

A Replicated Experiment of Pair-Programming in a 2nd-year Software Development and Design Computer Science Course

Emilia Mendes

Computer Science Department
The University of Auckland,
New Zealand

emilia@cs.auckland.ac.nz

Lubna Basil Al-Fakhri

Computer Science Department
The University of Auckland,
New Zealand

lubna@ww.co.nz

Andrew Luxton-Reilly

Computer Science Department
The University of Auckland,
New Zealand

andrew@cs.auckland.ac.nz

ABSTRACT

This paper presents the results of a replicated pair programming experiment conducted at the University of Auckland (NZ) during the first semester of 2005. It involved 190 second year Computer Science students attending a software design and construction course. We replicated the experiment described in [18], investigating similar issues to those reported in [32] and employing a subset of the questionnaires used in [32]. Our results confirm the use of pair programming as an effective programming/design learning technique.

Categories & Subject Descriptors

k.3.2 [Computer and Information Science Education]:
Computer Science Education.

General Terms

Experimentation, measurement.

Keywords

CS2, pair programming, collaboration, software design.

1 INTRODUCTION

Pair programming is a programming technique in which two programmers work together on the same task. Williams et al. [32] describe the process as follows:

“One partner, the driver, controls the pencil, mouse, or keyboard and writes the code. The other partner continuously and actively observes the driver’s work, watching for defects, thinking of alternatives, looking up resources, and considering strategic implications. The partners deliberately switch roles periodically.”

Permission to make digital or hand copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, require prior specific permission and/or a fee.

ITiCSE’06, June 26-28, Bologna, Italy.

Copyright 2006 ACM 1-XXXXXX-XXXXXX...\$5.00

We previously conducted a pair programming experiment during the first semester of 2004 at the University of Auckland (NZ). The subjects of our study were second year Computer Science students enrolled in a software design and construction course. Each student was assigned to either a control group (solo programmers) or an experimental group (paired programmers). Results showed that a significantly greater number of the paired students passed the course compared to solo students. In addition, exam scores, test scores, assignment scores and final grades for paired group were all significantly higher than those for the solo group. Further details of this experiment can be found in [18].

In 2005 we replicated the experiment described in [18] to verify the validity and generalisability of its results. This paper describes the replicated experiment and its results. Our contribution is therefore twofold: i) to replicate Mendes et al.’s experiment [18], also applied to 2nd year computer science students; ii) to add empirical evidence to the current body of knowledge on pair programming regarding this technique’s usefulness or not for improving educational outcomes and enjoyment of students.

Similar to [18], we looked at whether pair programming improves learning and the enjoyment of those students who participate in collaborative activities. We did not investigate the quality of the code produced through the pair-programming process. The hypotheses we investigated were (only the alternative hypotheses are presented):

H1 – An equal or higher percentage of students in paired labs will complete the course with a grade of C- or better compared to solo programmers.

H2 – Students in paired labs will earn exam scores equal to or higher than solo programming students.

H3 – Students in paired labs will earn assignment scores equal to or higher than solo programming students.

H4 – Students in paired labs will earn test scores equal to or higher than solo programming students.

H5 – Students in paired labs enjoy pair programming and will have a positive attitude towards collaborative programming settings.

Our results support all alternative hypotheses, and confirm results obtained in [18], thus providing additional empirical evidence of the benefits of pair programming for learning.

The remainder of this paper is organised as follows: Section 2 provides a detailed summary of the pair programming literature, followed by Section 3 where our experiment is described. Finally conclusions and comments on future work are given in Section 4.

2 PREVIOUS STUDIES

Mendes et al. [18], presented a summary of empirical research (i.e. those studies that described experiments/case studies) about the use of pair programming in the classroom. The summary was organized according to the framework suggested by Gallis et al. [7] in order to support empirical studies and meta-analysis for developing theories about pair programming. We continue to use the same framework here. A summary of the pair programming literature is given in Table 1.

A growing body of empirical evidence supports the use of pair programming as an effective pedagogical technique, especially in introductory CS courses where much of the research on pair programming has been conducted [2],[3],[10],[17],[19],[23],[32]. A few studies have investigated the effectiveness of pair programming with senior students [8],[22]. Only three of the studies in our literature review [4],[6],[18] have used second year students as subjects of a pair programming experiment.

A wide range of benefits are reported for students engaged in pair programming in the classroom, including improved quality of code, teamwork, communication skills, comprehension and learning [6],[17], [18],[19],[30],[31]. Students who work in pairs are less frustrated and have higher levels of enjoyment and satisfaction than those who work solo [8],[10],[15],[19],[31]. Pair programming also increases confidence levels and the likelihood that students will successfully complete the course [17],[24],[25]. However, only a few studies show that pair programming affect improved final exam scores [2],[18],[33].

Werner et al. [25] report that the degree of improvement in confidence levels for female students is substantially higher than for male students, and that the improved retention rate by paired females may help close the gender gap in Computer Science.

The benefits of pair programming also extend to teaching staff since there are typically fewer questions from students [1], [19],[28] and half as many projects to grade [5],[26].

Despite significant positive results, there are some reported disadvantages, including students schedule issues [3],[5], pair incompatibility [3],[16],[17] and unequal participation [23].

Although pair programming typically involves students sharing the same computer/desk/paper, recent studies have looked at the effectiveness of distributed pair programming [4],[9],[20]. Hanks [20] reported that there was no statistically significant difference between co-located paired students and distributed paired students with respect to the exam scores, programming scores, course completion, success rates and confidence.

Various studies have considered pair compatibility when forming pairs. McDowell et al. [17] allowed students to form their own pairs. Thomas, et al. [23], had students rate their programming abilities and assigned partners based on these ratings. Cliburn [5] assigned pairs by grouping students from different cultural/ethnic backgrounds or upper with lower ability students. A number of studies have investigated the effect of personality types on pair compatibility [1],[14],[19]. Katira et al. [14] suggest that pairs be formed between students who perceive themselves to be of equal ability.

Preston [21] discusses how research in collaborative learning activities can help us better understand the process of pair programming. Individual accountability and responsibility has been identified as being critical to cooperative learning activities [12]. Some pair programming studies have required students to formally evaluate their partners [5],[6],[8],[14],[15],[22], whereas others have not. Assiter [1] used process logs and Web portfolios to increase student accountability.

Beck et al. [2] used cooperative learning techniques similar to pair programming and reported significant improvement in exam results over students that worked solo.

Table 1 – Literature Review

Authors	Type of study	Sub.	N	Task	Duration	SP?	Independent variable(s)	Main dependent variables (metrics)
Assiter [1]	Exper.	Stud.	20.	Assignments	Academic semester	No	Students chose to pair or work solo for each assignment	Preference for pairing, identity as introvert/extrovert.
Beck et al. [2]	Exper	Stud.	84.	1 hour of exercises per week	Academic semester	Yes	Solo (48) versus teams (36)	Student success - test and exam scores
Bevan et al. [3]	Case study	Stud.	Unk.	Nine assignments	10 weeks	Yes	Evaluation of PP's effectiveness as a teaching technique and effect on student retention.	Total work time, satisfaction with partner, amount of time worked alone.
Canfora et al. [4]	Exper	Stud.	32.	Three maintenance tasks	Unk	Unk	Co-located (16) versus distributed (16)	Time taken to complete tasks
Cliburn [5]	Case studies	Stud.	14, 8, 17	Five assignments	Academic semester	Yes	All paired	Learning effect - Final course grade, PP enjoyment (qualitative survey)
DeClue [6]	Exper.	Stud.	24	Six assignments	Academic semester	No	Solo (22) and same 22 paired for 6 weeks.	Students attitude towards PP and role of PP on code/design quality (qualitative survey)
Gehring [8]	Exper.	Stud.	96	Three assignments	Academic semester	No	Solo (19,29,17) versus pairs (41, 23, 28 pairs)	Quality – score on assignment.
Hanks [9]	Exper.	Stud.	112	Five / Seven Assignments	Academic semesters	Yes.	Co-located (55) versus distributed (57)	Student success - assignment grades and final exam score, confidence.
Hanks et al. [10]	Expers.	Stud.	112, 50	Five Assignments	Academic semesters	Unk.	Solo (112) and paired (25 pairs) at different semesters	Program quality – complexity, length, subjective metrics, Students confidence and satisfaction -
Haungs [11]	Case study	Prof.	2	C3 system	Unk.	Yes	Investigate use of PP	Information and Knowledge transfer (qualitative assessment)
Katira et al. [13]	Exper.	Stud.	564	Up to six assignments	Academic semesters	Yes	All paired	Understand compatibility of pairs - survey

Katira et al. [14]	Exper.	Stud.	361	Multiple assignments	Academic semester	No	All paired	Understand compatibility of pairs - survey
McDowell et al. [15]	Expers.	Stud.	552	Five or four assignments	Four sections, 3 months each.	Yes	Solo (148) in Spring section pairs (202 pairs) in Fall and Winter sections.	Student success – final course grade and exam attendance. Gender completion rate – final course grade, Course performance - score on final exam and program quality.
McDowell et al. [16]	Expers.	Stud.	95, 19, 102	Unk.	Three months	Yes	Solo (47, 5,44) versus pairs (22, 7, 29 pairs)	Qualitative assessment of program quality, Learning effect – score on final exam.
McDowell et al. [17]	Exper.	Stud.		Five assignments	Two sessions of academic year	Yes	Solo (141) versus pairs (86 pairs).	Quality – score on programming assignment (functionality and readability), Learning effect – score on final exam.
Mendes et al. [18]	Exper.	Stud.	300	Three assignments	Twelve-week semester	No	Solo (186) versus pairs (114)	Course final grade, assignments/test/exam scores, enjoyment of PP
Nagappan et al. [18]	Expers.	Stud.	199, 502	Three programming projects	Academic semester	No	Solo (69, 102) versus pairs (22, 140 pairs)	Learning effect – course, assignments and exam scores, attitude towards PP (qualitative)
Srikanth et al. [22]	Exper.	Stud.	270, 140	Five assignments	Academic semesters	No	All paired	Students and teachers perceptions on pair rotation and students perceptions on peer evaluation (all surveys)
VanDeGrift [24]	Exper.	Stud.	546	Three projects and corresponding reports	Academic semester	Yes	All paired	Students processes – subjective assessment of reports, students perceptions on PP and written reports (survey)
Williams [26] [28]	Case study	Stud.	20	Web site development using CSP	Eleven-week semester	Yes	Evaluation of PP to gather experience with the process.	View of programming using CSP (qualitative assessment)
Williams et al. [31] Williams [27]	Exper.	Stud.	41	Four assignments	Six weeks	Yes	PSP (13 solo) versus CSP (14 pairs)	Total work time, productivity and programming quality
Unk. – Unknown Exper. – Experiment Expers. – Experiments Stud. – Students Sub. – Subjects SP? – Same Partner?								

3 PAIR PROGRAMMING EXPERIMENT

Similar to [18], we designed and conducted an experiment during the first semester of 2005 to evaluate whether pair programming was beneficial for learning and students attitude towards pair programming. Our experiment involved 190 CS students attending a second-year software design and construction course, where 78 paired and 112 worked solo.

3.1 Course Structure

The software design and construction course comprises three main areas¹: Area I introduces software design using Unified Modeling Language, and testing techniques. Area II furthers Java object-oriented programming skills acquired from the first year courses. Area III introduces basic client-server programming concepts and skills using Java and MySQL database. The sequence in which these three areas are taught varies. In 2004 the sequence used was I, II, III; in 2005 it changed to I, III, II.

As reported in [18], our semester lasts for 12 weeks. Students attended lectures three times a week for 50 minutes, and attended closed lab sessions, led by two teaching assistants, once a week for 11 weeks. Three lecturers, including one author, led the lectures. There were no specific in-lab assignments and attendance was not mandatory. Course assessment was based on three assignments (20%), one mid-semester test (15%) and one final exam (65%). Lab exercises, assignments, test and final exam were the same for all students. Although we aimed at keeping both experimental settings as similar as possible, there were unavoidable changes: our close labs lasted for 60 rather than 90 minutes; Lectures were taught by two of the 2004 lecturers, and a

new lecturer taught Area II; two new teaching assistants taught the closed labs.

3.2 Student Population

Of the 190 students enrolled, 81% intended to obtain a Bachelor of Science degree, 10% a co-joint degree in Science and Commerce, 5% a Graduate Diploma of Science degree, and the remaining 4% other degrees. 12% were females and 88% males.

3.3 Assignments

Students were given three assignments, one for each of the three course areas. Assignments had on average a three-week duration, and each contributed the same towards the final grade. The three assignments are briefly presented below:

A1: Students had to draw Use case, Class, Instance and Interaction diagrams describing a book processing application for the library.

A2: Students had to implement a Java client application, connecting to a database, which implemented the application designed as their A1 assignment.

A3: Students had to apply and implement design patterns and a testing framework on a large Java Swing-based application.

3.4 Experimental Design

Similar to [18], the experiment's treatment was the pair programming technique and our control was not to use the pair programming technique. Experimental units were the exercises given in the 11 labs and experimental subjects were the students attending lab sessions. The dependent variables were the learning effect (amount learnt), measured using assignments, test and exam scores and course grades and students enjoyment towards pair programming, measured using a qualitative questionnaire. Independent variables were individual programming/task versus paired programming/task.

¹ More information on the course can be found at <http://www.cs.auckland.ac.nz/compsci230s1c/>

During the first week of the term students had to sign up for one of the eight weekly labs offered, unaware of which labs were paired or solo. At the end of the first week we randomly chose the labs that would be paired and randomly assigned pairs within each paired lab in order to minimise the effects of uncontrolled variables.

Students attending paired labs were randomly allocated to a different pair every four weeks. During each lab sessions pairs had to swap roles every 20 minutes, reminded by teaching assistants. Both teaching assistants used the same lab exercises. During the first paired labs students were provided explanations on the pair programming techniques based on guidelines provided by Laurie Williams.

It should be noted that extreme care was taken to minimise the confounding effect that an enthusiastic teacher can bring to their classroom. All students attended the same lectures at the same time. The laboratory sessions were taught by teaching assistants who were not involved in the research project (beyond distributing and collecting questionnaires). Each teaching assistant taught both paired sessions and solo sessions. The laboratory exercises were the same for both groups, and did not count towards the final grade.

There were several factors (independent variables) that we were unable to control and that may have had a confounding effect on the results we obtained, e.g. “self-confidence level”, gender, ethnic group, communication skills, average grades on first year programming courses. Although recent findings suggest that pairs tend to be highly compatible and successful if paired randomly [13], we cannot guarantee that there were no threats to the validity of our results.

3.5 Results

The results presented here are based on data gathered from assignments, test, final exam scores and course grades, and a pair rotation questionnaire which is the same as that used in [18],[32]. This questionnaire had to be answered by each paired student three times, at the end of each four-week block. Ten students withdrew from the course, leaving 180 students (74 paired, 106 solo).

All statistical quantitative analysis was carried out using SPSS V10. Confidence limit was set at 0.05. Except for H1, all remaining hypotheses were tested using the two-tailed Student’s T-test to compare means between paired and solo scores. To test the statistical significance of differences in success rates between paired and solo groups we employed the Chi-square test, which is used to test the independence between categorical variables. A significant result indicates that success rates and group type (solo or paired) are not independent, i.e. that success rates for a particular group (solo or paired) did not occur by chance. Table 2 shows that a higher percentage of paired students passed the course, compared to solo students, and that their success rate is significantly related to the group they belonged to. This result supports the alternative hypothesis H1, and corroborates results obtained in [18].

Table 2 – Success Rate between paired and solo students

# Paired	% paired passed	# Solo	%Solo passing	Statistical significance
74	89.1%	106	82%	p = 0.003

The two-tailed Student’s T-test also showed statistically significant differences in scores between paired and solo students (favouring paired students) for all assignments, test and final

exam, which is remarkable. These results provided support for alternative hypotheses H2 to H4, and corroborated the results in [18]. We believe that a possible explanation for these results may be that the majority of second-year students come from an Asian ethnic group, where it seems that group work is always favoured as opposed to individualism and competition².

Regarding students enjoyment with pair programming (see Figure 1), on a scale from 1 (hated it) to 10 (loved it), our results showed that 42% of answers fell between 6 and 10, i.e. above average, 25% fell within the last three highest scale points (8 to 10), and 40% fell on the average. These results were not as optimistic as those reported in [18], however the majority of our students seemed to have liked it.

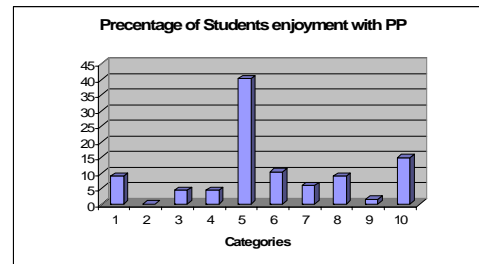


Figure 1 – Students enjoyment with pair programming

Finally, 51% of those students who paired would like to have pair programming as part of future Computer Science courses. Most students did not mind to pair with another student of different gender or ethnic group, and language skills also did not seem to be an issue.

4 CONCLUSIONS AND FUTURE WORK

We have reported on a replicated experiment which compares the performance of students who engaged in pair-programming during voluntary laboratory sessions with those who worked solo during voluntary laboratory sessions. Although the laboratory sessions did not contribute directly to the final grade, the effects of being involved in a pair programming experience appear to have improved the quality of independent assignment work, examination scores, and percentage passing overall. Furthermore, the majority of students enjoyed the experience and would like to have pair-programming used in future courses. Our results corroborate the results previously obtained in [18].

We have shown that experience with pair programming has positive outcomes, even when pair programming is not a compulsory course requirement. These results provide strong support for the use of pair programming in the Computer Science curriculum.

As part of our future work we intend to investigate the long-term impact of pair programming by looking at the performance of students in subsequent courses, and compare the effects found in this experiment with those obtained from experiments in introductory courses.

ACKNOWLEDGMENTS

We would like to thank Laurie Williams for providing us with the questionnaires used in our experiment and for her encouragement, Shawn Guo and Mind Li for their help running the labs and

² this data was obtained from a students statistics booklet published by the University of Auckland

finally those students who volunteered their time and grades for our research.

REFERENCES

- [1] K. Assiter, Balancing depth and breadth in the data structures course, *Journal of Computing Sciences in Colleges*, Feb. 2005, 20 (3), 255 - 271, 2005
- [2] L.L. Beck, A.W. Chizhik, and A.C. McElroy, Cooperative learning techniques in CS1: design and experimental evaluation. *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*, Feb. 2005, 470-474, 2005.
- [3] J. Bevan, L. Werner, and C. McDowell, Guidelines for the use of pair programming in a freshman programming class, *Proceedings 15th Conference on Software Engineering Education and Training*, 25-27 Feb. 2002, pages 100 - 107, 2002.
- [4] G. Canfora, A. Cimitile, and C.A. Visaggio, Lessons learned about distributed pair programming: what are the knowledge needs to address?, *Proceedings Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 9-11 June 2003, pages 314 - 319, 2003.
- [5] D.C. Cliburn, Experiences with pair programming at a small college, *Journal of Computing Sciences in Colleges*, October 2003, 19(1), 2003.
- [6] T.H. DeClue, Pair programming and pair trading: effects on learning and motivation in a CS2 course, *Journal of Computing Sciences in Colleges*, May 2003, 18(5), 2003.
- [7] H. Gallis, E. Arisholm, and T. Dyba, An initial framework for research on pair programming, *Proceedings International Symposium on Empirical Software Engineering*, 30 Sept.-1 Oct. 2003, pages 132 - 142, 2003.
- [8] E.F. Gehringer, A pair-programming experiment in a non-programming course, *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, October 2003, 2003.
- [9] B. Hanks, Student performance in CS1 with distributed pair programming, *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, June 2005, 2005.
- [10] B. Hanks, C. McDowell, D. Draper, and M. Krnjajic, Program quality with pair programming in CS1, *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education*, June 2004, 2004.
- [11] J. Haungs, Pair programming on the C3 project, *Computer*, 34(2), Feb 2001, pages 118 - 119, 2001.
- [12] R. Johnson and D. Johnson, An overview of cooperative learning, in J. Thousand, R. Villa and A. Nevin, Eds. *Creativity and Collaborative Learning*, Baltimore: Brookes Press, 1994
- [13] N. Katira, L. Williams, and J. Osborne, Towards increasing the compatibility of student pair programmers, *Proceedings of the 27th international Conference on Software Engineering*, May 15 - 21, 2005, 2005
- [14] N. Katira, L. Williams, E. Wiebe, C. Miller, S. Balik, and E. Gehringer, Paired programming/ collaborative learning: On understanding compatibility of student pair programmers, *Proceedings of the 35th SIGCSE technical symposium on Computer science education*, March 2004, 2004.
- [15] C. McDowell, L. Werner, H.F. Bullock, J. Fernald, The impact of pair programming on student performance, perception and persistence, *Proceedings 25th International Conference on Software Engineering*, 3-10 May 2003, pages 602 - 607, 2003.
- [16] C. McDowell, B. Hanks, L. Werner, Experimenting with pair programming in the classroom, *Proceedings of the 8th annual conference on Innovation and technology in computer science education*, June 2003, *ACM SIGCSE Bulletin*, 35(3).
- [17] C. McDowell, L. Werner, H. Bullock, and J. Fernald, The effects of pair-programming on performance in an introductory programming course, *Proceedings of the 33rd SIGCSE technical symposium on Computer science education*, February 2002, *ACM SIGCSE Bulletin*, 34(1), 2002.
- [18] E. Mendes, L. B. El-Fakhri, A. Luxton-Reilly, Investigating Pair Programming in a 2nd year Software Development and Design Computer Science Course. *Proceedings of ITiCSE'05*, June 2005, pages 296-300, 2005.
- [19] N. Nagappan, L. Williams, M. Ferzli, E. Wiebe, K. Yang, C. Miller, S. Balik, Improving the CS1 experience with pair programming, *Proceedings of the 34th SIGCSE technical symposium on Computer science education*, *ACM SIGCSE Bulletin*, January 2003, 35(1), 2003.
- [20] H. Natsu, J. Favela, A.L. Moran, D. Decouchant, and A.M. Martinez-Enriquez, Distributed pair programming on the Web, *Proceedings of the Fourth Mexican International Conference on Computer Science*, 8-12 Sept. 2003, pages 81 - 88, 2003.
- [21] D. Preston, Pair programming as a model of collaborative learning: a review of the research. *J. Comput. Small Coll.* 20, 4 (Apr. 2005), 39-45, 2005
- [22] H. Srikanth, L. Williams, E. Wiebe, C. Miller, and S. Balik, On pair rotation in the computer science course, *Proceedings 17th Conference on Software Engineering Education and Training*, 1-3 March 2004, pages 144 - 149, 2004.
- [23] L. Thomas, M. Ratcliffe, and A. Robertson, Code warriors and code-a-phobes: a study in attitude and pair programming, *Proceedings of the 34th SIGCSE technical symposium on Computer science education*, January 2003, *ACM SIGCSE Bulletin*, 35(1), 2003.
- [24] T. VanDeGrift, Coupling pair programming and writing: learning about students' perceptions and processes, *Proceedings of the 35th SIGCSE technical symposium on Computer science education*, March 2004.
- [25] L.L. Werner, B. Hanks, and C. McDowell, Pair-programming helps female computer science students. *Journal of Educational Resources in Computing*, 4, 1, Mar 2004, 2005
- [26] L. Williams, But, isn't that cheating? [collaborative programming], *Proceedings 29th Annual Frontiers in Education Conference*, 2, 10-13 Nov. , pages 12B9/26 - 12B9/27 vol.2, 1999.
- [27] L. Williams, Integrating pair programming into a software development process, *Proceedings 14th Conference on Software Engineering Education and Training*, 19-21 Feb 2001, pages 27 - 36, 2001.
- [28] L. Williams, and R.R. Kessler, The effects of "pair-pressure" and "pair-learning" on software engineering education, *Proceedings 13th Conference on Software Engineering Education & Training*, 6-8 March 2000, pages 59 - 65, 2000.
- [29] L.A. Williams, and R. Kessler, All I really need to know about pair programming I learned in kindergarten, *Communications of the ACM*, May 2000, 43(5), 2000.
- [30] L. Williams, and R.L. Upchurch, In support of student pair-programming, *Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education*, February 2001, *ACM SIGCSE Bulletin*, 33(1), 2001.
- [31] L. Williams, R.R. Kessler, W. Cunningham, and R. Jeffries, Strengthening the case for pair programming, *IEEE Software*, 17(4), July-Aug. 2000, pages 19 - 25, 2000.
- [32] L. Williams, C. McDowell, N. Nagappan, J. Fernald, and J. Werner, Building pair programming knowledge through a family of experiments, *Proceedings International Symposium Empirical Software Engineering*, 30 Sept.-1 Oct. 2003, pages 143 - 152, 2003.
- [33] K. Yang, Pair Learning in Undergraduate Computer Science Education, CS1 class, North Carolina State University, 2002.

Note. The data set can be made available to the reviewers for independent assessment of the statistical analyses presented in this paper but cannot be published for confidentiality reasons. Please contact Emilia Mendes at emilia@cs.auckland.ac.nz.