AUTHOR(S): ZARB, M., HUGHES, J. and RICHARDS, J.

TITLE: Evaluating industry-inspired pair programming communication guidelines with undergraduate students.

YEAR: 2014

Publisher citation: ZARB, M., HUGHES, J. and RICHARDS, J. 2014. Evaluating industry-inspired pair programming communication guidelines with undergraduate students. In Proceedings of the 45th ACM technical symposium on computer science education (SIGCSE 2014), 5 - 8 March 2014, Atlanta, USA. New York: ACM [online], pages 361-366. Available from: https://doi.org/10.1145/2539962.2538980.

OpenAIR citation: ZARB, M., HUGHES, J. and RICHARDS, J. 2014. Evaluating industry-inspired pair programming communication guidelines with undergraduate students. In Proceedings of the 45th ACM technical symposium on computer science education (SIGCSE 2014), 5 - 8 March 2014, Atlanta, USA. New York: ACM, pages 361-366. Held on OpenAIR [online]. Available from: https://openair.rgu.ac.uk.

Evaluating Industry-Inspired Pair Programming Communication Guidelines with Undergraduate Students

Mark Zarb
School of Computing
University of Dundee
Dundee, Scotland
+44 (0)1382 384 150

Janet Hughes
School of Computing
University of Dundee
Dundee, Scotland
+44 (0)1382 385 195

John Richards
IBM T.J. Watson Research Center
Yorktown Heights
New York, USA
+1 914 945 2632

markzarb, jhughes @computing.dundee.ac.uk
ajtr@us.ibm.com

ABSTRACT

A set of industry-inspired pair programming guidelines have been derived from qualitative examinations of expert pairs in order to aid novice programmers with their intra-pair communication. This research describes the evaluation of these guidelines with a set of student pairs, and demonstrates how novice pairs who were exposed to the guidelines were more comfortable communicating within their pairs.

Categories and Subject Descriptors

K.3.2 [**Computer and Information Science Education**]: Computer science education.

General Terms

Measurement, Performance, Experimentation, Human Factors, Standardization.

Keywords

Pair programming; communication skills; software engineering; collaboration; students.

1. INTRODUCTION

Pair programming is a method which describes two programmers working together on the same computer, sharing one keyboard. Typically, each member of the pair takes on a different role, swapping roles frequently: the driver creates the code while the navigator reviews it [16]. Pair programming requires its pairs to communicate frequently, which leads the pair to experience certain benefits over "solo" programming, such as a greater enjoyment, and an increased knowledge distribution [2].

Pair programming is one of the key aspects of Extreme Programming, which "favours both informal and immediate communication over the detailed and specific work products required by any number of traditional design methods" [14].

Novices find communication to be a barrier when they are pair programming, and industry-inspired guidelines have been presented as a possible solution [11, 16]. Initial student feedback with regards to their perceptions of the guidelines was positive. This research investigates an evaluation of the guidelines, in order to present evidence that they can be used to assist novice pairs in learning to communicate more effectively when working together.

2. BACKGROUND

Due to the nature of pair programming, *communication*, both verbal and non-verbal, occurs constantly. Williams and Kessler [13] write that effective communication is "paramount", whereas Sharp and Robinson [10] describe pairing as a highly communication-intensive process.

Within the classroom, pair programming is seen as being generally valuable [1, 13], with research showing that its use in educational settings has rapidly increased within the last decade, with reported usage in the US, the UK, Germany, New Zealand, India and Thailand [4]. Furthermore, students working in pairs can be seen to be more satisfied with their work output, solve problems faster than non-paired students, and have improved team effectiveness, with pairing students being more likely to complete CS courses when compared to their solo counterparts, as well as gaining an improved comprehension of unfamiliar topics, as well as increased levels of confidence [5, 7, 11, 14, 15].

Many programmers approach their first pairing experience with scepticism, having doubts about their partner's working habits and programming style, and about the added communication aspects that this programming style entails [14]. In a pilot study, roughly 50% of first-time novice pair programmers reported that they perceived communication to be the main problem with the pairing process [9].

Freudenberg et al. write that "the cognitive aspects of pair programming are seldom investigated and little understood" [3]. Furthermore, many authors simply attribute theoretical importance to *communication* as an issue – as a result, few studies have investigated the aspect of communication within an agile team in detail [10, 12].

3. PAIR PROGRAMMING GUIDELINES

The industry-inspired pair programming guidelines [18] were created with the aim of delivering concise, industry-derived instructions to novice pairs to improve their understanding of successful pair communication.

An analytic coding schema was derived from the observation of the communication of expert pairs working in industry, with over 35 hours of communication being analysed across 11 different pairs was reported in [17]. This coding schema was further analysed, and led to the creation of industry-inspired pair programming guidelines, as reported in [18]. A preliminary qualitative examination suggested that the guidelines could be used by novice pairs to aid them in their intra-pair communication; this was evaluated in the present paper, and found to be true.

The guidelines are presented in three separate sections: restarting, planning and action [18]; they are summarised in the following section:

3.1 Restarting Guidelines

These guidelines should be used when the novice pair is stuck, and cannot seem to make good progress:

- If your pair is stuck in a thinking/silent loop and cannot seem to progress, actively break your focus by discussing something completely off-topic and unrelated to the issues at hand. This will allow you to tackle the problem with a fresh outlook. Following this stage, attempt to:
    - Look back on your last couple of steps and review your previous work;
    - Try to suggest next steps related to your end-goal in order to make progress;
    - Identify a fresh thought process.
- If your partner is attempting to break focus, don't dismiss this. Breaking one's focus using jokes, private conversations, etc. can lead to a fresh perspective, which your partner may need.

3.2 Planning Guidelines

These guidelines should be used when the novice pair needs to review legacy code, or start to plot future steps:

- Suggestions and reviews are optimal states that will allow you to drive your work forward. When in these states, feel free to alternate (e.g. review previous code, suggest an improvement, review methods to be changed, suggest potential impact).
- At each stage, do not hesitate to ask your partner for clarification as to what they are working on, or suggesting.
- Think about what your partner is saying and doing. Offering an explanation of the current state can help move the work forward.

3.3 Action Guidelines

These guidelines should be used when the novice pair needs to discuss code logistics, or start to write code:

- NAVIGATOR: While the driver is coding, actively look to make suggestions that contribute to the code.
- NAVIGATOR: If the driver is muttering, use this opportunity to make sure your suggestions have been properly understood.
- DRIVER: While you are programming, or thinking about your code, voice your thoughts (even if it is just mumbling and muttering while you're typing). This helps the navigator know that you are actively working, and will allow for them to make suggestions based on your current actions.

4. EVALUATION

A class of undergraduate third years was exposed to the guidelines during one of their taught modules ("Agile Software Engineering"). Exposure occurred during a 60-minute lecture where the guidelines were introduced by the lecturer and discussed within the class. A printed copy of these guidelines was given to the students for reference whilst they completed their lab work in the following weeks. Initial feedback was largely positive, as evidenced by the following quote:

> *"There's a definite benefit in introducing this. In pair programming, we're told to 'work in pairs: go!', and there weren't formal steps, apart from the fundamentals. There wasn't a lot of what to do if you became stuck."*

At this stage, it is imperative to understand whether there is any quantitative value to be gained by using these guidelines with novice pairs, and whether this exposure can improve novice intra-pair communication. For this purpose, a study was devised with the aim of producing quantitative data, with the following hypotheses:

- Exposure to the pair programming guidelines positively impacts the pair's success rate.
- Exposure to the pair programming guidelines leads to an improvement in the pair's ease of communication.
- Exposure to the pair programming guidelines positively affects the way partners contribute to the pairing session.

4.1 Methodology

In 2010, Murphy et al. researched conversations within pairs, focusing on transactive statements that were primarily about debugging. Their initial study was carried out with ten undergraduate Java students, and the authors note that "the pairs that talked more […] attempted to solve more problems" [6]. Due to the study's focus on communication, the measurement of pair success, and a validated methodology, a similar process was proposed for this study, borrowing elements described in Murphy et al.'s research. The authors were contacted, and gave consent for permission to replicate the methodology, as well as providing a copy of the buggy code used in their original study.

Participants were recruited from the School of Computing at the University of Dundee, and a local college. An e-mail was circulated to all students, asking for their participation in exchange for a small compensation in the form of vouchers. Twelve students were recruited from Level 1, and eight students from Level 3, with six students studying at college-level.

All students were randomized into pairs – with each pair consisting of students who were matched to peers within the same level of study. A control group was set up, consisting of 12 students (6 pairs) from across all three levels of study, leaving an

experimental group of 14 students (7 pairs). Informed consent was obtained from all participants.

The study was carried out during a 2-week period during the students' second semester of study. Pairs were invited to the test room separately, and on different days. The test room was equipped with one laptop, and consisted of a camera and a voice recorder. In line with the methodology used by Murphy et al. [6], all pairs were given a list of programs which consisted of buggy code. It was explained to all pairs that each program would compile – but consisted of one logical error. The pair was given 45 minutes to sequentially locate and fix as many errors as possible. During this time, the recording equipment was switched on, and the researcher left the room.

Both the control group and the test group followed the process described above; prior to the task, pairs within the test group were exposed to the pair programming guidelines by means of a verbal presentation, as well as a short 3-minute video showing an experienced pair applying the guidelines to try and overcome various situations.

Following the test period, the researcher would return, log the number of programs completed by the pair, and distribute post-test surveys, which were completed individually by the members of the pair. Each survey consisted of Likert-scale questions relating to their experience with communication and partner contribution during the test, as well as questions on the student's experience with programming, which were used to measure central tendencies and variance within the groups, in order to ascertain that there were no significant difference between the groups which would lead to threats to validity.

4.2 Results

Several measures were taken for each pair: *success* was measured by the number of programs completed successfully (when compared to the number of programs attempted); *ease of communication* and *perceived partner contribution* were measured using post-test Likert scales as discussed above.

4.2.1 Previous Pairing Experience

The post-test surveys consisted of questions relating to the individual's experience with solo programming, pair programming, and previous pair programming experience with the session's partner. These results were analysed to understand group tendencies and variance, as reported in Table 1 below:

Table 1: Student programming experience.

	Experimental Group		**Control Group**	
	M	**SD**	**M**	**SD**
Solo Programming Experience (weeks)	192.4	203.031	123.3	88.108
Pair Programming Experience (weeks)	10.8	11.359	326.9	631.992
Previous Pair Programming Experience with this Session's Partner (weeks)	1.9	4.721	0.0	0.000

The data shows that there was differing experience between the groups; on average, more individuals in the experimental group (exposed pairs) had solo programming experience, but more individuals in the control group (unexposed pairs) pairs had pair programming experience. Furthermore, two pairs within the experimental group had previous experience in pairing together.

Statistical tests were therefore carried out in order to establish whether the differences between the two groups were significant, and whether they would establish threats to validity of the data.

Mann-Whitney U tests were used for this analysis, as these tests are not affected by outliers to the same degree as independent-samples t-tests.

Differences in 'Solo' Programming Experience

A Mann-Whitney U test shows that there were no significant differences in 'solo' programming experience (in weeks) prior to the test between students who were exposed to the pair programming guidelines, and students who were not; $U = 69.5$, $z = -0.414$, $p = 0.687$ ($p > 0.05$).

Difference in Pair Programming Experience

A Mann-Whitney U test shows that there were no significant differences in pair programming experience (in weeks) prior to the test, between students who were exposed to the pair programming guidelines, and students who were not; $U = 91$, $z = 0.386$, $p = 0.742$ ($p > 0.05$).

Difference in Pair Programming Experience with this Session's Partner

A Mann-Whitney U test shows that there were no significant differences between students who were exposed to the pair programming guidelines, and students who were not when looking at the data for *previous pair experience with this session's partner*; $U = 72$, $z = -1.336$, $p = 0.560$ ($p > 0.05$).

From the analysis, it can be seen that any differences between the two groups (with respect to solo and pair programming experience) were not significant, suggesting that there would be no validity issues when analysing data between the two groups.

4.2.2 Success

Following the pair's 45-minute test, the number of tasks attempted was noted by the researcher, and scored at a later date. Each attempt was scored by the researcher, and also compiled, to see if the whether the code successfully compiled and produced the correct result. The total number of successfully completed tasks was then noted for each pair.

Independent-samples t-tests were used to analyse completion rates, which in turn were used to determine whether there were any differences in success levels between the exposed pairs and the control group.

There were no outliers in the data, as assessed by inspection of a boxplot (Figure 1). The exposed pairs have a marginally higher number of tasks completed (2.71 ± 3.04) than the unexposed pairs in the control group (2.17 ± 2.14) – however, this difference is not statistically significant: $t(11) = 0.369$, $p = 0.718$ ($p > 0.05$).

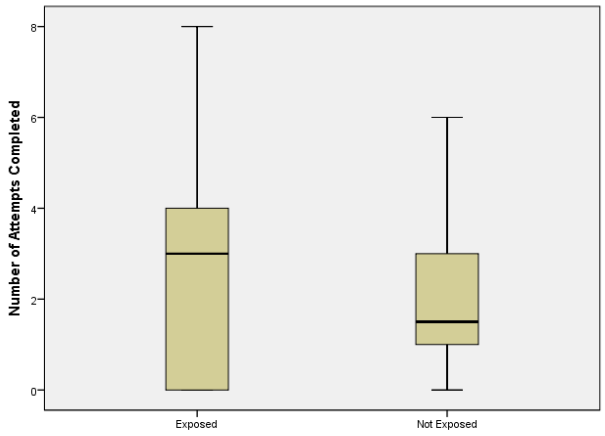

Figure 1: A boxplot showing the number of tasks completed between experimental group and the control group.

4.2.3 Ease of Communication

'Ease of Communication' was reported as a Likert scale on the post-test survey through the following statement: "*During this session, I found communicating with my partner to be easy*". The scale ranged from 1 (strongly disagree) to 5 (strongly agree).

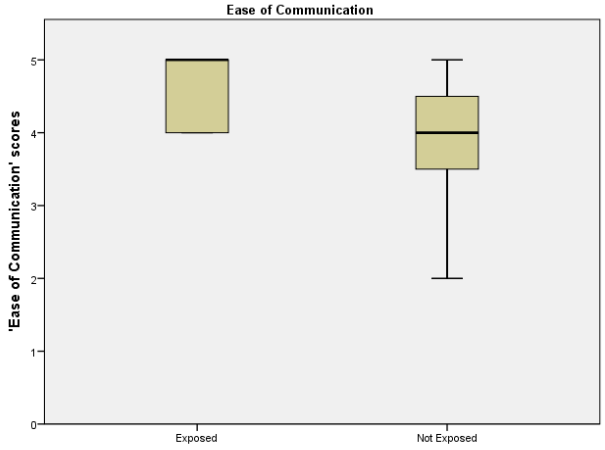

Figure 2: A boxplot showing the reported scores for ease of communication (ranging from 1 (strongly disagree) to 5 (strongly agree)) between the experimental group and the control group.

A boxplot (Figure 2) depicts the distribution of Likert scale scores reported by the individual students. It can be seen that students who were exposed to the guidelines generally reported a higher score (M=4.6; SD=0.514) than students who were not (M=3.9; SD=0.900).

As the data used is extracted from Likert scales, a Mann-Whitney U test was used for its analysis [8]. This test was run to determine any differences in *ease of communication* between the exposed group, and the control group.

There was a statistically significant difference in Ease of Communication scores between exposed students (*Mdn* = 5.00) and unexposed students (*Mdn* = 4.00), $U = 48$, $z = -2.037$, $p = 0.042$.

As $p < 0.05$, it can be seen that students who were exposed to the guidelines found their intra-pair communication to be easier than the students from the control group.

4.2.4 Perceived Partner Contribution

'Perceived Partner Contribution' was reported as a Likert scale on the post-test survey through the following statement: "*Rate your partner's contribution to today's session*". The scale ranged from 1 (no participation) to 5 (excellent).

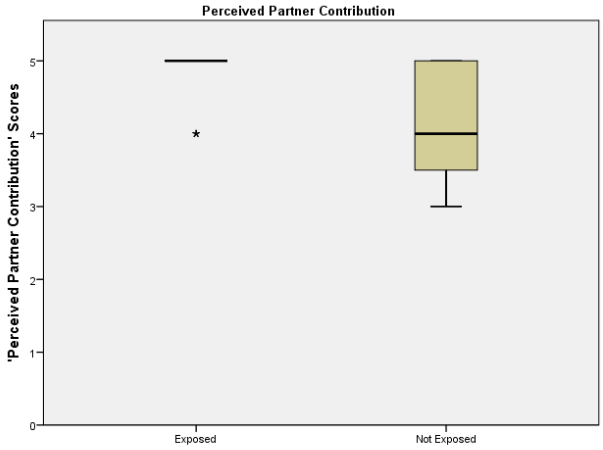

Figure 3: A boxplot showing the reported scores for the perceived partner contribution (ranging from 1 (no participation) to 5 (excellent) between the experimental group and the control group.

A boxplot (Figure 3) depicts the distribution of Likert scale scores reported by the individual students. It can be seen that students who were exposed to the guidelines generally rate their partner's contribution to be quite high, with low variance (M=4.8; SD=0.426) than students who were in the control group (M=4.1; SD=0.835).

The asterisk seen in the boxplot indicates outliers in SPSS, the statistics package used to carry out analyses for this study. The data in this case shows that 3 of the exposed students reported their perceived partner contribution to be '4' on the Likert scale, whereas the rest of the exposed group rated this as '5'.

A Mann-Whitney U test was run to determine if there were differences in Perceived Partner Contribution between the exposed and unexposed groups. There was a statistically significant difference in perceived partner contribution scores between exposed students (*Mdn* = 5.00) and unexposed students (*Mdn* = 4.00), $U = 48.5$, $z = -2.113$, $p = 0.035$.

As $p < 0.05$, it can be seen that students who were exposed to the guidelines were rated by their partners to have contributed more to the session, when compared to students from the control group.

4.3 Discussion

Following the analysis presented above, the following hypotheses are accepted:

1. The mean completion rate for pairs who were exposed to the guidelines and pairs who were not exposed is equal in the population.
2. The distribution of the pair's ease of communication scores differs by exposure to the guidelines.
3. The distribution of the pair's perceived partner contribution scores differs by exposure to the guidelines.

These results therefore suggest that exposure to the guidelines leads to the novice pair experiencing easier communication and improved perceived partner contribution – however, there are no changes to the task completion rates between the two groups.

Each pair had 45 minutes to debug as many programs as possible, following an ordered list of programs that had been presented to them. The fact that the exposed pairs did not perform significantly better than the pairs who were not exposed is perhaps to be expected: the students were exposed to the guidelines at the start of the 45 minute session, and whilst evidence shows that the guidelines were used to improve their intra-pair communication skills, one session might not have been enough time to completely take the guidelines on board and adopt them to improve their work output.

4.4 Feedback

All exposed students were invited to give detailed feedback on their experiences with using the guidelines by means of an online survey. A total of six students completed the survey. Their reported usage of the guidelines is reported in Figure 4 below.

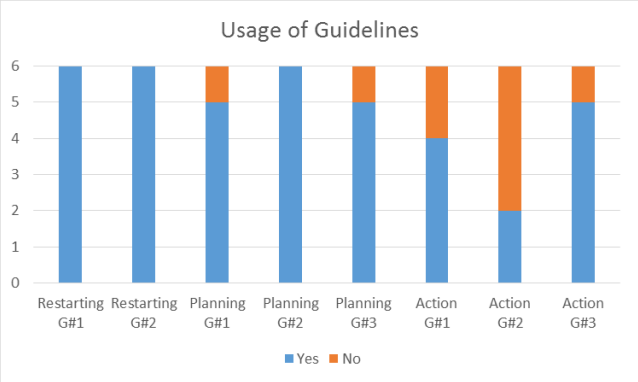

Figure 4: A bar chart plotting the answers to the question: "have you used this guideline?" from the online student survey.

It can be seen that most guidelines were used by most or all students who completed the survey. Two of the guidelines from the *Action* set were not as frequently used as the others (this is depictured as 'Action G#1' and 'Action G#2' in Figure 4 above, and correspond to the two *'Navigator'* guidelines from section 3.3 above).

As both these guidelines are only relevant to the navigator, it is possible that most of the respondents only contributed to the session as drivers, and therefore did not need to make use of these guidelines.

Furthermore, the second guideline actively points towards the driver muttering: if the respondents were navigating, it is also possible that their driver had not been muttering at the time, thus rendering the guideline ineffective.

4.4.1 Feedback regarding the Restarting Guidelines

All students who answered the survey indicated that they had made use of these guidelines whilst pair programming; "*When we were stuck, we lost focus and ended up going off-topic anyway before bringing it back to the task at hand*" / "*It was helpful to go for a walk, and then return less frustrated*."

Students indicated that they found this to "*usually work quite well*", commenting that the "*use of jokes or venting frustrations were helpful towards giving us a break*".

Comments were positive – "*This was a useful technique*" / "*Quite happy with using it; worked well*", with some students suggesting that these guidelines tended to occur naturally to them, without much planning required – "*Tended to use this naturally*" / "*We both used it intuitively without thinking about it*".

Some comments focused on the possibilities of having problems when working with a new partner: "*I can imagine if you do not know your partner very well it would be more tempting to dismiss an attempt to break focus*" / "*If one of us lost focus, the other was generally losing focus at the same time*". Other comments, on the other hand, discuss possible solutions to this issue: "*Sometimes identifying when your partner wishes to break focus for this purpose can be difficult, particularly if you do not know your partner well or feel uncomfortable working with them. Having said this, it can be as simple as merely saying, 'let's take a quick break' – this can provide a clear indication of intentions."*

4.4.2 Feedback regarding the Planning Guidelines

Feedback for these guidelines was positive – students felt that they had "*a natural flow*" and that "*group coding would be impossible without this*".

Students reported that "*[it was] helpful to know what [their] partner was thinking*", and explain that "*this can help to ensure that [both partners] understand the on-going work and are on the same page*."

One student felt that "*asking what the partner is doing at every stage can be irritating and detrimental; sometimes it's best to sit back and watch*", suggesting that for some students, the constant offer or request for explanations might prove to be distracting. This shows that in some cases, it might be more beneficial for the pair to discuss the guidelines between themselves prior to adopting them, and develop a way for them both to be comfortable with their usage in terms of distractions and interruptions.

4.4.3 Feedback regarding the Action Guidelines

The reported feedback was quite positive; students expressed that their muttering "*helped keep the navigator involved and encouraged them to contribute*", meaning that at times, the problem is solved "*before you waste time getting neck-deep in useless code*". By looking at overall comments, it can be seen that this guideline was considered beneficial in helping the survey respondents understand the underlying logic behind their code.

Students felt that reading the code as it was being typed by the driver "*helped save time*", and agreed that following the code

allowed them to be more proactive when helping, as they could make suggestions where the driver appeared to be struggling.

4.5 Further Work

As part of this study, it was not possible to determine a change in the pairs' success levels based on their limited exposure to the guidelines.

Further evaluations in the form of a longitudinal study are currently planned in order to determine whether repeated exposure to the guidelines can significantly affect the pair's measured success rate.

5. CONCLUSIONS

Research shows that novice pair programmers are sceptical about pair programming due to the added communication that will be required of them. This research has presented an evaluation of industry-inspired pair programming guidelines, to understand whether exposing novice pairs to the guidelines can have a positive impact on their success levels and communication.

Significant differences were identified between students who had been exposed to the guidelines and the control group when considering the individual members' perception of their experienced intra-pair communication, as well as their partner's contribution to the session. Furthermore, feedback obtained from the exposed students shows that the guidelines were found to be to be naturally-occurring, and complementary to the students' pairing approaches.

The results presented in this study show that the industry-inspired pair programming guidelines can be used to allow novices to communicate better within their pairs. This is of benefit to educators and students alike, as using these guidelines allows for a more structured introduction to pair programming, with clearer instructions on how to communicate with a new pairing partner.

6. ACKNOWLEDGMENTS

The authors would like to thank all the student pairs who have contributed time and effort towards giving feedback and evaluating these guidelines.

The research work disclosed in this publication is funded by the Strategic Educational Pathways Scholarship (Malta). The scholarship is part-financed by the European Union – European Social Fund (ESF) under Operational Programme II – Cohesion Policy 2007-2013, "Empowering People for More Jobs and a Better Quality of Life".

7. REFERENCES

[1] Begel, A. and N. Nagappan. *Pair programming: what's in it for me?* in *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*. 2008: ACM.

[2] Bryant, S., P. Romero, and B. du Boulay, *The Collaborative Nature of Pair Programming*, in *Extreme Programming and Agile Processes in Software Engineering*, P. Abrahamsson, M. Marchesi, and G. Succi, Editors. 2006, Springer Berlin/Heidelberg. p. 53-64.

[3] Freudenberg, S., P. Romero, and B. Du Boulay. *"Talking the talk": Is intermediate-level conversation the key to the pair programming success story?* in *Agile Conference (AGILE), 2007*. 2007.

[4] Hanks, B., S. Fitzgerald, R. McCauley, L. Murphy, and C. Zander, *Pair programming in education: a literature review.* Computer Science Education, 2011. **21**(2): p. 135-173.

[5] McDowell, C., B. Hanks, and L. Werner, *Experimenting with pair programming in the classroom.* SIGCSE Bull., 2003. **35**(3): p. 60-64.

[6] Murphy, L., S. Fitzgerald, B. Hanks, and R. McCauley. *Pair debugging: a transactive discourse analysis*. in *Proceedings of the Sixth international workshop on Computing education research*. 2010: ACM.

[7] Nagappan, N., L. Williams, M. Ferzli, E. Wiebe, K. Yang, C. Miller, and S. Balik. *Improving the CS1 experience with pair programming*. in *ACM SIGCSE Bulletin*. 2003: ACM.

[8] Ryu, E. and A. Agresti, *Modeling and inference for an ordinal effect size measure.* Statistics in Medicine, 2008. **27**(10): p. 1703-1717.

[9] Sanders, D., *Student Perceptions of the Suitability of Extreme and Pair Programming*, in *Extreme Programming Perspectives*, M. Marchesi, et al., Editors. 2002, Addison-Wesley Professional. p. 168-174.

[10] Sharp, H. and H. Robinson, *Three 'C's of agile practice: collaboration, co-ordination and communication*, in *Agile Software Development*. 2010, Springer. p. 61-85.

[11] Srikanth, H., L. Williams, E. Wiebe, C. Miller, and S. Balik, *On Pair Rotation in the Computer Science Course*, in *Proceedings of the 17th Conference on Software Engineering Education and Training*. 2004, IEEE Computer Society. p. 144-149.

[12] Stapel, K., E. Knauss, K. Schneider, and M. Becker, *Towards Understanding Communication Structure in Pair Programming*, in *Agile Processes in Software Engineering and Extreme Programming*, A. Sillitti, et al., Editors. 2010, Springer Berlin Heidelberg. p. 117-131.

[13] Williams, L. and R.R. Kessler, *Pair Programming Illuminated*. 2002: Addison-Wesley Longman Publishing Co., Inc. 288.

[14] Williams, L., R.R. Kessler, W. Cunningham, and R. Jeffries, *Strengthening the Case for Pair Programming.* IEEE Software, 2000. **17**(4): p. 19-25.

[15] Williams, L., E. Wiebe, K. Yang, M. Ferzli, and C. Miller, *In Support of Pair Programming in the Introductory Computer Science Course.* Computer Science Education, 2002. **12**(3): p. 197-212.

[16] Williams, L.A. and R.R. Kessler, *All I really need to know about pair programming I learned in kindergarten.* Communications of the ACM, 2000. **43**(5): p. 108-114.

[17] Zarb, M., J. Hughes, and J. Richards, *Analysing Communication Trends in Pair Programming Using Grounded Theory*, in *Proceedings of the 26th BCS Conference on Human-Computer Interaction*. 2012, British Computer Society: Birmingham, United Kingdom.

[18] Zarb, M., J. Hughes, and J. Richards, *Industry-inspired guidelines improve students' pair programming communication*, in *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*. 2013, ACM: Canterbury, England, UK. p. 135-140.