The Case for Pair Programming in the Computer Science Classroom

GRANT BRAUGHT and TIM WAHLS, Dickinson College
L. MARLIN EBY, Messiah College

Previous studies indicate that the use of pair programming has beneficial effects on student learning. In this article, we present a controlled study that directly measured students' acquisition of individual programming skills using laboratory practica (in which students programmed individually under exam conditions). Additionally, we analyzed other measures of student performance, attitudes, and retention. Our results provide direct evidence that pair programming improves the individual programming skills of lower SAT students, and that students who pair program are more confident in their work and are more likely to successfully complete the course. Results from the four other major studies of the effects of pair programming are reviewed and compared with those presented here in order to draw broader conclusions.

Categories and Subject Descriptors: K.3.2 [**Computer and Information Science Education**]: Computer Science Education; D.2.3 [**Coding Tools and Techniques**]:

General Terms: Experimentation, Human Factors

Additional Key Words and Phrases: Pair programming, collaborative learning

ACM Reference Format:
Braught, G., Wahls, T., and Eby, L. M. 2011. The case for pair programming in the computer science classroom. ACM Trans. Comput. Educ. 11, 1, Article 2 (February 2011), 21 pages.
DOI = 10.1145/1921607.1921609 http://doi.acm.org/10.1145/1921607.1921609

1. INTRODUCTION

As the name suggests, pair programming consists of two programmers working together at the same keyboard. One programmer (the driver) operates the keyboard and concentrates on lower-level details of the task at hand such as language syntax and control structures. The other programmer (the navigator) observes and offers suggestions, but is primarily concerned with higher level issues such as overall program design and integration. These roles are exchanged at regular intervals, and in practice both programmers share responsibility for all aspects of the program.

Although pair programming originated in industry and has become a key practice of the Extreme Programming (XP) development methodology [Beck and Andres 2004], we focus here on its use in educational settings. The Past Studies section of this article

Partial funding for this research was provided by the National Science Foundation under a CCLI-A&I grant (DUE-0511264).
Authors' addresses: G. Braught, Department of Mathematics and Computer Science, Dickinson College, Carlisle, PA; email: braught@dickinson.edu; T. Wahls, Department of Mathematics and Computer Science, Dickinson College, Carlisle, PA; email: wahlst@dickinson.edu; L. M. Eby, Department of Mathematical Sciences, Messiah College, Grantham, PA; email: eby@messiah.edu.

DOI 10.1145/1921607.1921609 http://doi.acm.org/10.1145/1921607.1921609

surveys four of the more significant controlled studies of the use of pair programming as a teaching tool. The proposed benefits for students include:

— greater mastery of programming skills,
— greater likelihood of successful course completion,
— increased confidence in programming abilities,
— improved attitudes toward programming and computer science in general,
— increased retention into subsequent computer science courses, and improved performance in those courses.

Additionally, some studies have suggested that pair programming is particularly effective for female and minority students [Carver et al. 2007; Werner et al. 2004; Williams et al. 2007].

The primary contribution of our study is a direct assessment of the effects of pair programming on the development of individual programming skills. During each semester of our study we ran two sections of our introductory course, one in which students pair programmed in lab and one in which they programmed individually in lab. We then administered laboratory practica (i.e., programming tasks to be completed under exam conditions) that all students completed individually. This approach allowed us to directly compare the individual programming abilities of students who used pair programming in lab with those of students who programmed individually in lab. Other studies have used metrics such as written exam scores to compare the performance of students who programmed individually to that of students who programmed in pairs, but to our knowledge, no other researchers have directly assessed programming performance with laboratory practica. In addition to our primary metric of laboratory practica scores, we have also collected and analyzed data on written exam, homework, and laboratory assignment scores; test coverage on laboratory practica; pre- and post-semester attitude surveys and retention into and performance in the subsequent computer science course. The analysis of these results and a comparison to the four studies reviewed in the next section are presented in the Results section of this article.

2. PAST STUDIES

A survey of the literature showed four major controlled studies of the effects of pair programming on aspects of student learning. Each of these studies involved more than 100 students, had both control (solo) and study (paired) groups and used rigorous statistical techniques for analysis. The solo groups completed selected programming tasks individually while the paired groups completed those tasks using pair programming. Various measures of performance and attitude were compared between the solo students and the paired students to measure the effects of pair programming. The results from several of these studies have been reported across multiple publications with no comprehensive report. This section collects the implementation details and results from these studies and presents them in a single location to facilitate comparisons with our study and with future studies of the effects of pair programming. Further, our review focuses on the strongest measures of the effects that pair programming has on individual performance. Thus, we have omitted a number of reported measures including those that compare work completed in pairs to work completed individually.

2.1 North Carolina State University

Researchers at North Carolina State University (NCSU) conducted the largest of the four studies [Nagappan et al. 2003a, 2003b; Williams et al. 2002a, 2002b]. Lab sections of the CS1 course were designated as either paired or solo after student registration.

Table I. Summary of the NCSU Pair Programming Study

Course	• CS1 in College of Engineering at large university (>31,000 students) • Two 50-minute lectures per week • One 3-hour closed lab per week ○ 24 students per lab section
Solo Sections	• Programmed individually during 3-hour closed lab • Completed written exams individually • Could elect to pair program on three out-of-class projects
Paired Sections	• Pair-programmed during 3-hour closed lab period ○ Pairs assigned randomly ○ Pairs changed every 3–4 weeks ○ Driver/Navigator role changes encouraged "periodically" • Completed written exams individually • Could elect to pair program on three out-of-class projects
Study Population	• 661 students across three semesters: ○ Fall 2001 (69 solo/44 paired) ○ Spring 2002 (102 solo/280 paired) ○ Fall 2002 (110 solo/56 paired) • Wide range of students, mostly 1st years in College of Engineering • Considered 1st and 2nd year students taking the course for a grade • Lab sections randomly selected as paired/solo ○ Analysis used SAT-MATH to account for differences in initial populations
Performance Measures	• Written mid-term and final exam scores • Success in the course (final grade of C or better) • Performance in CS2 course with solo programming: ○ Grade in CS2 ○ Change in grade from CS1 to CS2

Students in the paired lab sections used pair programming on assignments during a three-hour closed lab session once a week. Students in the solo sections completed the same lab assignments individually. Students in both the paired and solo sections could elect to use pair programming on three out-of-class programming projects. These out-of-class programming projects and the written exams were also identical for students in the paired and solo sections. Additional details of this study are summarized in Table I.

With respect to mid-term and final exams there were no statistically significant differences in scores between students in the paired sections and those in the solo sections. Reported results on the success of students in the CS1 course and also on performance in the subsequent course, which required solo programming, are mixed. Significantly more students from the Fall 2001 and Fall 2002 paired sections received a grade of C or higher than students in the solo sections. However, there was no significant difference between the section types in Spring 2002. An analysis of the combined Fall 2001 and Spring 2002 data showed that among non-CS majors, more students from the paired sections received a C or higher, while among declared CS majors there was no significant difference in grade between students from the paired sections and those from the solo sections.

Among those who took CS1 in Fall 2001, students from the paired section earned more As and Bs in CS2 than those from the solo section. In addition, students from the paired section experienced fewer grade decreases from CS1 to CS2. Among those who took CS1 in Spring 2002, students from the solo section earned more As and Bs in CS2 than those from the paired section. However, those from the paired section still saw fewer grade decreases than those from the solo section. Based on these results the authors conclude that pair programming in CS1 is most beneficial for non-CS majors and that pairing in CS1 has no detrimental effects on performance in future courses where solo programming is required.

Table II. Summary of the UCSC Pair Programming Study

Course	• Introductory programming for CS, Info. Sys. Mgmt. and CE at a large university (>15,000 students) • Three 50-minute lectures per week • One optional 90-minute TA help session (lab) per week
Solo Sections	• Completed all programming assignments, quizzes and exams individually
Paired Sections	• Pair programmed on five out-of-class programming assignments ○ Pairs assigned using expressed student preferences ○ Pairs fixed for entire term ○ Roles alternated from "hour to hour" • Completed all quizzes and exams individually
Study Population	• 552 students across one academic year ○ Fall 2000 (172 paired) ○ Winter 2001 (232 paired) ○ Spring 2001 (148 solo*) * 17 of these students were in the Fall 2000 or Winter 2001 sections but programmed individually and were included in the 148 solo programmers for analysis. • Primarily CS, Info. Sys. Mgmt and CE majors • Paired/solo populations drawn from different semesters ○ Statistical tests using SAT and prior GPA allowed the assumption that initial populations were similar.
Performance Measures	• Course completion (final exam was taken) • Final exam scores • Success in the course (final grade of C or better) • Retention in Computer Science ○ Registration for second programming course ○ Passing second programming course ○ Declaring CS major

2.2 University of California Santa Cruz

An extensive study of the effects of pair programming has been carried out at the University of California Santa Cruz (UCSC) [McDowell et al. 2002, 2003, 2006; Werner et al. 2004]. The primary study occurred during the 2000–2001 academic year. In the Fall 2000 term, an instructor required students in one section of the introductory course to complete five programming assignments using pair programming. In Spring 2001, the same instructor taught another section in which students were required to complete different but comparable programming assignments individually. The lectures and quizzes were also different but comparable in these two sections. The final exams were identical. In the Winter 2001 term two additional instructors taught sections of the course that used pair programming. Further details of this study are summarized in Table II.

With respect to course completion, students from the paired sections were more likely to complete the course than students from the solo sections. Among students who completed the course during the Fall 2000 and Spring 2001 sections (the Winter 2001 sections were not included in this analysis), there was no significant difference in final exam scores between the paired and solo sections. The authors hypothesized that weaker students dropping from the solo section might have caused this lack of a difference. To validate this hypothesis, they removed the same percentage of students who had dropped the solo section from the bottom (measured by course grade or final exam score) of the paired section and found that the remaining students in the paired section scored higher on the final exam than those from the entire solo section. When all students who completed the course were analyzed, no significant difference was found between the section types in the percentage of students receiving a final grade of C or higher in the course. However, if all students who began the course are considered

Table III. Summary of the UA Pair Programming Study

Course	• Second-year software design and construction at a large university (>38,000 students) • Three 50-minute lectures per week – common for all students • One closed lab session per week – attendance was optional ○ 90 minutes in 2004 ○ 60 minutes in 2005
Solo Sections	• Completed all assignments and exams individually
Paired Sections	• Completed in-lab assignments using pair programming ○ Pairs assigned randomly ○ Pairs changed every four weeks ○ Roles alternated every 20 minutes • Completed all other assignments and exams individually
Study Population	• 490 students across two semesters ○ Fall 2004 (172 solo/110 paired) 14 solo and 4 paired students dropped the course. ○ Fall 2005 (106 solo/74 paired) 6 solo and 4 paired students dropped the course. • Students who dropped the course were excluded from analysis. • Lab sections randomly selected as paired/solo ○ No check for differences in initial populations
Performance Measures	• Success in course (final grade of C- or better) • Project scores • Written test scores • Final exam scores

(i.e., including those who withdrew from the course), students from the paired sections were more likely to receive a final grade of C or higher.

Among those students who passed the first CS course, a greater percentage of those from the paired sections registered for the second CS course within a year. This was also true of the sub-population of students who entered the first course intending to pursue a CS-related major. With respect to passing the second CS course (which required solo programming), the results are mixed. When considering all students who passed the first course with a C or better, a larger percentage of students from the paired sections than from the solo sections also passed the second course. However, among the subset of these students who had expressed an intention to major in a CS related field, there was no significant difference in the percentage of paired and solo students who passed the second course. Of students who passed the first course and were still enrolled at the university one year later, a larger percentage of students from the paired sections had declared a CS major. This result also held when the sub-populations of men, women and those who entered the first course intending to pursue a CS-related major were analyzed separately.

2.3 University of Auckland

In the Fall 2004 and Fall 2005 semesters, researchers at the University of Auckland (UA) conducted two independent trials of an experiment to assess the effects of pair programming [Mendes et al. 2005, 2006]. In each semester all students attended a common lecture and signed up for one of a number of closed lab section that met once a week. After the students had signed up, a subset of the sections was chosen to use pair programming, while the others required solo programming. Students in the paired and solo lab sections completed the same in-lab assignments. In addition, the students also completed three individual programming projects, a written test and a written final exam, which were identical for all students. The scores on these individual assignments of students from the paired lab sections were compared to those of students from the solo lab sections. The details of this study are given in Table III.

Table IV. Summary of the PEC Pair Programming Study

Course	• Programming Languages, Internet Programming, Database Programming, Systems Programming • One 3-hour closed laboratory per week • Other aspects differed by course
Solo Sections	• Completed in-lab assignments individually
Paired Sections	• Completed in-lab assignments using pair programming ○ Pairs assigned with bias for similar ability and same gender ○ Roles alternated when required by TA
Study Population	• 214 students across four courses (98 solo/116 paired) • Students randomly assigned to paired/solo labs ○ No check for differences in initial populations
Performance Measures	• Scores on individual written test administered at end of lab period

A separate analysis was performed for each semester of this study. For both semesters, statistically significant differences were found between paired and solo programmers on all performance measures that were examined. Students assigned to the paired lab sections were more likely than those assigned to the solo labs to complete the course with a final grade of C- or better. Students assigned to the paired lab sections also earned higher test and final exam scores than those from the solo labs. In the most direct measure of student ability among the reviewed studies, students assigned to the paired labs received higher scores on their individual programming projects than did students in the solo labs.

Unfortunately, the validity of these findings is threatened by several factors. While the assignment of students to the paired/solo groups was effectively random, no tests were performed to verify the initial similarity of these groups. To satisfy the University's Human Ethics Committee, attendance at these lab sessions was optional, attendance data were not collected, and the work completed did not factor into the students' grades. Thus, the possibility exists that attendance at the paired lab sections was significantly greater than at the solo labs, giving students from the paired labs more practice with the material. In addition, the individual programming projects were completed outside of class and no information is available regarding the amount of assistance that was obtained in completing these projects.

2.4 Pondicherry Engineering College

Researchers at Pondicherry Engineering College (PEC) have studied the effects of pair programming on learning efficiency during the completion of short duration closed-laboratory exercises [Kuppuswami and Vivekanandan 2004]. Students in a variety of bachelor's and master's level courses were divided into paired and solo groups during weekly three-hour closed laboratory sessions. Students in the paired and solo groups completed the same exercises, though the exercises differed by course. Immediately after completing the in-lab exercises, all students individually took a 30-minute written test on the material addressed by the exercises. The scores achieved on these tests by students in the paired groups were compared to those achieved by the students in the solo groups. Additional study details are presented in Table IV.

The 30-minute written tests in each course were graded on a 20-point scale, and the scores from all four courses were aggregated into a single data set for analysis. This analysis showed that the mean score on the written tests for the paired groups was significantly higher than for the solo groups. While students were randomly assigned to paired or solo labs, the lack of a test for differences in the initial populations poses a threat to the validity of these results.

3. METHODLOLGY

We offered two sections of our course (COMP 131) each semester of the study period (Fall 2005–Spring 2007), for a total of eight sections. During each semester, students in the control (solo) section completed all assignments individually, while students in the paired section used pair programming for laboratory assignments and completed all other assignments individually. Students were not aware of the section type during the registration period, and were not allowed to switch between sections after the start of the semester. To control for instructor effects, each of the authors who taught study sections (Braught and Wahls) taught two paired and two solo sections during the period of the study.

In the paired sections, students were given a brief introduction to pair programming during the first lab period. Pairs were rotated every three to four labs as recommended by Srikanth et al. [2004]. Pairings for the first few labs were assigned randomly, while for later labs the pairings matched students of similar ability as measured by performance in the course to that point [Chapparo et al. 2005; Cliburn 2003]. During the lab period, we required pairs to exchange driver and navigator roles every 10 to 15 minutes. An automated timer that played amusing sound clips was used to remind pairs to exchange roles, even when the instructor was answering student questions.

Our key instrument for measuring student learning was the laboratory practica, which required students to program individually under exam conditions. The first lab practica given in each semester covered basic class construction (field declarations, constructors and method definitions) and nested conditionals, while the second emphasized using collections (arrays or ArrayLists) and iteration. All students (from both solo and paired sections) completed these practica individually, which allowed us to directly assess the individual programming skill of students who used pair programming. Additional measures of student learning that were examined include successful completion of the course (final grade greater than or equal to 70%), written exam scores, homework scores, and lab assignment scores. Student responses to pre- and post-semester surveys, registration in our second programming course and successful completion of that course were also analyzed. All of the materials for our course, the pre- and post-semester surveys and the raw data collected, are available via the project Web page.[1] The portions of the surveys relevant to this paper are also reproduced in Appendix A. Table V presents a summary of our study in the same format as the other reviewed studies.

3.1 Threats to Validity

Although we attempted to make the solo and paired sections as similar as possible, there were differences that may threaten the validity of our results. To control for instructor effects, we used a collaboratively developed set of lecture notes and examples, and all assignments and exams were the same across sections within each semester. Assignments were often revised between semesters and new exams and laboratory practica were written each semester. We used a detailed rubric for grading both written and laboratory practica, with frequent consultation on cases not directly covered by the rubric. In addition, because each instructor taught two paired and two solo sections, our statistical analysis was able to detect differences by instructor, as well as interactions of instructor with student ability (measured by SAT scores), section type (paired vs. solo) and gender. The few instances in which differences involving instructor were found are described in Sections 4.4 and 4.5.

[1]See http://users.dickinson.edu/~braught/NSFIntegrating/integrating.html.

Table V. Summary of the Dickinson College Pair Programming Study

Course	• CS1 course at a private liberal arts college (~2350 students) ○ Course fulfills a general education requirement and part of an elective for the math major • At most 24 students per course section ○ Three 50-minute lectures per week ○ One 2-hour open lab per week
Solo Sections	• Programmed individually during 2-hour open lab period • Completed homework, written exams and lab practica individually
Paired Sections	• Pair-programmed during 2-hour open lab period ○ Pairs random initially, and then by similar ability ○ Pairs changed every 3–4 weeks ○ Driver/Navigator change enforced every 10–15 minutes • Completed homework, written exams and lab practica individually
Study Population	• 176 students across four semesters: ○ Fall 2005 (19 solo/23 paired) ○ Spring 2006 (24 solo/23 paired) ○ Fall 2006 (24 solo/24 paired) ○ Spring 2007 (23 solo/16 paired) • Wide variety of students: ○ 94% non-majors taking the course as a general education requirement (~80%) or math elective (~15%) • Sections randomly assigned as paired/solo after registration ○ Statistical tests using SAT and SAT-MATH allowed the assumption that initial study populations were similar.
Performance Measures	• Written mid-term and final exam scores • Laboratory practica scores • Laboratory assignment scores • Homework scores • Success in the course (final grade of 70% or better) • Continuing in computer science: ○ Enrollment in CS2 ○ Successful completion of CS2

A second potential threat is that the common lab practica were administered to the control and paired sections on different days, which could allow information flow between the sections (i.e., cheating). For the first three semesters of the study, the solo section took the lab practica one day before the paired section did, while the paired section took the practica one day prior to the solo section during the final semester. During the first semester of the study period, we developed two versions of each lab practica, and administered each version to half of the students in each section. Close inspection revealed no evidence of cheating on lab practica during this (or any following) semester. Only one version of each practica was used in the remaining semesters of the study.

3.2 Statistical Methods

When designing our study we were interested in detecting not only if section type (paired vs. solo) had an effect on student learning and attitudes, but also whether such an effect differed depending upon instructor, gender, student ability (SAT), or some combination of these factors. Examples of the types of questions we wanted to address include: Did women who pair programmed successfully complete the course more often than men, or vice versa? Did the use of pair programming affect scores on the lab practica and if so did the effect vary based upon student ability?

In order to address such questions we used an interaction analysis. This analysis technique allows us to detect when differences in performance or attitudes were due

TOE00041 ACM (Typeset by SPi, Manila, Philippines) 9 of 21 February 18, 2011 14:1

to combinations (i.e., interactions) of the factors as well as when they were due to a single factor. For example, the detection of a significant section type × instructor interaction in homework scores would indicate that students who pair programmed performed differently on homework than students who programmed solo and that their difference in performance varied between instructors. When analyzing the metrics of student learning, lab practica scores, successful completion of the course, written exam scores, homework grades, and lab grades, we used a four-factor interaction analysis with the factors SAT, gender, section type, and instructor. When analyzing the data from student surveys we omitted SAT and used a three-factor analysis including only gender, section type, and instructor.

To further explain the interaction analyses used, we describe here the process used to analyze the pre- and post-semester survey data. The interaction analysis of each survey question proceeded by first testing for a three-way interaction among all of the factors (gender × section type × instructor). If that interaction was significant the analysis of the question stopped there because any simpler explanation involving fewer factors would be ignoring some factor that had contributed to the effect. If there was insufficient evidence to conclude a three-way interaction then all three two-way interactions were tested (gender × section type, gender × instructor, section type × instructor). If a two-way interaction was found to be significant, no tests were performed for the primary effects of the two factors making up that interaction because the significant interaction was the simplest explanation for the differences involving those two factors. If a two-way interaction was not found to be significant, then tests were performed for the primary effects of those two factors. When a primary effect is found for a factor it is an indication that that factor, and not any of the possible interactions, accounts for observed differences in the student responses. The analysis for student learning metrics was similar but began by testing for a four-way interaction and then, if appropriate, proceeding through all three three-way interactions, all six two-way interactions and finally to the four possible primary effects.

The statistical tests used in our interaction analysis varied as necessary to match the metric being analyzed. For all student learning metrics except successful course completion, a continuous-response ANCOVA test was used with SAT as the covariate. Because successful course completion is a binary variable, a binary-response ANCOVA (i.e., logistic regression analysis of covariance) was used in its analysis. In both cases the ANCOVA used allowed for a linear relationship between SAT score and the metric being analyzed. The pre- and post-semester survey responses were analyzed using standard (i.e., continuous-response) ANOVA tests. The advantage of using an ANOVA test over a non-parametric test, such as Wilcoxon Rank-Sum or Kruskal-Wallis, is that it allows us to test for interactions in addition to the primary effect of each factor. Further, this test ensures that each effect is evaluated after adjusting for all of the other effects. This is particularly beneficial due to the differences in student counts among the various section type/gender/instructor combinations (see Table VI).

When the results of these interaction analyses are reported we will typically only discuss the significant interactions or primary effects. Thus, if a significant two-way interaction is reported (e.g., SAT × section type) then the four-way and all three-way interactions were non-significant. When reporting test results we include the test statistic value, the degrees of freedom if appropriate, and the p-value parenthetically in the text. For example, an F-test with 2 and 30 degrees of freedom having a test statistic value 12.34 and p-value 0.0123 would be reported as ($F_{2,30} = 12.34$, 0.0123). Unless noted otherwise, all statistical tests were performed with a significance level of $\alpha = 0.05$.

Table VI. Breakdown of All Students by Section Type (Paired/Solo), Gender (M/F) and Instructor (I1/I2). Bold Values are Totals for the Respective Categories

		Total **(176)**			Starters **(151)**			Completers **(137)**		
		I1	I2		I1	I2		I1	I2	
Paired	M	28	26	**54**	26	22	**48**	23	20	**43**
	F	18	14	**32**	16	13	**29**	15	12	**27**
		46	**40**	**86**	**42**	**35**	**77**	**38**	**32**	**70**
		I1	I2		I1	I2		I1	I2	
Solo	M	26	30	**56**	21	28	**49**	18	24	**42**
	F	21	13	**34**	18	7	**25**	18	7	**25**
		47	**43**	**90**	**39**	**35**	**74**	**36**	**31**	**67**

4. RESULTS

4.1 Preliminaries

During the study period, a total of 176 students enrolled in COMP 131 and consented to participate in the study. Of those who consented, 151 had SAT or ACT[2] scores on file with the College and remained in the course through the first week. We will refer to this group of 151 students as the Starters. Of the 151 Starters, 137 completed both of the laboratory practica. This group of 137 will be referred to as the Completers. Table VI shows the breakdown of all students who consented, the Starters and the Completers by section type, by gender and by instructor.

Before any detailed analysis was performed, an ANOVA was run on the Completers to test for differences in the various initial populations with respect to SAT and SAT-MATH. An α of 0.10 was used here to make it easier to detect SAT and SAT-MATH differences in the initial populations. This test showed insufficient evidence for a difference in either SAT or SAT-MATH scores between students in the paired sections and those in the solo sections ($F_{1,133}$ = 0.01, 0.9278; $F_{1,118}$ = 1.79, 0.1838). Similarly insufficient evidence was found for differences in SAT or SAT-MATH across genders ($F_{1,133}$ = 0.01, 0.9257; $F_{1,118}$ = 1.58, 0.2113) or across the sections taught by the two different instructors ($F_{1,133}$ = 0.39, 0.5352; $F_{1,118}$ = 0.37, 0.5455). Based on this analysis we proceeded under the assumption that the initial populations were similar when divided by section type, gender or instructor.

Further tests were performed to determine if SAT scores were reasonable predictors of performance in our course. A binary response ANCOVA (i.e., logistic regression analysis of covariance) using the Starters indicated that successful completion of our course (final grade >= 70%) was positively correlated with SAT scores (χ^2_1 = 13.43, 0.0002). In addition, a continuous response ANCOVA using the Completers provided evidence of a positive correlation between students' SAT scores and their scores on written exams ($F_{1,126}$ = 40.50, 0.0000), laboratory practica ($F_{1,126}$ = 19.87, 0.0000),

[2]ACT scores were converted to equivalent SAT scores using the concordance provided by the College Board [Schneider and Dorans 1999].

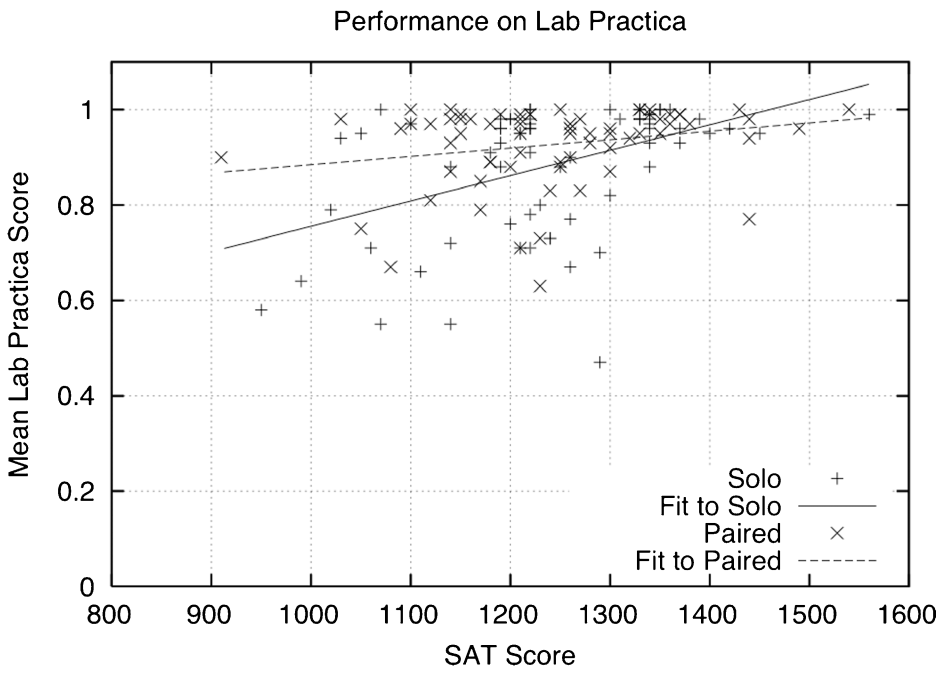

Fig. 1. Illustration of SAT x section type interaction.

laboratory assignments ($F_{1,132}$ = 11.03, 0.0012), and final grades ($F_{1,126}$ = 25.93, 0.0000). Only scores on the homework assignments failed to show sufficient evidence of a correlation with SAT scores ($F_{1,126}$ = 2.93, 0.0893). Given the correlation with SAT scores of performance on the majority of course components, our later analysis of these metrics included SAT score as a covariate.

4.2 Individual Programming Skills

Performance on the laboratory practica was our primary metric for assessing student learning with respect to individual programming skills. A four-factor interaction analysis of the Completers' average lab practica score using ANCOVA revealed evidence of an SAT × section type interaction ($F_{1,126}$ = 4.47, 0.0364), but showed insufficient evidence to infer the existence of any of the other possible interactions. Subsequent t-tests at multiple SAT levels showed, with $\alpha = 0.05$, that among students with SAT scores below 1265, students in the pair programming sections scored higher on the laboratory practica than students from the individual programming sections at the same SAT level. At SAT levels above 1265, differences in lab practica scores between section types were not statistically significant.

The SAT x section type interaction is visualized in Figure 1. Each point on the graph is the average lab practica score of a student. The fit lines represent the linear relationship between SAT and lab practica score that was allowed for in the ANCOVA. The detection of the SAT × section type interaction equates to finding sufficient evidence of a difference in the slope of the fit lines. The subsequent analysis for significant differences equates to testing for a difference in the y-coordinate of the fit lines at each SAT score.

As mentioned above, there was insufficient evidence to infer any interactions other than SAT x section type. Notably, there were no interactions involving gender nor were there any interactions involving instructor. The absence of sufficient evidence for an SAT × section type × gender interaction ($F_{1,122}$ = 0.46, 0.5012) argues against any conclusion that pair programming affects men and women differently with respect to the acquisition of individual programming skills. The lack of evidence for the existence of instructor effects suggests that the SAT × section type interaction was similar for both instructors. An earlier paper based on these results considered the possibility that the use of pair programming may help students adapt to different instructors [Braught et al. 2008].

In addition to the above analysis, we also examined the effects of pair programming on two sub-components of the lab-practica scores, the percent of reference tests passed and the percent of program statements covered by student-written unit tests. These two sub-components are of particular interest because they were machine graded, guaranteeing consistency across students and freedom from instructor grading bias. Unfortunately, we were not able to match these sub-components to specific students. Thus the analysis of these sub-components allows us to test only for a primary effect due to section type and does not, as all of our other analyses do, allow us to test for effects due to SAT, instructor, or gender, or their interactions. Despite this shortcoming, we feel that the objective nature of these two measures justifies their inclusion here.

For reference tests, we found sufficient evidence to conclude that on average students from the paired sections generated solutions to the lab practica that passed a higher percentage of reference tests than did students from the solo sections (92% vs 82%) (t_{163} = 2.64, 0.0091). For statement coverage, we found sufficient evidence to conclude that on average the unit tests submitted by students in the paired sections achieved a greater degree of statement coverage than did students from the solo sections (96% vs 90%) (t_{163} = 2.41, 0.0172).

The findings in this section provide direct evidence that the use of pair programming aids in the development of individual programming skills, particularly for lower SAT students. In addition, our results provide evidence that the use of pair programming can facilitate the acquisition of testing skills. Of the studies reviewed in the Past Studies section, only the UA study included a direct measure of students' programming ability. They found that students assigned to optional lab sessions where pair programming was used scored higher on individually completed programming projects than students assigned to lab sessions that used individual programming. All of the other measures of student programming ability in the reviewed studies are indirect measures such as scores on written exams. Therefore, a detailed comparison of the results from this section to past studies is not possible. However, the results from this section do reinforce the general conclusion of those studies that pair programming is beneficial for student learning.

4.3 Successful Course Completion

To analyze successful course completion, Starters who completed the course and received a final grade of 70% (C-) or higher were classified as having successfully completed the course while Starters who either dropped the course after the first week or received a final grade of less than 70% were classified as unsuccessful. The four-factor interaction analysis of this metric using a binary-response ANCOVA revealed insufficient evidence of interaction among the factors but sufficient evidence of a primary effect for both section type (χ^2_1 = 4.72, 0.0297) and gender (χ^2_1 = 4.56, 0.0327). Notably there was insufficient evidence of an instructor effect (χ^2_1 = 0.03, 0.8739).

Evaluation of the section type effect revealed that at a given SAT level, students who pair programmed were more likely to successfully complete our course than those who programmed individually. In our study, the odds of successful completion for those who pair programmed was 2.8 times that for those who programmed individually, where odds = probability of success / probability of failure. One reasonable explanation for this would be that students who pair programmed in lab received higher lab assignment scores than those who programmed solo. Thus, those who paired would succeed in the course at a greater rate because of their higher lab assignment scores. However, an ANCOVA showed insufficient evidence of a difference in the means of lab assignment scores between the paired and solo sections ($F_{1,132}$ = 2.66, 0.1052).

Evaluation of the gender effect indicates that at a given SAT level, female students were more likely to successfully complete our course than male students. It is important to note that even though both students who pair programmed and women had greater odds of successful completion of the course, an interaction analysis did not support a section type × gender interaction. This means that we cannot conclude that the effects of pair programming on successful course completion differ by gender or that the gender effect on successful course completion differs by section type. Thus, while women at a given SAT level have greater odds of successfully completing our course than men at the same SAT level, these increased odds cannot be attributed to the use of pair programming.

Three of the four major studies reviewed also used successful course completion as a metric for evaluating the impact of pair programming. However, each study's criterion for successful completion was slightly different. Our study and the UA study defined successful completion as a grade of C- or better while the NCSU and UCSC studies used a grade of C or higher. Our study and the UCSC and NCSU studies counted students who withdrew from the course among the unsuccessful, while the UA study omitted them from the analysis. Those differences aside, our results are in agreement with the studies at UCSC and UA which both found that pair programming increased the likelihood of successful course completion. The study at NCSU found this to be true for non-CS majors but not for CS majors. Given that the enrollment in our course is over 90% non-majors, our results are also consistent with those from the NCSU study.

4.4 Other Course Metrics

Four-factor interaction analyses using ANCOVA were also performed on the other course metrics (homework scores, lab assignment scores, written exam scores, final exam score and final course grade) revealing a number of instructor effects. There was sufficient evidence of a gender × instructor interaction for homework average ($F_{1,126}$ = 6.40, 0.0126), written exam average ($F_{1,126}$ = 5.77, 0.0178), and final course grade ($F_{1,126}$ = 4.28, 0.0407). For all three of these metrics, this interaction resulted from females at a given SAT level averaging higher than males at similar SAT levels in Instructor 2's sections. For the average lab assignment score, there was sufficient evidence of a primary instructor effect ($F_{1,132}$ = 5.07, 0.0260). On the average, students at a given SAT level obtained higher lab assignment scores from Instructor 2 than from Instructor 1. There was insufficient evidence for any effect, interaction or primary, involving section type, so we cannot conclude that any of these metrics are affected by the use of pair programming. This also suggests that the observed differences between instructors were consistent across section types. Thus, the observed instructor effects on these metrics do not impact the successful course completion results reported in the previous two sections.

These results along with those from the other four major studies paint a mixed picture of the effects of pair programming on metrics such as homework and written exam scores. The NCSU study found no difference in mid-term or final exam scores based on the use of pair programming. Similarly, the UCSC study found no difference in final exam scores between those who pair programmed and those who programmed individually. However, when accounting for a higher withdrawal rate among the solo group, the UCSC study did find that students from the paired sections scored higher on the final exam. The UA study found that students assigned to paired lab sections received higher scores on both mid-term and final exams. Similarly, the PEC study found that students from paired labs scored higher on written assessments of the material learned in labs than those from solo labs. These mixed results combined with our stronger and more direct results presented in the previous sections suggest that while pair programming may impact other areas of student learning, its benefits are realized primarily in the acquisition of programming and testing skills—precisely those things practiced while using pair programming.

4.5 Student Perceptions and Frustration Levels

We administered pre- and post-semester surveys in each section of the course offered during the study period. These surveys questioned students about their attitudes toward the course, their experiences in the course, and also their perceptions of their own abilities. Each of the survey questions discussed in this section has been reproduced in Appendix A.

One series of questions (#11–#17) on the post-semester survey asked students to report on their confidence in the correctness of their written exams, homework and lab assignments, and lab practica, and also on the thoroughness of their testing of homework, lab assignments, and lab practica. Each question asked students to express their degree of agreement with statements such as: “When I completed my laboratory practica, I was confident that they were correct” or “When developing tests for my homework assignments I was confident that I had tested thoroughly.” For each question students expressed their degree of agreement on a scale from 0 (strongly disagree) to 5 (strongly agree). The survey specifically asked students to respond based on their confidence levels at the time they submitted their work. However, because students had seen graded work throughout the semester it is possible that the differences in student confidence reported below may not be independent of the differences in actual performance reported above.

A three-factor interaction analysis using ANOVA was performed on the responses to each survey question. Because SAT score was not used as a factor, this analysis included all students who consented, not just those with SAT scores on file, and who also completed a post semester survey ($n = 141$, except on question #13 about correctness of lab practica where $n = 140$).

This analysis revealed evidence of several primary effects of section type on student confidence in their solutions. Students from the paired sections reported greater confidence than students from the solo sections in the correctness (3.8 vs. 3.3) ($F_{1,136} = 6.35, 0.0129$) and thoroughness of testing (3.8 vs. 3.2) ($F_{1,137} = 8.87, 0.0034$) of their lab practica solutions. Students from the paired sections also reported greater confidence in the thoroughness of their testing on homework assignments than did students from the solo sections (3.5 vs. 3.2) ($F_{1,137} = 4.16, 0.0432$). Thus, students from the paired sections not only performed better when programming individually, but also had a more positive view of their own individual performance than did students from the solo sections.

A section type × gender × instructor interaction was found in the responses to the question regarding confidence in testing thoroughness on lab assignments ($F_{1,133} = 4.53, 0.0351$). One way of viewing this interaction is as a difference in the effect of pairing between men and women for one of the instructors but not for the other. Specifically, Instructor 2’s paired males reported greater confidence than Instructor 2’s solo males (3.9 vs. 2.8) ($t_{133} = 3.48, 0.0007$) while Instructor 2’s females and all of

Instructor 1's students reported statistically similar levels of confidence. Insufficient evidence was found for any interactions involving section type or any primary effect of section type on students' confidence in the correctness of their solutions to homework assignments, lab assignments, or written exams.

Unlike the UCSC study [Werner et al. 2004], we did not find evidence that the effect of pair programming on the students' confidence in their solutions was greater for women than for men. A number of factors may account for this difference in findings including different student populations, different instructors, different course content, differences in the survey instruments, and the timing of the survey administration (end of the semester in our study vs. at time of assignment submission in Werner). The question of how the effects of pair programming on student confidence differ by gender is worthy of further study.

Another pair of questions (#7, #8) on the post-semester survey asked students to report their level of frustration when working on homework assignments and lab assignments. Frustration levels were reported on a scale from 1 (very low) to 5 (very high). Again three-factor interaction analysis with ANOVA was used to test for interactions among the factors and for a primary effect of each factor. Responses from all students who consented to participate in the study and completed a post-semester survey were also included in this analysis (n = 141). This analysis revealed insufficient evidence of any differences in students' reported frustration when working on homework assignments. In students' reported frustration when working on lab assignments a section type × gender × instructor interaction was found ($F_{1,133}$ = 6.18, 0.0142). Two of the significant differences driving this interaction relate to the effects of pair programming. First, among Instructor 1's female students, those from solo sections reported a greater degree of frustration than those from paired sections (4.3 vs 3.5) (t_{133} = 2.28, 0.0243). Second, in instructor 1's solo sections females reported greater frustration than males (4.3 vs. 3.4) (t_{133} = 2.82, 0.0055), while in his paired sections males and females reported similar levels of frustration (3.5 and 3.3). These results taken together suggest that pair programming may help to reduce the frustration level of female students.

4.6 Continuation in Computer Science

Our pre- and post-semester surveys each contained a question asking the student to complete the sentence "Right now my plan is to. . . " with one of five options. On the pre-survey the options were: Take just this course, Take at least two courses, Minor, Major, and Unsure. On the post survey the options were: Not take any more courses, Take another course, Take several more courses, Minor, Major, and Unsure. The responses to these questions were analyzed for all of the Completers who had filled out both surveys and were not graduating seniors (n = 123). Each students' pair of responses were categorized as negative (e.g., Pre: Major to Post: Unsure), neutral, or positive (e.g., Pre: Minor to Post: Major). As shown in Table VII, approximately one-half of the students had no change in intention. For those who had a change (n = 63), almost two-thirds had a positive change in intention. Using a proportion z-test, there was sufficient evidence to conclude that a student is more likely to have a positive change in intention than a negative change (z = 2.39, 0.0168). However, a binary-response ANOVA on the proportion with a positive intention change showed insufficient evidence of a difference by section type, gender, or instructor, or any interaction involving these three factors. Thus, while the course overall had a positive effect on students' intentions toward computer science, that effect cannot be attributed to instructor, gender or whether students pair programmed. A binary-response ANOVA on the proportion with a negative attitude change yielded similar results. However, with respect to no

Table VII. Student Intentions with Regard to Computer Science

Attitude	**Count**	**Percent**
Negative	22	17.9
Neutral	60	48.8
Positive	41	33.3

Table VIII. Student Continuation in Computer Science

	Enrolled in COMP 132	Completed COMP 132	Earned >= C- in COMP 132	Declared a CS Major
Paired in 131	23 (29%)	19 (83%)	17 (74%)	6 (26.1%)
Solo in 131	19 (25%)	16 (84%)	16 (84%)	5 (26.3%)

attitude change (i.e., neutral), we found that females were more likely than males to have no change in attitude ($\chi^2_1 = 4.21, 0.0401$).

We also studied students' actual behaviors regarding registration for and completion of the subsequent CS course, which required solo programming. Table VIII shows the number and percentage of students who took COMP 131 during our study who eventually continued on to our second course (COMP 132). Also shown is the number and percentage of these students who completed COMP 132, and for those who did complete COMP 132, the number and percentage who earned final grades of C- or higher, and the number who eventually declared a computer science major.

A slightly higher percentage of students who paired in COMP 131 enrolled in COMP 132 (29% paired vs. 25% solo). Of the students who enrolled in COMP 132, there was virtually no difference in the percentage of those who completed the course. However, a greater percentage of students who programmed solo in COMP 131 earned a C- or better when taking COMP 132 (84% solo vs. 74% paired). Not surprisingly, because of the small number of students continuing on to COMP 132, neither of these differences is statistically significant (Fisher's Exact Test p-values: 1.000 and 0.477).

That said, one possible explanation for the greater percentage of solo students who successfully completed COMP 132 is that pairing in 131 increased the likelihood of weaker students deciding to continue on to the subsequent course, in which they did not fare as well. This reasoning is analogous to that used in the UCSC study's analysis of final exam scores. In our case, of the five students who earned a C or worse in COMP 131 and continued on to COMP 132, four came from paired COMP 131 sections and none earned better than a C- in COMP 132.

Given the small numbers of students who continued on to COMP 132 and the mixed nature of our results, we cannot draw any conclusions regarding the effects of pair programming on continuation into and success in subsequent computer science courses. This is largely consistent with the results of the NCSU study, in which the relative success of paired and solo students in the following course was ambiguous. Our results are not consistent with those observed at UCSC, where paired students who were not computer science majors were more successful in the following course. As the majority of our students who continued on to COMP 132 were not majors, we had hoped to see similar results but did not. Another study by Carver et al. [2007] at Mississippi State University found that first-year declared CS/CE/SE majors were more likely to remain a major when they used pair programming in their Introduction to Programming Course. These inconsistencies present an area for further study. At this point, the evidence presented by our study and by those at NCSU, UCSC and Mississippi State University indicates that, at worst, pair programming in CS1 does not adversely affect student retention into and performance in a CS2 course that requires solo programming. At best, it may be beneficial, particularly for non-CS majors.

4.7 Instructor Observations

In addition to the above quantitative results, we observed several effects of pair programming on student (and instructor) enjoyment of the course. In the laboratory periods of the solo sections, students often had their hands in the air for several minutes at a time as the instructor and teaching assistant hurried from student to student answering questions. Except for these tense, low-voiced conversations and the clack of keys, the room was usually intensely quiet, and even students sitting at neighboring computers seemed completely isolated from one another. The atmosphere in lab periods of the pair programming sections was strikingly different. The room buzzed with spirited discussions between driver and navigator on the best way to approach some programming task. Pairs were typically able to find the solutions to lower level problems such as syntax and type errors with no outside assistance, and so questions for the instructor and teaching assistant were much less frequent. This is consistent with the empirical evidence reported by Müller [2007] and by Hanks [2008] indicating that pair programmers resolve more problems on their own. When the members of a pair did ask a question, it was often in regard to higher-level issues such as the elegance or efficiency of a particular approach, or the best way to achieve statement coverage when writing unit tests for a complex snippet of code. In short, lab was a much more enjoyable experience for both students and instructors. While these observations are anecdotal, they are consistent with our survey data on student frustration levels and consistent with reports by other researchers [McDowell et al. 2006; Simon and Hanks 2008; Williams and Kessler 2000; Williams et al. 2002a].

Another observation that we have made is on the importance of pairing students of like ability. Several studies have suggested that such a pairing mechanism is beneficial for student happiness [Chapparo et al. 2005; Cliburn 2003]. We believe that it is also beneficial for the acquisition of individual skills, particularly for weaker students. Conventional wisdom would pair a weaker student with a stronger student in hopes that the stronger student will teach the weaker student. However, in a pair programming session, what often happens is that the stronger student will either just do things correctly when driving or quickly point out how to correct errors when navigating. Thus, unless extremely diligent about asking why, the weaker student is prevented from struggling with the types of errors that he or she will encounter when programming alone. When paired with students of similar ability, weaker students encounter and work through problems typical of what they would see independently (e.g., mismatched {} or (), incorrect conditional logic—&& vs. ||). In addition, weaker students see that they are not alone in the difficulties that they are experiencing and thus may be less likely to become hopelessly discouraged. Our intuition on this issue has been strengthened by the experiences of several colleagues who taught our course using random pairing before switching to pairing by ability [Braught et al. 2010].

5. CONCLUSIONS

Our results indicate improvements in the acquisition of individual programming skill for students at lower SAT levels with no significant adverse effects for students at higher SAT levels. The consistency of the effect of pair programming on successful course completion across different courses, different types of institutions, different instructors and different student audiences argues strongly for its generality. In our study, students who pair programmed in lab were more successful in completing programming tasks individually and under exam conditions than students who programmed solo in lab—their lab practica contained fewer defects and were more thoroughly tested. They also reported greater confidence in the correctness of the programs they produced during the lab practica and in the thoroughness of their testing, and were in fact correct in this self-assessment. Although our analysis looked for, and we had anticipated, a number of measures where pair programming benefitted female students more than males, we observed only one; a reduction in frustration when working on lab assignments with one of the two instructors. This apparent inconsistency with other reported results [Werner et al. 2004] suggests that further investigation is required to determine if pair programming affects the performance and/or attitudes of men and women differently. We also found no reason to believe that the use of pair programming hindered the performance of students in subsequent courses that required individual programming, or negatively impacted the recruitment or retention of students in the computer science major.

Given the direct evidence resulting from our study and the indirect measures reported by us and by other researchers, the quantitative and qualitative benefits of pair programming appear to greatly outweigh any perceived harms. With five major studies of pair programming now complete and largely in agreement, we believe the case for the use of pair programming in the computer science classroom to be compelling.

APPENDIX A: SURVEYS

This appendix presents the questions from the pre- and post-semester surveys that have been discussed in this article. The text of the included questions and the scale used for responses has been reproduced here as they appeared in the surveys given to the students. The complete surveys can be found on the project web page.[3]

APPENDIX A.1 Pre-Semester Survey Question

4. Right now my plan is to:

Take just this CS Course	Take at least 2 CS Courses	Minor in CS	Major in CS	I'm unsure
☐	☐	☐	☐	☐

APPENDIX A.2 Post-Semester Survey Questions

1. Right now my plan is to:

Take another CS Course	Take several CS Courses	Minor in CS	Major in CS	I'm Unsure	Not take any more CS
☐	☐	☐	☐	☐	☐

7. My level of frustration while working on the homework for this class was:

Very high	Above average	About average	Below average	Very low
☐	☐	☐	☐	☐

8. My level of frustration while working on the laboratory assignments was:

Very high	Above average	About average	Below average	Very low
☐	☐	☐	☐	☐

[3]See http://users.dickinson.edu/~braught/NSFIntegrating/integrating.html.

9. My level of anxiety about taking the laboratory exams was:

Very high	Above average	About average	Below average	Very low
☐	☐	☐	☐	☐

10. My level of anxiety about taking the written exams for this class was:

Very high	Above average	About average	Below average	Very low
☐	☐	☐	☐	☐

The following questions ask about the confidence that you had in your work at the point when you turned it in, before receiving any feedback. Please try to answer these questions thinking only about how you felt about the work after you turned it in but before it was graded and returned to you.

11. When I submitted my laboratory assignments I was confident that they were correct.

Strongly Agree	Mildly Agree	Agree	Mildly Disagree	Disagree	Strongly Disagree
☐	☐	☐	☐	☐	☐

12. When I submitted my homework assignments I was confident that they were correct.

Strongly Agree	Mildly Agree	Agree	Mildly Disagree	Disagree	Strongly Disagree
☐	☐	☐	☐	☐	☐

13. When I completed my laboratory exams I was confident that they were correct.

Strongly Agree	Mildly Agree	Agree	Mildly Disagree	Disagree	Strongly Disagree
☐	☐	☐	☐	☐	☐

14. When I handed in my written exams I felt confident that I had done well.

Strongly Agree	Mildly Agree	Agree	Mildly Disagree	Disagree	Strongly Disagree
☐	☐	☐	☐	☐	☐

15. When developing tests for my lab programs I was confident that I had tested thoroughly.

Strongly Agree	Mildly Agree	Agree	Mildly Disagree	Disagree	Strongly Disagree
☐	☐	☐	☐	☐	☐

16. When developing tests for the laboratory exams I was confident that I had tested thoroughly.

Strongly Agree	Mildly Agree	Agree	Mildly Disagree	Disagree	Strongly Disagree
☐	☐	☐	☐	☐	☐

17. When developing tests for my homework assignments I was confident that I had tested thoroughly.

Strongly Agree	Mildly Agree	Agree	Mildly Disagree	Disagree	Strongly Disagree
☐	☐	☐	☐	☐	☐

ACKNOWLEDGMENTS

We would like to thank Dr. Stephen Edwards of Virginia Tech for supporting our use of WebCAT, which provided quantitative data on student program quality and testing.

REFERENCES

BECK, K. AND ANDRES, C. 2004. *Extreme Programming Explained: Embrace Change* 2nd Ed. Addison-Wesley Professional.

BRAUGHT, G., EBY, L. M., AND WAHLS, T. 2008. The effects of pair-programming on individual programming skill. In *Proceedings of the 39th Technical Symposium on Computer Science Education (SIGCSE'08)*. 200–204.

BRAUGHT, G., MACCORMICK, J., AND WAHLS, T. 2010. The benefits of pairing by ability. In *Proceedings of the 41st Technical Symposium on Computer Science Education (SIGCSE'10)*. 249–253.

CARVER, J., HENDERSON, L., HE, L., HODGES, J., AND REESE, D. 2007. Increased retention of early computer science and software engineering students using pair programming. In *Proceedings of the Conference on Software Engineering Education and Training (CSEET'07)*. 115–122.

CHAPPARO, E., YUKSEL, A., ROMERO, P., AND BRYANT, S., 2005. Factors affecting the perceived effectiveness of pair programming in higher education. In *Proceedings of the 17th Workshop of the Psychology of Programming Interest Group (PIPIG'05)*. 5–18.

CLIBURN, D. C. 2003. Experiences with pair programming at a small college. *J. Comput. Small Coll. 19*, 1, 20–29.

HANKS, B. 2008. Problems encountered by novice pair programmers. *J. Educ. Resour. Comput. 7*, 4, 1–13.

KUPPUSWAMI, S. AND VIVEKANANDAN, K. 2004. The effects of pair programming on learning efficiency in short programming assignments. *Informat. Educ. 3*, 2, 251–266.

MCDOWELL, C., WERNER, L., BULLOCK, H., AND FERNALD, J. 2002. The effects of pair-programming on performance in an introductory programming course. *SIGCSE Bull. 34*, 1, 38–42.

MCDOWELL, C., WERNER, L., BULLOCK, H. E., AND FERNALD, J. 2003. The impact of pair programming on student performance, perception, and persistence. In *Proceedings of the 25th International Conference on Software Engineering (ICSE'03)*. 602–607.

MCDOWELL, C., WERNER, L., BULLOCK, H. E., AND FERNALD, J. 2006. Pair programming improves student retention, confidence, and program quality. *Comm. ACM 49*, 8, 90–95.

MENDES, E., AL-FAKHRI, L. B., AND LUXTON-REILLY, A. 2005. Investigating pair-programming in a second-year software development and design computer science course. *SIGCSE Bull. 37*, 3, 296–300.

MENDES, E., AL-FAKHRI, L., AND LUXTON-REILLY, A. 2006. A replicated experiment of pair-programming in a second-year software development and design computer science course. *SIGCSE Bull. 38*, 3, 108–112.

MÜLLER, M. M. 2007. Do programmer pairs make different mistakes than solo programmers? *J. Syst. Softw. 80*, 9, 1460–1471.

NAGAPPAN, N., WILLIAMS, L., FERZLI, M., WIEBE, E., YANG, K., MILLER, C., AND BALIK, S. 2003a. Improving the CS1 experience with pair programming. In *Proceedings of the 34th Technical Symposium on Computer Science Education (SIGCSE'03)*. 359–362.

NAGAPPAN, N., WILLIAMS, L., WIEBE, E., MILLER, C., BALIK, S., FERZLI, M., AND PETLICK, J. 2003b. Pair learning: With an eye toward future success. In *XP/Agile Universe*. 185–198.

SRIKANTH, H., WILLIAMS, L., WIEBE, E., MILLER, C., AND BALIK, S. 2004. On pair rotation in the computer science course. In *Proceedings of the 17th Conference on Software Engineering Education and Training (CSEE&T'04)*. 144–149.

SCHNEIDER D. AND DORANS, N. 1999. Concordance between SAT I and ACT scores for individual students.

SIMON, B. AND HANKS, B. 2008. First-year students' impressions of pair programming in CS1. *J. Educ. Resour. Comput. 7*, 4, 1–28.

WERNER, L. L., HANKS, B., AND MCDOWELL, C. 2004. Pair-programming helps female computer science students. *J. Educ. Resour. Comput. 4*, 1, 4.

WILLIAMS, L. AND KESSLER, R. 2000. Experimenting with industry's "pair-programming" model in the computer science classroom. *J. Softw. Eng. Educ.*, 60–64.

WILLIAMS, L., LAYMAN, L., SLATEN, K. M., BERENSON, S. B., AND SEAMAN, C. 2007. On the impact of a collaborative pedagogy on African American millennial students in software engineering. In P*roceedings of the 29th International Conference on Software Engineering (SIGCSE'07)*. DC, 677–687.

WILLIAMS, L., WIEBE, E., YANG, K., FERZLI, M., AND MILLER, C. 2002a. IIn support of pair programming in the introductory computer science course. *Comput. Sci. Educ. 12*, 3, 197–212.

WILLIAMS, L., YANG, K., WIEBE, E., FERZLI, M., AND MILLER, C. 2002b. Pair programming in an introductory computer science course: Initial results and recommendations. In *Proceedings of the Symposium on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA'02)*.

Received July 2009; revised January 2010, June 2010; accepted July 2010