Thematic Analysis of Students' Reflections on Pair Programming in CS1

Mehmet Celepkolu Kristy Elizabeth Boyer
Computer & Information Science & Engineering
University of Florida
Gainesville, Florida, USA 32611
mckolu@ufl.edu, keboyer@ufl.edu

ABSTRACT

Pair programming is a successful approach for improving student performance, retention, and motivation toward computer science. However, not all students benefit equally from this approach. An open challenge for researchers is to develop a deep understanding of the student experience in pair programming, particularly for novices. This paper reports on a study of the cognitive, affective, and social experiences of students in an introductory programming course in which pair programming was utilized throughout the term. Students reported their experience through reflection essays written at the end of the semester. We analyzed 137 student reflection papers in a mixed-methods study. The quantitative results show that overall, students have a positive attitude toward pair programming. Looking more deeply at the reflection essays, thematic analysis revealed themes centered around cognitive, affective, and social dimensions. In the cognitive dimension, students expressed the importance of exposure to different ideas and developing deeper understanding. Affectively, students reported that working with a partner reduced their frustration and increased their confidence. Students also pointed out the social benefits of forming friendships and helpful connections. These results highlight the powerful benefits of pair programming and point to ways in which this collaborative approach could be adapted to better meet student needs.

ACM Reference format:

Mehmet Celepkolu and Kristy Elizabeth Boyer. 2018. Thematic Analysis of Students' Reflections on Pair Programming in CS1. In *SIGCSE '18: 49th ACM Technical Symposium on Computer Science Education, Feb. 21–24, 2018, Baltimore, MD, USA.* ACM, NY, NY, USA, 6 pages. https://doi.org/10.1145/3159450.3159516

1. INTRODUCTION

The high demand for computer science in a wide array of careers has driven a tremendous increase in computer science course enrollment at the postsecondary level. Every year, hundreds of thousands of students enroll in programming courses. Introductory courses are notoriously challenging, with a high failure rate [14]. Programming is a challenging task that requires high level thinking and abstraction that many students often struggle to achieve.

Previous research has suggested that student collaboration in introductory programming courses holds many benefits such as exposure to different ideas and increased motivation [10, 36]. One of the most prominent collaboration methods, pair programming, has been shown to be effective for teaching programming in introductory programming courses [9, 12]. In pair programming, two students collaboratively construct code by taking on different roles: The *driver* is responsible for writing the code and the *navigator* helps in catching mistakes and providing feedback.

Despite its effectiveness on the whole, not all pair programming interactions are successful [15]. Negative experiences during collaboration can discourage students from working in teams in the future [12]. Moreover, when a partnership does not function successfully, the pair is less productive and may fail to complete assignments [3,16]. This paper reports on a study of student reflections on pair programming, as recorded on students' reflection essays at the end of the semester. These students had gained substantial experience in pair programming, completing all weekly lab assignments in pairs (with partner assignments varying) over the course of fourteen weeks. This analysis focuses on two research questions:

Research Question 1: How positive is student sentiment toward pair programming?

Research Question 2: What are the cognitive, affective and social factors that emerge from the students' reflections about pair programming?

We have examined quantitative and qualitative (thematic) analysis of 137 students' reflection essays. Quantitative results showed that students have a positive attitude toward pair programming overall. Qualitatively, students reported many topics that affected their perception about pair programming such as being exposed to different perspectives, learning from their partners, becoming more efficient, and having less frustration. Their reflections on challenges in pair programming are also revealing, and suggest specific ways in which we may better support diverse learners through collaborative problem solving in our CS classes.

ACM ISBN 978-1-4503-5103-4/18/02...$15.00
https://doi.org/10.1145/3159450.3159516

2. RELATED WORK

Pair programming has become a well-known strategy for learning programming, and has been shown as an effective method over solo (individual) programming approaches [6,10,11,17,18]. Pair programming provides a productive environment where students produce higher quality code and typically enjoy the experience more [4], display increased confidence and perform better on exams [1]. Prior pair programming studies have focused on pairing students based on many different factors, including learners' achievement level, motivation, and gender [6,11]. There is recognition that pair programming does not work equally well in all instances and for all learners. This paper contributes to the body of research on pair programming with a deep thematic analysis of students' reflections after a full semester of pair programming in CS1.

3. METHODS

3.1 Participants

The data was collected from students who completed a CS1 course in Spring 2017 at a large public university in the southeastern United States. The class was taught in the Java programming language and had a total enrollment of 375 students. Of these 375 enrolled students, 278 students voluntarily agreed to have their data collected for research purposes. This consent was obtained at the beginning of the term. Students did not receive course credit or any incentive for consenting to data collection, and there was no penalty for declining to data collection. 180 students consented to have their data collected and we analyzed all the essays that mention pair programming, for a total of 137. The authors of the essays were 27 women (19.7%), 109 men (79.6%) and 1 unspecified (0.7%). Race/ethnicities were White (45%), Hispanic (18%), Asian (21%), Multiracial (11%), Black (2%), and Other (3%). Participants' mean age was 19.1 (range: 18-27) and there were 82 Freshmen (60.3%), 29 Sophomore (21.3%), 15 Junior (11%), 7 Senior (5.2%), and 3 graduate students (2.2%). The majority were from computing-related majors: 35% Computer Science, 26% Computer Engineering, 15% Other Engineering fields and 24% Others. 35% of students reported having no programming experience at the beginning of semester and 51% reported no prior Java programming experience.

3.2 Procedure

During the semester, students attended three one-hour lectures each week, had three exams, completed four projects and attended fourteen lab meetings. There was one lecture section and eighteen different lab sections with approximately 20 students each. In the labs, students completed pair programming exercises and these exercises comprised 20% of their overall course grade. Students were free to leave the two-hour lab after they completed the lab assignment, post-quiz, and post-survey. In the labs, each student worked with either a randomly assigned or self-selected partner depending on the lab structure of each week for the rest of semester. Students were paired with a variety of different partners throughout the semester. At the end of the course, students wrote a reflection essay about their experiences. These essays were announced at the start of the term and counted for two percent of the course grade. Students were given the following high-level prompt for the reflection paper.

The reflection essay prompt was intentionally left at a high level, to "*reflect on your learning process, your problem-solving approaches, and the course content.*" Thus, the essays cover a broad variety of topics including the instructor, course design, exams, projects, labs and teaching assistants. In this CS1 course, students who participated in at least two hours of human subjects studies received credit in lieu of writing the reflection essay, therefore not all students submitted a reflection essay.

3.3 Data

Of the 375 students, 262 wrote a reflection essay. This analysis examines essays from the 180 students who consented to have their data collected. Of these 180 essays, 137 mention pair programming. The essays that refer to pair programming as "peer programming," "partner programming" or "partnered programming" were also included in analysis. While some students discussed pair programming in 1-2 sentences, others went into detail and spent almost a page on pair programming: their positive and negative thoughts about the method, their partners and other factors. The 137 excerpts discussing pair programming were manually extracted by researchers who read the essays. The average length of a pair-programming-related excerpt was 165 words, with the longest being 861 and the shortest being 21 words.

3.4 Sentiment Analysis Method

We manually labeled each of the excerpts for positive or negative sentiment. Because labeling sentiment is a subjective task, we employed a standard inter-rater reliability methodology: Two human raters independently labeled the sentiment of each excerpt, and then the extent to which the raters agreed was computed. Sentiment was rated on a 5-point scale with 1 being the most negative and 5 being the most positive. Researchers rated sentiment based on the following criteria:

5: Completely positive. When the sentiment was completely positive and there were no any negative comments/thoughts. For example, "*I feel that the pair programming model is a very good practice to engage in. I found that it made me very comfortable working with a partner, which is an important quality to carry into the workplace. Also, explaining your code to your partner requires you to really understand your code and thus pushed me to really further understand concepts and techniques. Thus, I think lab was extremely helpful and allowed me to ask more questions than lecture.*"

4: Mostly positive. Positive overall, but reporting some concerns or dissatisfaction about some part of pair programming.

3: Neutral. When the student does not pick a side and reports that he/she can work with or without pair programming.

2: Mostly negative. These students report that they would do better without pair programming.

1: Completely negative. These excerpts report a high level of frustration and dissatisfaction, and do not report any positive sentiment. For example, "*My biggest complaint with lab was the pair programming. I HATED it. It was completely unproductive.*"

After the human raters scored all 137 extracted pair programming sentiments, we compared the scores and calculated the inter-rater reliability score. The ratings resulted in an inter-rater reliability kappa score of 0.714 and a weighted kappa of 0.817, indicating substantial agreement [8].

3.5 Thematic Analysis Method

Our second research question involves the cognitive, affective and social experiences that emerge from the themes in reflection essays; thus, we employed a qualitative approach, specifically thematic analysis. Thematic analysis has been recommended by scholars as "a method for identifying, analyzing and reporting patterns (themes) within data" [2]. Thematic analysis allows researchers to explore phenomenology, which examines individuals' perceptions and understandings of a phenomenon (or situation) through interviews, stories, or observations [5].

4. SENTIMENT ANALYSIS RESULTS

The majority of students (69%) reported positive attitudes (rated as 4 or 5) toward pair programming; other students reported negative (rated as 1 and 2) or neutral (3) attitudes. The results showed that high performing students report an overall neutral-to-positive sentiment toward pair programming (N = 65; M = 3.53; SD = 1.38) while other students display a positive sentiment overall (N = 72; M= 4.14; SD = 1.15). The difference is significant (p < 0.01, Mann-Whitney U).

5. THEMATIC ANALYSIS RESULTS

We used thematic analysis to explore student reflections on pair programming through their essays. First, we performed thematic coding on 62 randomly selected reflection excerpts, in which two graduate student researchers coded the excerpts independently and generated a total of 216 independent codes. For example, "conflict between students", "beneficial in programming", "learning new problem solving approaches" are some initial codes.

Then, two graduate student researchers met to discuss these codes and collectively sorted them into themes and dimensions, collapsing highly similar codes and creating a revised set of 82 codes. In the second round, they grouped thematically similar codes from this revised set, ultimately identifying seven major themes (e.g., "partner") and nine minor themes (e.g., "lower level partner", "equal level partner"), for sixteen themes in total. Minor themes would be "lower level partner", "equal level partner" and the major theme would be "partner". A sample of the thematic hierarchy is presented in Table 1. As per standard practice in this methodology, two researchers collaboratively extracted these themes. The themes were not selected based on importance or the frequency with which they were being mentioned by students. Instead, they represent different dimensions of the factors involved in pair programming activities.

After finalizing the categorization of the major themes, we sorted the themes into broader conceptual domains: cognitive, affective and social. In this paper, we focus on the most prominent major themes in each domain by discussing the related sub-themes and codes. We also present positive and negative reflections in student quotes.

Table 1. A portion of the thematic analysis hierarchy

Revised Codes	Themes	Dimensions
Comparing my ideas with a fellow peer	Exposure to different ideas	Cognitive
Observing others' solutions		
Learn different working styles		
Reduced workload	Efficiency	
Faster Completion		
Smoother Problem Solving		
Better Problem Solving	Deeper Learning	
Mastering skills		
Refine knowledge		
Extend knowledge		
Working with strangers	Social growth	Social
Leaving your comfort zone		
Meet with new people		
Bad when partners rush	Partner	
Bad when partners don't explain		
Enjoyment	Satisfaction	Affective
Motivation		
Partner Roles	Logistics	

5.1 Cognitive Dimension

The themes in the Cognitive domain are those involved in acquisition and understanding of knowledge, decision making and problem solving process [13]. The reflection essays revealed several themes of, "Exposure to new ideas", "Deeper Learning", and "Efficiency" (with sublevels "Reduced Workload", "Smoother Problem Solving" and "Faster Completion"). We also present how students with different achievement levels express different themes in pair programming.

Exposure to new ideas: This theme refers to how students perceive exchanging ideas with other students in pair programming activities. Students often mentioned that pair programming helped them to be exposed to different perspectives during the problem solving process.

> *"One of these benefits was getting multiple perspectives on problems. Sometimes I would not know how to approach a problem or I would not know if there was a better way to solve a problem. When this happened I sure was glad that I had a partner to program with."*
>
> -Male CS major who achieved an A in the course

However, some students indicated that they had conflicts during "exchanging ideas and" with other students as they encountered communication difficulties. For example, the following quote shows how the conflicts made the process cumbersome for this student when their partner did not agree on the solution:

"... trying to connect to thought processes in two different people in a way that can create one application was an absolute nightmare for me. Often my partner and I would come up with conflicting solutions on how to tackle a problem." –Male CE major with an A in the course

Deeper Learning: This theme refers to how students evaluate their perceived learning outcomes in pair programming. Most students mentioned that trying to explain the concepts to their peers helped them better understand the content.

"Also, explaining your code to your partner requires you to really understand your code and thus pushed me to really further understand concepts and techniques." –Male CE major who achieved a B+ in the course

However, some students explained that they failed to use the same knowledge when they later worked alone.

"During the first few weeks of lab this proved to be an effective method; however, I found when I was working on code myself, while I understood the majority of the material I was just not able to piece everything together and consistently made simple errors in my code that I really had no idea at the time how to solve." – Male, non-CS major who earned a D+ in the course

Efficiency: This theme refers to students' perceptions of the extent to which pair programming helped them to understand the concepts more easily, finish the assignment faster than they anticipated, and reduce their workload. In pair programming, students work with a partner, which is expected to reduce the workload. Also, students have a second eye examining the code, catching mistakes and providing feedback; thus, the majority of students pointed out how working together reduced their workload, and helped them work more efficiently:

"I found that along with the combined set of brains that working with a partner offers, a combined set of hands can further help to reduce the work load, increase efficiency, and help develop overall problem solving skills." –Male, CS major who achieved an A in the course.

Some students reported a smoother problem-solving process thanks to their partner's guidance:

"Pair programming engrained the techniques in an efficient way, unlike last semester where I struggled on several of the assignments because the project's level of difficulty outweighed my programming knowledge."

–Female non-CS major who earned an A- in the course

On the other hand, some students pointed out that pair programming actually increased their workload and made them slow down due to the free-riding problem:

"But sometimes this person who didn't know anything was not motivated to learn anything, and just wanted it done. So basically, I would be doing the whole lab (which is fine because I like working alone), but I would ALSO be spending my time teaching this person and explaining what I'm doing because we're supposed to be 'working together', and it would just be a complete waste of time." –Female CS major who achieved an A in the course

In addition, some students mentioned that time was an important factor in their learning process. When there was sufficient time, students stated that they were more willing to help their partner, discuss the problem and try different approaches.

"It is difficult to hand the reigns to a partner that just does all the work or doesn't know anything at all. It would not be a problem if lab was not timed but unfortunately it is so if one partner wasn't following along then that meant we would most likely fall behind." –Female CS major who earned a C+ in the course

"Being put together with a partner who also really wanted to learn and who wasn't in a hurry felt so relieving! I hate feeling like I have to try to be as clever and quick as possible so I'm not holding my partner back." –Female non-CS major who earned an A in the course

5.2 Affective Dimension

The themes in the Affective dimension are those that involve feelings or emotions toward pair programming experiences. In this section, we focus on the "Satisfaction" theme with Enjoyment and Motivation sublevels.

Satisfaction: This theme refers to the fulfillment of students' expectations, their motivation to pursue their goals, their reduced frustration and their enjoyment derived from pair programming. Most students reported high enjoyment from pair programming and referred to the activities as motivating, rewarding, engaging, less frustrating and imposing less pressure on them.

"Pair programming helped me get through the lab and learn from my partner instead of just sitting there frustrated and staring at my code, too afraid to ask for help." – Female CS major who earned a B in the course

However, some students reported strong negative feelings toward pair programming:

"I really did not enjoy the peer programming part of the lab. I feel that peer programming overall slowed down the work, created annoyance between both partners, and did not improve understanding." – Male CS major who achieved an B in the course

Logistics: This theme refers to the logistical issues derived from pair programming that affected students' affective state. Some students reported that working on the same computer with another student can be problematic.

"The biggest question I had for this when it was first introduced during class was how do two people code at the same time. While the solution to that was the driver-navigator dynamic, it seemed pointless in the end. The navigator might have just spent their entire turn sitting there. I know I had my share of moments where I just sat there watching my partner code during lab." – Female non-CS major who achieved an A- in the course

5.3 Social Dimension

The themes in the Social Dimension involve students' interactions with their partners in the pair programming environment. In this section, we focus on "Social Growth" and "Partner" themes.

Social growth: This refers to students' perception of how pair programming activities helped them to step out of their comfort zone to make new friends, express themselves to more people, enable them to work with strangers, interact with their classmates and make study groups for course projects:

> *"Another advantage of peer programming [sic] was leaving my comfort zone by meeting new people. Usually I am a very introverted person. The lab in this class forced me to be more social and express myself to more people. I think that this made me a better problem solver and a better communicator."* -Male CS major who achieved an A in the course

Partner: This theme refers to students' thoughts about their actual partners, their ideal partner, and the pair assignment procedures. Partnership is one of the core components of pair programming and it can have a big impact on the success of collaboration. While some students were in favor of random assignment, some students expressed a desire to select their own partner to avoid being paired with a much lower-achieving student.

> *"... choosing partners for us was irritating to say the least. As someone who knew what they were doing (relatively), during every randomly partnered class I was either doing the problem myself and then explaining my work or forced to sit irritated that my partner could not comprehend the problem. Choosing our own partners was a welcome change as I was able to work with someone of even skill level who was able to do their own work and follow mine."* – Male non-CS major with a B+ in the course

Pair programming literature often reports on conflicts between students. Previous studies have shown that when students work with a partner with a different knowledge level, the information usually flows from the higher-achieving student to the lower-achieving student [9], and higher-performing students usually dominate the problem solving activity.

> *"... the people I got who were better skilled than me just took over the whole thing and didn't even bother trying to explain to me what they were doing."* – Female CS major who earned an A in the course

> *"For the first few pair programming sessions I was assigned a partner with very little programming experience, which I feel slowed down the process and distracted me from thinking about the project."* – Female CS major who earned an A in the course

6. DISCUSSION

These student reflections point to several areas of the student experience that warrant attention. First, one student pointed out the frustration that can stem from partners arriving at "conflicting" solutions. While this is a natural way for a student to experience a difference in problem solving approaches, it is important to help students understand that two different viewpoints will actually help to produce a stronger product if the collaborators invest the time to talk through the relative merits of their approaches and determine which one (or a combination) is best suited for the task at hand.

Second, some students felt that when they were asked to program solo (such as on projects) that the transition away from pair programming was difficult. Providing appropriate on-demand support from teaching assistants or professors is a crucial step to support students in becoming competent solo programmers, though a plethora of empirical results have shown that pair programming does not inhibit solo programming performance.

Finally, one student reflected on experiencing irritation when working with a less knowledgeable partner. This sentiment is perhaps one of the most pervasive that we have seen expressed. There is no simple way to address this negative student comment. If we pair students with high skill together, it would mitigate this problem greatly but would create other problems such as failing to build the teamwork skill of working in a diverse team. On the other hand, if we can make a strong case for the importance of peer tutoring when a mismatch in skill is present, and provide incentives and proper training, we may be able to channel what could have been frustration into a sense of achievement that is beneficial to both students.

Supporting diverse learners is of paramount importance as we consider the student experience in our computer science classes. A full discussion from a broadening-participation viewpoint is beyond the scope of this paper, but as one example, we observe a phenomenon often noted between female and male students, when the difference in confidence is not commensurate with the students' achievement in the course.

> *In the labs, I was assigned to a partner who was way more advance in this stuff than I was. He did everything and made me feel so moronic. [...] I read the chapters before coming to lab, but this stuff just didn't click."* –Female non-CS major who earned an A- in the class

> *"... my partners were fresh, first-time programmers, and it was agonizing. They barely knew the syntax or how a method is structured and it would turn into me guiding them, line-by-line, what to type. At ten o'clock in the morning I'm not in the mood to hand-hold someone for two hours through something they should already know."* – Male CE major who earned a A in the course

Limitations and Treat to Validity. During the analysis process, we focused on three dimensions, cognitive, affective and social, which are widely studied and agreed upon as important aspects of programming. We made a simplifying assumption and placed each theme into one dimension, but some themes overlap categories. For example, exposure to different ideas is a socio-cognitive construct. Another limitation which is often pointed out in qualitative research is the subjective nature of the categorizations. However, the analysis presented here makes no claim that the categories are generalizable nor that the student experiences described here are shared by all (or even most) students with the same characteristics. The key point is that these essays and the themes discussed therein represent experiences that students actually had, and therefore they are worthy of our attention as we aim to serve all students well in our CS classes. Finally, roughly 25% of the students did not mention pair programming in their reflection because we intentionally left the reflection essay prompt high-level so students could focus on what they felt was important. However, this approach may have skewed the satisfaction results.

7. CONCLUSION

Our overarching goal for this study was to deepen understanding of why some pair programming collaborations are successful and others unsuccessful. We examined how different level students' attitudes differ towards pair programming and analyzed the emergent themes from students' reflections. We have presented quantitative and qualitative (thematic) analysis of 137 students' reflection essays. We focused on three dimensions of these reflections— cognitive, affective and social— and examined the related major themes. The quantitative results show that students have a positive attitude toward pair programming overall and that high-achieving students report significantly less positive sentiment and fewer perceived benefits compared to other students.

Qualitatively, students with different knowledge levels reported enjoyment and frustration about many different topics. In the cognitive dimension, most students expressed that pair programming helped them to be exposed to different perspectives, learn from their partners, develop deeper understanding by discussing the problem, and become more efficient by reducing the workload and finding more efficient solutions. Affectively, students described the pair programming activities as being motivating, rewarding, engaging, less frustrating, and putting less pressure on them than solo activities. Socially, pair programming contributed to social growth by helping students step out their comfort zone to make new friends, express themselves to more people, enable them to work with strangers, interact with their classmates and make study groups for course projects. Students also reported that their partner can define the success of the pair programming activity: too much difference in knowledge level demotivates high-achieving students by increasing their workload, distracting them from the activity and making them slower, while low-achieving students cannot keep up with the high-achieving students' pace and feel marginalized in the problem-solving activity. This may be why, with few exceptions, most students preferred to work with a similarly skilled partner.

Considering these three dimensions of pair programming, we begin to see there are many unresolved issues that students face during pair programming. The outcomes from this study can help instructors to increase enjoyment and improve learning outcomes in pair programming. For future studies, combining reflection essay data with more data sources, such as video recordings and students' programming outputs can uncover many other important issues and help computer science educators develop a deeper understanding of students' concerns and interactions in pair programming activities.

8. ACKNOWLEDGMENTS

The authors wish to thank the members of the LearnDialogue group at the University of Florida for their helpful input, especially Jennifer Tsan and Lydia Pezzullo for their invaluable help which made this work possible. This work is supported in part by Google through a CS Capacity Research Award and by the National Science Foundation through grant CNS-1622438. Any opinions, findings, conclusions, or recommendations expressed in this report are those of the authors, and do not necessarily represent the official views, opinions, or policy of the National Science Foundation.

REFERENCES

[1] G. Braught, T. Wahls, and L. M. Eby. 2011. The Case for Pair Programming in the Computer Science Classroom. ACM Transactions on Computing Education 11, 1: 1–21.

[2] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. Qualitative Research in Psychology 3, 2: 77–101.

[3] Joseph Chao and Gulgunes Atli. 2006. Critical Personality Traits in Successful Pair Programming. In AGILE 2006 (AGILE'06), 89–93.

[4] Alistair Cockburn and Laurie Williams. 2000. The Costs and Benefits of Pair Programming. Extreme Programming and Flexible Processes in Software Engineering XP2000: 223-247.

[5] Lynne M. Connelly. 2010. What is phenomenology? MedSurg Nursing 19, 2: 127–129.

[6] Timothy H. DeClue. 2003. Pair programming and pair trading: effects on learning and motivation in a CS2 course. Journal of Computing Sciences in Colleges 18, 5: 49–56.

[7] T. Dyba, E. Arisholm, D. Sjöberg, J. E. Hannay, and F. Shull. 2007. Are two heads better than one? On the efectiveness of pair programming. IEEE Software 6: 12–15.

[8] J. Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. biometrics.

[9] Sarah J. McCarthey and Susan McMahon. 1992. From convention to invention: Three approaches to peer interactions during writing. Interaction in cooperative groups: 17-35

[10] C. Mcdowell, B. Hanks, and L. Werner. 2003. Experimenting with Pair Programming in the Classroom. In Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education, 60–64.

[11] C. Mcdowell, L. Werner, H. E. Bullock, and J. Fernald. 2003. The impact of pair programming on student performance, perception and persistence. Proceedings of the 25th International Conference on Software Engineering: 602–607.

[12] J. L. Schultz, J. R. Wilson, and K. C. Hess. 2010. Team-based classroom pedagogy reframed: The student perspective. American Journal of Business Education.

[13] J. Sweller, P. Ayres, and S. Kalyuga. 2011. Cognitive load theory.

[14] Christopher Watson and Frederick W. B. Li. 2014. Failure rates in introductory programming revisited. In Proceedings of the 2014 conference on Innovation & technology in computer science education - ITiCSE '14, 39–44.

[15] Jon M. Werner and Scott W. Lester. 2001. Applying a team effectiveness framework to the performance of student case teams. Human Resource Development.

[16] L. L. Werner, B. Hanks, C. McDowell. 2004. Pair-Programming Helps Female Computer Science Students. ACM Journal of Educational Resources in Computing 4, 1.

[17] Laurie Williams and Richard L. Upchurch. 2001. In Support of Student Pair-Programming. In Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education, 327–331.

[18] B. Zhong, Q. Wang, and J. Chen. 2016. The impact of social factors on pair programming in a primary school. Computers in Human Behavior 64: 423–431.