direct than a command. A next step suggestion is a more high-level suggestion, indicating a task that may need to be completed next or in the near future.
## Table 3.4 The list and examples of suggestion types.

<table>
<thead>
<tr>
<th>Type</th>
<th>Example(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposal</td>
<td>“So this should be no.”; “How about let’s test?”; “Let’s try moving him up some.”; “…I think we should take the ask block out.”</td>
</tr>
<tr>
<td>Command</td>
<td>“When backdrop changes to wood, glide right here in front of her, right there. That’s one thirty-four, negative one thirty-five.”; “Try one ninety steps.”; “Go to Jack.”; “Now let’s go to scripts.”</td>
</tr>
<tr>
<td>Next Step</td>
<td>“We just n– need to make that turn into a horse.”; “…she needs to lay down and … she needs to wake up.”; “… when backdrop changes to woods, they have to not be showing.”; “And then we have the house hide.”</td>
</tr>
</tbody>
</table>

From reviewing the interactions in the excerpts we found four types of responses students had to their partners’ suggestions. The response types were: implement and verbally acknowledge; implement and do not verbally acknowledge; do not implement and verbally acknowledge; and do not implement and do not verbally acknowledge. Of these, perhaps the most productive response type is **implement and verbally acknowledge**, because it is clear that the student understood the partner’s suggestion and chose to follow it.

**Implement and do not verbally acknowledge** may still be a productive response because it at least implies that the partner heard and accepted the suggestion; however, without verbal acknowledgement it is unclear whether she actually heard her partner or if she took action on her own initiative. Additionally, if she heard the suggestion and implemented it without any discussion, it is difficult to evaluate whether she understood the reason for the action.

**Do not implement and acknowledge** is another type of response we observed. Although choosing not to implement a suggestion is an implicit rejection of the idea, the verbal acknowledgement of the suggestion confirms that the partner actually heard it and it may turn into a fruitful discussion about the suggestion’s merit.

**Do not implement and do not verbally acknowledge** is perhaps the least helpful type of
response in a collaborative context. If a partner does not acknowledge a suggestion in any way, it is unclear whether she did not hear it or simply chose to ignore it. Either way, such a response will likely deter the speaker from making another suggestion later.

3.5.1 Answering RQ1ba and RQ1bb

In this section we answered RQ1ba, How do elementary students make suggestions while engaging in pair programming?, and RQ1bb, How do elementary students respond to suggestions while engaging in pair programming? The answer to RQ1ba is that students in our dataset gave one of three types of suggestions: proposal, command, or next step. To answer RQ1bb, we found that the students responded to the suggestions in one of four ways: implementing and verbally acknowledge, implement and do not verbally acknowledge, do not implement and verbally acknowledge, do not implement and do not verbally acknowledge. More often than not, students accepted or rejected suggestions without acknowledging them.

3.6 Case Studies: Giving and Taking Suggestions

In order to answer our secondary research questions, we examined case studies of four pairs of students in the dataset. This led us to better understand the dynamics of the relationships and areas in which specific interactions could benefit from scaffolding.

3.6.1 Charlie and Quinn: “I think I can...”

We reviewed two sessions of Charlie and Quinn engaging in pair programming. In the first session Charlie appeared to be more open to collaborating and explained some of the code to Quinn. However, as time passed, Charlie also appeared to become frustrated with having to collaborate and at times he stopped verbally acknowledging Quinn’s suggestions.

Regardless of which role he was in, Charlie spoke much more than Quinn, and gave many more suggestions than Quinn did. He also showed impatience multiple times while Quinn was driving by repeatedly asking when it would be time to switch roles. We noticed that Charlie often gave specific suggestions in a series, and Quinn often followed those suggestions without verbally acknowledging them. Their behavior indicates that Charlie was the dominant member of the pair. The following excerpt further illustrates this point:

Excerpt 6 took place over about 30 seconds. In this time Quinn made one specific suggestion at turn 154, while asking for confirmation from Charlie when he says, “...right?” When Charlie verbally accepted the suggestion during turn 155, Quinn continued working and implemented it. Once Quinn was done, Charlie then gave him a string of suggestions in turns 158 and 159 which Quinn followed without speaking. The suggestions that Charlie gave were very direct and confident, leaving no question of what he wanted Quinn to do. In fact, these suggestions felt more like commands.
At first blush we might conclude that Charlie and Quinn worked relatively well together. They did not argue (a common problem for young collaborators), and they filled their respective roles roughly as expected. However, they were missing a crucial part of the collaboration process: discussion. The lack of discussion meant they simply followed one person’s vision without establishing common ground, a process described in Cazden and Beck [CB03], and may not have benefited as much from the collaboration as they could have. Throughout both sessions, Quinn implemented most solutions as a driver without questioning Charlie and he was a quiet navigator, especially in the second session where he seemed more distracted and withdrawn. There was usually more discussion during the rare times Quinn disagreed with Charlie, but in most cases, Charlie stood firm and seemed to have the final say.

### 3.6.2 Ian and Aaron: “I know.”

In this pair Ian seemed to prefer working independently while Aaron appeared to enjoy collaborating. Aaron gave the majority of the suggestions and Ian followed some of them and ignored others, often without verbally acknowledging them. Other times, when he acknowledged Aaron’s suggestions, he said, “I know,” before continuing to work. This is consistent with the behavior we mentioned above where Ian drove the majority of a session.

In Excerpt 7, Ian made some high-level suggestions, then solicited advice by saying he did not know how to implement those solutions. Aaron then gave him some low-level suggestions, which Ian did not verbally acknowledge, but did implement. When Ian implemented the suggestions, he did not seem to find what he needed, so he went to the “Scripts” category instead while stating it out loud. While Ian soliciting advice was a positive action to take, the lack of discussion is another instance of a failure to establish common ground.
Excerpt 7 (Driving: Ian, Navigating: Aaron)

[Ian opened their program which he had closed when the pair went to talk to their teacher.]

163 Ian I think we should do this.
164 Ian And then we have the house hide.
[Ian moved an existing block slightly and gestured towards a sprite using his mouse.]
165 Ian I don’t know how to.
166 Aaron You gotta go to stage.  
[Ian went to the Costumes tab.]  
167 Aaron Then you go to backdrops.  
[Ian went to the Backdrop tab.]  
168 Ian (()).
169 Ian Scripts.  
[Ian went back to the Scripts tab.]

3.6.3 Alonzo and Gigi: “Try it...”

For the most part Alonzo and Gigi got along well. In some ways Gigi seemed more dominant than Alonzo, but Alonzo gave more suggestions than she did and she generally followed those suggestions without disagreeing with him. His suggestions were often specific and she implemented them without problems. However, during the few times she disagreed with him, he stood his ground and they implemented his suggestions.

Excerpt 8 (Driving: Gigi, Navigating: Alonzo)

[Gigi went to the Backdrops tab.]  
253 Alonzo Now go to mine.  
[Gigi changed the backdrop and moved a sprite.]  
254 Alonzo My drawn one.  
[Gigi moved a different sprite.]  
255 Gigi No.  
256 Alonzo Yes bruh, we’re using that one.  
257 Gigi The ugly one?  
[Gigi goes back to Stage.]  
258 Alonzo Yes.  
[Gigi changes the backdrop.]  
259 Gigi You’re mean.
Excerpt 9 (Driving: Alonzo, Navigating: Gigi)

[Alonzo is looking in the Sounds library.]

439 Gigi What- whatever, (()) just pick one.

[Alonzo continues looking.]

440 Gigi Growl. Let’s (()) growl.

441 Gigi (())

442 Alonzo Alright, wait. I’m tryin think of what we need bruh.

[Alonzo goes to the animal category.]

443 Alonzo Alright, what do we need. An animal noise, right?

444 Gigi Yeah.

In Excerpt 8, Alonzo gave Gigi suggestions and, similar to Charlie, Alonzo gave suggestions that were closer to commands. Then Gigi disagreed with one and said, “No,” during turn 255. Alonzo responded by firmly stating that they were going to implement his idea in turn 256. Gigi protested again, but as she protested, she started to implement the change he suggested. In the end, despite the number of protests she made, they settled with his idea. This is an example of an interaction where a verbal acknowledgment and protest was not enough. The students would have benefited from self-explanation [Chi94], where each student presents an argument for or against the idea that was presented.

Excerpt 9 illustrates Alonzo and Gigi deciding which sound to add to their program. Gigi made a suggestion in turn 440. In turn 442, Alonzo responded by telling her to wait and that he was thinking about what they needed. He did not implement her suggestion; instead, he continued searching and made his own suggestion, to which Gigi agreed. Instead of discussing Gigi’s suggestion and why he did or did not agree with it, he decided not to implement it and came up with his own suggestion.

3.6.4 John and Mia: “Oh, that’s stupid.”

In the first part of our analysis, we observed that John gave more suggestions than Mia did. Our current observations of their conversation and programming process revealed that that Mia often did not react positively to John’s suggestions or give reasons for disagreeing with him. John also did not give reasons for his suggestions.

Overall, the relationship of these two students seemed to be much more competitive than collaborative. Mia often attributed the problem to her partner rather than the task at hand [Kel67]. They would sometimes even flaunt when one partner was correct and the other person was incorrect. Additionally, neither student explained their thoughts and ideas and Mia was not open to her partner’s suggestions. This undesirable behavior is illustrated in Excerpt 10:
Excerpt 10 (Driver: John, Navigator: Mia)

[John is making changes to the code.]

219 Mia What are you doing?
[John continues to work without saying anything.]

220 Mia We have to do the same thing that we did for the first one!
[John goes to the Sensing category.]

221 Mia What are you doin’?
[John grabs an ¡Ask¿ block.]

In Excerpt 10, John was making changes to the code when Mia noticed he was doing something she did not expect. Since he was working silently, she did not understand why he was making those changes and asked him what he was doing in turn 219. As John continued to work without answering her, she made a suggestion in turn 220. Once again, John did not answer and Mia repeated her question. Since John did not acknowledge her questions or her suggestions and he appeared not to have implemented her suggestion, Mia was obviously frustrated. It was not until the teacher noticed and asked John to explain what he was doing that Mia did John start talking to her again.

3.6.5 Harry and Greg: “No, ‘cause...”

Harry and Greg were both very engaged and excited about their project. While they did not always agree with each other’s ideas, they tended to follow similar trains of thought. This pair of students had perhaps one of the most successful collaborative interactions we observed from the participants. There were times when Harry had to make the same suggestion multiple times before Greg would implement it, but they had at least one situation where they gave a reason for disagreeing with each other. This situation is shown in Excerpt 11.
While Harry was working on a task in Excerpt 11, Greg gave a general suggestion in turn 255. Harry acknowledged the suggestion but disagreed and gave a reason for it in turn 256, which Greg accepted. Even though Greg had already accepted the explanation, Harry continued to explain his thoughts in turn 257 which Greg also agreed with. This interaction is encouraging, as previous work has found that students who approve suggestions or enter into discussion have more successful collaborations [Bar03].

3.7 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, RQ1a. How do young students balance their dialogue, turn-taking and control during collaborative computer science learning?, and RQ1b. How do young students coordinate their dialogue during collaboration for computer science? We answered RQ1a by using talk time and driving time as proxies for relationship balance. We measured whether the students’ talk and driving time were balanced or imbalanced in each pair and we found that most of the pairs in our dataset were not balanced according to these measures. In order to answer RQ1b, 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.

As we answered our main research questions, we noticed that a large number of the student utterances were suggestions, and previous work indicates that suggestions are vital to a collaborative interaction [Dow15]. This led us to ask the questions: RQ1ba. How do elementary students make suggestions while engaging in pair programming?; and RQ1bb. How do elementary students respond to suggestions while engaging in pair programming? We answered these...
questions by qualitatively reviewing the students’ transcripts to identify themes in the utter-
ances. We found that students often used three types of suggestions and responded in four ways
to their partners’ suggestions.

This analyses of elementary school students’ collaborative pair programming dialogues re-
vealed several patterns that provide useful insights as we work toward designing effective sup-
ports for collaborative learning 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 learning 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 collaborative computer science learning. 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 [ZB16], good collaboration strategies
have the particularly important potential to increase equity and broaden participation.

Finally, the students in our dataset gave four types of suggestions and reacted in one of
four ways to these suggestions. In addition, they often accepted or rejected suggestions with-
out explanation or explicit acknowledgement. In an empirical study, Barron [Bar03] found that
students were more successful in their collaborative process if they accepted or discussed sugges-
tions more than if they rejected or ignored them. Most commonly, the students in our dataset
do not verbally acknowledge suggestions, whether they implemented them or not. Our observ-
ations and Barron’s results support the notion that these actions prevent the students from
having, through self-expression [3] and grounding [2], the meaningful discussion that is vital
to the collaborative programming process. Therefore, it is important to encourage students to
explain their thought process to their partners.

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.
3.8 Conclusions and Future Work

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. However, much of the research in this area has focused on undergraduate students and there is much that is unknown about how elementary students collaborate while solving programming problems. In a study with 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, particularly suggestions and the actions that follow those suggestions. 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. We also found four types of suggestions, and four types of responses to those suggestions. In addition, we observed that more often than not, students either accepted or rejected suggestions from their partners without verbally acknowledging them. Behaviors like this hinder the collaboration process and we must consider ways to scaffold these interactions in order to help students have a better experience when working together.

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 chapter has only examined a few of its dimensions. By gaining insight into the collaborative practices of young students, we can identify strategies that are most beneficial to them, and work toward scaffolding that helps children achieve their full potential in collaborative learning.

3.8.1 Implications for My Dissertation Research

This research highlights two open research areas, which will be explored in later chapters of this dissertation.

Years of research in various domains have contributed to an emerging set of best practices for collaborative computer science learning: collaborators benefit from self-explanation [Chi94], question asking [Gra94], discussing the challenges of the task rather than criticizing each other [Wei80], and building upon each others’ ideas to establish common ground [CB03]. 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. This led to the formation of my first research question: To what extent do
upper elementary students use self-explanation, question generation, uptake, antagonization, and praise when collaboratively solving computer science problems? This research question is answered in Chapter 5.

Collaboration 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. In fact, this finding is not unique to children: it has been observed in undergraduates as well [Wil08]. From our results and from previously published work, we suggest that when conducting research on collaboration, researchers should ensure that students are trained in collaboration. 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. RQ2 is answered in Chapter 7., RQ2: To what extent do the following features in an adaptive programming environment support upper elementary students’ collaborative problem solving process?

- A built in timer for switching control
- A built in tasklist for switching control
- A reminder to use self-explanation
Chapter 4

CONFLICT EVOLUTION BETWEEN ELEMENTARY COMPUTER SCIENCE STUDENTS

One research question that came out of my overarching research goal is, *How do conflicts arise and evolve when elementary learners engage in pair programming?*. We investigated this question by analyzing videos of elementary students from two schools as they engaged in pair programming. This investigation led us to understand more about areas where collaboration between our participants broke down and provided insight into how we can support them. I have also identified conflicts between pairs to answer RQ2 of my dissertation. The work described in this chapter was completed in Fall 2018.

Like other researchers [LS15; Sha14; Ruv16], we were inspired by the evidence for the effectiveness of pair programming with novices in other age groups, and we implemented pair programming in our CS classroom activities with elementary school students. We began with the traditional pair programming approach: two learners at one computer, taking turns with the controls. The student controlling the laptop is the driver and the other student is the navigator and is tasked with planning ahead and looking for mistakes. However, we observed many conflicts between the students that were unlike those we had observed in our extensive studies of undergraduates and middle school students who pair programmed. Our observations of multiple classrooms led us to believe that studying conflicts is a crucial open area for research.

We categorized student actions from pair programming videos into four stages of conflict: initiation, escalation, de-escalation, and conclusion. In addition, some conflicts were resolved. Conflicts between the students were triggered by disagreements about who should control the keyboard and mouse, what changes should be made, and by other interpersonal concerns. As the conflicts unfolded, the students often argued or physically wrestled over equipment, with
conflicts ranging in length from four seconds to three minutes and four seconds. Finally, we observed that 71% of the students’ conflicts were never resolved; the students merely moved on to other topics.

4.1 Data and Methods

4.1.1 Participants

We collected data from two implementations of a pair programming intervention at different elementary schools in the Southeastern United States. The participating schools included a rural public school (Adams) and an urban charter school (Evans)\(^1\).

The curriculum, designed by the myself and another RA, covered CS topics such as conditionals, loops, and input using the Scratch programming platform. The lessons differed slightly between the schools: I taught hour-long lessons at Adams and adjusted lessons based on the students’ progress, while the lessons at Evans were taught by the local school technology instructor, who used a modified version of the curriculum that had been extended to include an open-ended game-design project. Lessons at Adams occurred once a week and covered seven total hours of instruction. At Evans, lessons were two hours long and occurred once a week for six weeks leading to twelve total hours.

A total of 40 4th and 5th grade students (typically 8 to 11 years old) participated in the interventions and consented to be observed, 20 in each school. We collected webcam video and screen capture data from each pair of students using *Open Broadcaster Software*\(^2\), which allowed us to record both streams of video in a single file (Figure 4.1).

\(^1\) All school and student names in this Chapter are pseudonyms.

\(^2\) https://obsproject.com/
4.1.2 Methods

4.1.2.1 Identifying Videos

To better focus our research, we selected videos from specific classroom lessons that students found challenging. While observing Adams, we noticed that the students particularly struggled during the last two sessions where we taught them how to make games using many of the concepts that the students learned earlier, such as conditionals, loops, and user input. We found that the students at Evans struggled the most when learning about input and broadcasting. Of the 30 possible videos for analysis, 20 videos met minimum requirements of audio clarity (both auditorily clear and spoken in the English language). Three videos were selected using purposeful stratification from each of the two schools. Each video from Adams was approximately one hour long. The videos from Evans were two hours each; we extracted 45 minutes from the middle of each video to analyze because that would most likely be when the students were most active and focused.

4.1.2.2 Extracting Episodes of Conflict from Videos

Our next step was to identify episodes of conflicts within each video. We developed a protocol for identifying episodes of conflict and annotating the stages of them. The protocol was based on definitions from prior research on conflict. The first definition is “One child attempting to influence another child who opposes or resists the first child” [Har88]. The second is a “perceived divergence of interest, a belief that the parties’ current aspirations are incompatible” [PK04]. After identifying conflict episodes, we subdivided each conflict episode into stages. Initiation was the start of the episode. Escalation was an “increase in the intensity of conflict as a whole” [Rub94]. De-escalation was a “reduction in the intensity of conflict” [Rub94] or the “adoption of new strategies following a perceptional adjustment” [Jeo08]. The conclusion was the end of the episode. Consistent with Rubin et al., we considered a conflict episode to be resolved when the students reached “substantive agreement dealing with most or all of the issues in a conflict” [Rub94]. In the protocol, we encouraged each researcher to use their best judgment in deciding whether an event was a conflict episode.

Then four other graduate research assistants and I independently reviewed the videos and identified conflict episodes; two researchers were assigned to each video, with each researcher reviewing two to four videos. This served as a way of crosschecking work and ensuring that the identification of conflict episodes remained as consistent and objective as possible. We used the students’ dialogue, physical actions, and facial expressions to determine whether an interaction was a conflict episode. We noted the start and end time of the potential conflict episode and what made the episode begin and end.

Every annotator disagreement was labeled. An annotator disagreement (AD) occurred when one of us identified an instance as a conflict episode while the other did not. All five of us met to resolve the ADs. During the session, we reviewed each video segment where there was
an AD, presented our reasons for believing the interaction was or was not a conflict, then recoded it accordingly. We reviewed about 50% of the ADs together. For the remaining ADs, we independently watched their video segments and voted on whether we believed each event was a conflict episode. Each person only voted on the ADs from videos we were not assigned to at the beginning of the process. The finalized conflict episodes were ones that did not have an AD or ones that had an AD and three votes for it being a conflict episode.

The final step involved tagging each conflict for escalation, de-escalation, and resolution. For this step, each of the researchers from the previous step was assigned to conflict episodes to tag the stages. The assignments for the stage tagging process was similar to the conflict tagging process, with two researchers tagging each conflict. The differences in tagging in this step were resolved using the same process as the previous step.

4.2 Results

4.2.1 Conflict Initiation and Progression

We identified a set of progressive stages that characterized the episodes. Although every conflict episode had a clear start and a conclusion, not all of them were escalated, de-escalated, or even resolved. We identified 38 total conflict episodes (simply ‘conflicts’ for short) in our analysis, with an average of 6.3 conflicts per video. Table 4.5 shows the distribution of conflicts over the videos as well as the average, maximum, and minimum percentage of conflicts that escalated, de-escalated, or resolved. The majority of the conflicts (22) lasted 15 seconds or less while 6 lasted from 16 to 25 seconds. Table 4.2 and Figure 4.2 display the total number of conflicts, average duration of conflicts, and the percentage of conflicts with escalation, de-escalation, and resolution per pair.

<table>
<thead>
<tr>
<th></th>
<th># Conflicts</th>
<th>w/ Escalation</th>
<th>w/ De-escalation</th>
<th>w/ Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>38</td>
<td>61%</td>
<td>55%</td>
<td>29%</td>
</tr>
<tr>
<td>Ave</td>
<td>6.33</td>
<td>57%</td>
<td>51%</td>
<td>29%</td>
</tr>
<tr>
<td>Max</td>
<td>9</td>
<td>89%</td>
<td>89%</td>
<td>60%</td>
</tr>
<tr>
<td>Min</td>
<td>4</td>
<td>20%</td>
<td>20%</td>
<td>0%</td>
</tr>
</tbody>
</table>

While reviewing the conflicts, we found that they were often triggered by one of three issues: disagreements about changes in the code, disputes over who should drive, and non-CS problems (e.g. camera position, personality/social disagreements). As the conflicts unfolded, the disagreements often evolved into verbal arguments and physically wrestling over the laptop,
camera, or each other. As noted above, we found that the majority of conflicts between the students, 71%, went unresolved. When resolution did occur, it was because one of the students directly acknowledged that the other was correct.

### 4.2.2 Case Studies

In order to better understand the dynamics of each pair’s relationship and to illustrate the nature of their disagreements, we summarize what we observed about each pair and describe selected conflicts in detail below.

#### 4.2.2.1 Pair 1: “The Average Case”

Pair 1 consisted of two males, Corey and Shawn. We found a total of seven conflicts lasting an average of 18 seconds. Four of the conflicts between them escalated, three de-escalated, and two were resolved. We believe this pair to be an average case.

The majority of Cory and Shawn’s conflicts started with disagreements about changes in the code. When we started watching the video, Shawn drove and was testing their code by getting the user’s input. Throughout the session, they switched back and forth regularly, which often occurred when one person had an idea.

One particularly representative example of their conflicts occurred about 35 minutes after
Table 4.3 An excerpt of Corey and Shawn’s conflict.

<table>
<thead>
<tr>
<th>Time</th>
<th>Speaker</th>
<th>Action/Dialogue</th>
</tr>
</thead>
<tbody>
<tr>
<td>35:33</td>
<td>Corey</td>
<td>Do z or.</td>
</tr>
<tr>
<td>35:38</td>
<td>Shawn</td>
<td>But it is p. Make it p.</td>
</tr>
<tr>
<td>35:39</td>
<td>Corey</td>
<td>So you have to do z. Say.</td>
</tr>
<tr>
<td>35:42</td>
<td>Shawn</td>
<td>No. No.</td>
</tr>
<tr>
<td>35:43</td>
<td>Corey</td>
<td>No. No no no no. No.</td>
</tr>
<tr>
<td>35:44</td>
<td>Corey</td>
<td>((()) um.</td>
</tr>
</tbody>
</table>

the start of class. Excerpt 4.3\(^3\) displays the transcript of the pair’s dialogue and actions. At the start of this conflict, Corey drove while the teacher was walking them through an example (35:33). The conflict escalated when Shawn reached for the laptop (35:42) and Corey started to sound more anxious and pushed Shawn’s hands out of the way (35:43). The conflict ended after 35:44 and did not seem to de-escalate or reach a resolution. This episode may have negatively affected their future interactions because of the escalation and lack of resolution.

4.2.2.2 Pair 2: Unresolved Conflicts

Hamzah and Arif, the two male students in Pair 2, had a total of eight conflicts during the video we analyzed, making them the second most fractious pair among all the groups. On average, the conflicts lasted 18 seconds, with five escalating, five de-escalating, and none having a resolution.

Half of the conflicts this pair engaged in included a physical component. In three cases, they wrestled for control of the laptop, physically pushing each other’s arms out of the way or preventing one another from controlling the trackpad. In another case, Hamzah left the space and could be heard on the video working with another group, at which point Arif called to him to return and help. Of note, when Arif commented that it was not his turn to drive, the teacher intervened and reminded the pair to work together and that they both needed to see the computer.

The conflict shown in Excerpt 4.4 began with Hamzah driving and Arif giving a suggestion (31:10). The conflict escalated around 31:23 with Arif saying “No” multiple times. The conflict de-escalated around 31:25 where Hamzah admitted he was messing up and continued to make changes. The conflict ended without a clear resolution as Arif seemed to disengage and Hamzah shifted the laptop away from his partner. It was clear that this pair of students struggled in their collaboration. Their unresolved conflicts likely caused more issues as time went on and they needed teacher intervention to be able to work productively.

\(^3\)((() stands for inaudible.
Table 4.4  An excerpt of Arif and Hamzah’s conflict.

<table>
<thead>
<tr>
<th>Time</th>
<th>Speaker</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:10</td>
<td>Arif</td>
<td>Then “say”, then “say”.</td>
</tr>
<tr>
<td>31:13</td>
<td>[Hamzah begins modifying the existing code.]</td>
<td></td>
</tr>
<tr>
<td>31:18</td>
<td>Arif</td>
<td>No no, say that one.</td>
</tr>
<tr>
<td></td>
<td>[Arif points at the screen.]</td>
<td></td>
</tr>
<tr>
<td>31:21</td>
<td>Arif</td>
<td>No, up here.</td>
</tr>
<tr>
<td></td>
<td>[Arif points at the screen.]</td>
<td></td>
</tr>
<tr>
<td>31:22</td>
<td>[Hamzah selects a large block of code and moves it]</td>
<td></td>
</tr>
<tr>
<td>31:23</td>
<td>Arif</td>
<td>No no no no no.</td>
</tr>
<tr>
<td>31:25</td>
<td>Hamzah</td>
<td>Uh, I’m messing it up... there you go</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(moves a block)... get out (moves a block)... get out (moves a block)...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>get out (moves a block)</td>
</tr>
<tr>
<td>31:30</td>
<td>Arif</td>
<td>No just leave it.</td>
</tr>
<tr>
<td>31:34</td>
<td>Hamzah</td>
<td>Happy birthday.</td>
</tr>
<tr>
<td>31:37</td>
<td>[Hamzah shifts the laptop to face only him and continues coding]</td>
<td></td>
</tr>
</tbody>
</table>

4.2.2.3  Pair 3: “The Longest Case”

Nia and Madison, two female students, engaged in five distinct conflicts over the course of the video, making them the second lowest among all the pairs. However, the average duration of their conflicts was high (53 seconds), and only two conflicts were resolved.

Their conflicts centered on code implementation issues or on issues completely unrelated to programming. For example, Nia laid her head on Madison’s shoulder partway through the video. Madison jokingly told her to stop but became more serious when Nia did not move. All of the conflicts about implementation issues were resolved. The implementation conflicts tended to last longer than the non-CS conflicts.

It is important to point out that the first implementation conflict lasted longer (around three minutes) but the two spent a nontrivial part of it joking and laughing. During this conflict, Madison also held Nia’s hand to prevent her from controlling the computer. This escalation occurred about 20 seconds into the conflict. Once the escalation began, the girls began to wrestle and joke around while giggling for over a minute. A minute and a half into the conflict, Madison stopped to test the code. Nia interrupted her and they continued wrestling again. Madison’s facial expression and tone of voice seemed to switch between amused and serious. Nia also seemed to express frustration at one moment stating that Madison was not doing what was on the dry-erase board. During this interaction, the pair did not seriously discuss changes in the code. The conflict started to de-escalate when the teacher came by, around two and a half minutes into the conflict, and gave them specific instructions. The resolution of the conflict occurred when Madison states, “Nia, I’m driving here.” Nia agreed but interjected that Madison is a terrible driver. Another implementation conflict occurred near the end of the video and was settled more peacefully, with little conversation but more serious coding. While the students
appeared to be friends, their familiarity with each other paired with disagreements about the code resulted in long conflicts that fluctuated from light-hearted bantering and wrestling to annoyance and frustration. These conflicts greatly affected the students’ productivity, as they lasted longer than the average case.

4.2.2.4 Pair 4: Camera Distraction

Pair 4 (Lindsey and Joey) had the fewest number of conflicts (4) but none of their conflicts resolved. Additionally, they had the second highest percentage of escalation (75%). Their conflicts lasted an average of 17 seconds and half of their conflicts de-escalated.

Half of Pair 4’s conflicts during the session involved the webcam placed by researchers. Twice, Joey moved the camera, one time telling Lindsey to shift her seat so she could be seen better. A third conflict involved the physical placement of the laptop itself; Joey and Lindsey pulled the laptop back and forth several times.

4.2.2.5 Pair 5: “The Problematic Pair”

Matt and Chris in Pair 5 have the highest number of conflicts (9) and the highest rate of escalated conflicts (89%). Interestingly enough, they also have the highest rate of de-escalated conflicts (89%) and second highest percentage of resolved conflicts (44%). On average, their conflicts lasted 35 seconds and three of their conflicts lasted over a minute.

The pair’s conflicts were focused on the programming task and on control of the laptop, with the majority of the conflicts starting from disagreements about control. Unlike some of the other pairs, every conflict in this pair that escalated also de-escalated. The only conflict that did not escalate and de-escalate was their shortest, which lasted for 9 seconds. Of the eight conflicts that escalated and de-escalated, four of them were resolved, including two of the conflicts that were longer than a minute. This is interesting since the number of conflicts and rate of escalation suggests that Matt and Chris struggled to get along.

A closer look at the stages of their conflicts reveal that although they encountered many conflicts, they may have been more skilled at collaboration, which allowed them to reach a resolution more often than the majority of their peers. For example in the following excerpt (Excerpt 4.5), Chris typed “Better go lower this time” into the “say” block (41:02) and Matt realized that it was incorrect (41:24). The conflict escalated in an argument from 41:27-41:37. Between the end of Matt’s argument and Chris’ next statement, Chris appeared to pause and think about what Matt stated. The conflict de-escalated when Chris agreed with Matt (41:45). The conflict was resolved with Matt’s last statement in the excerpt, in which he explained how he had originally thought the same way as Chris.
Table 4.5 An excerpt of Chris and Matt’s conflict.

<table>
<thead>
<tr>
<th>Time</th>
<th>Actor</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>41:02</td>
<td>[Chris typed in “Better go lower this time” in the “&lt;Say [] for 2 secs&gt;” block]</td>
<td></td>
</tr>
<tr>
<td>41:02</td>
<td>Matt</td>
<td>Better go lower...this...time. Better go lower this time. Better go lower this time.</td>
</tr>
<tr>
<td>41:14</td>
<td>[Chris added “your stupid” to the block and then deleted the text.]</td>
<td></td>
</tr>
<tr>
<td>41:23</td>
<td>Matt</td>
<td>If the num</td>
</tr>
<tr>
<td>41:27</td>
<td>Chris</td>
<td>It’s better go higher this time. Wait.</td>
</tr>
<tr>
<td>41:30</td>
<td>Chris</td>
<td>No, 'cause it is greater so you have to go lower.</td>
</tr>
<tr>
<td>41:37</td>
<td>Matt</td>
<td>No, the number- If your- if your answer is great- is less than the number.</td>
</tr>
<tr>
<td>41:45</td>
<td>Chris</td>
<td>Okay, so you have to go</td>
</tr>
<tr>
<td>41:51</td>
<td>Matt</td>
<td>Higher.[Moved the laptop toward Chris] Type what you wanna type.</td>
</tr>
<tr>
<td>41:54</td>
<td>[Chris typed “Better go higher”.]</td>
<td></td>
</tr>
<tr>
<td>41:58</td>
<td>Matt</td>
<td>I was thinking the same you were. I thought- I thought our number versus the answer. No, our answer versus the actual number, difference.</td>
</tr>
</tbody>
</table>

4.2.2.6 Pair 6: The “Low-Conflict” Case

Darla and Michael have the second lowest number of conflicts (5), the shortest average duration of conflicts (12 seconds), the lowest rate of escalated conflicts (20%), and the highest percentage of conflicts with resolutions (60%). They also had the lowest rate of de-escalated conflicts (20%). Overall, the pair seemed to have few issues with each other.

Three of the conflicts that occurred between this pair were control conflicts. The remainder were triggered by disagreements about changes to the code. Two of the conflicts neither escalated nor de-escalated but were resolved. Excerpt 4.6 displays the third conflict in their session. The conflict began when Michael made a change in the code and Darla asked him what he was doing (29:39). Throughout the conflict, the students’ tone of voice did not seem to differ. In the end, they both seemed to agree that the block was supposed to contain 10 instead of -10, and the conflict was resolved when the correct change was implemented (29:46). The resolution of the conflict likely improved future interactions between Darla and Michael, although the low number of conflicts suggests that this pair’s relationship may already be better than other pairs.

4.3 Discussion

We identified episodes of conflict within six pairs of elementary-age pair programmers. The results of these analyses have implications for practitioners and researchers. We formed initial
impressions on each of the six pairs of students based on their descriptive statistics. Looking across all pairs, Darla and Michael (Pair 6) were considered the “low conflict case” because they had the fewest conflicts, shortest average duration of conflicts, and highest resolution rate. In contrast, we considered Matt and Chris in Pair 5 to be the “problematic pair” because, in terms of the statistics on the number and length of conflict episodes, they were essentially the opposite of Darla and Michael. We found that each pair of students had a characteristic about their conflicts that stood out to us and could potentially define their collaborative success. Although these statistics on the number and length of conflict episodes are an efficient way to review the pairs, it is just as important to understand how the conflicts begin, what causes them to escalate, de-escalate, and to be resolved, and how long it takes the students to do so.

Going back to Darla and Michael, their first two and last conflicts were about control of the laptop and the first and last conflicts were not resolved. According to Jeong, “If no agreement is reached, the conflict may escalate again into a test of strength and resolve, or it may remain frozen at the status quo” [Jeo08][p. 99]. It is possible that the second conflict about control occurred because the first conflict was not resolved. The time between the conclusion of the first and the initiation of the second was less than three minutes. Additionally, the second lasted over twice as long as the first, 22 and 9 seconds, respectively. This may be an indication that the relationship between the students at that time was tumultuous. We have noticed similar patterns to this in our observations of young pair programmers in the past. This is a problem both practitioners and researchers could work to solve. It is important for practitioners to clearly define the pair programming roles and set expectations. It may also benefit the students to have lessons on good collaborative practices. Researchers in artificial intelligence and educational technology can develop software to identify when similar conflicts occur multiple times between the same students and either intervene or alert the teacher.

The case study of Matt and Chris was similarly enlightening. The majority of their conflicts were about control. However, they had the most conflicts and the highest rates of escalations and
de-escalations, which contrasts with Darla and Michael who had the lowest rates of escalations and de-escalations. During the beginning of the session, the pair’s conflicts were more spread out and were often resolved. Towards the end of the session, the conflicts arose more frequently and were resolved less often. This suggests that as time went on, the conflicts that occurred between the students began to strain their relationship with each other. However, the excerpt about the pair’s implementation conflict also brought to light that Matt used an important strategy, self-explanation [Chi94], which can help students improve their own understanding and communicate a concept to their partner. Matt also tried to relate to Chris by telling him they were originally thinking the same way. This interaction is promising since previous findings suggest that task-related conflicts have more of a positive effect on peer collaboration than process and relationship conflicts [Lee15].

Another interesting case was that of Lindsey and Joey in Pair 4. Although we only touched on this case briefly, they had unexpected conflicts, which were ones that mostly revolved around the webcam, which was on top of the laptop’s screen. These conflicts that occur around conditions that the researchers introduce are important to consider. Although they are not about the collaborative process, they affect the students’ relationships and learning. Researchers designing studies should be careful to minimize these effects.

### 4.3.1 Limitations

Several limitations are important to keep in mind as we interpret the findings of this work. First, we have examined a relatively small sample, though standard for this type of annotation-intense video analysis work. The findings we have presented are based on six videos, although there are approximately ten videos for each of the thirteen classroom sessions across the two schools. Second, even with theoretically-informed, working definitions of conflict and stages, and multiple researchers coming to agreement on how to label these events, it is still not possible to arrive at a 100% objective measure of what constitutes a conflict. We have followed methodological best practices to mitigate this limitation. Finally, a limitation and ongoing challenge for video- or audio-based classroom research is the amount of background noise, students’ use of varied languages, and scalability of equipment.

### 4.4 Conclusions

While pair programming has been shown to be beneficial to novices, much of the research has focused on undergraduate students and industry professionals [Wil00; WU01]. Supporting younger learners in collaborative CS learning brings new challenges that researchers must be cognizant of, including the students’ level of socio-emotional development. We identified episodes of conflict within six pairs of elementary-age pair programmers and calculated the summary statistics of the conflicts. This allowed us to understand how the overall conflicts evolved and how they evolved in each pair. Some pairs appeared to have productive collab-
orative interactions and were able to resolve their conflicts while others had difficulty either because one partner appeared not to listen or because both partners were distracted by factors unrelated to the activity. Though a small sample, our video analysis shows types of conflicts that are uncommon with older learners, specifically conflicts that lead to physically wrestling for control of the computer. Although our study was focused on the topic of collaborative learning in computer science, we believe that our findings can generalize to other problem solving contexts, especially our findings of the types of conflicts that exist and how the students worked towards resolving them.

Our findings highlight the great importance of better understanding how elementary students collaborate while learning CS if the potential of this pedagogical approach is going to be maximized. The results can inform both future classroom practice and the design of intelligent software support and can impact how students learn computing as a discipline and shape how they work with others in the future. As such, I have explore the conflicts between students to answer my second research question of my dissertation. The next chapter covers the first question in my dissertation.
Chapter 5

INVESTIGATING UPPER ELEMENTARY STUDENTS’ COLLABORATIVE DIALOGUE STRATEGIES AND CONFLICTS DURING PAIR PROGRAMMING

My initial investigation of upper elementary students’ collaborative programming dialogue and behaviors revealed ways inequity emerged in their relationships (Chapter 3). This was reflected in their driving time, talking time, and the way they accepted or rejected each other’s suggestions. This chapter includes a more detailed analysis of the students’ dialogue behaviors. Specifically, I investigated their dialogue using a framework that allowed me to examine their self-explanation, question generation, uptake, praise, and antagonization behaviors while pair programming. Moreover, I investigate how those behaviors see to relate to the students’ conflict behaviors, which I first investigated in Chapter 4.

Collaborative learning is a complex process that involves co-constructing knowledge and maintaining shared conceptions of the current activity [RT95]. Prior research has established that the amount of time students spend actively communicating is correlated with the success of their collaborative learning process [Tud92; Kru92; Bar03]. Prior work also suggests that taking an active part in the construction of solutions is important for learning [Chi09; CW14]. However, younger students do not necessarily have the required social skills or demonstrate good dialogue practices, which may hinder the success of their collaborative efforts [beau2010social; crook1998children].

Years of prior research across multiple domains have shown that collaborative learners benefit from engaging in self-explanation [Chi94], question asking [Gra94], and uptake (building upon each others’ ideas to establish common ground) [CB03]. Antagonistic actions and dia-
logue can be hurtful and may not foster a collaborative environment [Ruv16]. This leads to the research question, **RQ1**: To what extent do upper elementary students engage in self-explanation, question generation, uptake, and praise when collaboratively solving computer science problems? Conversely, to what extent do they antagonize each other? The detrimental effects of conflict can include avoidance of uncomfortable situations [Nay14] and fear of voicing concerns [PN19], but certain kinds of conflicts can also play an important productive role during collaboration. To our knowledge, this research is the first to deeply investigate conflicts with elementary students as they pair program. This gap leads to a second research question, **RQ1a**: How do upper elementary students’ collaborative dialogue practices relate to the number and type of conflicts that emerge during pair programming?

We investigated these questions by analyzing twelve videos of elementary school students as they engaged in pair programming in their classrooms. We annotated each video for students’ collaborative dialogue moves as well as instances of conflict. Then we annotated the conflict types and partitioned the pairs into high- and low-conflict pairs. The analyses showed that across all pairs, there was a much higher level of productive moves (23%) than antagonizing moves (1%). There were more instances of uptake with low-conflict pairs and were more instances of antagonization dialogue moves within the high-conflict pairs. Additionally, high-conflict pairs were more balanced in their talk distribution. The results of this study demonstrate the promise of our tagging scheme for better understanding of how antagonistic conflict relates to collaborative processes, and the findings may hold implications for building intelligent learning environments that support students in this context.

### 5.1 Methods

#### 5.1.1 Research Context

We collected data from a five-week pair programming intervention in which 16 Academically or Intellectually Gifted (AIG) 5th grade students in the Southeastern United States used NetsBlox [BL17] —a visual, block-based programming platform— to complete various coding challenges. Fifteen of those students consented to participate in data collection. The curriculum for the intervention was designed and taught by the authors, and covered topics such as loops, conditionals, and variables, and culminated in the students designing and implementing a simple game. As part of this curriculum, the students were taught about pair programming. They were introduced to the roles and responsibilities of the driver and navigator, and they were taught about the importance of talking through their decision making process. Some of the students received additional exposure to the curriculum as the intervention occurred during separate morning and afternoon sessions, with students who received gifted services for both math and English language arts attending both daily CS sessions. The activities during the sessions were similar with slight variances. The students were paired by the classroom teacher and the pairs
changed each session.

5.1.2 Data Collection and Sampling

We collected video of the students from the laptop’s webcam, audio, and screen captures of their pair programming activity within a single file; this permitted us to view their interactions and system actions synchronously. The students wore headsets with microphones to ensure higher quality audio data. The videos were then transcribed verbatim and students were assigned pseudonyms.

We collected 30 pair programming videos that were about 40 minutes each. Due to limited funding for transcriptions and the amount of time it takes to analyze videos and transcripts, we sampled the data using two main criteria: the audio had to be clear and the students had to properly follow the pair programming paradigm. Our final study sample was 12 videos.

5.1.3 Annotating Dialogue Moves

A “dialogue move” is a chunk of verbal or textual speech (used in the same way as making a ‘move’ on a chess board) [ginzburg1996dynamics]. We developed a dialogue move annotation scheme (Table 5.1) that was inspired by previous dialogue annotation schemes used for pair programming research [Rod17b; Tsa18a; Buf17; Ruv16; Gra94]. Many of the moves we are concerned with in this current work, such as explanation, question, antagonization, and directive/suggestion, were used in those schemes, while self-explanation, uptake, and praise were not included in the prior work. We included uptake and self-explanation because of the prominent role they play in effective collaboration and learning [Chi94; CB03]. We included praise because we noticed it in previous work as a counterpoint to antagonistic dialogue between young learners, and believe it plays a role in indicating positive collaborative dynamics.

In total, 8159 moves were tagged. We omitted student moves that were directed at the teacher, study facilitators, or peers who were not their partners. Moves were marked inaudible if they were not clear enough in the audio to transcribe (total of 70). Two researchers began with a training phase then independently tagged three randomly selected video sessions (about 23% of the data) from the corpus. Once the individual tagging was complete, we calculated the agreement ($\kappa = 0.65$), showing substantial reliability [LK77]. The first author annotated the remaining moves.

5.1.4 Annotating Conflicts

Our next step was to identify episodes of conflict within each video. We designed a novel protocol for identifying conflicts which was based on the following definitions drawn from prior research in this area: *One child attempting to influence another child who opposes or resists the first child* [Har88]. This is the same protocol that is used in Chapter 4 with an addition of conflict types.
<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
<th>Example(s)</th>
<th>Freq.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explanation</td>
<td>Statement explaining the student’s own idea, logic, or process.</td>
<td><em>I’m going to change it to 15 steps because it’s not moving far enough</em></td>
<td>11.79%</td>
</tr>
<tr>
<td>Self (Es)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Explanation</td>
<td>Statement explaining other components about the project, code, or problem solving process</td>
<td><em>The sprite is supposed to jump.</em></td>
<td>22.97%</td>
</tr>
<tr>
<td>Other (Eo)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Directive / Suggestion (D)</td>
<td>Statement telling the student’s partner to complete an action or offering an idea</td>
<td><em>Go to motion.; Delete that block.; Let’s make it jump.; How about making him dance?</em></td>
<td>16.92%</td>
</tr>
<tr>
<td>Question (Q)</td>
<td>Asking a partner a question about the task, process, their logic, or other relevant information about the process</td>
<td><em>What should we do next?</em>; <em>Where is the if block?</em></td>
<td>10.57%</td>
</tr>
<tr>
<td>Uptake (U)</td>
<td>Statement that builds upon the partner’s previous statement</td>
<td><em>Yeah, and we can also make her jump!; Let’s make the background change too.</em></td>
<td>0.12%</td>
</tr>
<tr>
<td>Praise (P)</td>
<td>Statement that emphasizes success</td>
<td><em>You’re smart; Great job!; We did it.</em></td>
<td>0.23%</td>
</tr>
<tr>
<td>Antagonization</td>
<td>Statement that appears to cause tension, including hurtful comments, instigating fights, prodding, putting down partner contributions, and showing annoyance with partner</td>
<td><em>You’re dumb; I’d rather work alone</em></td>
<td>1.09%</td>
</tr>
<tr>
<td>(A)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other - Related (OR)</td>
<td>Any move that does not fit under the categories above but still pertains to the activity</td>
<td><em>I don’t know why this isn’t working.; This is hard.</em></td>
<td>28.22%</td>
</tr>
<tr>
<td>Other - Unrelated (OU)</td>
<td>Any move that does not fit under the categories above and does not pertain to the activity</td>
<td><em>It’s almost lunch time; Um; Hmm</em></td>
<td>7.12%</td>
</tr>
</tbody>
</table>
The annotation for the entire dataset was completed by three researchers who were each assigned to independently review different subsets of the videos and to identify conflicts within them. Those subsets made up the entire dataset. Each video was assigned to two reviewers. We used the students’ dialogue, physical interactions, and facial expressions to determine whether an interaction was part of a conflict episode. We collaboratively discussed and re-tagged some target segments while reviewing the video clips. The remainder of disagreements were resolved by voting.

After episodes of conflict were tagged, we proceeded to label conflicts by type. We had noticed that conflicts could be centered around disagreements about changes in the code, disputes over who should drive, partner problems, and non-CS issues (e.g., camera position). A closer review of the conflict and student actions was used to determine the type. A task conflict is any conflict involving the task at hand (e.g., if Student A tries to make a change to the code, but Student B disagrees). A control conflict is any conflict involving the control of the computer (e.g., Student B asks if she could drive first, Student A refuses, they argue about who the driver should be). A partner roles/contribution conflict is any conflict involving one partner ignoring or downplaying their partner’s verbal contributions (e.g., Student A tries to make a change and Student B disagrees saying “I know better than you.”). Other encompasses any conflict involving anything other than Task, Control, or Partner Roles/Contribution (e.g., Student A and B argue about the camera because Student A was distracted by it). Annotators labeled 20% of the conflicts independently. Annotation disagreements were discussed until consensus was reached. One researcher labeled the remaining 80% of the conflict topics independently.

5.2 Results

We calculated descriptive statistics for each pair’s dialogue and conflicts. Table 7.5 displays those results. The number of moves spoken by the pairs ranged from 373 to 1133 and the average was 680. We identified a total of 79 conflicts with an average of 6.5 conflicts per pair. The fewest number of conflicts identified in a pair was 0 and the most was 17. About 96% of the conflicts were about the task, approximately 13% were about control of the computer, and 19% were about partner contributions/roles and control of the computer. The conflicts often started as task conflicts and some transitioned into other types as they continued.

5.2.1 Use of Dialogue Moves

To answer RQ1, we investigated the frequency of each dialogue move and the percentage of time that those moves occurred out of the 8159 moves. Explanation - Self (Es), and Question (Q) made up 11.79% and 10.57% of the data, respectively. Uptake (U) and Praise (P) comprised less than 1% of the tagged moves overall at 0.12% and 0.23%, respectively. It is clear that self-explanation, uptake, praise, and questions make up the minority of moves that upper elementary students in this dataset use in CS problem solving. In total, these types of moves
Table 5.2 Descriptive statistics of the conflicts and dialogue moves broken down by pair. Conflict category and balance category (see section 5.2.2), number of moves, number of conflicts, number of self-explanations, other explanations, directives, and questions, conflict types (task=T, control=C, partner=PC). Not shown are the percentage of antagonization, uptake, and praise. Antagonization was used 1% of the time overall. The maximum any pair used Uptake and Praise was 1% of the time. Finally this shows whether each pair used uptake (U), praise (P), or antagonization (A). The pairs are first ordered by conflict category and then balance category.

<table>
<thead>
<tr>
<th>Con. Cat.</th>
<th>Bal. Cat.</th>
<th>Pair ID</th>
<th>Pseudo. ID</th>
<th>Utts. (#)</th>
<th>Conflict Type</th>
<th>Exp-Self (%)</th>
<th>Exp-Other (%)</th>
<th>Direct. (%)</th>
<th>Ques. (%)</th>
<th>Con. Used Types</th>
<th>Used U, P, A?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Low</td>
<td>8</td>
<td>Rupert &amp; Anthony</td>
<td>492</td>
<td>1</td>
<td>9</td>
<td>23</td>
<td>15</td>
<td>12</td>
<td>T</td>
<td>U, P</td>
</tr>
<tr>
<td>Low</td>
<td>3</td>
<td>Sandy &amp; Melony</td>
<td>436</td>
<td>2</td>
<td>6</td>
<td>30</td>
<td>14</td>
<td>19</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low</td>
<td>9</td>
<td>Synthia &amp; Steve</td>
<td>843</td>
<td>4</td>
<td>16</td>
<td>29</td>
<td>17</td>
<td>7</td>
<td>T</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>Mid</td>
<td>10</td>
<td>Clara &amp; Rupert</td>
<td>410</td>
<td>3</td>
<td>12</td>
<td>35</td>
<td>20</td>
<td>6</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>High</td>
<td>6</td>
<td>Melony &amp; Mathew</td>
<td>483</td>
<td>0</td>
<td>9</td>
<td>17</td>
<td>16</td>
<td>16</td>
<td>T</td>
<td>U, P</td>
<td></td>
</tr>
<tr>
<td>High</td>
<td>4</td>
<td>Mathew &amp; Luke</td>
<td>373</td>
<td>2</td>
<td>5</td>
<td>19</td>
<td>23</td>
<td>14</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>High</td>
<td>Low</td>
<td>2</td>
<td>Sandy &amp; Anthony</td>
<td>572</td>
<td>12</td>
<td>16</td>
<td>28</td>
<td>11</td>
<td>10</td>
<td>T, C, PC</td>
<td>P, A</td>
</tr>
<tr>
<td>Mid</td>
<td>1</td>
<td>Luke &amp; David</td>
<td>800</td>
<td>5</td>
<td>11</td>
<td>16</td>
<td>26</td>
<td>10</td>
<td>T</td>
<td>U, P</td>
<td></td>
</tr>
<tr>
<td>Mid</td>
<td>5</td>
<td>Sandy &amp; Mitchell</td>
<td>732</td>
<td>17</td>
<td>15</td>
<td>26</td>
<td>15</td>
<td>14</td>
<td>T, C, PC</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>Mid</td>
<td>11</td>
<td>Dorothy &amp; Tylor</td>
<td>1133</td>
<td>8</td>
<td>8</td>
<td>25</td>
<td>12</td>
<td>7</td>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>High</td>
<td>7</td>
<td>Sandy &amp; Steve</td>
<td>872</td>
<td>17</td>
<td>14</td>
<td>21</td>
<td>11</td>
<td>9</td>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>High</td>
<td>12</td>
<td>Drew &amp; David</td>
<td>1012</td>
<td>8</td>
<td>14</td>
<td>15</td>
<td>23</td>
<td>9</td>
<td>T, C, PC</td>
<td>P, A</td>
<td></td>
</tr>
</tbody>
</table>
comprised 22.71% of the data. Antagonization occurred infrequently, making up 1.09% of the data. Explanation - Other made up 22.97% of the tagged moves. The most frequent dialogue move was Other Related (Or) at about 28% of the the tagged moves. Additionally, Other Unrelated (Ou) was used 7.12% of the time, which indicates that the students were on task the vast majority of their programming session. We did not examine these dialogue moves in our analysis. We also found that during conflicts, the students used self-explanation 15.11% of the time, questions 8.90% of the time, and antagonization 6.91% of the time. Additionally, the students did not use uptake or praise during conflicts.

5.2.2 Collaborative Dialogue Moves and Conflicts

To answer RQ1a, we investigated the ways in which dialogue balance, dialogue moves, and conflicts are related. We began by creating a stacked bar chart to display the pairs’ talk balance in terms of their moves as shown in Figure 5.1. We also categorized the pairs equally into low-, mid-, and high-balance with four pairs in each category as shown in Table 7.5. This was done based on overall talk distribution. Additionally, we divided the pairs into high- and low-conflict groups based on the median (4.5 conflicts per pair).

5.2.2.1 Low-Conflict

Of the six low-conflict pairs, none used antagonization, three pairs used uptake, and four pairs used praise in their collaboration. All low-conflict pairs only engaged in conflicts about the task. Three of these pairs were low-balance pairs, one was mid-balance, and two were high-balance.

Pair 8 (Rupert: 17%, Anthony: 83%) was the most unbalanced out of all the pairs in this dataset. Rupert spoke less than 20% of the time. An example of pair 8’s conflict is displayed in Table 5.3. Rupert was the driver and Anthony was giving him directions. While Rupert disagreed with Anthony, he still explained his reasoning to his partner on the third line. In the end, Anthony seemed to agree with Rupert’s reasoning.

The next most balanced pair was pair 3 where Sandy spoke about 78% of the time. This
Table 5.3 Conflict excerpt of Rupert and Anthony (pair 8).

<table>
<thead>
<tr>
<th>Student</th>
<th>Move</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anthony</td>
<td>Cut right here.</td>
<td>D</td>
</tr>
<tr>
<td>Anthony</td>
<td>Cut it right there.</td>
<td>D</td>
</tr>
<tr>
<td>Rupert</td>
<td>But you cut it before that though because it stops be-</td>
<td>ES</td>
</tr>
<tr>
<td>Anthony</td>
<td>Doesn’t he need to say oh look cake?</td>
<td>Q</td>
</tr>
<tr>
<td>Rupert</td>
<td>No because that’s not in the loop.</td>
<td>EO</td>
</tr>
<tr>
<td>Anthony</td>
<td>It’s.</td>
<td>OR</td>
</tr>
<tr>
<td>Anthony</td>
<td>That makes sense.</td>
<td>OR</td>
</tr>
</tbody>
</table>

pair did not have instances of uptake or praise. Pair 9 (Steve: 73%, Synthia: 27%), was slightly more proportionate than pair 3 and had 8 instances of praise. Pair 10 (Rupert: 36% and Clara: 64%) did not demonstrate uptake nor praise. Both pairs 6 and 4 demonstrated uptake (pair 6, 5 times; pair 4, 2 times) and praised each other once. In pair 6, Melony demonstrated five instances of uptake. Pair 4 had one instance of praise.

5.2.2.2 High-Conflict

With the high-conflict pairs, five out of six used antagonization, with a count of 89 uses total between the five pairs. The high-conflict pair that did not use antagonization also used both uptake and praise. Two other high-conflict pairs used praise as well as antagonization. Additionally, five of the pairs engaged in all three types of conflict. Only one pair was low-balance, three were mid-balance, and two were high-balance. We also noticed that these pairs often had a higher dialogue move count than their low-conflict peers.

The most balanced pair in the high-conflict group was pair 2 (Anthony: 68% and Sandy: 32%). They used both praise and antagonization; however, only Anthony praised (1 time) and only Sandy antagonized (1 time). Next in terms of balance is pair 1 (David: 67% and Luke: 33%). David used uptake (1 time) and praise (4 times). With pair 5 (Mitchell: 66% and Sandy: 34%), Sandy uttered 65% of the antagonization (33 total). In terms of balance, they are in the mid-balance category, but they are tied in first place with the number of conflicts. An example of pair 5’s conflict is show in Table 5.4. This conflict was about the task, control of the computer, and partner contribution. In comparison to pair 8, Sandy did not agree with her partner, but she did not explain her thoughts.

With pair 11 (Tylor: 38% and Dorothy: 62%), Tylor uttered 75% of this pair’s antagonization (4 total). Pair 7 had no instances of praise and 49 of antagonization. Table 5.5 displays an excerpt from a partner contribution/role conflict pair 7 engaged in. This is an example of a more lighthearted antagonization exchange. Pair 12 had one instance of praise and two of antagonization.
Table 5.4 Conflict excerpt of Sandy and Mitchell from pair 5.

<table>
<thead>
<tr>
<th>Student</th>
<th>Move</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mitchel</td>
<td>Wait I need to change</td>
<td>ES</td>
</tr>
<tr>
<td>Sandy</td>
<td>No no no.</td>
<td>A</td>
</tr>
<tr>
<td>Mitchel</td>
<td>No click turtle because ...</td>
<td>D</td>
</tr>
<tr>
<td>Sandy</td>
<td>No.</td>
<td>A</td>
</tr>
<tr>
<td>Mitchel</td>
<td>No but I’m blue.</td>
<td>EO</td>
</tr>
<tr>
<td>Sandy</td>
<td>That’s nice.</td>
<td>A</td>
</tr>
<tr>
<td>Mitchel</td>
<td>But I am blue.</td>
<td>EO</td>
</tr>
</tbody>
</table>

Table 5.5 Conflict excerpt of Sandy and Steve from Pair 7.

| Sandy   | You broke it                             | A   |
| Steve   | No I didn’t                              | A   |
| Sandy   | Oh                                       | OU  |
| Steve   | Look it says “Switch costume” and since we have- have no costume | EO  |
| Sandy   | It’s that right                          | Q   |
| Steve   | Yeah we just have to do that             | ES  |
| Sandy   | But it should already have the           | EO  |
| Steve   | Oh he doesn’t have any costumes other than this one | EO  |
| Sandy   | Oh he should already have that           | EO  |
| Steve   | Yeah so we have to make a new costume for the next costume | ES  |
| Sandy   | Can I make it                            | Q   |
5.3 Discussion

Collaborative learning is a complex process with many dimensions. By labeling dialogue moves and conflict types, we can gain insight into the ways in which young learners utilize collaborative dialogue and experience conflicts of different types as they code together.

**Use of collaborative dialogue moves.** The analysis for RQ1 revealed that, across all dyads, students used self-explanation, questions, uptake, praise, and antagonization about 23% of the time. The remaining were other explanation, directive/suggestions, other related, and other unrelated. Because the first four are moves that are important in collaboration [Chi94; Gra94; CB03], the community should continue to investigate these dialogue moves and whether students need more explicit instruction and scaffolded support as they communicate during pair programming.

**High-conflict pairs.** The analysis for RQ1a revealed that most high-conflict pairs used antagonization, and also engaged in conflicts about control and partner contribution while low-conflict pairs did not. This pattern held true for five out of six high-conflict pairs (Table 7.5). One high-conflict pair only engaged in task conflicts and did not use antagonization. We found that antagonization was often uttered in the form of insults (“You’re boring me to death.”) and sarcasm (“That’s nice.”). Some high-conflict pairs may use antagonization as a form of lighthearted banter (Table 5.5). Previous researchers have found that such impoliteness can be a signal of close rapport between friends [Wan12]. However in our observations we examined the tone and body language of students on the videos and we believe that other instances of antagonization indicate a genuine conflict between two students that may be related to issues with their personal compatibility or strong disagreements about design. This issue that has been noted in prior work with older learners [nagappan2003improving]. These results suggest that by analyzing collaborative dialogue, it may be possible to detect whether pairs may engage in more conflicts. If so, an important line of work will be to develop and investigate intelligent systems that detect conflict and alert teachers that the pairs need intervention.

**Low-conflict pairs.** The analysis for RQ1a also revealed that most low-conflict pairs used uptake and praise, but tended to be less talkative overall. Praise often took the form of short moves, such as “good job” or “we did it!” Uptake was more difficult to detect and often contained “Yeah like...” or “and...” after their partner gave a suggestion or directive. While uptake is a productive collaboration move, in some cases, it appears that one partner may perceive themselves to be less knowledgeable than the other partner, and may use uptake deferentially. Deeper investigations may reveal further nuance to the productive move of uptake and how best to mitigate overly deferential behavior by a partner who perceives themselves as less knowledgeable.

Some pairs with high dialogue balance and low conflict, such as pairs 4 and 6 (Table 7.5), seemed to have especially productive conversation. Their dialogue balance suggested that both students were contributing equally compared to other pairs. Additionally, both pairs used uptake
and praise. Their low number of conflicts were about the task rather than about control or each other’s contributions.

Additionally, low-conflict pairs did not engage in any conflicts about control of the computer or partner contributions/roles. These may be instances of one member being perceived as having more knowledge so the other student defers [SL19], reducing the likelihood of conflict and increasing the dialogue imbalance. This suggests that even low-conflict pairs may need teacher intervention to prevent one student from overpowering another.

Finally, many of the pairs that had a lower number of dialogue moves and did not use antagonization also engaged in fewer conflicts. While on the surface level it may appear that these learners are better collaborators, prior studies suggest that these learners may not have encountered many conflicts because they were not discussing their ideas or did not seek to resolve their differences [mugny1978socio]. Those learners that encountered more conflicts would ideally be discussing the project they were working on. In light of findings from previous work that indicate that the amount of interaction between learners affects learning gains [Chi09; CW14], future research should investigate how to encourage quieter learners to engage with their partners more. This can be completed in a variety of ways, such as adding dialogue reminders in the learning environment or asking teachers to make announcements.

5.4 Conclusions and Future Work

Although pair programming has been shown to be beneficial to novices, much of the research has focused on undergraduate students and industry professionals [Wil00; WU01]. Supporting younger learners in collaborative CS learning brings new challenges where we must be cognizant of the students’ level of socio-emotional development. In this study, we analyzed pair programming videos to better understand upper elementary students’ collaborative dialogue and conflicts. Of the specific collaborative dialogue moves we focused on, we found that they comprised about 23% of the dialogue used by these learners. The results also revealed that high-conflict pairs sometimes antagonized, while most used uptake and praise. High-conflict pairs engaged in conflicts over task, computer control, and partner contributions; low-conflict pairs engaged in only task-related conflicts and were less talkative overall. Finally, high-conflict pairs were often mid- or high-balanced dialogue pairs. These findings include both productive and unproductive conflict; however, high-conflict pairs seem to be more likely to engage in certain kinds of conflict or antagonistic dialogue moves overall. These findings suggest that through dialogue analysis, we may be able to detect conflict and provide supports to students. They also reinforce the nuances of the social dynamic within collaboration and the importance of empowering each student to contribute.

In future work, it is important to continue investigating collaborative dialogue as well as how to detect both productive and unproductive conflicts. With a larger dataset of tagged dialogue and conflicts, we can model dialogue and conflicts starting with the lessons learned in
this work about uptake, praise, and antagonization. Also of interest would be a deeper investigation of the relationship between dialogue and learning outcomes, as well as the relationship between learners’ self-perceptions and their dialogue. This research can impact how students learn computing as a discipline and shape how they work with others in the future. We will continue investigating how to support elementary CS learners’ collaborative processes.
Chapter 6

UPPER ELEMENTARY STUDENTS’ PERCEPTIONS AND USE OF COLLABORATIVE SUPPORT IN BLOCK-BASED PROGRAMMING: A PILOT STUDY

Since research on how elementary students collaborate in computer science has only emerged recently, there has been little work on how we can use adaptive environments to support them. In this work I investigated two adaptive support features and their efficacy in supporting collaboration between students in upper elementary school. This chapter details the pilot study and focus group analysis of the initial version of those features.

6.1 Motivation

Effective collaboration is challenging for children who are still developing necessary cognitive [Gok95; LD96], affective [Bra08], and social skills [Pia13; Lem15]. Researchers are actively investigating how children collaborate in various contexts and how they can be best supported, such as through audience interaction and collaborative learning [Apo18], addressing special needs [NBK19], and overcoming socio-cultural challenges through the use of collaborative applications [sharma2018overcoming].

In the domain of computer science and programming, there is emerging insight into how upper elementary students use block-based programming environments [Fra17; Han16] and comparing block-based, text-based, and hybrid block/text programming environments [WW18].
For collaborative programming, comparisons of teenager and younger learners’ collaborative processes have been made, [Pap17], and the importance of affect during collaboration has also recently been investigated [Sha19]. This recent body of work leaves open many questions around how we can best support elementary-aged learners collaboratively coding within classrooms.

One open question involves how to support learners in taking turns within collaborative roles during structured collaboration. Active contribution is important for learning and affective outcomes, but young learners often struggle to take turns within collaborative roles such as “driver” and “navigator” in pair programming [Dei16; Tsa18a]. During pair programming, two learners construct one piece of code together and take on two roles: the driver, who controls the computer, and the navigator, who directs the driver and looks for mistakes [WK02]. Although pair programming has been shown in a sizeable body of literature to be an effective approach for computer science collaboration, for young learners, imbalances in roles sometimes persist even when the teacher attempts to scaffold a balance.

A second open question involves the amount of time students spend engaging in dialogue with each other, a factor that is influential in the success of their collaborative learning [Tud92; Kru92; Bar03]. Without scaffolding or instruction, younger students do not necessarily follow productive dialogue practices for collaboration, which may hinder the success of their collaborative efforts [LS15; Sha14].

I address these open questions in the current work by developing features to support students’ pair programming within a block-based programming environment. To address the time-balance problem for actively constructing their code, I developed a software feature which I call switching-reminder, to remind the students when it is time to switch roles. To encourage students’ dialogue, I developed a talking-reminder, which reminds the students to discuss their ideas with each other. I investigated the use of these two features through a multi-day pilot study with 4th and 5th grade students and focus group interviews completed after the study. I added the features to the NetsBlox block-based coding interface [BL17]. My theory of action is that the talking reminder and switching reminder features will increase the talk balance of the students’ relationships, decrease conflicts, and foster self-efficacy.

For the switching-reminder, I prototyped two mechanisms: one time-based and the other task-based. Time-based role switching has been used in prior studies. Each coder takes a role and after a determined amount of time, they switch. This mechanism can result in role switching in the middle of a task, so I also prototyped a task-based switching-reminder, which prompted students to switch after they completed each task. For the talking-reminder, I reminded students to talk after sizeable additions to the code within a short time, and if no other reminder had been given in the past minute. Finally, the third novel feature I implemented was a task list, which was necessary to support task-based role switching and which allowed students to view their learning tasks within the coding interface as they worked. My overarching research question for these features is RQ2: To what extent do the following features in an adaptive programming environment support upper elementary students’ collaborative problem
I observed how students engaged with these features and asked students about their experiences with them. This chapter investigates the following research questions: **RQ2.1 How do students feel about the support features added to NetsBlox? RQ2.2 How did the students use the support features? RQ2.3 What would the students like to change about the support features?** My analysis of the focus group interviews revealed strengths and areas of refinements of the features as well as design principles that can inform the design of future collaborative support for young learners. This line of investigation can lead to more effective support features that empower young learners to collaboratively create and learn with computer science.

### 6.2 Collaborative Support Tool

Findings from my work and work by other researchers in upper elementary [LS15; Sha14; Tsa18a] and high school [Dei16] pair programming contexts indicate that students of this age resist switching roles even when directed to do so by their teacher or in response to an audible alarm. My previous classroom-based observations have included incidents such as students physically fighting for control of the laptop or verbal arguments over who the driver is. These experiences have led me to conclude that it is important to simplify the role switching process for younger learners and it is necessary to make explicit who should and should not be driving.

I have also observed that the ways in which students converse with one another varies considerably within and across classes. Some groups are highly talkative and collaborative whereas others seem to simply work quietly and independently, only speaking to their partner when necessary. I aimed to encourage partners to talk more readily so that each felt they were valued members of the pair. This goal is also supported by previous research [LS15; Sha14; Tsa18a].

Additionally, I noticed that students often asked the teacher for multiple reminders of their assignment or goal. This was not noticeably lessened when steps were written on the board or the teacher’s computer screen was widely shared. I hoped to find a way for students to maintain their focus on their coding work and encourage them to track their own progress as they work on their activities. This section describes the three features I prototyped to address those challenges: task list, role switching support, and dialogue support. Figure 6.1 shows the NetsBlox interface with my changes.
6.2.1 Task List

The task list served as a way for the students to keep track of what they have completed during each lesson and to support task-based role switching. Figure 6.2 shows the task list. For each lesson, I divided the activities into tasks. Each task was displayed in a tab. The tabs displayed the title of the task, a checkbox for the students to check if the task had been completed, and the description of the task. Only one task was displayed at a time and a task could only be displayed if the previous tasks had been marked as completed. At any time, the students could go back to a previous task.

I decided to place the task list in the bottom right hand corner of NetsBlox because this area is used for visual design tools that are sparsely used in my activities.

6.2.2 Role Switching Support

Previous research has emphasized the importance of students' participation in the tasks they collaborate on. Without adaptive support, elementary students often have an imbalance in their interactions when pair programming [Dei16; LS15; Sha14; Tsa18a].
Traditionally, instructors and researchers asked pairs to switch based on time (e.g., switch every 15 minutes) [Lew11; Lew10; Wer13; Ruv16; Wer05]. One benefit of this method is that if the pairs follow this guidance, they will each spend an equal amount of time in each role. However, if they switch right when they are told to, they may be in the middle of a thought or task and the switch may temporarily hinder their progress. Although there is a large amount of research on pair programming, to my knowledge, there has been no research that specifically
evaluated these and other methods to identify the best method for role changes.

In this work, I implemented two mechanisms for the collaborative support feature to remind students to switch roles when pair programming in NetsBlox. One mechanism used time-based switching with a fixed time per turn. The other used task-based switching, in which the students switched roles after completing a task. I refer to those mechanisms as time-based role switching and task-based role switching, respectively. With both mechanisms, students saw the same reminder: a pop-up was displayed that told the students to switch. The pop-up features an icon which was explained by myself at the beginning of the class period. The icons were also displayed on posters; the posters were placed on the desks in front of the students. Figure 6.3 shows the switch roles pop-up and Figure 6.5 is the poster the students saw.

When students first logged in, they determined who was the driver and who was the navigator first by entering their IDs into a pop-up form. Because I noticed that many of the students argued about who would drive first, on the final day of the study I modified the system to select the first driver randomly. At all times, the current driver’s ID was displayed in the top right corner above the stage (i.e., “Driver: student1”). This served as a reminder to the students in case there were any arguments about who should be driving. For the time-based version of the switching-reminder, I first determined how much time the students should have in each role. I reviewed prior research and observed the times other researchers used in their studies with elementary and middle school students: every five minutes [Lew11; Lew10], every ten minutes [Ruv16], every fifteen minutes [Wer05], and every twenty minutes [Wer13]. The average time between switches in this prior literature is 12 minutes, so I configured my time-based switching reminder for 12 minute intervals. In contrast, the task-based switching reminder appeared after the students clicked “complete” on one task and activated the tab of the next task.
Switch Roles

Switch driver and navigator!

Tell your partner what you are thinking and why!

Figure 6.5 Posters of the features with their explanations.
6.2.3 Dialogue/Communication Support

To encourage collaborative talk, I developed a talking-reminder. The logic behind the mechanism to remind students to talk is that if students add many blocks with very little pause, they likely were not discussing these changes with their partners. To compute a threshold for number of blocks added, I calculated the average number of times students added blocks to the coding area within a minute based on existing logs ($N=26$) collected during a prior study on pair programming with the same age group. The mean was two blocks per minute. Then, I set the threshold two standard deviations above that average ($2 \times SD = 5.5$). If the students reached the threshold and there had not been any popups about switching or talking within the past minute, then the talking-reminder would pop up. Figure 6.4 shows the talking-reminder pop-up.

6.3 Pilot Study

The participants in this study were 4th and 5th grade students in a suburban elementary school in the southeastern United States. In total, there were 14 and 9 children in the 4th and 5th grade classes, respectively. 11 students (2 girls) in the 4th grade class and all students (6 girls) in the 5th grade class had consent and assent to participate in data collection. Both were English Language Arts classes for academically or intellectually gifted students. Eight out of nine 5th grade students also participated in a previously held intervention when they were in 4th grade.

The students participated in a four-day Computer Science curriculum that used poetry as a coding domain. For example, when learning about input and conditionals, the students had to write a program to give a rhyme when the user pressed ‘r’ and an alliteration when the user pressed ‘a’. The curriculum covered algorithms, input, conditionals, loops, and broadcasting. The lessons were approximately one hour long and were taught by the myself. The students worked in pair programming teams for all of the activities. In both classes, the pairs were formed by the teacher and remained stable for the entire study. Each pair was randomly assigned to one of the four prototypes: time-based role switching, task-based role switching, time-based role switching with talking-reminder, task-based role switching with talking-reminder. The students were not informed of which prototype they used until the focus group interviews. During the study, I collected audio and video data of their pair programming process.

After completing the final lesson, consenting students participated in focus group interviews that lasted an average of 25 minutes. The researchers asked a list of questions and were encouraged to follow up on answers the students provided. The protocol and interview questions are in Appendix A. The students were assigned to focus groups based on the prototype they used. The focus group information is displayed in Table 6.1.
Table 6.1 Focus group information. *The data for group 5 did not save correctly and could not be analyzed.

<table>
<thead>
<tr>
<th>Group Num.</th>
<th>Grade</th>
<th>Girls</th>
<th>Boys</th>
<th>Approx. Duration in Mins</th>
<th>Switch Con.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>32</td>
<td>Task</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>28</td>
<td>Time</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>24</td>
<td>Time</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>17</td>
<td>Task</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>*</td>
<td>Task</td>
</tr>
</tbody>
</table>

6.4 Findings

Here, I present the findings from the students’ responses to the interview questions. The quoted responses were taken from transcriptions of the focus groups.

6.4.1 Focus group question 1: What went well with the reminder to talk and the reminder to switch?

Students expressed a wide range of responses to this question. Often mentioned was balance or fairness with regards to role-switching. Overall, the students found that this feature allowed both partners to experience both roles. One student stated, “Um switch role-roles one was so that everyone can have a turn.” Another student said, “I liked how everyone got a chance to do both things and we both had fun working together. You know one person would be navigator and the other person would be driver.” One student enjoyed that the switch happened regularly. “I think it went really frequently and that was good.” The students using the time-based switching-reminder appreciated the fact that they got an equal amount of time in each role. One student stated, “And I liked how everybody got like an even amount of time.” Although the students enjoyed driving the most, some also enjoyed having a break from driving, “because their hands were getting tired.”

The students felt that having this feature helped to reduce the amount of arguing that occurred and helped satisfy a desire for agency, “So nobody was like fighting” while they used the feature. Another student felt that there were fewer “two-driver” situations where both students were trying to make changes at once, “I like- I like how we didn’t - two people didn’t have to stick their hands in at once and try to type.”

During this study, the talking-reminder did not appear often because the students rarely added enough blocks in a minute to reach the threshold that triggered the talking-reminder. In the focus groups, I discovered that the reminder only appeared for one group during the last day of the study. This group said, “It helped us to actually...to start talking more.”
6.4.2 Focus group question 2: What did not go well with reminder to talk/reminder to switch?

While switching prompts seemed to enhance the sense of fairness and gave all students agency in enacting the programs, it was not a solution to all conflict. Although some students felt that the feature reduced arguing, others still had disputes with their partners about switching roles. When asked about what did not go well, one student answered, “We were fighting. We were fighting.”

The majority of students expressed enjoy the driving role more than the navigator role, and some considered being the navigator a downside of the switching-reminder, “Cause the driver had to switch to navigator and the navigator was happy because they had switched driver.” One student expressed feeling impatient when he was not driving, “When I was navigator, I was getting really impatient.” Their frustration with switching roles also became a problem when students tried to prevent switching from happening. One student said, “He [partner] hid it... he like hid it so like I couldn’t see what was it because I was next.”

Regarding the task-based switching reminder, many students were dissatisfied with the time variability of task-based switching. One fourth grade student who used the task-based switching-reminder prototype stated that, “It shouldn’t be every single task, I don’t think...I think it should be more on a time limit.” He continued on to suggest, “Say you’re doing it faster than five minutes. That’s time limit. And like you’re already done with the task...Then you can already switch it to the task. It can be two different things.” Thus, some students felt that the time was too long whereas others felt that it was too short. This was also related to the students who felt tired of driving after a while and wanted a break.

Other comments included one student who disliked that the icon was of a boy and a girl. He suggested making the icons gender neutral, “shouldn’t you make someone neutral. So like if it’s all boys then we can then give it as a boy. If it’s all- if it’s a girl, then they can think of it as a girl.”

6.4.3 Focus group question 3: What would you improve about the reminder to talk/reminder to switch?

In a recurring theme, students who used the task-based switching-reminder prototype experienced being in each role for an unequal amount of time. One student said, “The last time, me and my partner um kind of took us a while for one of-one of the tasks.” Since one of the tasks took longer, they were in their respective roles longer.

Perhaps not surprisingly, the theme of fairness and balance returned with the time-based switching. When queried about changes the students would like to see to the switching-reminder, some of the students using the task-based switching-reminder prototype suggested it be time-based and not task-based to ensure equal time in each role. However, students did recognize the advantages of switching at the end of tasks rather than at an arbitrary time point. One group
of students supported the idea of reducing the time each programmer gets in each role from 12 minutes to 10 minutes as they felt this allowed for both more frequent switches and sufficient time to complete “one decent code.”

Students offered additional suggestions on how to improve the switch feature. One group of students proposed alerting the programmers to how much time remained before they would switch roles. A lively debate among the students resulted in some supporting the idea of having a timer available, but not always visible, “Maybe it’s one of those things where you can push a button and it will show the time and then like when you let go of it it doesn’t.”, with others advocating for a pop-up notification of time left before switch, “Like a little message in the corner or something that is like hey, heads up like get to an ending point.”.

Again with regards to fairness, the final recommendation the students offered was to change how roles are assigned for the first driver. One student encouraged me to make it “like a game to choose... like a number one to ten... Because it was unfair ’cause every single time I had to be navigator [first].”

Regarding changes that mainly affected the talking-reminder, the students suggested a reminder to talk after the students completed the final task of the lesson. They said that this would help the students talk about their thinking process, “Talk to like- how did I think about that.” as well as work through any parts that they did not understand, “Even if they still don’t understand like. Oh, this was what I was thinking.”.

Some student feedback related to both the switching-reminder and talking-reminder. The first was to have the talking-reminder displayed before the switching-reminder, “So that way your partner can um go along with what you’re thinking on the program, when they switch to driver, if they do” and the driver can, “show [their] thinking.” The students also felt that receiving a warning before they were told to switch would be a helpful change since one of the pairs reached a conflict when a partner did not see the switching-reminder. A suggestion was to use it as a, “Heads up and like talk with your partner.” This was interwoven with the discussion about having a pop-up notification of the time remaining before switching.

The reminders were displayed within the system as icons, and accompanying posters were hung in the room to elaborate on those icons (Figure 6.5). Some students made suggestions regarding the wording of these posters. One of the wording changes the students suggested was to tell the students to stop working on their code before talking, “Maybe it should also say like um stop what you’re doing and then talk instead of talk while you’re still doing it.” Another wording change the students suggested was to tell the students to talk specifically about programming, “And just tell what you’re thinking. And they have two things on their mind. They don’t know what to talk about, this or that.” The final wording change that was suggested was to tell the students to talk about what they are thinking and doing, “Um, it says, just what you’re thinking,” one student said, “but...What if you’re um doing it?”
6.4.4 Focus group question 4: What do you think would have been different without the reminder to talk/reminder to switch?

Reinforcing earlier themes, the students felt that without the switching-reminder, there would have been more conflicts over the roles, “You would argue for the computer.”, “It would be more arguing.”. With regards to fairness and balance, they also felt that there would be instances where the students stay in the same roles the entire time, “I would just be driver the whole time.”. One student felt that both problems would occur, “Yeah, people more-would fight more often. One person might not even get the chance to be navigator or driver.” Even if both students had the chance to drive, one student was still concerned about equal access to the roles, “One would be on it way more than the other and it would be unfair.”

6.4.5 Focus group question 5: Did you ever decide not to follow the reminder to talk/reminder to switch?

When asked about how often students followed the reminders, every student in two of the focus groups said they followed them all the time, and one pair in each of the other two groups said they disregarded the reminders only once. From my observations and the focus group answers, I found that there were at least two students who tried to hide the switching-reminder from their partners. One 4th grade student turned the laptop away from his partner and put his hand up to hide the screen, which was apparent in the video from that session. Although he appeared to be laughing, his partner needed to take the laptop away in order to enact his role and mentioned this problem in the focus group interviews. I had observed this behavior in previous studies as well. Another 5th grade student saw the reminder when her partner was reading a poem book for the task, and clicked “Ok” without alerting her partner; this interaction was witnessed by myself during the study. In the focus group, the partner stated that he only knew it was his turn when he saw that the ID that was displayed was his, “I was looking through one of the books...I look over and then it was like hey it’s my turn already.”

Regarding the reminder to switch, the reason for not following was that they wanted to finish something first before following the talking-reminder or switching-reminder. As one of the students said, “I’m gonna be confessing. That sometimes you don’t want to give it up when you are the driver because you like- you want to finish something.”

In terms of the talking-reminder, one group said they did not follow it once because they were already talking when the talking-reminder popped up. In that instance, the reminder popped up twice and they said “It went like tell then when we’re talking it says that tell so then we had to click pause to press ok.”
Focus group question 6: We programmed NetsBlox to tell you to switch after every task. We had other students switch every 12 minutes. Which do you think you prefer and why?

The opposite was asked about the students in the time-based conditions. In alignment with a synthesis of earlier responses, when asked to choose their preference, the majority of the students chose time-based switching because it allowed for equal time in each role (“So that you could get the same amount of time.”). Some students preferred time-based switching even if they did not get equal time, stating that they would like it, “As long as I’m the one going three times because I’m the driver.” In their discussion, they divided their class time of 60 minutes by the current time in each role (12 minutes) and realized that this would result in an unequal amount of time. However, they did not account for instruction time, which reduced the time they spent on the computer. They also stated that time-based role switching “gives a chance for both partners if you don’t finish [a task].” Another student felt the same sentiment and said he wanted time-based switching because “[my partner] was the driver the entire activity because we didn’t complete the task.”

One student stated that he would prefer task-based role switching because “You know when you’re gonna get it. You know like when they finish it.” For him, it was important to be able to predict when he would need to switch.

Again echoing the desire for fairness, many students were worried that their partners would purposefully extend a task in task-based role switching to prevent them from driving. One student said, “Cause one person could purposely take a long time on their task and the other person could be like oh I’m gonna get this done quick.” Another student who expressed the same concern, said a partner could “mess around with it and not do it and then it would and then when the time was up, nobody else have got a turn.” One student who used the task-based switching-reminder prototype experienced this and described it in the focus group interviews, “He was the driver because we didn’t complete the task. And like he took over the driver for the entire time and I was just sitting there bored to death.”

Focus group question 7: (5th Grade Groups only) What was different between your pair programming experience last year and your pair programming experience this year?

During the focus group interviews, some students compared the features in NetsBlox to the analog equivalent of the features, which they had experienced the previous year. For instance, students compared the switching-reminder to the teacher timing them and calling out when it was time to switch. They preferred seeing the switching-reminder to having the teacher remind them when to switch, “I like the switch roles thing.” because it “[t]ells you exactly when [to switch].” The students liked that the computer prompted them when it was time to switch, “I liked that this year the um teacher didn’t have to call out switch now every few minutes.”
Perhaps alluding to issues of balance again, they also felt that, “It [switching-reminder] really kept track of time better.”

6.4.8 Focus group question 8: What went well with using the task list?

The positive aspects of the task list focused on both the cognitive facilitation and the agency provided by the tool. Overall, the students enjoyed the task list feature because the task list showed the information they needed in a convenient location. As one student said, “You can easily look down and back...”

As a cognitive support tool, some students enjoyed the fact that the tasks are broken down for them, “I like this but has separate ones.” Some students thought that it helped them in case they forgot what to do, e.g., “It gave me something to do and it told us what to do in case we forgot what it was.”, “The task list is good because you don’t have to like remember the task.” Another student felt it helped them know what to do, “You wouldn’t know what to do.” One of the students just wanted some variety in the work she did in NetsBlox, “I just wanted to click something other than codes.”

Some students felt that the task list provided agency, as it allowed them to work at their own pace rather than having to wait for instructions from the teacher, “And then- when our task was over, we could just press complete and then go to the next task”, “And after that task you didn’t have to explain it to us, we already knew because we just hit complete go onto the next task and know what we do.”

The students compared this to having the tasks on the board/projector. The students enjoyed the fact that they could move at their own pace, they did not need to look up at the board to see their instructions, “And we don’t have to like turn around and stop focusing just to look at what we’re supposed to do.”, and they felt like they were given more responsibility, “I like that more cause I feel like-I feel like it makes you have more responsibility for your thing and I think that just makes kids feel special”. Another student discussed that when the task was projected on the board, there would be times when something “would also be kinda blocking [the board] so we couldn’t really see what we were supposed to do.”

6.4.9 Focus group question 9: What did not go well with using the task list?

Negative feedback on the task list pointed to some students not liking having to complete the tasks in order, “I didn’t like it where you had to complete the one task before you go to the next. Because I like to know what I’m going to do after.” Some students thought that they could not go back to the previous task, “So he pressed task two but we couldn’t go back to task one because it was already completed so I had to go to the sprite. The other sprite.”

One student said he had trouble understanding the tasks sometimes and a video would be helpful, “I think if you put like a video or like a picture down there. It would have been nice.” Their logic is that the visual could provide an example of what the stage and sprites should look like when the task is complete.
In a similar vein, some students did not like that the interface did not create a new project for the next task. One student elaborated that the stage was still the same after they clicked on the next tab, “It’s how um when you do one task, and you go onto the next, the sprite is still gonna be there.”

6.4.10 Focus group question 10: What would you improve about the task list?

Addressing perhaps both the desire for greater agency and better cognitive supports, some students said that they wanted to be able to see all the tasks at once, “I would change it so that you can freely look at the tasks at what you’re supposed to do-what we do when we’re finished.” In order to do so during the study, some students clicked all the way through the list to try to see what each task was, “One time I just wanted to see what it said so I just kept hitting complete.”

Some suggestions focused on cognitive offloading through automation. The suggestion was made by the students for the list to automatically go to the next task after clicking the complete button for one task, “Like, it had to more be like you have to click on the next slide instead like once you’re done and you press complete...Well, I didn’t really like that you had to like go over [to the tab] and click it.” When the researcher asked if the students would like to switch automatically to task two after finishing task one. The student confirmed that he would like that change.

One student wanted NetsBlox to check their code after they clicked on the complete button, “Um, when they click completed...the computer or the browser would verify if they had completed it or not. Like it would verify and then say yay you completed it.” Another student requested that NetsBlox start new project after the student clicks complete, “So maybe if you would do this like maybe when it presses...When you press completed it would like go back to normal.” In addition, one student suggested adding highlights for the important parts of instructions, “I would highlight the w- the important parts.” Finally, the students wanted NetsBlox to prevent them from starting a task until they press complete on the previous tasks, “You want them to not be able to start another task um until they press-press complete on the first task.”

Many of the other suggestions focused on usability and aesthetic improvements. A student felt that there was a lot of empty space in the task list area, “It’s like pretty boring so it could like show you video of like what you’re really supposed to do like a little screen over where the sprite is.” This was similar to a suggestion made by a peer in a different group.

One student also mentioned that it may be beneficial for students to have the option of the tasks being read out loud.”They, they might want to like listen to it like. There’s a listen tab instead of you reading it.”
6.5 Discussion and Implications

The students in this study participated in collaborative programming activities, covering movement, conditionals, loops, and broadcasting over the course of four days. During these activities the students were exposed to a switching-reminder and a talking-reminder, which were designed to support their collaboration. Additionally, they used a task list to keep track of where they were in the activities that needed to be completed. I conducted focus group interviews to understand how the students felt about the features, how they used them, and what they thought should change about them.

Overall, I found that students expressed a mixture of positive and negative feelings about the features; however, they clearly preferred having the features integrated into the software than to have the analog equivalent of those features delivered by the teacher. For the most part, the students followed the reminders, and they gave many suggestions on how they would improve each feature.

A prominent theme that was visible throughout the focus group interviews was the interrelated ideas of agency and fairness. That is, there was a strong desire on the part of students to be active participants in the construction of creative programming solutions to the problems posed by their teacher, and that students deserved equal access to actively participate in their creation. The students found it important that everyone should have a turn and that they should get equal time in each role. This was emphasized in every focus group by multiple students. Active participation in a collaborative learning activity has been shown to improve students’ learning [Chi09]. However, I found that children often participated unequally in a pair programming relationship and that students recognize this imbalance. In their interactions, disagreements on code changes and who has control of the computer often led to the escalation of conflicts into a physical form. This trend has also been discovered in previous research with children during pair programming [Dei16; LS15; MI18; Sha14; Tsa18a] as well as research in other contexts that involve collaboration using a single access point [Mar08; SN03]. These findings suggest that designers of collaborative educational environments should design learning support tools that ensure the students have equal chance to participate in meaningful and substantive ways. Additionally, researchers should ensure that the children are aware of that design choice.

Students also mentioned wanting to have more influence on who drove first. My observations in this study and previous studies showed that when given the autonomy to choose their order, many pairs struggled to do so in a way that both students were happy. This topic also arose in the focus groups when a student stated that he always navigated first. This is not the first time I encountered students of this age group expressing an interest in independence. My work with various collaborative paradigms in programming revealed that a benefit students found from one paradigm was their ability to have more control over their code. Based on these results, I conclude that during the design of of collaborative support features, researchers should be
careful to maintain a balance of autonomy/independence for the students, and making sure they still have equal opportunity in each role.

Despite the general consensus that equal participation is important, there were students who expressed the desire to have more time driving than their partner, such as a 4th grade student who said that he would be alright with an unequal amount of time in each role as long as he would have the chance to drive more. From my past experiences, I found that it is common to see students’ actions reflect a different attitude from what they believe to be best behavior. I found that students can often list good and bad collaborative behaviors; however, scaffolding is necessary in ensuring that good collaborative behaviors are followed. These results highlight the importance of designing environments that encourage and support students in holding themselves and each other accountable. Not only is this important in turn taking, it is important in terms of the students’ communication. Both in this work and my prior classroom experiences, I heard from students that their pair programming experience was challenging because there was arguing and their partners did not listen to them. Although one group of students felt that the talking-reminder helped them talk more, I must keep in mind how to support students in talking effectively.

Wanting to have more responsibility and having the autonomy to move forward with tasks at their own pace also shaped their response to the task list feature. There were differing reactions as to whether the tool met their needs for increased agency. Similarly, the tool had not yet met its full potential to provide cognitive support for guiding their work, as reflected by their critiques and suggestions. However, on the whole, the task list provided a simple tool to help students mark their completed tasks.

Future collaborative support tools must work around more than just students sharing one computer. Although my studies and findings are centered around students working on a single device, roles and distribution of work can also be a factor in multi-computer collaboration. This is an important open area of research in designing collaborative learning support tools that can lead to empowering children as they learn together.

Although I designed these collaborative support features specifically for 1-computer pair programming, my work can also inform the design of features in other collaborative contexts as well. Guided by the interrelated themes of agency and fairness, my design and findings can help inform instructors and researchers who desire to support students in a collaborative paradigm where they have to take turns. Future improvements will also inform those in the field about how to encourage students to communicate with their partners while they are collaborating on the computer.

From a practical standpoint, the students articulated an overall positive response to using these tested features. In fact, they felt that there was a difference between seeing the switching-reminder and having the teacher time them. This finding coupled with the chance of lessening instructors’ burdens is an argument for using that feature in more collaborative learning contexts. The rich, thoughtful discussion of time versus task-based switching points to the nuanced
response that students need such a feature, perhaps not surprising since it strikes at the center of this agency/fairness nexus. Taken together, these results highlight the importance of continuing to research the many ways in which I can support equitable and empowering collaborative learning opportunities between children.
Because research on how elementary students collaborate in computer science has only emerged in the past six years, there has been little work on how we can use adaptive environments to support them. Chapter 6 detailed the first iteration of adaptive support features I built for upper elementary students. The features were a role switching reminder and a talking reminder. These features were piloted with 4th and 5th grade students and I conducted focus group interviews to better understand the students’ experiences with those features. My analysis of the focus group interviews revealed areas of promise and improvements as students expressed both positive and negative feedback about the features. Additionally, the students made suggestions that led me to the improvements that will be described in this chapter. In this chapter I also analyze the data to discover whether there is evidence that the features had the intended benefits by reviewing the students’ dialogue and conflict patterns. Finally, I analyse the focus group interviews that I conducted after this study iteration. The research question I answer in this chapter and my hypotheses are listed below.

RQ2: To what extent do the following features in an adaptive programming environment support upper elementary students’ collaborative problem solving process?

- A built in timer for switching control
- A built in tasklist for switching control
• A reminder to talk to their partner

H2.1: An adaptive programming environment that offers talking reminders will lead to an increase in students’ reported enjoyment of the collaborative process when compared to students who use the programming environment without the scaffold.

H2.2: An adaptive programming environment that offers talking and switching reminders will reduce the number of conflicts that arise between students compared to students who use the programming environment without the scaffolds.

H2.3: An adaptive programming environment that offers talking reminders will foster increased use of collaborative dialogue strategies more frequently compared to students who use the programming environment without the scaffold.

7.1 Collaborative Support Tool Refinement

Based on the focus group analysis described in the preceding chapter, I made the following refinements to the support features. First, I created a warning that occurs one minute before the time-based switching reminder. This was a feature suggested by some of the students. Although it was met with mixed responses in the group where it was suggested, given the fact that some students also mentioned wanting to finish what they were doing before switching roles, I decided to test the warning. The warning popup is shown in Figure 7.1. Second, I lowered the threshold for the talking reminder to one standard deviation above the average number of blocks that are added in a minute instead of two standard deviations above. I made this decision because only one pair of students received the talking reminder and they only saw it once. Third, I decided to take the students’ suggestion about allowing them to view any of the tasks at any point in time. My goal was to allow the students to have more autonomy and encourage them to plan ahead. Additionally, I implemented the students’ idea to automatically move to the next tab once the student clicked complete on the previous task. Finally, I added the warning icon along with an explanation to the poster shown to the students. Additionally, I changed the wording of the poster as suggested by the students. The description for the talking reminder was changed to “Make sure you keep talking about your work and what you are thinking.” And right below the message, I added subquestions as guidance, “Ideas? Questions? Plans? Changes?” This was done to clarify any confusion the students had and to give them more ideas of what they should discuss.
7.2 Methods

7.2.1 Research Context and Data Collection

The participants in this study were 4th and 5th grade students in an urban charter elementary school in the Southeastern United States. In total, there were 76 students in the 4th and 5th grade of this school. I worked with 4 classes of students. There were 19 students in each class, however, the classes in each grade were co-managed by two teachers. In total, 61 students consented and assented to participate in data collection.

The students participated in a Computer Science curriculum that has been piloted and refined in previous studies [Tsa20; Zak19] and described in Chapter 5. The curriculum covered algorithms, input, conditionals, loops, and broadcasting. The lessons were approximately one hour long and were taught by the Connected World teacher. Connected World is a class all of the 4th and 5th grade students took. The course encompassed many types of lessons and activities, including some Scratch programming before the students participated in the study. However, the teacher stated that she had not taught programming before and the Scratch activities were exploratory. The students used pair programming for all of the study activities. Each pair was randomly assigned to one of the four prototypes: time-based switching reminder; task-based switching reminder; time-based switching reminder with talking reminder; task-based switching reminder with talking reminder (Table 7.1). The students were not informed of which prototype they used until the focus group interviews. The pairs were formed by the teacher and remained stable after the first week of the study. Some pairs encountered problems in the first week and the teacher rearranged them afterwards. Due to the rearrangement, I did not have an equal number of students in each condition.
Table 7.1 Focus group information.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Number of Pairs</th>
<th>Grade</th>
<th>Girl-Girl</th>
<th>Boy-Boy</th>
<th>Girl-Boy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time No Talk</td>
<td>6</td>
<td>3 4th; 3 5th</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Task No Talk</td>
<td>7</td>
<td>3 4th; 4 5th</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Time and Talk</td>
<td>8</td>
<td>6 4th; 3 5th</td>
<td>6</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Task and Talk</td>
<td>8</td>
<td>4 4th; 4 5th</td>
<td>1</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

During the study, I collected audio and video data reflecting the students’ pair programming processes. While I had initially planned on a six-week study, weather conditions and COVID-19 only allowed most students to participate for five weeks and one 4th grade class only had four weeks of the programming class.

The students completed pre- and post- surveys and assessments. While there were 76 total students, 61 of whom consented to the study, not all consenting students were present during the pre-assessment and survey days. The post-assessment and surveys were given to the students after the classes had moved online due to COVID. The students completed pre- and post-assessments of their computer science concept knowledge. This assessment also recorded demographic and technology/programming experience (Appendix C). Thirty students completed the pre- and post-assessment. Additionally, they completed pre- and post- computer science attitudes (Appendix E) and collaboration attitudes surveys (Appendix D). Twenty-nine and 28 students completed the pre- and post- CS and collaboration attitudes surveys, respectively. The computer science attitudes survey has been validated with this age group [Van20] and the computer science concepts assessment is in the validation process. While the collaboration attitudes survey has not been validated, we have used and refined this survey with the same age group in the past.

After completing the lessons, consenting students participated in focus group interviews that lasted an average of 23 minutes. The researchers asked a list of questions and were encouraged to follow up on answers the students provided. The protocol and interview questions are in Appendix B. The students were assigned to focus groups based on the prototype they used and I prioritized pairs that had fewer absences. The focus group information is displayed in Table 7.2. Given our time constraints, I was only able to conduct the interviews with six 5th grade and three 4th grade groups. The groups consisted of three to four students.

7.2.2 Sampling Data

Due to the limitations on the time of annotating dialogue and video data, I sampled 12 pairs from my full dataset. My goal was to have a sample that had an equal number of 4th and 5th grade pairs, and an equal number in each switching condition. Additionally, I sought to have an equal number of girl-girl, boy-boy, and girl-boy pairings to increase the generalizability of my findings. My next criteria was that the videos should be auditorily clear enough for transcription. I also sought videos where I could easily view both partners a majority of the
Table 7.2 Focus group information.

<table>
<thead>
<tr>
<th>Group Num.</th>
<th>Grade</th>
<th>Girls</th>
<th>Boys</th>
<th>Approx. Duration in Mins</th>
<th>Switch Con.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5th</td>
<td>2</td>
<td>2</td>
<td>28</td>
<td>Time</td>
</tr>
<tr>
<td>2</td>
<td>5th</td>
<td>1</td>
<td>3</td>
<td>29</td>
<td>Task</td>
</tr>
<tr>
<td>3</td>
<td>5th</td>
<td>1</td>
<td>3</td>
<td>19</td>
<td>Time</td>
</tr>
<tr>
<td>4</td>
<td>5th</td>
<td>3</td>
<td>1</td>
<td>18</td>
<td>Time</td>
</tr>
<tr>
<td>5</td>
<td>5th</td>
<td>3</td>
<td>1</td>
<td>26</td>
<td>Task</td>
</tr>
<tr>
<td>6</td>
<td>5th</td>
<td>0</td>
<td>4</td>
<td>20</td>
<td>Task</td>
</tr>
<tr>
<td>7</td>
<td>4th</td>
<td>4</td>
<td>0</td>
<td>17</td>
<td>Time</td>
</tr>
<tr>
<td>8</td>
<td>4th</td>
<td>3</td>
<td>1</td>
<td>12</td>
<td>Task</td>
</tr>
<tr>
<td>9</td>
<td>4th</td>
<td>3</td>
<td>0</td>
<td>36</td>
<td>Time</td>
</tr>
</tbody>
</table>

Table 7.3 Pairs in each condition

<table>
<thead>
<tr>
<th></th>
<th>Task</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dialogue</td>
<td>7, 11</td>
<td>2, 3, 10</td>
</tr>
<tr>
<td>No dialogue</td>
<td>1, 5, 6, 9</td>
<td>4, 8, 12</td>
</tr>
</tbody>
</table>

session. Finally, I sought to have about 40 minutes of video for each pair and for six pairs, I needed two videos, one lesson each. In total, I sampled 18 videos of 12 pairs of students. Table 7.3 displays which pairs are in which conditions.

7.2.3 Annotating Data

For dialogue act annotation I used the same annotation scheme that was described in Chapter 5. This annotation scheme was inspired by previous dialogue annotation schemes used for pair programming research [Rod17b; Tsa18a; Buf17; Ruv16; Gra94]. Many of the moves we are concerned with in this current work, such as explanation, question, antagonization, and directive/suggestion, were used in those schemes, while self-explanation, uptake, and praise were not included in the prior work. We included uptake and self-explanation because of the prominent role they play in effective collaboration and learning [Chi94; CB03]. We included praise because we noticed it in previous work as a counterpoint to antagonistic dialogue between young learners, and we believe it plays a role in indicating positive collaborative dynamics. Additionally, I omitted utterances that were directed at the teacher, study facilitators, or peers that were not their partners. I marked utterances that were not clear enough in the audio to transcribe reliably, marking these as inaudible (total of 159). In total, 10,057 utterances were tagged in the corpus.

For the conflict annotation, I used the same annotation schemes that were presented in Chapters 4 and 5. With those annotation schemes, I tagged the students’ conflicts and the types of conflicts that they engaged in. In total, the students in this dataset engaged in 131 conflicts. 84 (64%) were about the task, 22 (17%) were about control of the computer, and 24 (18%) were about partner contributions/roles.
Table 7.4 Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Chapter</th>
<th>Pairs/Students</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conflicts and Dialogue Acts 1</td>
<td>7</td>
<td>12 pairs</td>
</tr>
<tr>
<td>Assessment</td>
<td>7</td>
<td>30 students</td>
</tr>
<tr>
<td>CS attitudes</td>
<td>7</td>
<td>29 students</td>
</tr>
<tr>
<td>Collaboration attitudes</td>
<td>7</td>
<td>28 students</td>
</tr>
<tr>
<td>Conflicts and Dialogue Acts 2</td>
<td>5</td>
<td>12 pairs</td>
</tr>
<tr>
<td>Conflicts</td>
<td>4</td>
<td>6 pairs</td>
</tr>
</tbody>
</table>

In an initial review of the video data beyond my 12 pairs, I found that the majority of students did not verbally acknowledge the talking reminder. Instead, they often clicked “ok” immediately and went back to their work. Sometimes they even left the reminder up and continued working. For those that did acknowledge it, they often talked about the instructions but did not follow them. I developed a coding scheme to formally analyze how the students reacted to the reminders. First, I labeled whether they verbally acknowledged the talking reminder (acknowledged and did not acknowledge). Then for the switching reminder, I labeled whether they switched (switched, did not switch, willingly switched later, or teacher had them switch). I decided on “willingly switched later” because during the focus groups, students mentioned finishing up activities before switching. And I used the word “willingly” because sometimes the teacher noticed and required them to switch.

Finally, for the focus group interviews, I used the same annotation scheme as the one we reported in chapter 6 [Sal15].

7.2.4 Datasets Used

In the following subsections, I present the findings in relation to my hypotheses. When presenting these findings, I will be referring to six datasets. Two were presented in previous chapters and four will be described in this chapter. Datasets “Conflicts and Dialogue Acts 1”, “Assessment”, “CS attitudes”, and “Collaboration attitudes” are the dataset described in this chapter, Dataset “Conflicts” is the dataset described in chapter 4 and Dataset “Conflicts and Dialogue Acts 2” is the the dataset described in chapter 5. Table 7.4 displays these datasets, their chapter, and the n.

7.3 Results

I calculated the descriptive statistics for each pair’s dialogue and conflicts. Table 7.5 displays those results. The number of utterances spoken by the pairs ranged from 542 to 1398 and the average was 838. The majority of the pairs used uptake and praise less than 1% of the time, and used antagonization less than 2% of the time. I identified a total of 131 conflicts in my videos.
Table 7.5 Descriptive statistics of the conflicts and dialogue tags broken down by pair.

<table>
<thead>
<tr>
<th>Pair ID</th>
<th>Pseudonyms</th>
<th>Utterances (#)</th>
<th>Con. (#)</th>
<th>Es (%)</th>
<th>EO (%)</th>
<th>D (%)</th>
<th>Q (%)</th>
<th>U (%)</th>
<th>P (%)</th>
<th>A (%)</th>
<th>OR (%)</th>
<th>OU (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Penny &amp; Kendra</td>
<td>669</td>
<td>8</td>
<td>19.28</td>
<td>8.22</td>
<td>21.08</td>
<td>9.42</td>
<td>0.15</td>
<td>1.20</td>
<td>1.49</td>
<td>30.49</td>
<td>8.37</td>
</tr>
<tr>
<td>2</td>
<td>Surya &amp; Evan</td>
<td>713</td>
<td>13</td>
<td>20.34</td>
<td>9.12</td>
<td>17.39</td>
<td>5.89</td>
<td>0.42</td>
<td>0</td>
<td>1.54</td>
<td>26.79</td>
<td>17.67</td>
</tr>
<tr>
<td>3</td>
<td>Paula &amp; Mary</td>
<td>944</td>
<td>8</td>
<td>20.87</td>
<td>6.25</td>
<td>20.34</td>
<td>7.20</td>
<td>0.95</td>
<td>0</td>
<td>0</td>
<td>41.95</td>
<td>2.01</td>
</tr>
<tr>
<td>4</td>
<td>Molly &amp; Noah</td>
<td>1002</td>
<td>7</td>
<td>21.06</td>
<td>6.99</td>
<td>21.16</td>
<td>12.38</td>
<td>0.10</td>
<td>0</td>
<td>1.20</td>
<td>26.25</td>
<td>9.18</td>
</tr>
<tr>
<td>5</td>
<td>Lincoln &amp; Daniel</td>
<td>1398</td>
<td>24</td>
<td>15.74</td>
<td>8.08</td>
<td>13.95</td>
<td>7.73</td>
<td>0</td>
<td>0.14</td>
<td>0.93</td>
<td>25.97</td>
<td>24.96</td>
</tr>
<tr>
<td>6</td>
<td>Annie &amp; Larry</td>
<td>571</td>
<td>13</td>
<td>14.01</td>
<td>13.49</td>
<td>19.96</td>
<td>8.93</td>
<td>0</td>
<td>1</td>
<td>1.58</td>
<td>29.25</td>
<td>11.03</td>
</tr>
<tr>
<td>7</td>
<td>Ellen &amp; Marshall</td>
<td>1072</td>
<td>14</td>
<td>17.26</td>
<td>9.51</td>
<td>26.21</td>
<td>6.72</td>
<td>0.28</td>
<td>0</td>
<td>0.75</td>
<td>29.85</td>
<td>9.42</td>
</tr>
<tr>
<td>8</td>
<td>Charlotte &amp; Ariel</td>
<td>697</td>
<td>9</td>
<td>25.25</td>
<td>9.90</td>
<td>22.81</td>
<td>6.60</td>
<td>0</td>
<td>0.57</td>
<td>0.72</td>
<td>22.09</td>
<td>9.18</td>
</tr>
<tr>
<td>9</td>
<td>Abigail &amp; Angie</td>
<td>842</td>
<td>13</td>
<td>21.63</td>
<td>10.10</td>
<td>33.37</td>
<td>5.46</td>
<td>0</td>
<td>0.36</td>
<td>0.12</td>
<td>28.74</td>
<td>0.12</td>
</tr>
<tr>
<td>10</td>
<td>Aria &amp; Rachel</td>
<td>900</td>
<td>13</td>
<td>18.22</td>
<td>7.67</td>
<td>22.22</td>
<td>8.44</td>
<td>0.22</td>
<td>0.22</td>
<td>0.56</td>
<td>28.22</td>
<td>13.78</td>
</tr>
<tr>
<td>11</td>
<td>Jordan &amp; Martin</td>
<td>712</td>
<td>6</td>
<td>5.76</td>
<td>2.95</td>
<td>8.85</td>
<td>3.79</td>
<td>0</td>
<td>0.28</td>
<td>0.14</td>
<td>14.47</td>
<td>57.30</td>
</tr>
<tr>
<td>12</td>
<td>Amy &amp; Sarah</td>
<td>542</td>
<td>4</td>
<td>17.71</td>
<td>6.27</td>
<td>15.13</td>
<td>9.23</td>
<td>0</td>
<td>4.06</td>
<td>1.11</td>
<td>29.89</td>
<td>15.50</td>
</tr>
</tbody>
</table>

Table 7.6 Overall descriptive statistics of the conflicts.

<table>
<thead>
<tr>
<th></th>
<th>Num. Con.</th>
<th>Duration</th>
<th>Task</th>
<th>Control</th>
<th>Partner</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>131</td>
<td>44 min 56 secs</td>
<td>84 (64%)</td>
<td>22 (17%)</td>
<td>24 (18%)</td>
<td>40 (31%)</td>
</tr>
<tr>
<td>Avg</td>
<td>11</td>
<td>21 secs</td>
<td>7</td>
<td>1.8</td>
<td>2</td>
<td>3.3</td>
</tr>
<tr>
<td>Max</td>
<td>24</td>
<td>1 min 9 secs</td>
<td>13</td>
<td>4</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>Min</td>
<td>4</td>
<td>4 secs</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

with an average of 10.92 conflicts each. The smallest number of conflicts identified in a pair was 4 and the largest was 24. Of the 131 conflicts, 64% were about the task, 17% were about control of the computer, 18% was about partner contributions/roles, and 31% were about other problems, such as partner distractions and one partner wanting to ask for help from the teacher while the other did not.

For the students’ actions after the reminders, I found that in the 12 pairs, there were 33 switching reminders in response to which 26 (78.79%) “switched immediately”, 3 (9.09%) “switched later”, 3 (9.9%) “did not switch”, and 1 (3.03%) “teacher had them switch.” There were 23 talking reminders. 14 (60.87%) were “verbally acknowledged” and 9 “did not verbally acknowledge.” I also discovered that many of the “verbally acknowledged” were utterances of frustration or utterances like, “Shut up, we don’t care about that.”

As stated before, my theory of action is that the features will increase the equity of the relationships, decrease conflicts, and foster self-efficacy. The results presented here will present evidence to support or contradict my theory of action.

7.3.1 H2.1

H2.1: An adaptive programming environment that offers talking reminders will lead to an increase in student reported enjoyment of the collaborative process compared to students who use the programming environment without the scaffold.
In order to test this hypothesis I draw on the following data: pre- and post- collaboration attitudes surveys; pre- and post- CS attitudes surveys; and pre- and post- CS assessment (Datasets Assessment, CS attitudes, Collaboration attitudes). This data was all introduced in this chapter. I conduct the following comparisons:

**Talking reminder ≠ non-talking reminder for pre/post collaboration.**

I compare the pre- and post- collaboration attitudes between the talking reminder and non-talking reminder conditions and the results showed that there was not a statistically significant difference ($Z = 0.27$; ran a Wilcoxon Rank Sums Test to compare). The evidence does not support this hypothesis.

**Time-based switching reminder ≠ task-based switching reminder for pre/post collaboration.**

I compare the pre- and post- collaboration attitudes between the time-based role switching and task-based role switching conditions and the results showed that there was not a statistically significant difference ($Z = -1.7$; ran a Wilcoxon Rank Sums Test to compare). The evidence does not support this hypothesis.

**Talking reminder ≠ non-talking reminder for pre/post CS attitudes.**

I compare the pre- and post- CS attitudes between the talking reminder and non-talking reminder conditions and the results showed that there was not a statistically significant difference ($Z=-0.39$; ran a Wilcoxon Rank Sums Test to compare). The evidence does not support this hypothesis.

**Time-based switching reminder < task-based switching reminder for pre/post CS attitudes.**

I compare the pre- and post- CS attitudes between the time-based switching reminder and task-based switching reminder conditions and the results showed that there was a statistically significant difference ($Z = -2.11$, $p=0.04$; ran a Wilcoxon Rank Sums Test to compare) with the task-based switching reminder students reporting higher increase in enjoyment of CS. The evidence does not support this hypothesis.

**Talking reminder ≠ non-talking reminder for pre/post assessment.**

I compare the students’ pre- and post- assessment data between the students who received the talking reminder and the students that did not. The results showed that there was not a statistically significant difference ($Z = 0.88$; ran a Wilcoxon Rank Sums Test to compare). The evidence does not support this hypothesis.

**Time-based switching reminder ≠ task-based switching reminder for pre/post assessment.**

I compare the students’ pre- and post- assessment data between the time-based and task-based switching reminder students. The results showed that there was not a statistically significant difference ($Z = -0.87$; ran a Wilcoxon Rank Sums Test to compare). The evidence does not support this hypothesis.
7.3.2 H2.2

H2.2: An adaptive programming environment that offers talking and switching reminders will reduce the number of conflicts that arise between students compared to students who use the programming environment without the scaffolds.

In order to test this hypothesis I draw on conflict data. The analysis includes conflict data from Chapter 4, Chapter 5, and this chapter (Datasets Conflicts and Dialogue Acts 1, Conflicts and Dialogue Acts 2, Conflicts). I conduct the following comparisons:

Non-scaffolded pairs < talking and switching reminder pairs for number of conflicts.

I compare conflict data from all three chapters. On average, pairs without the talking or switching reminders (Chapter 4 and Chapter 5) encountered 6.5 conflicts in about 45 minutes. The maximum number of conflicts was 17 and the minimum was 0. On average, pairs with the scaffolds (Chapter 7) encountered 11 conflicts in about 45 minutes. The maximum number of conflicts was 24 and the minimum was 4. Additionally, when I ran a Wilcoxon Rank Sums Test to compare the pairs’ conflict numbers, the results were statistically significant (Z = 2.361, p=0.02). The non-scaffolded pairs had a lower number of conflicts than the talking and switching reminder pairs. The evidence does not support the hypothesis.

Non-scaffolded pairs ≠ talking and switching reminder pairs for number of control conflicts.

On average, pairs without the talking or switching reminders (Chapter 4 and Chapter 5) encountered conflicts that were about the control of the computer 31% of the time. On average, pairs with the scaffolds (Chapter 7) encountered conflicts that were about the control of the computer 17% of the time. However, when I conducted a Wilcoxon Rank Sums Test comparing the pairs’ usage, the results were not statistically significant (Z = 0.218). The evidence does not support this hypothesis.

Task-based switching reminder > time-based switching reminder for number of conflicts.

I compare the conflict data from Chapter 7 to find out whether switching condition affects the number of conflicts students engaged in. On average, time-based switching reminder pairs encountered 9 conflicts (maximum of 13 and a minimum of 4). On average, the task-based switching reminder pairs encountered 13 conflicts (maximum of 24 and a minimum of 6). Table 7.7 displays this information. The evidence supports this hypothesis.

Task-based switching reminder < time-based switching reminder for number of control conflicts.

I compare conflict data from Chapter 7 to discover whether students in the time-based switching condition have encountered fewer control conflicts than their task-based switching peers. On average, time-based switching reminder pairs encountered conflicts that were about the control of the computer about 30% of the time while the task-based switching reminder pairs encountered control conflicts about 8% of the time. Every time-based switching reminder
Table 7.7 Comparing conflicts between conditions. The top half is task-based role switching and the bottom half is time-based role switching.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Pair</th>
<th>Conflicts</th>
<th>Task Con.</th>
<th>Control Con.</th>
<th>Partner Con.</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>63%</td>
<td>0%</td>
<td>50%</td>
<td>75%</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>24</td>
<td>46%</td>
<td>13%</td>
<td>0%</td>
<td>46%</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>92%</td>
<td>15%</td>
<td>62%</td>
<td>38%</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>14</td>
<td>86%</td>
<td>7%</td>
<td>14%</td>
<td>14%</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>13</td>
<td>100%</td>
<td>0%</td>
<td>23%</td>
<td>0%</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>6</td>
<td>17%</td>
<td>0%</td>
<td>83%</td>
<td>83%</td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>13</td>
<td>62%</td>
<td>31%</td>
<td>0%</td>
<td>0%</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>88%</td>
<td>13%</td>
<td>0%</td>
<td>0%</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>14%</td>
<td>57%</td>
<td>14%</td>
<td>57%</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>44%</td>
<td>33%</td>
<td>0%</td>
<td>44%</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>13</td>
<td>69%</td>
<td>15%</td>
<td>8%</td>
<td>15%</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>50%</td>
<td>50%</td>
<td>0%</td>
<td>25%</td>
<td></td>
</tr>
</tbody>
</table>

pair had at least 1 and at most 4 conflicts about control while only three of the task-based switching reminder pairs had control conflicts; they had at most 3 control conflicts in a pair. The time-based switching reminder pairs had more conflicts about control than the task-based switching reminder pairs. The evidence does not support this hypothesis.

**Non-scaffolded pairs ≠ talking and switching reminder pairs for shorter conflicts.**

I compare conflict data from all three chapters to discover whether pairs with the talking and switching reminders for collaborative dialogue will engage in shorter conflicts than their peers. For non-scaffolded students (Chapter 4 and Chapter 5), the majority of the conflicts (86 conflicts, 74%) last 30 seconds or less, 21 conflicts (18%) last from 30 seconds to a minute, and 10 conflicts (9%) last a minute or more. For scaffolded students (Chapter 7), the majority of the conflicts (104 conflicts, 79%) last 30 seconds or less, 24 conflicts (18%) last from 30 seconds to a minute, and 4 conflicts (3%) last a minute or more. The times of the conflicts were similar between the non-scaffolded pairs and the pairs that used the talking and switching reminders. The evidence does not support this hypothesis.

**7.3.3 H2.3**

**H2.3: An adaptive programming environment that offers talking reminders will foster increased use of collaborative dialogue strategies more frequently compared to students who use the programming environment without the scaffold.**

In order to test this hypothesis I draw on dialogue data. This includes data from Chapter 5 and this chapter (Datasets Conflicts and Dialogue Acts 1 and Conflicts and Dialogue Acts 2). I conduct the following comparisons:

**Talking reminder ≠ non-talking reminder for collaborative dialogue strategies.**

I compare dialogue data from Chapter 7 between students who received talking reminders
Table 7.8

<table>
<thead>
<tr>
<th></th>
<th>ES</th>
<th>Q</th>
<th>U</th>
<th>P</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turn</td>
<td>19.30%</td>
<td>8.51%</td>
<td>0.02%</td>
<td>0.93%</td>
<td>0.95%</td>
</tr>
<tr>
<td>Dialogue and Turn</td>
<td>16.06%</td>
<td>6.41%</td>
<td>0.38%</td>
<td>0.12%</td>
<td>0.62%</td>
</tr>
</tbody>
</table>

Table 7.9

<table>
<thead>
<tr>
<th>Condition</th>
<th>Avg. Talk</th>
<th>Max. Talk</th>
<th>Min. Talk</th>
<th>Avg. ES</th>
<th>Max. ES</th>
<th>Min. ES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dialogue and Turn</td>
<td>867.8</td>
<td>1072</td>
<td>711</td>
<td>16.06%</td>
<td>20.72%</td>
<td>5.49%</td>
</tr>
<tr>
<td>Turn</td>
<td>728.4</td>
<td>1398</td>
<td>542</td>
<td>19.30%</td>
<td>23.80%</td>
<td>14.01%</td>
</tr>
<tr>
<td>None</td>
<td>679.83</td>
<td>1133</td>
<td>373</td>
<td>11.31%</td>
<td>16.13%</td>
<td>5.36%</td>
</tr>
</tbody>
</table>

and students who did not. The pairs are from the dataset described in Chapter 7 without the talking reminder uttered more self-explanations, questions, and praise. However, they antagonized each other more than the dialogue scaffolded pairs. The talking reminder pairs used uptake more. Table 7.8 compares the pairs from Chapter 7 with the talking and switching reminders to the pairs with only the switching reminder. This evidence does not support the hypothesis.

**Talking reminder + switching reminder and switching reminder > non-scaffolded for self-explanation.**

I compare the utterances and self-explanation between the talking reminder and switching reminder pairs, the switching reminder pairs, and the non-scaffolded pairs from Chapter 7 and Chapter 5 (Table 7.9). It appears that the students from both conditions in the dataset described in Chapter 7 used self-explanation more than the students in the dataset described in Chapter 5. This suggests that the switching reminder may help scaffold the student’s collaborative dialogue strategies. This evidence supports this hypothesis.

**Non-scaffolded pairs < talking and switching reminder pairs for balanced talk time.**

The work presented in chapter 3 and by other researchers [Sha14; Dei16] showed that pairs of students were often unbalanced in talk time. I am interested in whether students who received the talking and switching reminders had more balanced talk times than students who did not. For this, I compare dialogue data from datasets presented in Chapter 7 and Chapter 5. From Figures 7.2 and 7.3, it appears that the pairs with the reminders were generally more equitable in talk time than the pairs without the reminders. I then compare the most talkative partners’ proportion of talk (e.g. .67 for Pair 1 and .68 for Pair 2 of Figure 7.3 between the datasets and found that the difference was statistically significant ($Z = 2.87$, $p = 0.004$; ran a Wilcoxon Rank Sums Test to compare). This evidence supports this hypothesis.
**Figure 7.2** Scaffolding datasets Total Talk Distribution

**Figure 7.3** Non-scaffold dataset Total Talk Distribution
7.3.4 Qualitative Analysis and Findings

This subsection details my qualitative findings that allow me to build off the analysis I completed in the previous subsection. First, I investigated how the students behaved when they received each type of reminder. Then, I will investigate why the hypotheses were not supported and why it is possible that turn-taking scaffolding supported the students’ dialogue strategies.

7.3.4.1 Reminder actions

The majority of students followed the switching reminder immediately. Most instances seemed to involve the driver saying something similar to, “Your turn to drive,” and turning the laptop towards their partner. Other times, the navigator would immediately take the laptop from their partner. In one instance, a student in pair 8 took the laptop as her partner was still typing. She said, “Stop I can do it.” and pushed her partner’s hands away. Pairs 1, 10, and 12 each had one instance of follow later. With pair 1, they were getting help from myself and they switched after they saved their project. With pair 10, the students were each working out a math problem by hand to the same question they would ask the user. The navigator was still working and they switched about 7 minutes after the reminder came up. Finally with pair 12, the students were also asking the teacher for help and switched afterwards. Pair 2 had two instances of “Did not follow”. With pair 2, in one instance of the driver saying he wanted to finish something but they never ended up switching back. Then, in the other instance, the navigator did not say anything and the driver stated, “We don’t care.” Pair 12 had one instance where the driver and navigator switched before the reminder and stayed the same when the reminder showed. With pair 2, the teacher was present when the switching reminder showed and enforced switching between students. With this pair, they had switched halfway through one student’s turn and the other student did not want to switch when the reminder came up. In the focus groups, most students stated that they switched immediately, however, others admitted that they would wait to switch or switched on their own based on how they perceived their and their partner’s skills.

Most students seemed to follow the switching reminders with little protest and pairs 2 and 12 who had instances where they did not switch were in the time-based condition. These results may help explain the findings for the work completed for hypotheses H2.1.

Pairs that acknowledged the talking reminder seemed to express frustration through utterances such as “We don’t care” or “We are talking.” Pairs that did not acknowledge the reminder either clicked “ok” immediately or left it on the screen while they worked. Additionally, multiple students in the focus groups mentioned that the talking reminder came up too often. They include pairs that showed frustration or aggression towards the reminder. These findings may help explain why some results did not support hypotheses H2.1, H2.3.

It appears as though the students did not enjoy receiving the talking reminder and became frustrated when the reminder appeared while they were talking. Even in instances where they were not talking, some pairs showed aggression and hostility towards the feature.
7.3.4.2 Comparing collaborative dialogue and conflicts between switching conditions

To further investigate the data, I first analyzed the students’ dialogue strategies. The time-based switching reminder pairs used the strategies more than the task-based switching reminder pairs. Additionally, they engaged in antagonization more than the task-based switching reminder pairs. This is displayed in Table 7.10. Next, I investigated the interactions between pairs of students in each condition. Below, I presented one excerpt from each switching condition for each pairing type.

**Girl-Girl Examples**
For the girl-girl examples, I present excerpts from pairs 9 and 10. Pair 9 was in the task-based switching reminder condition. I sampled one video for each pair of students.

Pair 9 engaged in 12 conflicts; all of the conflicts were about the task and 3 were also about partner contributions/roles. They had no instances of uptake, 3 of praise, and 1 of engaging in antagonization. Angie and Abigail were pair 9. Table 7.11 displays an excerpt that shows how the pairs negotiated some of their differing perspectives. In this excerpt, the students disagreed about how difficult should the math question that they ask the user be. To come to an agreement, the girls went back and forth with different suggestions on a question until the very end where they settled on asking, “What is four times six?”

Pair 10 engaged in 13 conflicts, 9 of which were about the task, 2 were about control of the computer, 1 was about their partner, and 2 were about other problems. This pair exhibited 2 instances of uptake, 2 of praise, and 5 of engaging in antagonization. The conflict displayed in Table 7.12 occurred at the beginning of their session. In this conflict excerpt, the girls were receiving help from the researcher and when Rachel realized she was right, she stated that what the researcher said was the same thing she has been trying to tell her partner. Aria disagreed with Rachel. Rachel then expressed her frustration by saying “It’s mind blowing!” However, the pair moved on quickly and continued to work without seeming to agree on what happened before.

With these students, pair 9 (task-based) seemed to come to an agreement via negotiation whereas pair 10 (time-based) did not. This shows that the use of specific dialogue strategies is not enough for students to engage in productive conflicts even if they engage in fewer conflicts.

**Boy-Boy Examples**
For the boy-boy examples, I present excerpts from pairs 2 and 5. For both pairs I had two sessions of video to review. While reviewing the data, I noticed commonalities between them.

<table>
<thead>
<tr>
<th></th>
<th>ES</th>
<th>Q</th>
<th>U</th>
<th>P</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>19.83%</td>
<td>8.29%</td>
<td>0.28%</td>
<td>0.75%</td>
<td>0.88%</td>
</tr>
<tr>
<td>Task</td>
<td>14.79%</td>
<td>7.00%</td>
<td>0.07%</td>
<td>0.37%</td>
<td>0.83%</td>
</tr>
</tbody>
</table>
### Table 7.11

(Driver: Abigail, Navigator: Angie)

<table>
<thead>
<tr>
<th>Angie</th>
<th>What is five plus five</th>
<th>Directive/Suggestion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angie</td>
<td>Let’s do that one</td>
<td>Directive/Suggestion</td>
</tr>
<tr>
<td>Abigail</td>
<td>No, let’s do something hard</td>
<td>Directive/Suggestion</td>
</tr>
<tr>
<td>Angie</td>
<td>No</td>
<td>Other - Related</td>
</tr>
<tr>
<td>Abigail</td>
<td>Like 12 times 12</td>
<td>Directive/Suggestion</td>
</tr>
<tr>
<td>Angie</td>
<td>No, let’s do ”what’s five times four?”</td>
<td>Directive/Suggestion</td>
</tr>
<tr>
<td>Angie</td>
<td>What..</td>
<td>Directive/Suggestion</td>
</tr>
<tr>
<td>Abigail</td>
<td>Is..</td>
<td>Other - Related</td>
</tr>
<tr>
<td>Abigail</td>
<td>Let’s do something hard, like 12 times 12</td>
<td>Directive/Suggestion</td>
</tr>
<tr>
<td>Angie</td>
<td>No</td>
<td>Other - Related</td>
</tr>
<tr>
<td>Abigail</td>
<td>It’s for-144</td>
<td>Explanation - Other</td>
</tr>
<tr>
<td>Angie</td>
<td>No, let’s do like, something easier</td>
<td>Directive/Suggestion</td>
</tr>
<tr>
<td>Angie</td>
<td>Four times five</td>
<td>Directive/Suggestion</td>
</tr>
<tr>
<td>Abigail</td>
<td>Four times eight</td>
<td>Directive/Suggestion</td>
</tr>
<tr>
<td>Angie</td>
<td>Four times six</td>
<td>Directive/Suggestion</td>
</tr>
<tr>
<td>Abigail</td>
<td>What is four times six</td>
<td>Explanation - Self</td>
</tr>
</tbody>
</table>

### Table 7.12

(Driver: Aria, Navigator: Rachel)

<table>
<thead>
<tr>
<th>Rachel</th>
<th>That’s what I was saying this whole time</th>
<th>Explanation - Self</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aria</td>
<td>No, you were saying put it in there for the seconds</td>
<td>Explanation - Self</td>
</tr>
<tr>
<td>Rachel</td>
<td>No, I was saying put it in there</td>
<td>Explanation - Self</td>
</tr>
<tr>
<td>Rachel</td>
<td>Aria, it’s mind blowing</td>
<td>Other - Related</td>
</tr>
<tr>
<td>Aria</td>
<td>(laughs) No</td>
<td>Other - Related</td>
</tr>
<tr>
<td>Rachel</td>
<td>No, hit stop</td>
<td>Directive/Suggestion</td>
</tr>
<tr>
<td>Aria</td>
<td>No, because we have to go over-</td>
<td>Explanation - Self</td>
</tr>
<tr>
<td>Rachel</td>
<td>Yeah</td>
<td>Other - Related</td>
</tr>
</tbody>
</table>
Table 7.13

<table>
<thead>
<tr>
<th>(Driver: Surya, Navigator: Evan)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Evan</td>
<td>No Surya stop</td>
</tr>
<tr>
<td>Evan</td>
<td>I’m driver you idiot</td>
</tr>
<tr>
<td>Surya</td>
<td>Make him 350</td>
</tr>
</tbody>
</table>

Table 7.14

<table>
<thead>
<tr>
<th>(Driver: Daniel, Navigator: Lincoln)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Lincoln</td>
<td>I’ll draw your car</td>
</tr>
<tr>
<td>Daniel</td>
<td>No, wait</td>
</tr>
<tr>
<td>Daniel</td>
<td>Lincoln</td>
</tr>
<tr>
<td>Daniel</td>
<td>Lincoln, seriously</td>
</tr>
<tr>
<td>Daniel</td>
<td>You’ve got to put in the thing</td>
</tr>
<tr>
<td>Lincoln</td>
<td>No, who cares</td>
</tr>
<tr>
<td>Daniel</td>
<td>Lincoln, we can’t start until you do that</td>
</tr>
<tr>
<td>Lincoln</td>
<td>You remember last time, I had such a great attitude but look what happened</td>
</tr>
<tr>
<td>Daniel</td>
<td>Yeah</td>
</tr>
</tbody>
</table>

For the second session, both pairs had one student that was off-task and distracted during the majority of the session. With pair 2, the distracted student often disengaged from the task and began talking to another peer. With pair 5, the distracted student stayed engaged with their partner but not their task. This then frustrated his partner. In fact, early on, Lincoln, the distracted student, even stated, “You remember last time, I had such a great attitude but look what happened.”

Pair 2 engaged in 13 conflicts, 8 were about the task and 4 were about control of the computer. Pair 2 had 3 instances of uptake, none of praise, and 12 of engaging in antagonization. The excerpt displayed in Table 7.13 is an example of one of their control conflicts. Surya often took control over the computer and Evan sometimes did not notice for a while and other times did not appear to care (e.g., he did not ask for the computer back). Both students in pair 2 participated in the focus group interviews but did not suggest that they had any problems working together. However, they mentioned that they often waited to switch until the other person wrapped up their thoughts and by that point they had forgotten that they were supposed to switch.

Pair 5 engaged in 24 conflicts, 11 were about the task, 3 were about control, and 11 were about other problems. They did not have instances of uptake and had 2 instances of praise and 13 instances of engaging in antagonization. This conflict occurred early on when Daniel was driving and Lincoln stated “I’ll draw it, since I’m your copilot and you’re busy driving.” Table 7.14 displays the conflict.

There are two takeaways from these examples. With pair 2, the boys engaged in fewer conflicts than pair 5, however, they seemed to interact with each other a lot less. This suggests
Table 7.15

(Driver: Noah, Navigator: Molly)

<table>
<thead>
<tr>
<th>Molly</th>
<th>Let me try</th>
<th>Directive/Suggestion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noah</td>
<td>Shut Up</td>
<td>Antagonization</td>
</tr>
<tr>
<td>Molly</td>
<td>Cause I can be stupid sometimes.</td>
<td>Explanation - Self</td>
</tr>
<tr>
<td>Noah</td>
<td>Why do we have to have this on</td>
<td>Other - Unrelated</td>
</tr>
<tr>
<td>Molly</td>
<td>Sometimes I can be stupid but I can also figure it out</td>
<td>Explanation - Self</td>
</tr>
<tr>
<td>Noah</td>
<td>Shut up</td>
<td>Antagonization</td>
</tr>
<tr>
<td>Molly</td>
<td>This is why I don’t like having Noah as a partner, ’cause he’s always control of everything apparently</td>
<td>Explanation - Other</td>
</tr>
</tbody>
</table>

that their collaboration was not as productive as it could have been. With pair 5, their conflicts were often about other problems, such as Lincoln distracting Daniel. This supports H2.3 and provides examples of unproductive conflicts.

**Girl-Boy Examples**

For the girl-boy pairs, I present excerpts from pairs 4 and 6. I had videos of two sessions for each pair. They both had fewer conflicts in their second session. Additionally, both pairs displayed dislike for working with each other. Molly, the girl in pair 4, was more distracted with her peers than the students in pair 6.

Pair 4 engaged in 7 conflicts. 1 was about the task, 4 were about control, and 4 were about other problems. They had one instance of uptake, none of praise, and 11 of engaging in antagonization. Table 7.15 displays the conflict example. Noah often expressed frustration with Molly and in this case, he told her to “Shut up” multiple times. This may have caused her to disengage with the task at hand and talk to her peers in other groups instead.

Pair 6 engaged in 13 conflicts. 12 were about the task, 2 were about control, 8 were about partner contributions/roles, and 5 were about other problems. They had no instances of uptake or praise and had 9 of engaging in antagonization. This pair often had long conflicts and even engaged in a conflict during the focus group interviews. When asked about the switching reminder, Annie had stated, “I think I preferred being the navigator but I think that it was still pretty helpful.” Larry replied with, “Then why did you always hog the driver?” When navigating, Larry seemed to expect Annie to follow his directives and suggestions exactly. Larry also expected very specific directives and suggestions from Annie and expressed his frustration with Annie by raising his voice.

After reviewing these six pairs, I investigated the frequency of their use of uptake, praise, and antagonization since we found in Chapter 5 that these were indicators of high- and low-conflict pairs and Table 7.10 indicated that time-based role switching had a higher percentage of uptake, praise, and antagonization use. I found that 2 pairs of the task-based role switching pairs used uptake (4 times total) and 4 pairs of the time-based role switching pairs used uptake (15 times total). 5 pairs of task-based used praise (17 times total) while 3 time-based pairs used praise (26 times total). Finally, all but one time-based pair used antagonization; 42 instances
Table 7.16

<table>
<thead>
<tr>
<th>(Driver: Annie, Navigator: Larry)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Larry</td>
</tr>
<tr>
<td>Annie</td>
</tr>
<tr>
<td>Larry</td>
</tr>
<tr>
<td>Annie</td>
</tr>
<tr>
<td>Larry</td>
</tr>
<tr>
<td>Larry</td>
</tr>
<tr>
<td>Larry</td>
</tr>
<tr>
<td>Annie</td>
</tr>
<tr>
<td>Larry</td>
</tr>
<tr>
<td>Larry</td>
</tr>
<tr>
<td>Annie</td>
</tr>
<tr>
<td>Larry</td>
</tr>
<tr>
<td>Annie</td>
</tr>
<tr>
<td>Larry</td>
</tr>
</tbody>
</table>

Table 7.17

<table>
<thead>
<tr>
<th>Condition</th>
<th>Total</th>
<th>ES</th>
<th>EO</th>
<th>D</th>
<th>Q</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>630</td>
<td>20.48%</td>
<td>33.81%</td>
<td>24.13%</td>
<td>12.22%</td>
<td>9.37%</td>
</tr>
<tr>
<td>Scaffolded</td>
<td>835</td>
<td>31.02%</td>
<td>23.47%</td>
<td>30.06%</td>
<td>9.82%</td>
<td>5.63%</td>
</tr>
</tbody>
</table>

in task-based and 40 instances in time-based). This suggests that uptake and praise may also play a part in reducing the number of conflicts that occur between pairs and may explain my earlier findings. Specifically, it could explain why the time-based switching reminder groups encountered fewer conflicts than the task-based switching reminder groups.

7.3.4.3 Comparing collaborative dialogue and conflicts between datasets

Table 7.17 displays the dialogue moves of the conflicts in both datasets Conflicts and Dialogue Acts 1 and Conflicts and Dialogue Acts 2. The results are consistent with Table 7.9 in that the scaffolded pairs showed a higher use of self-explanations than the non-scaffolded pairs. In the remainder of this section, I will present excerpts as examples of the way the pairs differ in their interaction to explain the differences in the dialogue of the two datasets.

Self Explanation Examples

Self explanation was used more by the students in Dataset Conflicts and Dialogue Acts 1. This excerpt (Table 7.18) is from pair 5 in Dataset Conflicts and Dialogue Acts 1 during the second week of the curriculum. Daniel and Lincoln worked well together the previous week. However, this week, Lincoln was distracted. In this excerpt, Lincoln was driving and this conflict started when Lincoln made a noise a fellow classmate was making. When Daniel did not make the noise, Lincoln asked him if he was going to join. Daniel’s response was “No, cause I want to work.” Lincoln was driving and continued to make changes. During this conflict, both students
used self-explanation and they appeared reach an agreement at the end of the conflict.

Table 7.19 displays an excerpt of a conflict from pair 8 in Dataset Conflicts and Dialogue Acts 1. At the beginning of the conflict, Ariel was driving and Charlotte disagreed with her changes. Charlotte tried to explain that those changes would not work. Ariel used self-explanation but did not elaborate on her thoughts, just her future actions. This shows that although self-explanation was used, more scaffolding was needed for the students to use it effectively.

Directive/Suggestion Examples

Directives and suggestions were used more by students in Dataset Conflicts and Dialogue Acts 1. In this excerpt (Table 7.20), we see how Angie from pair 9 in Dataset Conflicts and Dialogue Acts 1 tried to use directives/suggestions to influence her partner. They disagreed with each other and tried to explain why copying the current code is better or worse than starting the next task from the beginning. In the end Abigail did not listen to Angie and Angie seemed to follow Abigail’s lead.

Table 7.21 displays an excerpt of pair 10 from Dataset Conflicts and Dialogue Acts 1. Aria is the driver and Rachel made a suggestion. Aria disagreed with it, saying they had already tried that and pointed at where the block was. Rachel countered with a new idea of putting it in a different part of the code. Aria did not follow her idea and asked the teacher for help less than a minute later.

This shows that using directives/suggestions by itself is not enough. Students need to listen to their partner’s ideas and discuss them in order to have a productive collaboration process.

Explanation - Other Examples

Other explanations were used more by the pairs in Dataset Conflicts and Dialogue Acts 2. Table 7.22 displays an excerpt of pair 1 from Dataset Conflicts and Dialogue Acts 2. Luke sought to use the move block but David disagreed and gave a different directive/suggestion. Luke tried to explain that it’s the same thing. Then David explained how it should have worked. They went back and forth for a little while before the conflict ended. Even though the students...
Table 7.19

<table>
<thead>
<tr>
<th>(Driver: Ariel, Navigator: Charlotte)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Ariel</td>
<td>We have to say “hello Ariel”</td>
</tr>
<tr>
<td>Charlotte</td>
<td>Oh my goodness, that is not what we are supposed to do, trust me</td>
</tr>
<tr>
<td>Charlotte</td>
<td>Go to “green”</td>
</tr>
<tr>
<td>Ariel</td>
<td>I’m just going to try this</td>
</tr>
<tr>
<td>Charlotte</td>
<td>It will work, but not the way you want it to</td>
</tr>
<tr>
<td>Ariel</td>
<td>I am just, I am just going to try</td>
</tr>
<tr>
<td>Charlotte</td>
<td>It will work, but not the way we want...are supposed to</td>
</tr>
<tr>
<td>Ariel</td>
<td>Yeah but</td>
</tr>
<tr>
<td>Charlotte</td>
<td>No type in my name</td>
</tr>
<tr>
<td>Ariel</td>
<td>What</td>
</tr>
<tr>
<td>Charlotte</td>
<td>Type in my name here</td>
</tr>
<tr>
<td>Ariel</td>
<td>It won’t work</td>
</tr>
<tr>
<td>Ariel</td>
<td>I know</td>
</tr>
<tr>
<td>Ariel</td>
<td>That’s why you type in your name here</td>
</tr>
<tr>
<td>Charlotte</td>
<td>You are not supposed to do that</td>
</tr>
<tr>
<td>Charlotte</td>
<td>It is supposed to be she works[inaudible 00:08:15]</td>
</tr>
<tr>
<td>Charlotte</td>
<td>Change that</td>
</tr>
<tr>
<td>Ariel</td>
<td>Change this</td>
</tr>
<tr>
<td>Ariel</td>
<td>Oh fine</td>
</tr>
<tr>
<td>Charlotte</td>
<td>No, no, no</td>
</tr>
<tr>
<td>Charlotte</td>
<td>Okay go to ”green” again, I guess</td>
</tr>
<tr>
<td>Ariel</td>
<td>Go to ”green”</td>
</tr>
</tbody>
</table>

Table 7.20

<table>
<thead>
<tr>
<th>(Driver: Abigail, Navigator: Angie)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Abigail</td>
<td>So, we can Copy this</td>
</tr>
<tr>
<td>Angie</td>
<td>What...</td>
</tr>
<tr>
<td>Angie</td>
<td>No, let’s just make it again, because we know how</td>
</tr>
<tr>
<td>Angie</td>
<td>So, get a ”forever”..</td>
</tr>
<tr>
<td>Abigail</td>
<td>No wait, let’s just copy this</td>
</tr>
<tr>
<td>Abigail</td>
<td>It’s be easier</td>
</tr>
<tr>
<td>Angie</td>
<td>No, ’cause then it’ll copy the whole thing</td>
</tr>
<tr>
<td>Abigail</td>
<td>I know, and then we can just change this to down arrow</td>
</tr>
<tr>
<td>Angie</td>
<td>I think that’ll be a little hard</td>
</tr>
<tr>
<td>Angie</td>
<td>Let’s just..</td>
</tr>
<tr>
<td>Abigail</td>
<td>No, it’s, it’s..</td>
</tr>
<tr>
<td>Angie</td>
<td>Do, make it again</td>
</tr>
<tr>
<td>Abigail</td>
<td>It’ll be easy</td>
</tr>
<tr>
<td>Angie</td>
<td>Just make it again, Abigail</td>
</tr>
<tr>
<td>Abigail</td>
<td>It’ll be easy</td>
</tr>
<tr>
<td>Angie</td>
<td>Abigail come on, please</td>
</tr>
<tr>
<td>Angie</td>
<td>Duplicate</td>
</tr>
</tbody>
</table>
Table 7.21

<table>
<thead>
<tr>
<th>(Driver: Aria, Navigator: Rachel)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rachel</td>
</tr>
<tr>
<td>Rachel</td>
</tr>
<tr>
<td>Aria</td>
</tr>
<tr>
<td>Aria</td>
</tr>
<tr>
<td>Rachel</td>
</tr>
</tbody>
</table>

Table 7.22

<table>
<thead>
<tr>
<th>(Driver: Luke, Navigator: David)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Luke</td>
</tr>
<tr>
<td>David</td>
</tr>
<tr>
<td>David</td>
</tr>
<tr>
<td>David</td>
</tr>
<tr>
<td>Luke</td>
</tr>
<tr>
<td>David</td>
</tr>
<tr>
<td>David</td>
</tr>
<tr>
<td>Luke</td>
</tr>
<tr>
<td>David</td>
</tr>
<tr>
<td>Luke</td>
</tr>
<tr>
<td>David</td>
</tr>
<tr>
<td>Luke</td>
</tr>
</tbody>
</table>

...were not using self-explanation, they were still explaining how they thought the program would work and was able to move on in their project.

Table 7.23 displays an excerpt of pair 5 from Dataset Conflicts and Dialogue Acts 2. Similar to the previous group, the students explained what would happen if they made certain changes to the program and Sandy seemed to agree with Mitchell at the end of the conflict.

**Question Examples**

Questions were used more often by pairs from Dataset Conflicts and Dialogue Acts 2. While pairs from Dataset Conflicts and Dialogue Acts 2 already used explanation often, their use of questions often prompted the use of self and other explanation as well. Table 7.24 displays an excerpt of pair 7 from Dataset Conflicts and Dialogue Acts 2. When Sandy asked, “It’s that right?” Steve confirmed and explained that it’s all they needed to do. They then went into a discussion about the next steps of the task.

Table 7.25 displays an excerpt of pair 8 from dataset Ch5-conflictDialogue. With this pair, Anthony asked a question because he needed clarification on what the sprite needed to say. Rupert’s explanation helped clear up his confusion and Anthony stated, “That makes sense.”
### Table 7.23
(Driver: Mitchell, Navigator: Sandy)

<table>
<thead>
<tr>
<th>Mitchell</th>
<th>Sandy</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>wait if that’s the left arrow then this is the right then I need D-</td>
<td>You need to make the same thing again except-</td>
<td>Self</td>
</tr>
<tr>
<td>That’s going to be D.</td>
<td></td>
<td>Other</td>
</tr>
<tr>
<td>When D clicks it goes that way.</td>
<td></td>
<td>Other</td>
</tr>
<tr>
<td>But then that would completely erase player one.</td>
<td></td>
<td>Other</td>
</tr>
<tr>
<td>No because look this is sprite two and sprite one has the arrows.</td>
<td></td>
<td>Other</td>
</tr>
<tr>
<td>Ok</td>
<td></td>
<td>Unrelated</td>
</tr>
</tbody>
</table>

### Table 7.24
(Driver: Steve, Navigator: Sandy)

<table>
<thead>
<tr>
<th>Sandy</th>
<th>Steve</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>You broke it</td>
<td>No I didn’t</td>
<td>Antagonization</td>
</tr>
<tr>
<td>Oh</td>
<td></td>
<td>Antagonization</td>
</tr>
<tr>
<td>Look it says ”Switch costume” and since we have- have no costume</td>
<td></td>
<td>Other</td>
</tr>
<tr>
<td>It’s that right</td>
<td>Yeah we just have to do that</td>
<td>Question</td>
</tr>
<tr>
<td>But it should already have the</td>
<td></td>
<td>Self</td>
</tr>
<tr>
<td>Oh he doesn’t have any costumes other than this one</td>
<td></td>
<td>Other</td>
</tr>
<tr>
<td>Oh he should already have that</td>
<td></td>
<td>Other</td>
</tr>
<tr>
<td>Yeah so we have to make a new costume for the next costume</td>
<td></td>
<td>Self</td>
</tr>
<tr>
<td>Can I make it</td>
<td></td>
<td>Question</td>
</tr>
</tbody>
</table>

### Table 7.25
(Driver: Rupert, Navigator: Anthony)

<table>
<thead>
<tr>
<th>Anthony</th>
<th>Rupert</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>cut right here.</td>
<td>But you cut it before that though because it stops be-</td>
<td>Suggestion</td>
</tr>
<tr>
<td>Cut it right there.</td>
<td></td>
<td>Suggestion</td>
</tr>
<tr>
<td>Doesn’t he need to say oh look cake?</td>
<td></td>
<td>Self</td>
</tr>
<tr>
<td>No because that’s not in the loop.</td>
<td></td>
<td>Other</td>
</tr>
<tr>
<td>It’s.</td>
<td></td>
<td>Related</td>
</tr>
<tr>
<td>That makes sense.</td>
<td></td>
<td>Related</td>
</tr>
</tbody>
</table>
7.3.5 Focus Group Results

The focus group interviews that were held after the curriculum was completed helped bring to light some new changes the students would like for the features.

7.3.5.1 Talking Reminder

First, the students felt that the talking reminder showed up too often, “It popped up way too much.” They felt annoyed by the reminder after a while, “But then thought it was kind of annoying cause once it would appear, it would be up every minute or so. Like three in a row.” Another student said, “It’s just so annoying.”

They also felt that it interrupted their flow of work, “Just kind of if you were in the middle of doing something and one of those popped up it kinda interrupted what you were doing.” Some students did not want to talk while they were in the middle of working because they might forget their thoughts, “...if you’re like really into something and then it tells you to talk and then you kind of forget about it.”

Some students were already talking “... during the coding, so when that came up, we were like just ignoring it ’cause like it, we were already talking.” Another student stated, “there’s no need to add a talking thing when the navigator’s constantly talking to the driver.” One student stated, “Well like, whenever it like, the thing appeared, is just like you had to just push the like the okay thing when you weren’t actually doing anything else.”

Some students suggested that the talking reminder could still be helpful though, “Well, if we didn’t have the talk thing then we probably wouldn’t talk.” A different student stated, “I thought it was helpful because it reminded me to keep asking questions.” Another student said, “I think the talking is helpful because in case one person like doesn’t, doesn’t know how to do something and you know how it works it can like, help that person.”

In terms of the modifications that could be made, students generally wanted the talking reminder to occur when they are not working, “I, maybe at the beginning it says talk with your partner, so then like.” Another student finished, “So it’s not like while you’re working...Or maybe at the end, like the very end.” Other students suggested having it show up in the corner instead of the middle of the screen, “Bottom right, like under the task.” One student suggested, “… Do the microphone where it senses, like, um, senses you talking.”

7.3.5.2 Switching Reminder

Overall, the students had positive views of the switching reminder. There was also focus on equal driving time as we found in chapter 6, “Um, that we both get to do the same amount of coding and stuff.” Another student stated, “Each person gets the ability to do it.” A different student said, “The switch one, it was really helpful to, like, separate, um, who could be the navigator because we... because i-if we didn’t have that, we would probably like, argue.” This was also a point that students brought up in the previous round of focus group interviews.
With the switching reminder, most changes students wanted to make was to switch more or less often. Different students had different desires about how often they would like to switch. We also had varying responses when we asked whether they would prefer task-based or time-based role switching.

In terms of the one minute warning, students felt that it changed their behavior, “Like you’re doing something and then you realize you have one minute left. So you’re trying to like, like, like you’re trying to wrap [it] up.” One student felt that it messed them up, “I don’t really like it ‘cause it kinda messes you up from what you’re doing and you start rushing, so you don’t do as good-” Another student stated, “So the warning is good so that it’s not, like, a surprise-” Some students suggested giving a two or five minute warning instead of a one minute warning.

7.3.5.3 Icons and Poster

This round we also asked the students for feedback about the icons and posters. Some students did not like the colors. One students suggested adding headphones to the icons. Students seemed to prefer more gender neutral icons. And students wanted other small changes to the wording of the posters and to the icons.

7.4 Discussion

In my work I found that those who used the time-based switching reminder will encounter fewer conflicts compared to students who use the task-based switching reminder. However, there are still areas of interaction that need to be considered. First, I found out that Pair 1 had forgotten to check off their tasks in at least one full class session. Which occurred before the video sessions that I sampled. Given that they are a task-based role switching pair, that means only one student was able to drive for that session. This could be true of other pairs that was not part of the sample dataset. Additionally, if some tasks took more time than others, then pairs may not drive for similar amounts of time. Both issues could have been points of contention with the task-based role switching pairs.

The second area that should be discussed is in relation to the pairs I examined qualitatively. Within three pairs of students, I found that one partner from each pair was distracted for one of the sessions. When those partners were distracted, it seemed that the way they addressed their distraction was important. For example, pairs 2 and 4 (time-based) had students that started interacting with other classmates. However, with pair 5 (task-based), the distracted partner resorted to talking like a pilot and distracting his partner by giving him various nicknames. These distractions were often a point of conflict for this pair. I posit that the students who were disengaged but distracted with other peers had less of a chance at engaging in conflicts. This is consistent with our previous finding in Chapter 5 that low-conflict pairs also often had fewer utterances.

Finally, the third area of discussion with findings related to H.2.3 is that there were more
instances of uptake in time-based role switching pairs. This suggests that the students built off of each other’s ideas more often and might have had fewer conflicts as a result of their willingness to use uptake. Future work should also investigate how productive each conflict was in order to better understand the whether the time-based or task-based switching reminder was best for students’ conflicts.

Additionally, this analysis shows that an adaptive programming environment that scaffolds collaborative dialogue and role switching will lead to more balanced dialogue distribution compared to students who use the programming environment without the scaffold. When comparing dialogue from chapters 5 and 7, I found that there was a significant difference in the balance of pairs in chapter 7’s dataset (talking and switching reminder) than there were in chapter 5’s dataset (no features). This indicates that students using the scaffolds that I implemented, were positively affected in terms of balance. The most imbalanced group was pair 1 at 64% for the most talkative partner in comparison to the 83% for the most talkative partner in Chapter 7’s dataset. Additionally, with Chapter 7’s dataset most pairs (10) did not have a partner that talked 60% of the time or more. In contrast, Chapter 5’s dataset only had 3 pairs that were as balanced. This finding is promising because in previous work with learning companions as an intervention, researchers did not find evidence of the intervention affecting the distribution of talk between the students [Buf17]. I believe that this could an effect of the switching reminder. That is because some students seemed to believe that the navigator should talk more than the driver, “And one of them’s the navigator, so it’s supposed to talk.” Additionally, there has been reported instances of students feeling passive and being disengaged while being the navigator[Bra19; CB18a]. Researchers hypothesized that “Because the driver has the goal of building the program, drivers may not engage in dialogue as easily as navigators.” However, they found that driver dialogue is an important part of the collaborative learning process [Rod17b]. This suggests that if the switching reminder allows both students fairly similar driving times, then they may also have similar talking times. This could include both the collaborative processes between the students and their learning.

Note that this finding is based upon a comparison between two populations in two separate studies. In the dataset from Chapter 7, we investigated both 4th and 5th grade students that were in an urban charter school. In the dataset from Chapter 5, I worked with 5th grade AIG students at a suburban public school. However, the imbalance observed in our study described in Chapter 5 is a result that is consistent with my findings in Chapter 3, where most pairs of students exhibited imbalanced talk and driving time. This was in the context of 5th grade students in an urban public school. Similar imbalances have also been found in other work on CS education by Shah, Lewis, and Caires [Sha14], as well as Deitrick, Shapiro, and Gravel [Dei16]. This has also been found in group work outside of CS [SH18].

My analysis also showed that the task-based switching reminder pairs reported a statistically significant higher increase in enjoyment of CS than the time-based switching reminder pairs. The time-based switching reminder condition seemed to have more instances of students not
switching or switching later. This was not always agreed upon by both partners. Additionally, I found that the time-based switching reminder pairs engaged in more control conflicts than task-based switching reminder pairs. These findings may have contributed to the time-based switching reminder students having a smaller increase in enjoyment of CS than their peers in the other condition. Future work can follow up with more open-ended survey questions about factors that made them enjoy or not enjoy CS. Additionally we may consider interviewing students about their responses and what factors affect their answers.

When comparing the conflicts of pairs in Chapter 7’s dataset with pairs in chapter 4 and chapter 5’s dataset, I found that pairs in Dataset Conflicts and Dialogue Acts 1 had more conflicts and the difference was statistically significant. I also found that the datasets from Chapter and Chapter 5 had control conflicts 31% of the time while the dataset Chapter 7 had control conflicts 17% of the time; however, these results were not significant. With partner contribution/roles conflicts, the dataset from Chapter 7 had 18% while the datasets from Chapter 4 and Chapter 5 had 34%. With other conflicts, Chapter 7’s dataset had 31% and chapters 4 and 5’s datasets had 12%. This suggests that the students in Chapter 7’s dataset engaged in conflicts that the features were not designed to help with. However, the switching reminder may have helped with reducing control and partner contribution/role conflicts, even if the difference was not significant. With advancing technology, we may be able to detect types of conflict and alert teachers when the students engage in conflicts unrelated to the task and the system can intervene as usual to improve the number of control and partner roles/contribution conflicts.

When comparing the switching reminder pairs to the switching reminder plus talking reminder pairs, I found that the students with the talking reminder did not use the collaborative dialogue strategies more frequently. However, looking further in a comparison between the scaffolded students from Chapter 7’s dataset and the un-scaffolded students in Chapter 5’s dataset, I found that the students in Chapter 7’s dataset appeared to use self-explanation more than the students in Chapter 5’s dataset. This holds true for students in the talking reminder and the non-talking reminder conditions. This may also be explained by our findings about the students’ dialogue balance. The switching reminder may encourage students to use more self-explanation simply by having them drive at more of an equal amount of time. This suggests that the switching reminder may be of more use than first expected, however, more research is needed before reaching this conclusion.

Additionally, the productiveness of a conflict does not seem to depend on self-explanation, directives, or questions. The conflict outcome depends more on how the students use each collaborative dialogue strategy. This suggests that a more refined tagging scheme may be necessary for detecting productive and unproductive conflicts. For example, both “I know” and “I am making it move left” are tagged as self-explanation. A more detailed tagging scheme may separate the two and also have more subcategories for each type of dialogue move.

In terms of my theory of action, the talking and switching reminders seemed to increase the equity of the students’ talk time. Additionally, the task-based switching reminder improved
students’ CS attitudes more than the time-based switching reminder. The CS attitudes survey included questions about the students’ CS self-efficacy. The reminders did not reduce the number of overall conflicts, however, the students with the reminders had a smaller percentage of control conflicts than students who did not have the reminders.

7.4.1 Reminder Usage

From my qualitative annotation of the students’ use of the reminders, I found that the students often became frustrated with the talking reminder while they often followed the switching reminder. During the focus group interviews I found that some students did not like to be prompted to talk while they are working because it interrupted them or their flow of thought. Previous researchers found that it may be difficult for novices to think aloud while they are solving programming problems [WK14]. This may be akin to asking students to explain to their partner while they are working. This may also be one reason why students were frustrated with the reminder in the pair programming videos. These findings support the argument for the students’ suggestions to present the reminder at the beginning of the task rather than throughout the task. This would prevent the reminder from showing up when they are working and hopefully, also prevent the frustration of also being interrupted while they are working. Additionally, different activities require more or fewer programming actions in a period of time. For example, with the loops lesson, we have students copy code from a piece of paper after modifying it to use loops. With this activity, the talking reminder is likely to get triggered more often than other activities. In addition, some students work faster than others. Therefore, using a fixed trigger rather than an action trigger for the reminder may prevent these problems from occurring. Another option would be to adjust the threshold of actions depending on the lesson and/or pair. But that would require more data. Therefore, in the next iteration, I would have followed one of the students’ suggestions and either have them talk at the beginning or have them talk after each task.

With the task switching reminder, I found that students did not always click on the task completed button either on purpose to avoid switching, or by accident. Additionally, I noticed two pairs of students that clicked the completed button intentionally to switch, even if they had not completed the task. This suggests that the features may benefit from an automated system to detect when a task is completed. This has also been suggested by the students in both focus group interview rounds, “It would be, like, a self-checking system.”

In order to provide students with autonomy but still provide them with scaffolds, researchers should continue to research switching methods and how students’ learning and enjoyment are affected if they are given the choice on which switching method to use. Since I did not find a significant difference in learning or collaboration attitude between the switching reminder conditions, researchers should continue to investigate changes in the students’ attitudes and learning based on the choice the students make. We may also consider letting them choose how quickly they switch and if they want a warning for those that choose the time-based switching
reminder. Giving them choices is important because in Chapter 6, I found that the students enjoyed using the task list because it gave them autonomy and allowed them to move forward at their own pace. This could improve their collaboration and CS attitudes and encourage them to continue to collaborate and learn CS in the future.

7.5 Study 2 Plan

I had planned on collecting data from another iteration of the features. However, due to the coronavirus, the study had to be cancelled. A total of 84 4th grade elementary students at a science charter school would have participated in the class. At the time of the cancellation, the STEAM teacher had data collection consent and assent forms returned from 50 students.

7.6 Conclusion

In this chapter, I investigated how a talking reminder and switching reminder supported upper-elementary students’ collaborative problem solving process. I did not find differences in the students’ collaboration attitude or learning. The task-based switching reminder students reported a higher increase in CS attitude than the time-based switching reminder students. I found that students in the time-based switching reminder condition engaged in fewer conflicts than the task-based switching reminder condition. However, the students with the scaffolds engaged in more conflicts than students in a previous study. Additionally, the switching reminder appeared to scaffold the students’ talk time and use of self-explanation, possibly due to differences in how drivers and navigators talk. This chapter provides insight into how to design features to support upper-elementary students while they are pair programming.
Chapter 8

CONCLUSION

My goal for this work was to continue to research how students at the elementary age level collaborate to solve computer science problems and how we can support them. The knowledge generated from this work can be used to support many students in their future endeavors. The students participating in the study will benefit from learning computer science and best collaboration practices that they will need to use in the future.

8.1 Summary

My research for this dissertation is interdisciplinary and draws from the computer science education, computer supported collaborative learning, and artificial intelligence in education communities. I have investigated the processes that elementary students use when they solved computing problems in pairs and implemented adaptive collaborative support features into an existing block-based programming environment to examine the efficacy of those features. Specifically, I have investigated the research questions:

RQ1: To what extent do upper elementary students engage in self-explanation, question generation, uptake, and praise when collaboratively solving computer science problems? Conversely, to what extent do they antagonize each other?

RQ1a: How do upper elementary students’ collaborative dialogue practices relate to the number and type of conflicts that emerge during pair programming?

RQ2: To what extent do the following features in an adaptive programming environment support upper elementary students’ collaborative problem solving process?

- A built in timer for switching control
- A built in tasklist for switching control
- A reminder to talk to their partner

In Chapter 2, I synthesized relevant literature to provide background information into my work. Chapters 3 and 4 laid the groundwork necessary for the remainder of my chapters. They
detailed my initial investigations into elementary students’ collaborative dialogue (Chapter 3) and how they engaged in conflicts (Chapter 4).

Chapter 5 detailed my investigation into my first research question. This research question was built upon my early work in chapters 3 and 4. The work completed in that chapter revealed that high-conflict pairs sometimes antagonized one another, while most used uptake and praise. High-conflict pairs engaged in conflicts over task, computer control, and partner contributions; low-conflict pairs only engaged in task-related conflicts and were less talkative overall. Finally, high-conflict pairs were often mid- or high-balanced dialogue pairs. These findings include both productive and unproductive conflicts; however, high-conflict pairs seem to be more likely to engage in certain kinds of conflict or antagonistic dialogue moves overall. These findings suggest that through dialogue analysis, we may be able to detect conflicts and to provide support to students. They also reinforce the nuances of the social dynamic within collaboration and the importance of empowering each student to contribute.

In Chapter 6, I analyzed the data I collected from the first iteration of my adaptive support features. After a multi-day pilot study, I conducted focus group interviews to understand the students’ experiences with the features and what improvements they would like to see. I identified important overarching themes of agency and fairness that were expressed frequently throughout the children’s feedback. These learners strongly desire equality in the amount of time they spend in each role, and the autonomy to make their own choices in terms of turn taking and task completion. The findings from this work directly informed the changes I made to the features for the next iteration of studies.

My next iteration was detailed in Chapter 7. I conducted a multi-day study with the second iteration of my features. Using a pre- and post-assessment, pre- and post-surveys on students’ attitudes, pair programming video data, and focus group interviews, I found that pairs without the scaffolded supports had significantly fewer conflicts, time-based role-switching pairs had fewer conflicts than task-based role-switching pairs, the pairs using the scaffolded supports were significantly more balanced than pairs that did not use the scaffolded supports. Additionally, the students had a positive viewpoint of the role-switching feature and they often immediately followed the reminder. On the other hand, they expressed frustration and annoyance with the talking-reminder and explained that it came up too frequently and interrupted them while they worked. This work lends itself to further research on how to iteratively design scaffolded supports for upper elementary students.

8.2 Contributions

This work contributes to multiple fields of study, including collaborative learning and problem-solving, computer science education for elementary students, block-based programming environments, and environments with adaptive support for collaboration. The contributions of this dissertation is as follows:
• **A large and rich dataset of elementary students’ collaboration process while solving computer science problems.** While there may be large and rich datasets of elementary students collaborating, there has yet to be such a dataset (81 students) for students of this age group collaborating in computer science. These datasets contain videos from multiple studies. Additionally, I have collected focus group, assessment, and survey data to allow for more detailed analysis in terms of their experiences, learning, and attitudes.

• **Dialogue move annotation scheme.** I developed a dialogue move annotation scheme for analyzing the collaborative practices that students use while pair programming. To develop this scheme, I referenced related work on pair programming dialogue annotation schemes and other collaborative dialogue best practices. These include: [Rod17b; Tsa18a; Buf17; Ruv16; Gra94]

• **Conflict annotation scheme.** I developed an annotation scheme to identify conflicts and the types of conflicts students engaged in during pair programming. To develop this annotation scheme, I reviewed work on conflicts and used references such as [Har88].

• **Dialogue and conflict analysis of multiple elementary classrooms of pair programming relationships.** Research on elementary students pair programming has often focused on the balance of the pairs. My work adds to this line of research with a novel annotation scheme to investigate their best collaborative dialogue practices and a conflict annotation scheme to identify when they engage in conflicts. Additionally, I have investigated how their dialogue practices related to their conflicts. This has revealed findings in an area that has yet to be explored with regards to upper-elementary students learning CS collaboratively.

• **A set of curricula and activities that are appropriate for teaching computer science concepts to elementary students.** With the rise of computer science being taught at the elementary school level, there has also been an increase in computer science curricula, activities, and resources. However, few of these have been implemented in formal classrooms for more than a few days. The activities and curricula that we have designed for these studies will likely be made available as online resources for those who wish to implement them in classrooms, clubs, or other activities.

• **Empirical data and analyses of the effectiveness of an adaptive system to support elementary students collaborating to solve computer science problems.** To my knowledge, there has yet to be research on supporting upper elementary students using an adaptive support system while pair programming. My work reveals how students interact with such a system, how it affects their attitudes and learning, and how they felt about their experiences with the system. This dissertation makes a novel contribution to the computer supported collaborative learning and computer science education fields in this way.

• **Design recommendations for adaptive support for elementary students pair programming.** This dissertation produces insight into how future researchers can develop such a system
for this age group and context. My work on the students’ use of the system and their recommendations can provide guidelines on the next iteration of this system or how to build new features for elementary pair programmers.

8.3 Future Work

From here, there are many avenues of future work that can be followed. The first is a deeper analysis on the balance of programming pairs in regards to using 1-computer pair programming versus 2-computer pair programming. Does one paradigm lend itself to more balanced editing and talking with the pairs of students? Additionally, how does the pattern of collaborative dialogue differ between students using each paradigm? Can my annotation scheme capture the nuances of student dialogue in both paradigms? In an investigation of whether individual, collaborative, or both was more effective in the tutoring environment, Olsen et. al. [Ols17; Ols19] found that the students who learned both individually and collaboratively had higher learning gains than those that only learned individually or collaboratively. Does one paradigm allow for both individual and collaborative learning and will we find similar results in CS? Finally, we should investigate what types of adaptive support are needed with 2-computer pair programming and how those differ from those that are needed for 1-computer pair programming.

Second, while annotating my data for chapter 7, I noticed differences in the students’ dialogue based on their gender. For example, girls seemed to express uncertainty more often and state that they do not know how to solve a problem. Additionally, they uttered more self-deprecation statements about their intelligence. With boys, they seemed to be more aggressive in their use of antagonization than girls. Girls seemed to use more sarcasm and boys liked to use direct insults. More detailed analysis into their dialogue and their conflicts are necessary to closing the gender gap in CS. We should also investigate the differences in their collaboration and CS attitudes to better understand how those affect the students’ interactions with each other and how it relates to their learning.

Third, Celepkolu [Cel20] has developed visualizations that encourage and scaffold middle school students to reflect on their pair programming behaviors. This has shown promising results and the students saw the value in such an intervention. Elementary students may also benefit from this type of dashboard. Future work can focus on the best way to visualize this type of information to 4th and 5th grade students. We may begin with the stacked bar charts that appear in chapters 3, 5, and 7. An iterative refinement approach with students’ feedback could create a valuable tool that benefits both teachers and students.

Fourth, my work focuses on the students and how to support their collaborative work. However, an important component to the students’ success in learning CS collaboratively is their teachers. As such, I argue that we should begin interviewing elementary school teachers and their perceptions of pair programming. Additionally, we can learn about the types of support they want and need in order to successfully implement pair programming in their classrooms.
Finally, during COVID times, it has become increasingly apparent that access and equity is a problem that must receive more attention in CS education. Students learning at home have vastly different resources and some families and school districts may not have the funds and knowledge necessary to allow all students to learn CS. Therefore, researchers must work towards solving this problem before we can truly have CS for all.

In a research program, the first avenue would pair well with the second, third, and fourth avenues. The first and second avenues could be investigated together. This would allow for an investigation of which pair programming paradigm works best, which gender pairing works best, and possibly which gender pairings works best with specific pair programming paradigms. The third avenue could be investigated after the first and second. This would allow researchers to first decide on which paradigm to use and how they might design their system differently for girls versus boys. The fourth avenue should be investigated after each of the other avenues: how do they teachers feel about each pair programming paradigm; what differences do they see in the behaviors of boy and girl pairs; and how do they think the visualizations affect the behaviors of the students. The final avenue is the most important and should be investigated right away.
BIBLIOGRAPHY


134


**Partnership for 21st Century Learning.**


APPENDICES
Appendix A

FOCUS GROUP INTERVIEW QUESTIONS AND PROTOCOL: VERSION 1

The students in each focus group will have been in one of four conditions where they tried one of the following features:

- Time-based switching
- Task-based switching
- Time-based switching plus dialogue support
- Task-based switching plus dialogue support

All students also used a version of NetsBlox that had a task list of what they were supposed to complete.

A.1 Protocol

1. Turn on the camera.

2. Introduce yourself and explain that this focus group interview is to improve the features. Encourage them to be honest and tell them that their feedback is important and nobody’s feelings will get hurt.

   Example script:

   Hi, my name is Jen. This focus group is to help us understand how you feel about the reminders to talk/reminders to switch. Your feedback is very important to making these better. Please be as honest as you can. Nobody’s feelings will be hurt by your feedback.
3. Hand out the posters and remind the students of the features.
   
   Example script:
   
   Here are some popups you might have seen while you were programming. The first is the
   reminder to switch roles and the second is the reminder to talk to your partner. You also
   had a task list at the bottom, right-hand corner of NetsBlox. Do you remember those?

4. Ask the questions. For all questions, follow up on any interesting points the students
   talked about.

A.2 Task/Task+ Questions

1. What went well with the reminder to talk/reminder to switch? Why? Ask for examples.

2. What did not go well with reminder to talk/reminder to switch? Why? Ask for examples.

3. What would you improve about reminder to talk/reminder to switch?

4. What do you think would have been different without the reminder to talk/reminder to
   switch?

5. Did you ever decide not to follow the reminder to talk/reminder to switch? Why? Ask for
   examples of times that they did not.

6. Read Verbatim: We programmed NetsBlox to tell you to switch after every task. We
   had other students switch every 12 minutes. Which do you think you prefer and why?

7. 5th Grader Groups only: What was different between your pair programming experi-
   ence last year and your pair programming experience this year?

If there is time:

8. What went well with using the task list? Researcher hands out pictures with a screenshot
   of the task list. Why? Ask for examples.


10. What would you improve about the task list?

Other:

11. Is there anything else you would like to discuss?
A.3 Time/Time+ Questions

1. What went well with the reminder to talk/reminder to switch? Why? Ask for examples.

2. What did not go well with reminder to talk/reminder to switch? Why? Ask for examples.

3. What would you improve about reminder to talk/reminder to switch?

4. What do you think would have been different without the reminder to talk/reminder to switch?

5. Did you ever decide not to follow the reminder to talk/reminder to switch? Why? Ask for examples of times that they did not.

6. Read Verbatim: We programmed NetsBlox to tell you to switch every 12 minutes. We had other students switch after every task. Which do you think you prefer and why?

7. 5th Grader Groups only: What was different between your pair programming experience last year and your pair programming experience this year?

If there is time:

8. What went well with using the task list? Researcher hands out pictures with a screenshot of the task list. Why? Ask for examples.


10. What would you improve about the task list?

Other:

11. Is there anything else you would like to discuss?
Appendix B

FOCUS GROUP INTERVIEW QUESTIONS AND PROTOCOL: VERSION 2

The students in each focus group will have been in one of four conditions where they tried one of the following features:

- Time-based switching
- Task-based switching
- Time-based switching plus dialogue support
- Task-based switching plus dialogue support

All students also used a version of NetsBlox that had a task list of what they were supposed to complete.

B.1 Protocol

1. Turn on the camera.

2. Introduce yourself and explain that this focus group interview is to improve the features. Encourage them to be honest and tell them that their feedback is important and nobodys feelings will get hurt.

   Example script:
   Hi, my name is Jen. This focus group is to help us understand how you feel about the reminders to talk/reminders to switch. Your feedback is very important to making these better. Please be as honest as you can. Nobodys feelings will be hurt by your feedback.
3. Hand out the posters and remind the students of the features.

   Example script:
   
   *Here are some popups you might have seen while you were programming. The first is the reminder to switch roles and the second is the reminder to talk to your partner. You also had a task list at the bottom, right-hand corner of NetsBlox. Do you remember those?*

4. Ask the questions. For all questions, follow up on any interesting points the students talked about.

B.2 Task/Task+ Questions

1. What went well with the reminder to talk/reminder to switch? Why? Ask for examples.

2. What did not go well with reminder to talk/reminder to switch? Why? Ask for examples.

3. What would you improve about reminder to talk/reminder to switch?

4. What do you think would have been different without the reminder to talk/reminder to switch?

5. Did you ever decide not to follow the reminder to talk/reminder to switch? Why? Ask for examples of times that they did not.

6. Read Verbatim: *We programmed NetsBlox to tell you to switch after every task. We had other students switch every 12 minutes. Which do you think you prefer and why?*

7. What do you think about the posters? Would you change anything about the icons or the sentences?

   If there is time:

8. What went well with using the task list? Researcher hands out pictures with a screenshot of the task list. Why? Ask for examples.


10. What would you improve about the task list?

Other:

11. Is there anything else you would like to discuss?
B.3 Time/Time+ Questions

1. What went well with the reminder to talk/reminder to switch? Why? Ask for examples.

2. What did not go well with reminder to talk/reminder to switch? Why? Ask for examples.

3. What would you improve about reminder to talk/reminder to switch?

4. What do you think would have been different without the reminder to talk/reminder to switch?

5. Did you ever decide not to follow the reminder to talk/reminder to switch? Why? Ask for examples of times that they did not.

6. Read Verbatim: We programmed NetsBlox to tell you to switch every 12 minutes. We had other students switch after every task. Which do you think you prefer and why?

7. What do you think about the posters? Would you change anything about the icons or the sentences?

If there is time:

8. What went well with using the task list? Researcher hands out pictures with a screenshot of the task list. Why? Ask for examples.


10. What would you improve about the task list?

Other:

11. Is there anything else you would like to discuss?
Appendix C

CS CONCEPTS ASSESSMENT

. What is your first AND last name?

. What is the name of your school?

Q29. Who is your teacher for this class?

Q30. What is your current grade level?
   - 3rd grade
   - 4th grade
   - 5th grade
Q31. Which gender do you identify with?

- Female
- Male

Q32. Which best describes you?

- Black/African American
- White
- Hispanic/Latino
- Native American/American Indian
- Middle Eastern
- Asian
- Multiracial
- Other: Please describe

Q33. How old are you currently?

- 8
- 9
- 10
- 11

Q34. How often do you participate in activities that involve coding? For example, using Scratch, Snap, Alice, NetsBlox, Osmo, CodeSnaps, etc...

- Never
- Sometimes
- Frequently
- Every Day
Q35. How often do you play computer or video games?
- Never
- Sometimes
- Frequently
- Every Day

Q36. How often do you use technology (computer, smartphone, tablet, etc.) at home?
- Never
- Sometimes
- Frequently
- Every Day

Q37. How often do you use technology (computer, smartphone, tablet, etc.) at school?
- Never
- Sometimes
- Frequently
- Every Day

Q13. If you want to write code that asks a user to type in a sentence, then reports back to the user the number of times the letter ‘e’ appears in that sentence, which of these things would your blocks NOT need to be able to do:
- Compare two letters to each other to determine if they are the same
- Display text on the screen
- Convert letters into numbers and numbers into letters
- Store user entered information
Q23.
Clap
Clap
Cheer
The above instructions can be written using loops as:
(Clap) x 2
Cheer

Which of these answers show the instructions rewritten using loops?

Do three pull-ups.
Do three pull-ups.
Drink water.
Do three pull-ups.
Do three pull-ups.
Drink water.
Do three pull-ups.
Do three pull-ups.
Drink water.
Have a snack.

○ (Do three pull-ups.
  Do three pull-ups.
  Drink water,) x 4
  Have a snack.

○ (((Do three pull-ups.) x 2
  Drink water,) x 3
  Have a snack.

○ (Do three pull-ups.
  Do three pull-ups.
  Drink water.
  Have a snack,) x 3

○ (((Do three pull-ups.
  Do three pull-ups.) x 2
  Drink water,) x 3
  Have a snack.
Q19.
A robot is going to deliver a package to an owner. Below are the steps the robot needs to take to deliver the package.
1. Locate the owner of the package
2. Follow the fastest path from the robot location to the owner’s location
3. Find the fastest path from the robot location to the owner’s location
4. Drop the package
However, there might be a small mistake in the order of the steps. Can you find the mistake?

- The order of the steps is just right
- Step number 2 should be after step number 3
- Step number 2 should be after step number 4
- Step number one should be after step numbers 2 and 3

Q22. Select the instructions IN ORDER for going to recess as your teacher would like using three steps.

Step 1
- Push in my chair

Step 2
- Line Up

Step 3
- Run wildly down the hall
- Yell "It's recess!"
- Walk with class to recess
- Bounce Basketball in classroom
Q10.

```
set x to "yes"
set y to 0
set z to 0
set y to maybe
```

Which of the following can be used to replace ??? so that the code will have z equal to no?

- [ ] set x = to 0
- [ ] set y = to 0
- [ ] set z = to 0
- [ ] does not need an additional block

Q25.

```
when green flag clicked;
if (5 > 0) {
    move 100 steps;
}
else {
    move 10 steps;
}
```

How many steps will the sprite move?

- [ ] 5
- [ ] 10
- [ ] 100
- [ ] 110
Which of these answers runs the same blocks in the same order?

- when clicked
  - repeat 3
  - move 100 steps
  - wait 1 secs
  - turn 90 degrees

- when clicked
  - wait 1 secs
  - repeat 3
  - move 100 steps
  - turn 90 degrees

- when clicked
  - repeat 3
  - move 100 steps
  - turn 90 degrees
  - wait 1 secs
  - move 100 steps
  - turn 90 degrees

- when clicked
  - repeat 4
  - move 100 steps
  - turn 90 degrees
  - wait 1 secs
Q16.
The following code needs to say 'strawberry' six times. What changes need to be made, if any, for this to happen?

- Nothing, the sprite will say 'strawberry' six times
- Block number 2 should come out of the repeat block
- Another 'say strawberry' should be added to the repeat block
- The block number 3 should go inside the repeat block

Q11.
What does this code do?

- Makes sure the value of x is not equal to 10
- Makes sure the value of x is less than 5
- Makes sure the value of x is between 10 and 5
- It always sets x equal to 5
- This code will cause an error
Q3.

```python
when green flag clicked;
set (month) to 2;
set (day) to 1;
if ((month) = 1) {
    if ((day) = 1) {
        say "Happy New Year!" for 2 secs;
    }
}
else {
    say "Happy Day!" for 2 secs;
}
```

What will the sprite say after the code is run?

- The sprite says "Happy New Year!"
- The sprite says "Happy Day!"
- The sprite says "February."
- The sprite says "January."
Q2. When green flag clicked;
set (age) to 16;
if ((age) < 16) {
    say "You can't drive yet" for 2 secs;
}
if (age > 16) {
    say "You can drive!" for 2 secs;
}

What happens after the code is run?

- The sprite says "You can't drive yet."
- The sprite says "You can drive!"
- Both of the above.
- None of the above.
What will the sprite say after code A runs? What will the sprite say after code B runs?

- Sprite A: Hello! Hello! Hello! Hello!
  Sprite B: Hello! Hello! Hello!

- Sprite A: Hello! Hello! Hello! Hello!
  Sprite B: Hello! Hello! Hello!

- Sprite A: Hello! Hello! Hello! Hello!
  Sprite B: Hello! Hello! Hello!

- Sprite A: Hello! Hello! Hello! Hello! Hello! Hello!
  Sprite B: Hello! Hello! Hello! Hello! Hello! Hello!
Q6.

Which of the following should replace ??? so that the code will say "Hello Girls and Boys"?

- □ join z and y x
- □ join y z and x
- □ join y x and z
- □ join y and z x

Q5.

a, b and c are variables with values. Which of the following can be used to replace ??? so that the code will switch the values of a and b?

- □ set tmp to a
- □ set tmp to b
- □ set a to b
- □ set b to a
Q1.

| set x to 10 | set x to 10;  
| set y to 5 | set y to 5;  
| set y to x | set y to x;  

What are the values of \( x \) and \( y \) after the above code runs?

- \( x \) is equal to 10; \( y \) is equal to 5
- \( x \) is equal to 5; \( y \) is equal to 5
- \( x \) is equal to 10; \( y \) is equal to 10
- \( x \) is equal to 5; \( y \) is equal to \( x \)
Q4. Which lines of code will result in the output saying 'ABABABCD'?

- repeat 3 {
  say "A" for 2 secs;
  say "B" for 2 secs;
  say "C" for 2 secs;
  say "D" for 2 secs;
}

- say A for 2 secs
say B for 2 secs
repeat 3 {
  say "A" for 2 secs;
  say "B" for 2 secs;
  say "C" for 2 secs;
  say "D" for 2 secs;
}

- say C for 2 secs
say D for 2 secs
repeat 3 {
  say "A" for 2 secs;
  say "B" for 2 secs;
  say "C" for 2 secs;
  say "D" for 2 secs;
}
Q8.

What will be said when this code is run?

- apple
  - apple
  - apple
  - orange
  - orange
  - orange

- apple
  - orange

- apple
  - orange
  - apple
  - orange
  - apple
  - orange

- Nothing will be said

- It will be different each time you run it
Q14.
The following code is supposed to say "15."

<table>
<thead>
<tr>
<th></th>
<th>1. set (x) to 10;</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2. set (2) to 5;</td>
</tr>
<tr>
<td>3</td>
<td>3. set (x) to ((x) + (y));</td>
</tr>
<tr>
<td>4</td>
<td>4. say (x);</td>
</tr>
</tbody>
</table>

What needs to be changed in this code for this to happen?

- [ ] Change the block number 3 to

- [ ] Change the block number 2 to

- [ ] Change the block number 2 to

- [ ] Nothing needs to be changed
Appendix D

COLLABORATION ATTITUDES SURVEY

1. What is your first AND last name?

   Please select how much you agree with the following statements.

2. I learn more when I work with others.

3. When I work with other people, we get more done.

4. I have more fun when I work with others.

5. When I work with others, everyone shares the work.

6. When more than one person works on a project, we do better.

7. I prefer to work alone.
Appendix E

COMPUTER SCIENCE ATTITUDES SURVEY

1. What is your first AND last name?

   Please select how much you agree with the following statements.

2. I would like to use coding to make something new.

3. If I learn coding, then I can improve things that people use every day.

4. I am good at building code.

5. I am good at fixing code.

6. I am interested in how code makes computer programs work.

7. Using code will be important in my future jobs.

8. I want to use coding to be more creative in my future jobs.

9. Knowing how to code computer programs will help me in math.

10. Knowing how to code computer programs will help me in engineering.

11. Knowing how to code computer programs will help me in science.

12. I believe I can be successful in coding.