How and Why Collaborative Software Development Impacts the Software Engineering Course

Lucas Layman [1], Laurie Williams [2], Jason Osborne [3], Sarah Berenson [4], Kelli Slaten [5], and Mladen Vouk [6]

Abstract - **This paper presents the results of an initial quantitative investigation to assess a variety of factors that potentially affect the collaborative software development experience. This research was conducted with 119 students in two undergraduate software engineering classes at North Carolina State University. A survey was administered where students could reflect on their collaborative experiences. We analyzed these factors for interrelationships as well as for correlations with performance in the course, grade point average, and SAT scores. Our findings support the components of the proposed Social Interaction Model of Pair Programming. The substantiation of the Social Interaction Model of Pair Programming values suggests that they should be considered in course planning. We also find that work ethic and self-perceived programming ability positively correlate with GPA. Our results also suggest that collaborative software development may improve student perceptions of software engineering.**

Index Terms - Collaborative development, Education research, Software engineering.

INTRODUCTION

In the junior or senior year, many computer science students will take their first software engineering course. Software engineering courses are typically meant to reflect large-scale, team-oriented software development. For many students, a software engineering course is the first time that they encounter the collaborative nature of software development.

Collaboration is an everyday part of professional software development [4]. Previous work [1, 2] has suggested that collaboration and group work can have a positive influence on the experiences of women in software engineering courses. This work led to the development of the Social Interaction Model of Pair Programming (SIMPP) (see Figure 1), which suggests that collaboration may lead to higher productivity, higher quality, less time spent, and increased confidence. These factors are important values for women, and it is suggested that increasing these values will increase women's interest in IT careers.

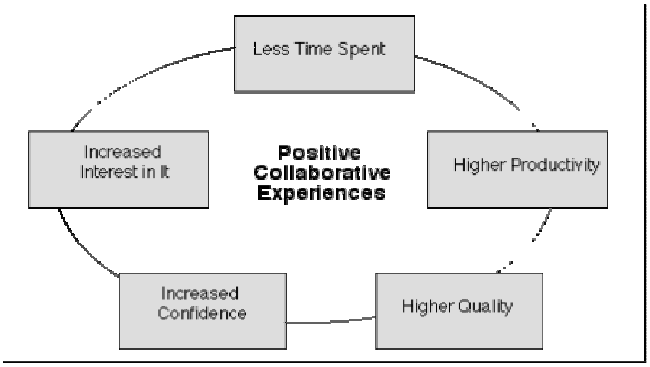

FIGURE 1
A PROPOSED SOCIAL INTERACTION MODEL OF PAIR PROGRAMMING THAT CONNECTS WOMEN'S VALUES TO THEIR INTERESTS IN IT CAREERS.

The goal of this study is to better understand these proposed values and to generate empirical evidence in support or refutation of the proposed SIMPP. Additionally, we wish to determine if this model applies not only to women, but to software engineering students in general. We also seek to understand the role of self-confidence in the students' classroom performance. We present results from a survey administered to 119 software engineering students at North Carolina State University (NCSU). Data was collected from two separate software engineering courses over two semesters, Spring 2004 and Fall 2004. Students were surveyed to identify factors that contribute to students' preferences to work together or alone.

BACKGROUND AND RELATED WORK

In this section, we provide information on prior work with pair programming and on the use of collaborative, situated cognition techniques in education.

1. Pair Programming in Education

Extensive studies of student pair programmers have been conducted at NCSU [9, 13, 14] and the University of California – Santa Cruz (UCSC) [3, 8]. Those studies consistently report, to varying degrees, the following observations relative to the use of pair programming in

[1] Lucas Layman, Department of Computer Science, North Carolina State University, lmlayma2@ncsu.edu
[2] Laurie Williams, Department of Computer Science, North Carolina State University, williams@csc.ncsu.edu
[3] Jason Osborne, Department of Statistics, North Carolina State University, osborne@stat.ncsu.edu
[4] Sarah Berenson, Department of Mathematics, Science & Technology Education, North Carolina State University, berenson@unity.ncsu.edu
[5] Kelli Slaten, Department of Mathematics, Science & Technology Education, North Carolina State University, kmslaten@unity.ncsu.edu
[6] Mladen Vouk, Department of Computer Science, North Carolina State University, vouk@csc.ncsu.edu

0-7803-9077-6/05/$20.00 © 2005 IEEE **October 19 – 22, 2005, Indianapolis, IN**
35th ASEE/IEEE Frontiers in Education Conference

introductory computer science classes. An equal or higher percentage of pair programming students completed an introductory programming class with a grade of C or better when compared with solo programmers. Student participation in pair programming leads to at least similar performance on exams, when compared with solo programming students. Students that use pair programming on programming projects produce better projects than solo programming students. If pair programming is required only for a closed lab, there is no discernable impact on programming projects produced outside of the closed lab. Students in paired labs have a positive attitude toward collaborative programming settings. Students who use pair programming in an introductory computer science course are not hampered in future solo programming courses. Students who use pair programming in an introductory programming course are significantly more likely than solo-programming students to pursue Computer Science related majors one year later.

2. Collaboration and Situated Cognition

The guiding conceptual frameworks for our study stem from a perspective of situated cognition [6]. In situated cognition, students learn by participating in a collaborative apprenticeship. This apprenticeship enables social constructivism through univocal and dialogic discourse [11]. Univocal thinking is limited to the conversations that individuals have with themselves, while dialogic communications are those that occur in interactions with others. Situated cognition is focused on the apprenticeship nature of the students' learning. In this software engineering course, students move from the peripheral apprenticeship positions of their prior two years of study toward assuming more realistic, collaborative tasks of an IT worker. In our study, we examine students' perceptions of their learning in a software engineering course as they move beyond their initial apprenticeships, acquiring and taking on more responsibilities of a job in software development. These contexts are the focus of study along with the univocal and dialogic nature of the collaborative learning context.

RESEARCH SETTING AND METHOD

This section describes the two classes involved in our study as well as our research method.

1. Research Setting

Data was gathered from two software engineering classes at NCSU (see Table 1) that were taught by different instructors. Students attended two, 50-minute lectures and one, two-hour closed lab each week.

TABLE I
CLASS INFORMATION

Class	Female	Male	Total
Spring 2004	10	112	122
Fall 2004	7	63	70

The pedagogical approaches of the two instructors differed. The Spring 2004 class had two pair programming assignments, two paired requirements assignments, two group design exercises (3-4 students), one test plan assignment, and one solo programming assignment. The group assignment involved the creation of a design document and was not substantial in length. The Fall 2004 class had two, two-week paired assignments followed by one three-week solo assignment. All of the assignments emphasized understanding requirements, creating a design, and writing code and tests. The Fall 2004 class cumulated in a large, group project that spanned six weeks in which the students followed an iterative development methodology. The positive benefits of pair programming were emphasized by the instructor and the teaching assistants in both classes.

2. Research Method

Our findings are based primarily on the results of surveys administered during each class. The survey questions differed slightly between semesters. The questions from both surveys are provided in this section. The Spring 2004 survey was a written survey where the students were asked to respond to 14 questions on a five-point Likert scale. The survey was administered during a lecture period at the end of the semester and 63 out of 122 (51.6%) students took the survey. The survey was anonymous and students were not asked to record their gender. Not all students responded to each question in the Spring 2004 survey. For the Spring 2004 survey, the questions were:

1) I really enjoy the content of software engineering in this course.
2) I prefer to work on assignments with another student.
3) Working with another student saves homework time.
4) I would rather work alone on large projects.
5) I learn more from working problems out on my own.
6) I am more organized when I work with others on assignments.
7) When solving a difficult problem, I ask other students' advice.
8) I could avoid a lot of coding errors if I was paired with another student.
9) The instructor handled the problem of team slackers very well.
10) If given a choice, I would always work alone.
11) I get new ideas about solving problems from other students.
12) I tend to procrastinate when I work by myself.
13) An ideal homework partner is responsible and reliable.
14) If my partner is a drag, then I don't mind telling the instructor.

The Fall 2004 survey was an online web application and the students responded on a five-point Likert scale. At the end of the last lab session of the semester, students were asked to volunteer to take the survey and were offered snacks and drinks if they did so. Fifty-six out of 70 students (80.0%) took part in the Fall 2004 survey. The survey was anonymous and students were asked to specify their gender. Students who elected to take the survey had to respond to all of the questions. Question 13 was replaced with an open-ended

prompt for students to describe their ideal partner, but all other questions were identical. The questions were posed in a different order than in the Spring 2004 survey: questions 3 and 4 were swapped, and question 12 appeared immediately before question 10. The following two questions were inserted into the Fall 2004 survey:

15) When I pair program, I feel responsible for my partner's success. (inserted before question 2)

16) When I explain my logic to my partner, I sometimes find errors in my thinking. (last question)

Students were asked to rate their programming self-confidence on a scale from 1-9, with one being the weakest and nine being the strongest. The question, which was used in prior studies of pair programming [10] asked, "When it comes to software development, do you think that…" The students were provided with two reference points for the 1-9 scale:

- 1: I don't like programming and I don't think I am any good at it. I can write simple programs but have trouble writing new programs for solving new problems.
- 9: I have had no trouble at all completing programming tasks to date, in fact they weren't challenging enough. I love to program and anticipate no difficulty with this course."

At the beginning of the Fall semester, students were asked to rate their work ethic and procrastination tendencies on a similar scale. The work ethic question asked, "In your class, do you work hard enough to..." where one corresponded to "Just barely get by" and nine corresponded to "Get the best grade you possibly can." The procrastination question asked, "When you have homework, when do you get started?" where one corresponded to "Very early" and nine to "Very late."

We wished to look for relationships between the responses to separate questions in the surveys. Our analysis was performed separately for the Spring and Fall data. We performed bivariate correlations on the survey responses using Kendall's tau-b method. Statistical tests for significance of the correlations were two-tailed tests with $p < 0.05$. The data in our study were analyzed using SPSS[7].

We also wished to look for relationships between students' responses and performance. For the Fall 2004 class, we used Spearman rank order correlations to investigate relationships of self-perceived programming confidence, work ethic, and procrastination tendencies to the following factors: midterm exam grade; final exam grade; class grade; total GPA, cumulative CS GPA, age, SAT, SAT-Math, and SAT-Verbal. Statistical tests for significance of the correlations were two-tailed tests with $p < 0.05$. Twenty-five students (35.7% of the Fall class) elected not to have their grades and SAT scores involved in this part of the study.

Finally, we performed a Mann-Whitney U test on the Spring and Fall survey responses to determine if there was a statistically significance difference in the responses to common questions between semesters. It was also used to assess changes in programming self-confidence across the Fall 2004 semester.

FINDINGS

In this section, we discuss the findings of our statistical analyses. First, we discuss correlations between questions. We then examine the four of the five factors of the SIMPP followed by other interesting results that emerged from the data. The SIMPP factor not discussed is Increased Interest in IT because no survey question addressed this factor.

1. Inter-question correlations

Table II summarizes the significant correlations between the answers to two questions.

TABLE II
CORRELATIONS

Correlation	Spring τ_b (sig.)	Fall τ_b (sig.)
Less time spent		
"Prefer to collaborate" and "Pairing saves time"	0.579 (0.0005)	0.435 (0.0005)
"Pairing avoids coding errors" and "Pairing saves time"	0.320 (0.003)	0.437 (0.0005)
Higher productivity		
"I get new ideas from others" and "Pairing saves time"	0.248 (0.028)	0.338 (0.001)
"I procrastinate by myself" and "Pairing saves time"	0.227 (0.010)	0.345 (0.002)
"Prefer to collaborate" and "More organized"	0.421 (0.0005)	0.384 (0.001)
Higher quality		
"Prefer to collaborate" and "Pairing finds coding errors"	0.427 (0.0005)	0.457 (0.0005)
Confidence		
"Prefer to work with another student" and programming self-confidence	N/A	-0.246 (0.026)
"Pairing avoids coding errors" and programming self-confidence	N/A	-0.238 (0.037)
"Learn more alone" and programming self-confidence	N/A	0.280 (0.011)
"I would always work alone" and programming self-confidence	N/A	0.275 (0.014)
Collaborative problem solving		
"I get new ideas from others" and "I ask other students' advice"	0.268 (0.019)	Not significant
"Pairing avoids coding errors" and "I ask other students' advice"	0.290 (0.008)	0.298 (0.011)
"Pairing avoids coding errors" and "I get new ideas from others"	0.440 (0.0005)	Not significant
"Prefer to collaborate" and "I get new ideas from others"	0.415 (0.0005)	0.283 (0.015)
"Prefer to collaborate" and "I ask other students' advice"	0.415 (0.0005)	0.408 (0.0005)
Working alone		
"Prefer to work alone on large projects" and "Learn more alone"	0.485 (0.0005)	0.465 (0.0005)
"Would always work alone" and "Learn more alone"	0.427 (0.0005)	0.443 (0.0005)
"Would always work alone" and "I get new ideas from others"	-0.419 (0.0005)	-0.310 (0.009)
"Prefer to work alone on large projects" and "Pairing saves time"	-0.391 (0.0005)	-0.356 (0.002)

[7] http://www.spss.com

By significant, we mean statistically significant as well as meaningful. In total, there were 35 inter-question correlations and we present 19 in Table II. The correlations not shown are due to space considerations and the relevance of the correlations. We acknowledge that multiplicity may cause false positives in inter-question correlations. Potential false positives can be ruled out through repeated, consistent findings. The high number of correlations in the samples suggests that there is a non-trivial amount of multicollinearity in the data. This is to be expected since many of the questions relate to the same thing (e.g. questions 4, 5, and 10 above). The questions on the survey were not designed to be orthogonal to one another.

We organize the correlations according to the values in the SIMPP conceptual framework (see Figure 1) and two additional categories, Collaborative Problem Solving and Working Alone. The grouping is somewhat subjective and arguments can be made that inter-question correlations belong to more than one group. This is due to the nature of the questions and to the relationships between the values in the SIMPP. The inter-question correlations presented in the table show support for the values in the SIMPP, and provide more evidence about impact of collaboration on students.

2. Less Time Spent

Overall, students' perceptions of whether or not collaboration saved time varied. The responses to question three above are shown in Table III. Neither class favored positive or negative responses. Furthermore, neither distribution is normal (Kolmogorov-Smirnov, $p < 0.05$). The varied responses of the students suggests that the perception of whether pairing saves time or not is dependent on multiple factors.

TABLE III
SURVEY RESPONSE S FOR "PAIRING SAVES TIME"

Class	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Spring 2004	4.9%	25.8%	12.9%	38.7%	17.7%
Fall 2004	8.9%	21.4%	28.6%	30.4%	10.7%

Both Spring and Fall survey responses showed a positive correlation between the students' preference to collaborate and their belief that working in pairs saves time. Similarly, there is a correlation between their belief that pairing avoids coding errors and the belief that working in pairs saves time. The collaborative, group-oriented nature of software engineering work draws upon the skills of multiple individuals to solve a coding problem. These results suggest that overcoming coding errors is an important part of the collaborative process because it saves time.

3. Higher Productivity

Less time spent on a problem leads to higher productivity. Higher productivity also infers that the time spent on a task is not wasted, and that the participants are active. Both survey samples show positive correlations between saving time in pairing and a tendency to procrastinate when students work on their own. Similarly, we see a correlation between students' preference to collaborate and their belief that they are more organized when working with a partner. Therefore, the data suggests that higher productivity is an important element of the collaborative experience. This supports the idea of "pair pressure" [12] which encourages collaborators to be more focused and productive.

4. Higher Quality

Higher quality in collaboration is achieved through an active and continual peer review. In the case of pair programming, as one developer writes a design or codes a program, the other developer watches for mistakes and offers ideas. A correlation exists between students' preference to collaborate and their belief that pair programming finds coding errors. Table IV shows the students' responses to questions on the survey pertaining to quality and pair programming. The students' responses are skewed such that most do believe that pairing can help them reduce errors. Students with high confidence do not feel that pairing helps them to avoid coding errors. However, it is interesting to note that more students see pair programming as beneficial in uncovering logic errors than in uncovering coding errors. Furthermore, there is no correlation between confidence and the usefulness of pairing to find logic errors. Together, this information suggests that even the most confident students, who do not benefit from pair programming to find code errors, can still benefit by explaining their thoughts to a partner to uncover errors in their logic.

TABLE IV
SURVEY RESPONSE S REGARDING QUALITY

Class	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
"Pairing avoids coding errors"					
Spring 2004	3.2%	19.0%	14.3%	47.6%	15.9%
Fall 2004	5.4%	8.9%	35.7%	44.6%	5.4%
"I find logic errors when explaining my thinking to my partner"					
Fall 2004	1.8%	5.3%	17.9%	67.9%	7.1%

5. Confidence

One value of the SIMPP that is not explicitly supported by our study is that pair programming increases confidence. Programming confidence was assessed at the beginning of the Fall 2004 semester and again at the end of the semester. Since the end of semester survey was anonymous, we could not track individual changes in esteem across the semester. We found no statistically significant difference between the confidence scores from the beginning and the end of the semester. We note that 22 out of 56 respondents (39.3%) had previous pair programming experience in their freshman lab courses. While there are no statistically different responses between the confidence of those who had previous pair programming experience and those who did not, the mean ranking of confidence for those who had pair programmed previously was higher than those who had not.

The data does suggest that programmer confidence is related to a student's preference to pair. Students that are more confident would prefer to work alone, and less confident students prefer to collaborate with others. This is consistent with similar findings in [10]. Also, we see a correlation between those students with higher self-confidence and those students who learn more alone. As previously mentioned, we see that students with high confidence also do not believe that pairing helps avoid coding errors. These correlations seem to suggest that the most confident students, perhaps the best programmers, feel that they can be held back by less apt partners. Conversely, pairing seems to benefit those with lower programming self-confidence.

6. Collaborative Problem Solving

Another theme that emerges from the data involves collaborative problem solving. The data shows correlations between students' preference to collaborate and their tendencies to get new ideas from other students as well as their tendency to ask other students for advice. This suggests that there are students who learn collaboratively and benefit from the ideas and opinions of others. Prior studies have shown that collaborative learning is important among women in engineering [5, 7].

7. Working Alone

The survey data also provides some insight into why students might prefer to work alone. The correlations suggest that students who learn more from working on their own would always work alone, even on large projects. Furthermore, we see that students who would always work alone also do not get new ideas from others. These same students also disagree that collaborating saves time.

8. Differences Between Classes

We analyzed the differences in the common survey questions between the Spring and Fall semesters to see if any interesting differences arose. Table V lists the significant differences in survey responses between classes as well as the mean rank score of the question. There were four statistically significant differences between the samples.

TABLE V
SURVEY RESPONSE DIFFERENCES BETWEEN CLASSES

Question	U (sig.)	Mean rank (Spring)	Mean rank (Fall)
"Prefer to work alone on large projects"	1256.5 (0.003)	51.94	69.06
"I ask other students' advice"	1332.5 (0.013)	66.85	52.29
"I get new ideas from others"	1394.0 (0.015)	65.87	53.39
"Slackers were handled"	1214.0 (0.005)	50.73	66.82

The significant difference in the Fall class's preference to work alone on large projects may be due to several reasons. Some of the Fall semester students reported a negative experience where team members did not do their share of the work. Also, the Spring class did not have a final group project nor do the prerequisite classes have large group projects. Therefore, Spring 2004 students were more likely to have been reflecting on their conjectures rather than experiences.

The difference between the students' likelihood to ask others for advice or to get new ideas from others students is interesting. In the Fall class, which emphasized collaboration, we would expect these numbers to be higher, but this is not the case. We believe that the Spring numbers are higher due to the pedagogical differences between the two instructors as well as the course structure. The Spring course was regarded by students and TAs as being more abstract and hard to follow at times. Therefore, the students may have had to rely on one another to understand the course material more so than in the Fall course.

The final observed difference makes an important side note. In the Fall 2004 class, it was emphasized throughout the course that collaboration partners and group mates must do equal work. An online system was also in place where students would rate their partners and leave comments to the TA and the instructor if partners were not participating. The instructor also made it clear that failure to participate in the collaborative assignments would negatively affect a student's grade. These policies impacted the students' perceptions of the way non-participants were handled in the course.

9. Class Performance Analysis

For the Fall 2004 class, we analyzed the students' self-reported programming confidence, work ethic, and procrastination ratings against their course performance, grade point averages, and SAT scores. Table VI shows the Spearman rank-order coefficients for significant correlations. No significant, meaningful correlations were found with SAT scores, so these are omitted from the table.

TABLE VI
CLASS PERFORMANCE ANALYSIS

Factor 1	Factor 2	R_s (sig.)
Programming confidence	Total GPA	0.386 (0.014)
Programming confidence	CS GPA	0.332 (0.032)
Work ethic	Procrastination	-0.591 (0.0005)
Work ethic	Final exam grade	0.288 (0.0005)
Work ethic	Total GPA	0.592 (0.0005)
Work ethic	CS GPA	0.568 (0.0005)
Procrastination	Final exam grade	-0.241 (0.05)
Procrastination	Total GPA	-0.424 (0.006)
Procrastination	CS GPA	-0.424 (0.005)
Midterm grade	Final exam grade	0.422 (0.0005)
Midterm grade	Total GPA	0.443 (0.004)
Midterm grade	CS GPA	0.352 (0.022)
Final exam grade	Total GPA	0.382 (0.015)
Final exam grade	CS GPA	0.430 (0.004)

The bivariate correlations reveal some intuitive and some non-intuitive results. First, we see a positive correlation between confidence and overall GPA and CS GPA. An argument can be made that the most confident students are so because they have performed well in their courses. Conversely, it might be that the most confident students perform well in their courses due to their natural abilities. The analysis also suggests that work ethic is positively correlated with overall GPA and CS GPA. This lends weight to the

adage that hard work in the classroom is rewarded. Furthermore, we see that midterm and final exam grades both positively correlate with overall and CS GPAs, as is to be expected.

The students self-reported procrastination tendencies are somewhat confounding. All of the correlations involving procrastination are negative correlations. This suggests that the students who perform best and have the best GPAs are also those who procrastinate the most. Also, procrastination is negatively correlated with work ethic. This may suggest that those students with high work ethic are better at distributing their workload.

One more point of interest is that programming confidence is not a predictor of performance in the course. This may be because the software engineering courses are not programming- or technology-centric. Instead, the software engineering course focuses on higher level, more abstract concepts rather than technical details.

CONCLUSIONS

Software development has been stereotyped as a reclusive, all-consuming career, and this stereotype may dissuade students from pursuing information technology degrees. Software engineering courses are important because they represent a microcosm of the software development profession that many students will undertake. A model of the collaborative experience, the SIMPP, has been proposed in prior research to better understand values important to women. The goal of this study is to better understand these proposed values and to generate empirical evidence in support or refutation of the proposed SIMPP. Furthermore, we wished to determine if these values generalize to all software engineering students.

This paper presents the results of an initial quantitative investigation to uncover the elements that affect the collaborative software development experience. We surveyed students in Spring and Fall 2004 software engineering classes to assess a variety of factors that potentially impacted the collaborative experience. These factors were examined for interrelationships as well as for correlations with performance in the course, grade point average, and SAT scores.

Statistical analyses reveal several interrelated factors that influence collaborative development, such as programming efficiency, procrastination, self-confidence, and personal responsibility. We find that students with lower self-confidence in their programming abilities prefer to work with other students, that procrastinators believe collaboration saves time, and that some students prefer to work alone at all times despite the size of the project. We also find that work ethic and self-perceived programming ability positively correlate with GPA. These results support the findings of prior studies that suggest that collaboration can have a positive influence on work ethic and programming confidence [7].

Our findings lend support to the SIMPP by showing the interrelationships among its values. The substantiation of these important values suggests to educators that they should consider them in planning their courses. Furthermore, our results suggest that collaborative software development may improve student perceptions of software engineering. Finally, our findings suggest that the values of the SIMPP, originally created to model the values of women in collaboration, apply to software engineering students regardless of gender.

ACKNOWLEDGMENT

The authors would like to thank the NCSU software engineering reading group for their comments. This material is based upon the work supported by the National Science Foundation under the Grant No, 00305917. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Berenson, S. B., K. M. Slaten, L. Williams and C. Ho, "Voices of Women in a Software Engineering Course: Reflections on Collaboration", Journal on Educational Resources in Computing, TBD, TBD, 2005, pp. to appear

[2] Berenson, S. B., L. Williams and K. M. Slaten, "Using Pair Programming and Agile Development Methods in a University Software Engineering Course to Develop a Model of Social Interactions", Crossing Cultures, Changing Lives Conference, 2005, pp. to appear.

[3] Bevan, J., L. Werner and C. McDowell, "Guidelines for the Use of Pair Programming in a Freshman Programming Class", Conference on Software Engineering Education and Training, February 25-27 2002, pp. 100-107.

[4] DeMarco, T. and T. Lister, Peopleware, New York: Dorset House Publishers, 1977

[5] Felder, R. M., G. N. Felder and M. Mauney, "A Longitudinal Study of Engineering Student Performance and Retention. III. Gender Differences in Student Performance and Attitudes", Journal of Engineering Education, 84, 2, 1995, pp. 151-163

[6] Lave, J. and E. Wenger, Situated Learning: Legitimate peripheral participation, New York, NY: Cambridge University Press, 1991

[7] Margolis, J. and A. Fisher, Unlocking the Clubhouse: Women in Computing, Cambridge, Mass: MIT Press, 2002

[8] McDowell, C., L. Werner, H. Bullock and J. Fernald, "The Effect of Pair Programming on Performance in an Introductory Programming Course", ACM Special Interest Group of Computer Science Educators, 2002, pp. 38-42.

[9] Nagappan, N., L. Williams, M. Ferzli, K. Yang, E. Wiebe, C. Miller and S. Balik, "Improving the CS1 Experience with Pair Programming", SIGCSE 2003, 2003, pp. 359-362.

[10] Thomas, L., M. Ratcliffe and A. Robertson, "Code Warriors and Code-a-Phobes: A study in attitude and pair programming", SIGCSE, 2003, pp. 363-367.

[11] Wertsch, J. L. and C. Toma, "Discourse and Learning in the Classroom: A Sociocultural Approach", in Constructivism in Education, L. Steffe and J. Gale, Eds., Mahwah, NJ: Lawrence Erlbaum, 1995, pp. 159-174.

[12] Williams, L. and R. Kessler, Pair Programming Illuminated, Reading, Massachusetts: Addison Wesley, 2003

[13] 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, 12, 3, 2002, pp. 197-212

[14] Williams, L., K. Yang, E. Wiebe, M. Ferzli and C. Miller, "Pair Programming in an Introductory Computer Science Course: Initial Results and Recommendations", OOPSLA Educator's Symposium, 2002, pp.