The Microgenetic Analysis of Staged Peer Collaboration for Introductory Programming

Maria Altebarmakian, Richard Alterman,
Anna Yatskar, Kendall Harsch, Antonella DiLillo
Computer Science
Brandeis University
Waltham, MA

***Abstract*—A positive peer learning collaboration is one where each participant engages with their partner's point of view. The negotiations that result from such engagements are the key to a productive collaborative learning experience. This paper presents a framework for in-class programming exercises done by dyads of students whose activity was mediated by an online learning platform. To maximize the amount of positive collaboration that occurs, it is crucial to consider the design of the interaction. The overall structure of the interaction for the platform described in this paper follows the think-pair-share (TPS) paradigm. The focus of the design is on the pair stage, which is where the heart of the collaboration lies. A model is developed of the progress of interaction during this stage, which is used to explain the data collected from a study of dyads of students engaged in problem-solving activities for short periods of time. A microgenetic analysis shows that students actively engage in collaboration through considering, comparing, and negotiating alternate viewpoints for a given exercise.**

1. Introduction

Oftentimes, online learning environments attempt to integrate collaborative elements. A key to any kind of productive peer collaboration is the design of the interaction amongst the learners; including collaborative elements does not necessarily translate into productive collaborative work. The literature defines a positive collaboration to be one where participants view and discuss alternative viewpoints for solving a problem and work to negotiate the different viewpoints in order to deepen their understanding of the material [1], [2]. Even "good" learners can struggle in virtual collaborative settings if the interaction is not effectively designed [3]. A poorly designed interaction will result in extra work and frustration for the student collaborators, which may lead students to adopt divide-and-conquer strategies or avoid collaborating altogether, and thereby significantly reduce learning. Thus, designing the collaboration is critical to producing constructive peer-enabled learning.

This paper will explore a framework for dyads of students working together in class, during short problem-solving sessions to improve their programming skill. We will show that the framework we propose encourages negotiation to occur between students as they solve programming exercises. We will present a model of the progress of interaction: simple correction, understanding a partner's point of view, comparing points of view, and negotiation. A study applies this model to the analysis of dyads solving problems. The platform used is based on the notion of think-pair-share (TPS), which mixes individual and collaborative elements. Our focus is on the *pair* stage where students actively engage in peer collaboration during a short period of time.

The data shows all dyads achieve a significant level of collaboration, varying from the syntactic and semantic elements of Java to the algorithmic aspects of the problem they are solving. The discussion sums up the findings and describes the use of the platform to support dyads doing logic puzzles. The point here is that the conversational structure imposed by our version of the TPS frame, continues to support the achievement of "true" collaboration regardless of the problem to be solved.

2. Background

Both learning individually and learning collaboratively have their advantages [4].

Learning individually gives students a sense of ownership over their work and knowledge and allows students to work within their particular learning style. As such, it may leave students feeling more invested in the work and potentially more engaged.

There are also advantages to learning collaboratively [5]. Learning collaboratively allows for students to work within their "zone of proximal development" [6]. The zone of proximal development is defined as the space of problems that a learner can complete within a collaboration that he could not complete alone.

> "It is the distance between the actual developmental level as determined by independent problem solving and the level of potential development as determined through problem solving under adult guidance or in collaboration with more capable peers."
> [6, p. 86]

Within a peer collaboration, learners bring in various skills that are potentially complementary and, thus, different learners are "more capable" during various stages of the collaboration. As such, they have the capacity to learn from each other and increase the sum of their skills. As students collaborate, they externalize and construct different viewpoints about a concept as they develop their skill [7], [8], [9]. Through this process, they are exposed to multiple perspectives for the problem at hand and can sample from the differing perspectives to deepen and solidify their individual understanding.

Collaboration is defined as "a coordinated, synchronous activity that is the result of a continued attempt to construct and maintain a shared conception of a problem" [1]. Computer-supported collaborative learning features technology that mediates the learning process. The collaboration between students is emphasized and the "learning takes place largely through interactions among students" [1]. Students co-construct a joint problem space through their interaction [10] as they work to make sense of the problem and the relevant concepts for solving the problem. In this context, the technology should foster the collaboration and allow students to engage and interact with their peers as they learn a given concept.

The two crucial features of a successful peer collaboration are that the students have the opportunity to consider different points of view [8], [11] and engage in negotiation [2].

A key feature of a successful collaboration centers on the idea of negotiation of different viewpoints between participants of the collaboration. Dillenbourg argues that "a main difference between collaborative interactions and an hierarchical situation is that one partner will not impose his view on the sole basis of his authority, but will - to some extent - argue for his standpoint, justify, negotiate, attempt to convince" [2]. Participants in the collaboration may negotiate on the task itself or they may negotiate on how to proceed in the interaction and it is this negotiation that enhances the quality of the collaboration and boosts the learning of participants.

It is not necessary for individual and collaborative learning to be mutually exclusive. As each has its own specific advantages and drawbacks, a balance of individual and collaborative features may provide the student with the largest potential for learning.

3. Design of the Collaboration

The focus of this paper will be on a specific implementation for the middle stage, *pair*, within the context of the think-pair-share (TPS) framework [12]. In the TPS approach, students develop individual thoughts during the *think* stage before they *pair* up and discuss their ideas with a partner and, ultimately, *share* their thoughts with the entire class. Studies show the TPS method improves the depth of in-class discussions [13].

This paper explores whether, within a TPS model implemented for an introductory programming class, students regularly achieve the collaborative ideal of the negotiation of alternate points of view. A conversational structure for TPS is presented where the *pair* stage has two phases. The data shows that within that structure, in a limited amount of time, students are regularly considering each others' point of view and negotiating their different understandings of the syntax and semantics of Java and problem-solving over the algorithmic elements of the assigned problem.

3.1. TPS for In-class Programming Exercises

Pair programming is an example of an alternate design for how, during an in-class programming assignment, collaborating students could work together to improve their skill [14]. With a pair programming model, students work in pairs to come up with a single solution to a given programming problem. One student writes code while his partner provides a commentary of ideas and questions as they go along. At some point during the interaction they may switch roles, but at any one time, one partner is the coder and the other partner provides comments for the coder.

Although pair programming has proven to be a very productive form of collaboration, an issue for its classroom use is striking the right balance between individual and collaborative learning. The collaboration between some pairs of student programmers will be balanced but others will be dominated by one partner. In the interpretation of the TPS paradigm that is discussed in this paper, there is a balance between opportunities for individual and collaborative learning.

As an approach to learning to program, this study interprets the TPS model in the following manner: the instructor presents a coding problem, the students work on the problem individually, pairs of students discuss their individual solutions, and finally the instructor steps through a final solution with the entire class. All parts of the peer interaction are mediated by the platform built for this study. The instructor sets up coding exercises in advance of class. During class, partners are randomly matched and communication between dyads occurs vis-a-vis a chat feature of the platform; this arrangement makes it easier for students to later review their in-class work and leaves open the future possibility of matching students based on their profiles.

First, students are given time to *think* about their approach to the problem and use a simple text editor on the platform to individually write code for the exercise. As this is meant for in-class exercises, the instructor will dictate the amount of time given to each stage and can move the students forward through the stages.

Once the *think* stage is complete, students are anonymously placed into pairs and are shown the code of another student in the classroom. The *pair* stage (see Figure 1) is broken up into two parts: commenting, which includes reading, and discussing. First, partners proceed by providing line-by-line comments on their partner's code. In doing so, the partner doing the critical reading gets to choose the entry points for the subsequent discussion – an *entry point*

is defined as "a structure or cue that represents an invitation to do something" [15].

1. Read and comment on each others' code *(creates an entry point)*.
2. Review and discuss each others' comments.
   a. Partner A reviews and discusses Partner B's comments on Partner A's code.
   b. Partner B reviews and discusses Partner A's comments on Partner B's code.

Figure 1. The pair stage

Then the discussion phase of the pair begins in which partners have the ability to chat regarding the comments their partner left on their code. First the code and comments of one partner is discussed, then the code of the other partner. The discussions are anchored in entry points provided by the comments and the author of the code chooses to *uptake* [16] by beginning a conversation.

Finally, the instructor *shares* a final solution and students have the ability to take notes. This stage provides the instructor with an opportunity to discuss the correct solution with her students.

To summarize: this paper excavates the critical features of the design that contribute positively to peer collaboration during the *pair* stage. The comments created during the first phase of the *pair* stage serve as entry points into the code that was written by each of the students. Each entry point can initiate a set of collaborative learning events [17] during the discussion phase. The choice to explore an entry point constitutes an uptake [16].

4. Study

For this study, transcripts of 12 dyads of students completing exercises within the framework described in section 3.1 were collected. Of those dyads, 8 completed a coding exercise in Java and 4 completed a logic puzzle. The purpose of the logic puzzle is to confirm that the significant elements of the design generalize to other types of problems. All participants in the study were undergraduate and graduate computer science students. Dyads consisted of two undergraduate students, two graduate students, or one undergraduate student and one graduate student. Dyads completed either a coding exercise or a logic puzzle.

The platform was built using Ruby on Rails which was configured to collect all of the necessary data. Complete transcripts of their work on the platform was collected including clickstream data.

A model of the collaboration is extracted that characterizes the interactions observed amongst the dyads as they completed the coding exercises. The analysis of the collaboration to complete logic puzzles confirms that the model has relevance in other domains of joint problem solving activity that can be scaffolded by the platform developed for this study for in-class problem solving.

A microgenetic analysis is presented that will document many of the ways in which the students work together and will show that partners engage in meaningful discussion of each others' code, gain practice critically reading the code of others, and receive answers to specific questions about topics in which they may be unclear of in their understanding. The conversational structure provided by the TPS platform provides a basis for significant collaboration. The dyads of students consider each others' points of view, make comparisons to their own, and negotiate their individual understanding of the problem and its solution. These elements of the collaboration occur throughout the *pair* stage.

The following list of statements are the significant findings in the data with regards to peer collaboration (see Figure 2). These findings characterize how the "conversational infrastructure" of the interaction design functions to provide a basis for successful peer collaboration during class over short periods of time.

i. Partners regularly consider each others' point of view and compare it to their own.
ii. The consideration of alternate points of view occurs in both *phases* of the *pair* stage.
iii. The consideration and comparison of alternate points of view occurs before negotiation.
iv. Conversation is a necessary but not sufficient condition for negotiation.
v. Negotiation depends on the prior consideration and comparison of alternate points of view.

Figure 2. Five significant findings in the data.

5. Model of Interaction Progression

This section presents a model for the progression of a collaborative interaction during the *pair* stage: simple correcting, understanding the partner's point of view, comparing points of view, and negotiating. Each subsequent step, or *level*, represents progress and a deepening of the collaboration between the students. For example, negotiation between points of view is more complex and potentially more rewarding than simply correcting or understanding a partner's point of view. Nevertheless, the data shows that the earlier steps of collaboration provide a fertile ground for the latter steps.

For each level, for expository reasons, we begin with a very simple, made-up example that will make it easier for the reader to grasp the concept. These simple examples are followed by more complicated segments of text extracted from the data. The simple, made-up examples are italicized; examples from the data are <u>not</u> italicized.

5.1. Simple Correcting

The first level of the model centers on simple corrections made between partners. With coding problems, this is oftentimes centered on minor syntax errors in the language or

corrections of the conventions traditionally expected when writing code in this language. An example of a simple correction would be:

You forgot a semi-colon at the end of this line.

The comment signals an error in the code. The error could be a result of lack of knowledge by the coder or perhaps an innocent mistake as a result of writing code in a short amount of time. While these types of conversations can signal learning events and are important for the knowledge of the student, they do not involve negotiation and do not serve as a precursor to achieving negotiation.

Below are some examples of comments from the data that fall within the scope of a simple correction:

> "need to clarify return type and use '{' instead of ':'"

or

> "for loop should be in form for (int i = 0; i >somenumber; i ++) {"

In both of the above examples, we see one partner sharing syntactic knowledge with the other. Generally, in the discussion phase, these comments are not selected for uptake, or, if they are, the conversation is a simple "Thank you" or some form of confirmation and agreement with the correction.

5.2. Understanding the Partner's Point of View

The next level toward negotiation of ideas revolves around each partner understanding the approach their partner took. At this level, partners begin to work through the approach of their partner. They try to make sense of the logic of and thought-process behind the code they are critically reading. An example of this would be:

Using substrings instead might be more efficient.

Understanding the work (i.e. their code) or contribution (i.e. comment) of a partner is the first step toward negotiation. In order to negotiate viewpoints, partners must, at some level, understand the reasoning behind each others' work.

An example of a comment from the data evidencing the effort one partner is making to understand the other partner's point of view:

> "This works fine as well, but you could have also solved it with the .contains() method so you don't have to sort both strings."

Here, the commenter is providing evidence that they understand what their partner was trying to do. They do this by providing an alternative idea as to how to proceed in a potentially simpler way. In order to be able to provide an alternative, they must first make sense of the approach their partner is taking and suggest an alternative that is reasonable.

An example of a conversation from the data that falls within this scope follows:

1 **Partner Comment:** It's good to consider the empty string case.
2 **Author of Code:** This is actually not the empty string case. I just created a new string object to store the reversed string.
3 **Partner:** Yes, it is an initialization. And I later thought that you consider the empty string case in the base case
4 **Author of Code:** Yes, right.

Line 1 is a comment left on the code that shows evidence that the commenter is working to understand the code presented to him. This serves as an entry point. In line 2, the author of the code chooses to uptake in order to clarify her approach. In lines 3 and 4 the partners further solidify both of their understandings of the approach. By the end of the conversation, both partners have agreed that they both understand the approach of the author of the code.

5.3. Comparing Partner's Point of View to Your Own

A subtle, but important level up from developing an understanding of the approach taken by your partner is to begin comparing the approach taken by your partner to your own approach. Perhaps your partner's approach is more efficient or perhaps your approach incorporates the use of a valuable data structure that simplifies the code. Pointing out portions of the code and drawing comparisons between the approaches marks an important step toward negotiating your ideas. An example of this would be:

This is much clearer than the method I wrote.

These comparisons mark entry points into a potential negotiation about the variations between the approaches of each partner.

One example of such a comparison taken from the data follows:

> "I see that you first put the last character to the front, and reverse for the rest. I just did the opposite direction."

Here, the first part of the comment conveys understanding of the partner's approach and transitions into a comparison with their own view of the problem.

For example, from the data:

1 **Author of Code:** I was going to make another method
2 **Partner:** Ohh alright. I got stuck writing my own with a lot of for loops. It was quite messy
3 **Author of Code:** That's smart, though
4 **Author of Code:** I mean I was going to have a bunch of for loops in different methods

Here we see that the partners have two different approaches. In line 2, the partner alludes to their individual approach, going a step beyond simple stating that he understands the approach of the author of the code. They discuss the different choices they made in solving the problem in lines 2-4 but there is no evidence that they are integrating their approaches and reaching a negotiation at this point.

5.4. Negotiation

The final (highest) level in this model is negotiation. At this level, partners work together to compare ideas and

negotiate their different approaches in order to arrive at a deeper understanding of the problem at hand. For example:

> *Partner A: This is much clearer than the method I wrote.*
> *Partner B: I think your method was very elegant. I wish I had thought of using substrings instead of trying to use recursion.*
> *Partner A: Perhaps using substrings within a recursive method would produce a more efficient piece of code.*
> *Partner B: It could be.*

In the data, we observe that achieving the negotiation level requires that the partners engage in a conversation; the data shows negotiation occurs only during the conversation of the second partner's code. Partners go beyond a simple comparison of their approaches. They begin to contribute and integrate their individual knowledge as they proceed to learn more about the problem at hand.

In the discussion section, three extended examples extracted from the programming exercise data are shown.

6. Summary of Results

6.1. The Eight Dyads who Completed Java Exercises

Of the dyads that completed a coding exercise there were 52 total comments written and 27 discussions on those comments. Below we organize the results in terms of the 5 claims made in Figure 2.

Partners regularly consider each others' point of view and compare it to their own. Of the 8 coding dyads, there were 49 occasions where a student considered her partner's point of view or compared her partner's point of view to her own: 36 of these occurred during the commenting and 13 of these occurred during the conversation. The team that spent the most time considering and comparing points of view did so 11 times in the commenting and conversation phases combined. The least collaborative team by this measure did it 3 times.

The consideration of alternate points of view occurs in both phases of the pair stage. There is evidence of consideration of alternate points of view and comparison between points of view in both the commenting and discussion phases. Of 52 comments, 36 show evidence of considering and comparing alternate viewpoints between partners. Of the 27 discussions, 13 show evidence of this.

The consideration and comparison of alternate points of view occurs before negotiation. Of the 8 dyads, 5 negotiated. Of those 5, all of them showed evidence of consideration and comparison of points of view prior to reaching the point of negotiation.

Conversation is a necessary but not sufficient condition for negotiation. None of the dyads showed evidence of negotiation until the discussion phase of the pair stage. While all of the dyads engaged in conversation in the discussion phase, not all of the conversations showed evidence of negotiation. Of 27 discussions, 5 conversations showed partners negotiating.

Negotiation depends on the prior consideration and comparison of alternate points of view. Of the 8 dyads, all reached the point of considering alternate points of view, 6 reached the level of explicitly comparing points of view, and 5 reached the level of negotiation. None of the dyads showed evidence of negotiation who didn't previously do some level of understanding or comparing of points of view.

6.2. The Four Dyads who Completed Logic Puzzles

The students who completed logic puzzles on the platform also reached negotiation levels. While the problem is slightly different and the interaction proceeds differently as a result, all of the pairs for the puzzle problems displayed evidence of negotiation in the discussion phase.

Of the dyads that completed logic puzzles there were 24 total comments written and 8 discussions on those comments. Below we organize the results in terms of the 5 claims made in Figure 2.

Partners regularly consider each others' point of view and compare it to their own. Of the 4 coding dyads, there were 30 occasions where a student considered her partner's point of view or compared her partner's point of view to her own: 23 of these occurred during the commenting and 7 of these occurred during the conversation.

The consideration of alternate points of view occurs in both phases of the pair stage. There is evidence of consideration of alternate points of view and comparison between points of view in both the commenting and discussion phases. Of 24 comments, 23 show evidence of considering and comparing alternate viewpoints between partners. Of the 8 discussions, 7 show evidence of this.

The consideration and comparison of alternate points of view occurs before negotiation. Of the 4 dyads, all 4 negotiated and all of them showed evidence of consideration and comparison of points of view prior to reaching the point of negotiation.

Conversation is a necessary but not sufficient condition for negotiation. None of the dyads showed evidence of negotiation until the discussion phase of the pair stage. Of the 8 conversations, 4 show evidence of negotiation.

Negotiation depends on the prior consideration and comparison of alternate points of view. Of the 4 dyads, all of them explored each level of the model. They all considered alternate points of view, compared points of view, and negotiated. In each case, the consideration and comparison levels occurred before the negotiation was observed.

7. Discussion

7.1. Understanding and Comparing Alternate Viewpoints is Preparatory for Negotiation

The data shows that negotiation, which is the highest level of collaboration (see Section 5), is not necessarily

preceded by all three of the lower levels. For example, in the following interaction, only two comments were written prior to the negotiation and both of them provided evidence of understanding their partner's point of view (level 2). Prior to the negotiation, there is no evidence of comparing points of view (level 3). In other words, the interaction at level 2 is sufficient preparation for the negotiation that subsequently occurs between the partners' different points of view:

1 Author of Code: The for loop just runs through each letter of word1 and word2.contains() just checks that if that index of word1 is in word2. If it's not, then they aren't anagrams, but if it makes it to the end of the loop without returning false, they are
2 Partner: Very good and clever!
3 Author of Code: Thanks.
4 Partner: Actually what if word 1 has ababab and word two is bbbbbb. Wouldn't the method fail to work?
5 Partner: sorry word1 and word2 reversed
6 Author of Code: yeah, that's true. if both words have the same characters then it will say that it's an anagram even through they're in a different order
7 Partner: right, but they may not be anagrams because if word one is bbbb and word two is abba, the words won't be anagrams, but word two will contain b for all bs.
8 Author of Code: yeah, that's true! overlooked those cases

While these partners are both working to understand the code written, the partners don't show any evidence of comparing their two points of view. In lines 1 to 3 the partners believe the code they are discussing serves its purpose and solves the problem at hand (level 2). However, in line 4, the partner presents a case that may not have been considered before, i.e., they begin to negotiate (level 4). At this point, the partner is negotiating the knowledge she has and the concept conveyed through the code and realizes there is a mistake. When she points it out to the author of the code, he must negotiate between his understanding of the concept and the new information provided by his partner. They enter a negotiation in which they are both contributing individual knowledge about the problem in order to make progress toward critically examining the code.

More often, however, conversations proceed through each level before reaching the negotiation level.

1 Comment by partner: I think you means to put p1m == p2m instead of phrase1 = phrase2
2 Author of Code: yes, that's something that would have killed it no matter which language it was, good catch
3 Partner: I'm confused because I don't know python, did you create a set or an array in lines 3 or 4?
4 Partner: Because your code made me realize a set that allows for duplicates would be a very fast way to solve this.
5 Author of Code: It was meant to be a dict, where each key-value pair would be a letter and the number of times it appears, so if all the same letters occur in each phrase at the same rate, it would return true
6 Partner: Would the java equivalent of that be a map?
7 Author of Code: I think so, yeah
8 Partner: Interesting idea. That's a much better way than I did it
9 Author of Code: Yea, something I just realized about your method is that it evaluates phrases that are identical (or written backwards? i can't remember), but not phrases that are anagrams with letters in a different order

The comment in line 1 is a simple correction (level 1) which the author of the code immediately agrees to and acknowledges in line 2. In line 3, the partner is working toward understanding the approach taken by the author (level 2). The author of the code in this interaction wasn't comfortable with Java syntax so she wrote out her individual code in pseudocode that resembled Python. She explains her strategy for the problem in line 5 and her partner integrates it with his knowledge about the Map data structure in Java in line 6 and, through the conversation, both partners increase their knowledge related to how to solve the problem at hand. They also compare approaches as evidenced in lines 8 and 9 (level 3). They are sharing, negotiating, and integrating information about the concepts behind solving the given coding exercise (level 4).

7.2. Negotiation Only Occurs in the Conversation of Partner B's Code

With the coding dyads, negotiation was only observed in the data during the discussion of Partner B's code. To illustrate this point, we will walk through the stages of the interaction of one dyad of students, Partner A and Partner B:

First, both partners critically read and comment on the code of each other. The comments are largely minor corrections (level 1) on the code such as:

"don't forget to capitalize your class name!"

or evidence of one partner working to understand their partner's approach (level 2):

"I assume you were going to check if each char of s2 existed in letters?"

Then the partners proceed to discuss the comments on Partner A's code.

At this point, the partners converse about 5 different comments and most of those conversations are simple confirmations of minor errors that are pointed out by the comments (level 1). For example, in response to the first comment above, Partner A responds with:

"Ah! You're right!"

There is also evidence of comparing approaches (level 3) in this phase:

1 Comment by Partner B: I couldn't remember if "toArray" was a method (or if that's what it was called)
2 Partner A: It's not. I was going to make another method
3 Partner B: Ohh alright. I got stuck writing my own with a lot of for loops. It was quite messy
4 Partner A: That's smart, though
5 Partner A: I mean I was going to have a bunch of for loops in different methods

As discussed previously, partner B leaves a comment that alludes to a shortcoming of his memory of available Java functions. This provides some evidence that he is comparing his approach to that of his partner's (level 3). This becomes even more explicit in line 3 where Partner B conveys understanding of what Partner A was trying to do and continues to draw parallels between that and what he did in his solution. In lines 4 and 5, Partner A proceeds by comparing what she did with the idea that Partner B just conveyed.

However, negotiation (level 4) is not reached until Partner B's code is discussed:

1 Comment by Partner A: Oh right charAt is a thing! I was going to use substring
2 Partner B: I wanted to use substrings to replace each letter in the original string to avoid creating another data structure, but I couldn't remember enough about string operations
3 Partner A: How would you have done that to find anagrams, though?
4 Partner A: like, replace each corresponding letter as it came up?
5 Partner B: Once I found a common letter, I was going to replace it with a symbol or something
6 Partner B: yeah exactly. then if a letter came up twice I could find it a second time
7 Partner A: ohh and then see if there were any characters that weren't the symbol in the end?
8 Partner B: yup! I was going to try to do something similar with the array, changing each value to null as I found it
9 Partner A: that's smart
10 Partner B: but I think that would have caused a lot of extra loops versus doing it with a substring
11 Partner A: but that accounts for the "what if they have the same letters and same length but different repeated letters" question
12 Partner A: You could have used a map maybe
13 Partner A: idk how well it would work
14 Partner B: yeah that would have been good. like some kind of data structure that holds a letter and a count for that letter
15 Partner A: you could have made an array size 32
16 Partner A: and put in numbers for each location
17 Partner A: with each being a letter
18 Partner B: oh that's smart
19 Partner A: so like abcc would be [1, 1, 2, 0, 0, ...]
20 Partner B: I like that. That seems a lot more efficient
21 Partner A: and then count down as you come across those numbers

In line 1, the comment provides evidence that Partner A is understanding the approach of Partner B (level 2) and comparing it to her own approach (level 3). As they move through their conversation, it becomes clear that they are negotiating their individual knowledge to come up with an efficient solution to the problem at hand. For example, in line 12, Partner A presents the idea of using a Map data structure but, in line 13, admits that she's not sure how well it would work. In line 14, Partner B goes on to say that he thinks it would be a good idea. From there they continue developing the idea of a data structure that maps characters with a count of the number of times the character appears in a word (level 4). In this conversation, they are clearly negotiating their own ideas and consolidating their knowledge in order to make progress.

With all of the coding dyads, negotiation was only observed in the later conversations that occurred on Partner B's code. The transcript reads as though both partners needed to be recursively aware [18] that they had been exposed to each others' point of view before they could begin negotiation. In other words, real-time, explicit feedback during the interaction is necessary prior to negotiation.

Interestingly enough, this pattern of interaction was not replicated in the logic puzzle data. With the logic puzzles, partners began negotiating in the discussion on Partner A's solution. The phases of the pair stage function differently. The commenting phase was sufficient for each partner to independently reach an understanding of their partner's approach; a conversation was not required to achieve this depth of understanding. Thus, beginning with the discussion of Partner A's solution, the dyad could begin to negotiate from scratch an improved approach to solving the puzzle. We believe this is a result of the complexity of the puzzle-solving versus Java code and not a function of the particular skills and talents of the students.

8. Concluding Remarks

In order to maximize the amount of positive collaboration that occurs on an online learning platform, it is crucial to consider the design of the interaction. For a potential collaboration to be positive, it must allow students to consider and negotiate alternate viewpoints. This paper explored a framework for dyads of students working together in short sessions to complete in-class exercises. A microgenetic analysis showed that students actively engaged in collaboration

through considering, comparing, and negotiating alternate viewpoints for a given exercise. We are currently planning a larger study with a more systematic collection of data for student work using the platform throughout a semester.

Of the 8 coding exercises completed by pairs of students 5 of them reached a negotiation level by the end of the exercise. All of them achieved this level toward the end of their discussion phase. The commenting phase largely consisted of simple corrections (level 1) and attempts to understand the viewpoint of a partner (level 2). In the discussion phase, partners tended to move toward the higher levels of intersubjectivity (levels 3 & 4). A similar progression of the interactions are observed with the dyads that completed logic puzzles.

References

[1] G. Stahl, T. Koschmann, and D. Suthers, "Computer-supported collaborative learning," in *The Cambridge Handbook of the Learning Sciences*, R. Sawyer, Ed. New York, NY: Cambridge University Press, 2006.

[2] P. Dillenbourg, "What do you mean by collaborative learning," *Collaborative-learning: Cognitive and computational approaches*, vol. 1, pp. 1–15, 1999.

[3] B. Barron, "When smart groups fail," *The journal of the learning sciences*, vol. 12, no. 3, pp. 307–359, 2003.

[4] R. Alterman and K. Harsch, "Collaborative and individual learning: Mixing the two," in *Proceedings of CSEDU 2015*, 2015.

[5] R. E. Slavin, E. A. Hurley, and A. Chamberlain, "Cooperative learning and achievement: Theory and research," *Handbook of psychology*, 2003.

[6] L. S. Vygotsky, *Mind in society*. Cambridge, MA: Harvard University Press, 1980.

[7] G. Stahl, *Group cognition: Computer support for building collaborative knowledge*. MIT Press, 2006.

[8] D. Suthers, "Technology affordances for intersubjective meaning making: A research agenda for cscl," *International Journal of Computer-Supported Collaborative Learning*, vol. 1, no. 3, pp. 315–337, 2006.

[9] S. Ludvigsen, G. Stahl, N. Law, and U. Cress, "Collaboration and the formation of new knowledge artifacts," *International Journal of Computer Supported Collaborative Learning*, vol. 10, no. 1, pp. 1–6, 2015.

[10] S. Teasley and J. Roschelle, "The construction of shared knowledge in collaborative problem solving," *Computers as cognitive tools*, pp. 229–258, 1993.

[11] G. Stahl, "Meaning and interpretation in collaboration," in *Designing for change in networked learning environments: Proceedings of the international conference on computer support for collaborative learning (CSCL'03)*, 2003, pp. 523–532.

[12] S. Kagan, "The structural approach to cooperative learning," *Educational Leadership*, vol. 47, no. 4, pp. 12–15, 1989.

[13] N. N. Azlina and A. Nik, "Cetls: Supporting collaborative activities among students and teachers through the use of think-pair-share techniques," *IJCSI International Journal of Computer Science Issues*, vol. 7, no. 5, pp. 18–29, 2010.

[14] C. McDowell, L. Werner, H. E. Bullock, and J. Fernald, "Pair programming improves student retention, confidence, and program quality," *Communications of the ACM*, vol. 49, no. 8, pp. 90–95, 2006.

[15] D. Kirsh, "The context of work," *Human–Computer Interaction*, vol. 16, no. 2-4, pp. 305–322, 2001.

[16] D. Suthers, N. Dwyer, R. Medina, and R. Vatrapu, "A framework for conceptualizing, representing, and analyzing distributed interaction," *International Journal of Computer-Supported Collaborative Learning*, vol. 5, no. 1, pp. 5–42, 2010.

[17] K. VanLehn, "Rule-learning events in the acquisition of a complex skill: An evaluation of cascade," *The Journal of the Learning Sciences*, vol. 8, no. 1, pp. 71–125, 1999.

[18] M. Tomasello, A. Kruger, and H. Ratner, "Cultural learning," *Behavioral and brain sciences*, vol. 16, pp. 495–495, 1993.