Collaborative Dialogue and Types of Conflict: An Analysis of Pair Programming Interactions between Upper Elementary Students

Jennifer Tsan, Jessica Vandenberg,
Zarifa Zakaria, Danielle C. Boulden,
Collin Lynch, Eric Wiebe
jtsan@ncsu.edu,jvanden2@ncsu.edu,zzakari@ncsu.edu,
dmboulde@ncsu.edu,cflynch@ncsu.edu,wiebe@ncsu.edu
North Carolina State University, Raleigh, North Carolina

Kristy Elizabeth Boyer
keboyer@ufl.edu
University of Florida, Gainesville, Florida

ABSTRACT

In successful collaborative paradigms such as pair programming, students engage in productive dialogue and work to resolve conflicts as they arise. However, little is known about how elementary students engage in collaborative dialogue for computer science learning. Early findings indicate that these younger students may struggle to manage conflicts that arise during pair programming. To investigate collaborative dialogue that elementary learners use and the conflicts that they encounter, we analyzed videos of twelve pairs of fifth grade students completing pair programming activities. We developed a novel annotation scheme with a focus on collaborative dialogue and conflicts. We found that student pairs used best-practice dialogue moves such as self-explanation, question generation, uptake, and praise in less than 23% of their dialogue. High-conflict pairs antagonized their partner, whereas this behavior was not observed with low-conflict pairs. We also observed more praise (e.g., "We did it!") and uptake (e.g., "Yeah and…") 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 partner's contributions. The results presented here provide insights into the collaborative process of young learners in CS problem solving, and also hold implications for educators as we move toward building learning environments that support students in this context.

CCS CONCEPTS

• **Social and professional topics → K-12 education**.

KEYWORDS

K-5, pair programming, dialogue, conflict

ACM Reference Format:
Jennifer Tsan, Jessica Vandenberg,, Zarifa Zakaria, Danielle C. Boulden,, Collin Lynch, Eric Wiebe, and Kristy Elizabeth Boyer. 2021. Collaborative Dialogue and Types of Conflict: An Analysis of Pair Programming Interactions between Upper Elementary Students. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (SIGCSE '21), March 13–20, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3408877.3432406

SIGCSE '21, March 13–20, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8062-1/21/03…$15.00
https://doi.org/10.1145/3408877.3432406

1 INTRODUCTION

Collaborative learning is a complex process that involves co-constructing knowledge and maintaining shared conceptions of the current activity [39]. Prior research has established that the amount of time students spend actively communicating is correlated with the success of their collaborative learning process [3, 26, 44]. Prior work also suggests that taking an active part in the construction of solutions is important for learning [14, 15]. 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 [4, 17].

Pair programming, a paradigm in which two programmers work on one computer while taking turns at the controls, has been used in introductory CS courses and in the programming industry for over two decades [5, 6]. In pair programming, the person controlling the keyboard and mouse acts as the *driver*, while the person who is tasked with planning ahead and looking for mistakes acts as the *navigator*. Researchers have discovered the educational efficacy of collaborative programming for undergraduate novice programmers [45, 47], such as increased undergraduate retention in CS courses [35]. In light of these results, educators and researchers have begun to incorporate this approach into learning activities for middle and upper-elementary school students as well (e.g.,[11, 29]). There is reason to believe that younger students may need more support to effectively leverage the advantages of pair programming [10, 42]; for example, Lewis found that elementary students might have less interest in CS after difficult pair programming experiences [29].

The CS education community has begun to investigate how young students pair program and how to support them [10, 11, 29, 30, 42]. However, there is still little known about how elementary students' collaborative dialogue relates to CS problem solving. Years of prior research across multiple domains have shown that collaborative learners benefit from engaging in self-explanation [13], question asking [22], and uptake (building upon each others' ideas to establish common ground) [12]. Antagonistic actions and

dialogue can be hurtful and may not foster a collaborative environment [41]. 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?***

Conflict, defined in this context as one child attempting to influence the actions of another who resists or directly opposes that influence [23], can occur in any collaborative interaction. The detrimental effects of conflict can include avoidance of uncomfortable situations [34] and fear of voicing concerns [36], but certain kinds of conflicts can also play an important productive role during collaboration. For example, by prompting students to rectify differences in their perspectives which can help to improve rapport [40]. The benefits of conflict may also depend on the topic of the conflict: 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 arguments [16]. On the other hand, conflicts during the collaboration process and those related to interpersonal relationships can negatively impact members of a group [28]. 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, ***RQ2: 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 12 videos of elementary school students as they engaged in pair programming in 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 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.

2 RELATED WORK

2.1 Collaboration in CS Education

Early pair programming research focused on understanding the advantages and disadvantages of pair programming for industry professionals and undergraduate students. In prior work, researchers found that engaging in pair programming improved the overall quality of code produced by industry engineers [7], increased the academic performance of undergraduate students in introductory CS courses [46], and increased productivity in active learning labs within a CS course [33]. However, this research has also highlighted some potential challenges of implementing pair programming within these environments. One such challenge is that individual characteristics can lead to incompatibility between the partners and to conflicts [33]. In contrast to this prior work, our analysis moves from the more distal data sources such as surveys and final grades to focus directly on the students' collaborative processes using video data to better understand the finer-grained characteristics of collaborative interaction in programmer pairs.

More recent work on pair programming and group work has been focused on CS students' collaborative processes. Methodology for this type of research has ranged from annotation of students' dialogue for various acts such as expressions of uncertainty, directives, suggestions, and transactive statements [9, 30, 32, 37, 38, 41], as well as analyzing the balance of the relationships through talk time and driving [2, 18, 30, 42].

Although collaboration in CS is popular in industry, using collaborative problem solving at the elementary-school level can raise several concerns. One of the primary concerns is ensuring balance in the students' collaborative learning. In prior studies on balance of upper-elementary pair programming relationships, the researchers measured balance by documenting the distribution of different types of moves between the partners considering the participation of partners [2, 30], the skill level of each partner [42] and the way in which the learners socially positioned themselves relative to their partners [18]. Israel et al. sought to understand the content of elementary students' conversations while they collaborated in a problem-solving environment and found that students engaged in both problem-specific discussions and discussions of their achievements, as well as more general off-topic conversations [24]. Although these results provide vital information about the relationships between elementary students as they collaborate in CS, they have not yet revealed how dialogue unfolds and conflicts develop in this age group.

2.2 Conflict

Prior research has shown that some types of peer conflict during collaborative activities can be beneficial both to group dynamics and learning outcomes [1, 25, 28, 40]. Fawcett and Garton [20] found that young children with low sorting ability often improve more when paired with high sorting ability peers, based on a pre-test about sorting. They argued that this might be because of cognitive conflict that occurred during the interaction. Cognitive conflict may be productive if there is dialogue and discussion between peers, which enables learners to reorganize their knowledge and to fill in the necessary gaps. Rubin et al. [40] argued that conflict can help group members reconcile differing perspectives and ideas, and help to foster group solidarity. Lee, Huh, and Reigeluth [28] distinguished among three types of intra-group conflict (i.e. task-related, process-related, and relationship-related). They found that task-related conflict can have a positive influence on peer collaboration, while process and relationship conflicts typically have negative impacts. In particular, task-related conflict facilitates reflection, idea building, and coordination. In summary, this research suggests that task-related conflict during group processes should be encouraged, whereas process and relationship conflicts should be carefully regulated. In this work, we define conflict broadly as one child attempting to influence the actions of another who resists or directly opposes that influence [23]. The definition reflects the perspective that the "influence" in the definition can be a learner influencing their partner's perspectives. The "resistance" can be the partner opposing the perspective and the learners should work towards

resolving their differing perspectives. This definition of conflict was used to manually identify conflicts in our pair programming dataset and it encompasses all types of conflict.

3 METHODS

3.1 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 [8]—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 pairs changed each session.

3.2 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.

3.3 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) [21]. We developed a dialogue move annotation scheme (Table 1) that was inspired by previous dialogue annotation schemes used for pair programming research [2, 9, 22, 37, 41]. 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 praise because of the prominent role they have been shown to play in effective collaboration and learning [12, 13]. 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 [27]. The first author annotated the remaining moves.

3.4 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* [23].

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. 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.

4 RESULTS

We calculated descriptive statistics for each pair's dialogue and conflicts. Table 2 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.

4.1 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

Table 1: The dialogue move annotation scheme.

Tag	Description	Example(s)	Freq.
Explanation - Self (Es)	Statement explaining the student's own idea, logic, or process.	*I'm going to change it to 15 steps because it's not moving far enough*	*11.79%*
Explanation - Other (Eo)	Statement explaining other components about the project, code, or problem solving process	*The sprite is supposed to jump.*	*22.97%*
Directive / Suggestion (D)	Statement telling the student's partner to complete an action or offering an idea	*Go to motion.; Delete that block.; Let's make it jump.; How about making him dance?*	*16.92%*
Question (Q)	Asking a partner a question about the task, process, their logic, or other relevant information about the process	*What should we do next?; Where is the if block?*	*10.57%*
Uptake (U)	Statement that builds upon the partner's previous statement	*Yeah, and we can also make her jump!; Let's make the background change too.*	*0.12%*
Praise (P)	Statement that emphasizes success	*You're smart; Great job!; We did it!*	*0.23%*
Antagonization (A)	Appears to cause tension, including hurtful comments, instigating fights, prodding, putting down partner contributions, and showing annoyance with partner	*You're dumb; I'd rather work alone*	*1.09%*
Other - Related (Or)	Any move that does not fit under the categories above but still pertains to the activity	*I don't know why this isn't working.; This is hard.*	*28.22%*
Other - Unrelated (Ou)	Any move that does not fit under the categories above and does not pertain to the activity	*It's almost lunch time; Um; Hmm*	*7.12%*

Table 2: Descriptive statistics of the conflicts and dialogue moves by pair. Conflict category and balance category (see section 4.2), num. of moves, num. of conflicts, num. 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 used 1% of the time overall. The maximum any pair used Uptake and Praise was 1% of the time. 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.

Conflict Cat.	Balance Cat.	Pair ID	Pseudonyms	Utt. (#)	Conflict (#)	Exp-Self (%)	Exp-Other (%)	Directive (%)	Question (%)	Conflict Types	Used U, P, A?
Low	Low	8	Rupert & Anthony	492	1	9	23	15	12	T	U, P
	Low	3	Sandy & Melony	436	2	6	30	14	19	T	
	Low	9	Synthia & Steve	843	4	16	29	17	7	T	P
	Mid	10	Clara & Rupert	410	3	12	35	20	6	T	
	High	6	Melony & Mathew	483	0	9	17	16	16	T	U, P
	High	4	Mathew & Luke	373	2	5	19	23	14	T	U, P
High	Low	2	Sandy & Anthony	572	12	16	28	11	10	T, C, PC	P, A
	Mid	1	Luke & David	800	5	11	16	26	10	T	U, P
	Mid	5	Sandy & Mitchel	732	17	15	26	15	14	T, C, PC	A
	Mid	11	Dorothy & Tylor	1133	8	8	25	12	7	T, C, PC	A
	High	7	Sandy & Steve	872	17	14	21	11	9	T, C, PC	A
	High	12	Drew & David	1012	8	14	15	23	9	T, C, PC	P, A

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 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.

4.2 Collaborative Dialogue and Conflicts

To answer RQ2, 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

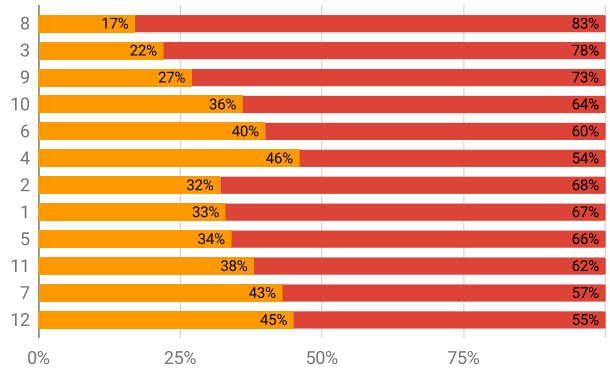

Figure 1: Talking balance of all pairs.

Table 3: Conflict excerpt of Rupert and Anthony (pair 8).

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

their moves as shown in Figure 1. We also categorized the pairs equally into low-, mid-, and high-balance with four pairs in each category as shown in Table 2. 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).

4.2.1 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. Rupert spoke less than 20% of the time. An example of pair 8's conflict is displayed in Table 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.

Next in terms of balance was pair 3 where Sandy spoke about 78% of the time. This 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.

4.2.2 High-Conflict. With the high-conflict pairs, five out of six used antagonization, with a count of 89 uses total between the five

Table 4: Conflict excerpt of Sandy and Mitchel from pair 5.

Student	Move	Tag
Mitchel	Wait I need to change	ES
Sandy	No no no.	A
Mitchel	No click turtle because ...	D
Sandy	No.	A
Mitchel	No but I'm blue.	EO
Sandy	That's nice.	A
Mitchel	But I am blue.	EO

Table 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

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 unbalanced 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 was pair 1 (David: 67% and Luke: 33%). David used uptake (1 time) and praise (4 times). With pair 5 (Mitchel: 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 shown in Table 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 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.

5 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 [12, 13, 22], 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 RQ2 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 2). 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). Based on tone of voice and body language, we believe other instances of antagonization indicate a problem that may be related to issues with personality compatibility, an issue that has been noted in prior work with older learners [33]. 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 RQ2 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!" Uptake was more difficult to detect and often contained "Yeah 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. 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 [43], 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.

Some pairs with high dialogue balance and low conflict, such as pairs 4 and 6 (Table 2), seemed to have especially productive conversation. Their dialogue balance suggested that both students were contributing equally compared to others. 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.

Finally, many of the pairs that had a lower number of dialogue moves also engaged in fewer conflicts. 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 [31]. 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 [14, 15], 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.

Limitations and Threats to Validity. Some limitations are important to keep in mind as we interpret the findings. First, though standard for this type of annotation-intense video analysis work [19, 30], we have examined a relatively small sample; the findings we have presented are based on 12 videos. Second, the students in our dataset are all Academically or Intellectually Gifted students, which may affect the generalizability of our findings. Finally, as with all video- and audio-based classroom studies, our dataset contains background noise which can complicate the analysis.

6 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 [45, 47]. 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 and 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.

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation through the grant DRL-1721160. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

REFERENCES

[1] Allen C Amason. 1996. Distinguishing the effects of functional and dysfunctional conflict on strategic decision making: Resolving a paradox for top management teams. *Academy of management journal* 39, 1 (1996), 123–148.
[2] Author. 2018. (2018).
[3] Brigid Barron. 2003. When smart groups fail. *The Journal of the Learning Sciences* 12, 3 (2003), 307–359.
[4] Miriam H Beauchamp and Vicki Anderson. 2010. SOCIAL: an integrative framework for the development of social skills. *Psychological bulletin* 136, 1 (2010), 39.
[5] Kent Beck. 1998. Extreme Programming: A Humanistic Discipline of Software Development. In *International Conference on Fundamental Approaches to Software Engineering*. Springer, 1–6.
[6] Kent Beck. 1999. Embracing Change with Extreme Programming. *Computer* 32, 10 (1999), 70–77.
[7] Andrew Begel and Nachiappan Nagappan. 2008. Pair programming: what's in it for me?. In *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*. ACM, 120–128.
[8] Brian Broll and Akos Ledeczi. 2017. Distributed Programming with NetsBlox is a Snap!. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. ACM, 640–640.
[9] Philip Sheridan Buffum. 2017. *Design and Analysis of Virtual Learning Companions for Improving Equitable Collaboration in Game-based Learning*. Ph.D. Dissertation. North Carolina State University.
[10] Shannon Campe, Jill Denner, Emily Green, and David Torres. 2020. Pair programming in middle school: variations in interactions and behaviors. *Computer Science Education* 30, 1 (2020), 22–46.
[11] Shannon Campe, Jill Denner, Emily Green, and Linda Werner. 2018. Pair Programming Interactions in Middle School: Collaborative, Constructive, Dismissive, or Disengaged?. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. ACM, 1093–1093.
[12] Courtney B Cazden and Sarah W Beck. 2003. Classroom discourse. *Handbook of Discourse Processes* (2003), 165–197.
[13] Michelene T.H. Chi, Nicholas Leeuw, Mei-Hung Chiu, and Christian LaVancher. 1994. Eliciting self-explanations improves understanding. *Cognitive Science* 18, 3 (1994), 439–477.
[14] Michelene TH Chi and Ruth Wylie. 2014. The ICAP framework: linking cognitive engagement to active learning outcomes. *Educational Psychologist* 49, 4 (2014), 219–243.
[15] Michelene T. H. Chi. 2009. Active-Constructive-Interactive: A Conceptual Framework for Differentiating Learning Activities. *Topics in Cognitive Science* 1, 1 (2009), 73–105.
[16] Ann-Marie Clark, Richard C Anderson, Li-jen Kuo, Il-Hee Kim, Anthi Archodidou, and Kim Nguyen-Jahiel. 2003. Collaborative reasoning: Expanding ways for children to talk and think in school. *Educational Psychology Review* 15, 2 (2003), 181–198.
[17] Charles Crook. 1998. Children as computer users: The case of collaborative learning. *Computers & Education* 30, 3-4 (1998), 237–247.
[18] Elise Deitrick, R Benjamin Shapiro, and Brian Gravel. 2016. How Do We Assess Equity in Programming Pairs? Singapore: International Society of the Learning Sciences.
[19] Elise Deitrick, Michelle Hoda Wilkerson, and Eric Simoneau. 2017. Understanding student collaboration in interdisciplinary computing activities. In *Proceedings of the 2017 ACM Conference on International Computing Education Research*. ACM, 118–126.
[20] Lillian M Fawcett and Alison F Garton. 2005. The effect of peer collaboration on children's problem-solving ability. *British Journal of Educational Psychology* 75, 2 (2005), 157–169.
[21] Jonathan Ginzburg. 1996. Dynamics and the semantics of dialogue. *Seligman, Jerry, & Westerst ahl, Dag (eds), Logic, language and computation* 1 (1996).
[22] Arthur C Graesser, Cathy L McMahen, and Brenda K Johnson. 1994. Question asking and answering. (1994).
[23] Willard W Hartup, Brett Laursen, Mark I Stewart, and Amy Eastenson. 1988. Conflict and the Friendship Relations of Young Children. *Child development* (1988), 1590–1600.
[24] Maya Israel, Quentin M. Wherfel, Saadeddin Shehab, Oliver Melvin, and Todd Lash. 2017. Describing Elementary Students' Interactions in Puzzle-based Environments using the Collaborative Computing Observation Instrument (C-COI). In *Proceedings of the Thirteenth Annual International Conference on International Computing Education Research*. ACM, 110–117.
[25] Karen A Jehn. 1997. A qualitative analysis of conflict types and dimensions in organizational groups. *Administrative science quarterly* (1997), 530–557.
[26] Ann Cale Kruger. 1992. The effect of peer and adult-child transactive discussions on moral reasoning. *Merrill-Palmer Quarterly* 38, 2 (1992), 191–211.
[27] J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics* 33, 1 (1977), 159–174.
[28] Dabae Lee, Yeol Huh, and Charles M Reigeluth. 2015. Collaboration, intragroup conflict, and social skills in project-based learning. *Instructional Science* 43, 5 (2015), 561–590.
[29] Colleen M Lewis. 2011. Is pair programming more effective than other forms of collaboration for young students? *Computer Science Education* 21, 2 (2011), 105–134.
[30] Colleen M Lewis and Niral Shah. 2015. How equity and inequity can emerge in pair programming. In *Proceedings of the eleventh annual International Conference on International Computing Education Research*. ACM, 41–50.
[31] Gabriel Mugny and Willem Doise. 1978. Socio-cognitive conflict and structure of individual and collective performances. *European journal of social psychology* 8, 2 (1978), 181–192.
[32] Laurie Murphy, Sue Fitzgerald, Brian Hanks, and Renée McCauley. 2010. Pair debugging: a transactive discourse analysis. In *Proceedings of the Sixth international workshop on Computing education research*. ACM, 51–58.
[33] Nachiappan Nagappan, Laurie Williams, Miriam Ferzli, Eric Wiebe, Kai Yang, Carol Miller, and Suzanne Balik. 2003. Improving the CS1 experience with pair programming. *ACM SIGCSE Bulletin* 35, 1 (2003), 359–362.
[34] Piia Näykki, Sanna Järvelä, Paul A Kirschner, and Hanna Järvenoja. 2014. Socio-emotional conflict in collaborative learning—A process-oriented case study in a higher education context. *International Journal of Educational Research* 68 (2014), 1–14.
[35] Clem O'Donnell, Jim Buckley, Abdulhussain E. Mahdi, John Nelson, and Michael English. 2015. Evaluating Pair-Programming for Non-Computer Science Major Students. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15)*. 569–574.
[36] Andrew Phuong and Judy Nguyen. 2019. Evaluating an Adaptive Equity-Oriented Pedagogy on Student Collaboration Outcomes Through Randomized Controlled Trials. (2019).
[37] Fernando J Rodríguez, Kimberly Michelle Price, and Kristy Elizabeth Boyer. 2017. Exploring the pair programming process: Characteristics of effective collaboration. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. ACM, 507–512.
[38] Fernando J Rodríguez, Kimberly Michelle Price, and Kristy Elizabeth Boyer. 2017. Expressing and addressing Uncertainty: A study of collaborative Problem-solving dialogues. Philadelphia, PA: International Society of the Learning Sciences.
[39] Jeremy Roschelle and Stephanie D Teasley. 1995. The construction of shared knowledge in collaborative problem solving. In *Computer supported collaborative learning*. Springer, 69–97.
[40] Jeffrey Z Rubin, Dean G Pruitt, and Sung Hee Kim. 1994. *Social conflict: Escalation, stalemate, and settlement*. Mcgraw-Hill Book Company.
[41] Omar Ruvalcaba, Linda Werner, and Jill Denner. 2016. Observations of Pair Programming: Variations in Collaboration Across Demographic Groups. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. ACM, 90–95.
[42] Niral Shah, Colleen Lewis, and Roxane Caires. 2014. Analyzing equity in collaborative learning situations: A comparative case study in elementary computer science. Boulder, CO: International Society of the Learning Sciences.
[43] Niral Shah and Colleen M Lewis. 2019. Amplifying and attenuating inequity in collaborative learning: Toward an analytical framework. *Cognition and Instruction* 37, 4 (2019), 423–452.
[44] Jonathan R. H. Tudge. 1992. Processes and Consequences of Peer Collaboration: a Vygotskian Analysis. *Child Development* 63, 6 (1992), 1364–1379.
[45] Laurie Williams, Robert R Kessler, Ward Cunningham, and Ron Jeffries. 2000. Strengthening the case for pair programming. *IEEE software* 17, 4 (2000), 19–25.
[46] Laurie Williams, Charlie McDowell, Nachiappan Nagappan, Julian Fernald, and Linda Werner. 2003. Building Pair Programming Knowledge Through a Family of Experiments. In *Proceedings of the International Symposium on Empirical Software Engineering*. IEEE, 143–152.
[47] Laurie Williams and Richard L Upchurch. 2001. In support of student pair-programming. In *ACM SIGCSE Bulletin*, Vol. 33. ACM, 327–331.