Using Collaboration to Overcome Disparities in Java Experience

Colleen M. Lewis Nathaniel Titterton Michael Clancy
Computer Science Division, EECS
University of California, Berkeley
Berkeley, CA 94720
{colleenl, xtitter, clancy}@cs.berkeley.edu

ABSTRACT
The lower-division CS curriculum at the University of California, Berkeley includes a version of CS 2 that is intended to introduce students to Java as well as data structures and programming methodology. Some students in the course already have Java experience. In one course offering, students without previous Java experience received final grades that were 0.27 standard deviations below their peers who already had some Java experience ($d=0.27$, $p<0.05$). In a subsequent offering, the instructor adopted course policies and teaching strategies that made student collaboration more frequent in hopes that students without Java experience could learn from their peers with Java experience. In this highly-collaborative offering, there were no statistically significant differences in average final grades between students with and without Java experience ($d=0.12$, $p>0.1$). A smaller percentage of students dropped the highly-collaborative offering than the less-collaborative offering. This decrease in attrition was most notable for female students, from 37 percent to 5 percent.

Categories and Subject Descriptors
K.3 [Computers and Education]: General

Keywords
Collaboration, group work, pair programming, competition, gender, satisfaction, lab-centric instruction, CS 2, attrition, programming experience

1. INTRODUCTION
This research compared two offerings of CS2 to examine the extent to which course policies that require and support collaboration can provide opportunities for students without Java experience to learn from and catch up with students that already had Java experience.

This study presents a comparison between two offerings of a data structures and programming methodology course (CS2) at U.C. Berkeley. The first offering (N=166), in the spring of 2008, emphasized individual work on homework assignments and multi-week programming assignments (projects). The second offering (N=227), in the spring of 2010, emphasized group work: lab exercises were done with pair programming, all small programming assignments could be completed with a partner, and all projects were completed in three- to four-person groups. We call this offering in 2010 the "highly-collaborative" offering. The first offering in 2008 where collaboration was less systematic will be referred to as the "less-collaborative" offering.

A direct comparison of learning outcomes between the two courses was not possible, because the two offerings did not use the same assessments. Instead we investigated our hypotheses using the relative performance of different groups across the two offerings and patterns of attrition between the two offerings.

In this study we compared the relative performance and attrition rates of three populations:

- students with and without Java experience,
- male and female students, and
- students who reported a preference for individual or collaborative work.

We found that differences in performance between students with and without Java experience were larger and were statistically significant in the less-collaborative offering. In the less-collaborative offering, students without previous Java experience received lower final grades than their peers who already knew some Java ($d=0.27$, $p<0.05$). In the highly-collaborative offering, there were no statistically significant differences in average final grades between students with and without Java experience ($d=0.12$, $p>0.1$). This suggests that prior experience with Java was less relevant to success in the highly-collaborative offering. We attribute this to the use of collaboration and believe it allowed students without Java experience to catch up with their peers with Java experience.

A smaller percentage of students dropped the highly-collaborative offering, but this drop in attrition was similar for students with and without Java experience.

Female students were overrepresented in the population of students without Java experience and on most assessments male students averaged a higher score than female students. However, there were no statistically significant differences in male and female students' final grades in either offering and differences in male and female students' scores were smaller in the highly-collaborative offering. The decrease in attrition was also most notable for female students, from 37 percent to 5 percent.

Although we expected differential benefits for students in the highly-collaborative offering who reported a preference for collaboration, we found no meaningful performance or attrition patterns for these populations between the two offerings.

ICER'12, September 9–11, 2012, Auckland, New Zealand.

2. PREVIOUS RESEARCH

There is a multitude of research investigating benefits of collaboration at both the college [4][13] and pre-college level [2]. For example, Springer, Stanne, and Donovan [13] report on a meta-analysis of studies of small-group learning in undergraduate STEM courses. They summarize studies of achievement, persistence, and attitudes with 3,417, 2,014, and 1,293 undergraduate students sampled in each. They report a positive effect of small-group learning on all three outcomes with an effect size of 0.51 on achievement, a reduction of attrition by 22%, and an effect size of 0.55 on students' attitudes. They report from their meta-analysis of these studies that "*all* average effect sizes are positive and only two… are not statistically significant."

This meta-analysis compared treatments with and without small-group learning, but both treatments in the current study include small-group learning opportunities. Springer *et al.* [13] call for development of a more unified theoretical base to explain the effects of small-group learning including studies that, like this one, compare various levels of small-group learning.

A collaborative environment may also support a less competitive or defensive community [5]. This may be particularly advantageous for female students, who research has found are less likely to prefer competitive environments [9]. Werner, Hanks, and McDowell [14] also discuss some of the benefits of pair programming for female students in dispelling beliefs that computer science is not collaborative and that it is socially isolating [11].

Pair programming has become a popular collaboration technique for research in computer science education and many studies have shown positive results (*e.g.*, [1][7][8][14]). Pair programming is the technique whereby two students work together, using a single computer. Typically the students take turns using the keyboard and mouse, but work collaboratively to create and debug programs.

It is hypothesized that the use of pair programming increases persistence among students who would have otherwise dropped the course, possibly through creating feelings of mutual dependence among collaborators [15]. The authors of [8] claim that a lack of significance between a pair-programming treatment and non-pair-programming control may be because pair programming allowed students to complete the course who would have otherwise dropped the course. They assume that these students contribute generally lower outcomes, lowering the average of the pair-programming condition.

A collaborative environment may also support a less competitive or defensive community [5]. This may be particularly advantageous for female students, who research has found are less likely to prefer competitive environments [9]. Werner, Hanks, and McDowell [14] also discuss some of the benefits of pair programming for female students in dispelling beliefs (described in [11]) that computer science is not collaborative and that it is socially isolating.

3. RESEARCH CONTEXT

This study considered two offerings of CS2 at U.C. Berkeley, one in the spring of 2008 and a second in the spring of 2010. The course is the second in the sequence for computer science majors at U. C. Berkeley and introduces Java, data structures, and programming methodology. Both offerings were taught by the same instructor, covered the same content, and used similar assessments; however, the second offering was modified to include additional opportunities for collaboration. The following sections describe the course format, the intervention, and differences that were independent of the intervention.

3.1 Course format: lab-centric instruction

Both offerings implemented an instructional approach called lab-centric instruction [10]. This approach trades lecture and discussion time for supervised closed-lab time. The format used in these offerings included one hour of lecture and six hours of scheduled lab sessions per week (two sessions of three hours each). Seven of the first eight lab sessions focused on Java constructs and programming techniques normally covered in an objects-early Java-based CS 1. Both offerings included three exams, three projects, three surveys, and numerous homework assignments and quizzes.

In lab-centric instruction, all lab activities are delivered online. They include conventional programming activities (*e.g.*, invention, modification, debugging, and testing) as well as collaborative activities and numerous embedded assessments. The lab instructor can monitor students' work as they proceed through the online activities, and can intervene immediately with targeted tutoring if a student is confused.

The lab-centric approach facilitates on- and off-line collaboration [10]. However, in the first offering considered in this research, off-line collaboration was not systematically encouraged.

3.2 Intervention: Increasing collaboration

The second offering of the course was intended to systematically increase opportunities for collaboration. Below, we discuss the elements of collaboration that were added in this highly-collaborative offering.

3.2.1.1 Group projects

There were three projects in both offerings of the course. In the less-collaborative offering, the first two projects were done individually and the third in a partnership of two. In contrast, students in the highly-collaborative offering worked in a static three- to four-person group on all projects.

In the highly-collaborative offering, project groups were formed in each lab section after six weeks of instruction. The teaching assistant (TA) was responsible for forming the groups, with the primary goal of creating heterogeneous groups, based upon scores on the first exam, and a secondary goal to avoid, when possible, isolating female students in a group with three male students. (These goals were recommended by [4] but other research [1] found that students from the lowest quartile were most successful if they pair-programmed with another student from the lowest quartile.)

TAs were charged with monitoring group dynamics. TAs had the opportunity to monitor group processing through observing group interactions during lab and meetings. The TAs met with each group before each project to discuss the team's technical and collaboration plans, which were determined by members of the group.

In the highly-collaborative offering, all group members confidentially rated their fellow group members' contribution to project. On projects 2 and 3, individuals that were identified by their teammates and TA as not contributing to the project received only a fraction of the team's project score. Similarly, a few

individuals that contributed far beyond their share of the project received additional points above the score earned by their team's project. Ratings were combined into a percentage that was multiplied by the points for code and write-up to get the project grade, which are summarized in Table 1.

Table 1. Range of project score multipliers in the highly-collaborative offering determined by teammate and TA ratings

	Min %	*Max %*	*Median %*
Project 2	19%	110%	101%
Project 3	12%	110%	100.5%

The sum of all project scores, including these adjustments, is used as an outcome in the study and accounts for 20 percent of each student's final grade.

3.2.1.2 Pair programming

In lab-centric courses, students engage in on- and off-line whole class discussions or small group discussions with the TA. Although collaboration is a primary component of the lab-centric pedagogy [10], in the less-collaborative offering collaboration was typically *ad hoc*. The online lab curriculum frequently encouraged students to work with a partner, but it was relatively infrequent for students to engage in pair programming in the less-collaborative offering.

In the highly-collaborative offering, pair programming received significantly more emphasis. TAs were encouraged to help all students find a partner with whom to pair-program. After project groups were formed, students were encouraged to pair-program during lab with another member of their project group. If no other members of one's project group were present, a student could partner with another classmate or work individually.

3.2.1.3 Collaborative and individual assignments

In both offerings of CS2, lab exercises were not graded, but homework assignments were graded and accounted for 10 percent of a student's final grade. Homework assignments were completed individually in the less-collaborative offering, but could optionally be submitted with a partner in the highly-collaborative offering. Students submitting a homework assignment in partnership submitted only a single copy of the assignment and both partners received the same grade.

3.3 Differences independent of intervention

There were additional differences between the offerings that did not relate directly to collaboration.

The lab-centric curriculum evolves with each offering of the course. Between the less-collaborative offering and the highly-collaborative offering, changes were made to the curriculum by a second instructor. Additional modifications were made throughout the highly-collaborative offering. All modifications were intended to improve the clarity and quality of the lab curriculum and were not intended to modify the content of the course. We do not expect that these improvements to the curriculum differentially affect different populations in the course and as we are not directly comparing these two offerings this does not limit the findings of our study.

A characteristic of lab-centric instruction is that the majority of learning takes place in the weekly six hours of lab. Therefore TAs had the most direct contact and possibly the most influence on student learning. Both offerings were taught by the same instructor, but none of the graduate and undergraduate teaching assistants (TAs) who led the lab sections were the same in the two offerings. Thus it was not possible to control for TAs between course offerings.

While the content covered in the course was the same between the two offerings, the individual assessment items differed to varying degrees. There were three exams in both offerings that were worth 12, 18, and 30 percent of students' grade. The exams given in each offering covered the same topics but had no overlapping questions. These exams were all completed individually.

In both offerings, individual quizzes were given in lab and were worth 10 percent of students' grade. To discourage academic dishonesty, students received all of the points for the quiz portion of their grade if they averaged 70 percent across these quizzes. Most students received full credit for the quiz portion of their grade and it is therefore not used as an outcome in our study.

There were three multi-week programming assignments (projects) in each offering. However, only the third project, the "Sliding Blocks Puzzle" from the Nifty Assignments archive [12], was used in both offerings. The first two projects in the less-collaborative offering (a number-line interval data type; simulation of a UNIX directory), where students worked individually, each required approximately 800 lines of code. In the highly-collaborative offering, where students worked in groups of 3 or 4, the first project (a variant of the "DNA Splicing" project in the Nifty Assignments archive [12]) required 350 lines of code and the second project (a proof checker for propositional logic) required 1700 lines of code.

4. METHODS

4.1 Outcome measurements

As outcome measurements we use students' scores on the three exams (exam 1, exam 2, and final), students' total adjusted scores on the projects, and students' course point total (grade). The exams between the two offerings were similar and were designed by an experienced instructor to test the same content.

There were no common assessments that could be used for direct measurements of differences in learning between the courses. Therefore we cannot simply compare students' performance on the exams between the two offerings. However, even if this were possible, other uncontrolled factors such as the refinement of the curriculum and the content of the projects may have affected total learning gains.

4.2 Survey data collection

The first lab's curriculum included a demographic survey. This survey was not anonymous and all questions on this survey were optional. Students' responses to the following three survey questions are featured in the analysis.

- "What is your gender?" Radio-button options: "male," "female," "I decline to state."
- "Do you prefer to work on large projects in a group or alone?" Radio-button options: "I prefer to work in a group," "I prefer to work alone." "I don't know/can't say."
- "In which of the following have you written programs?" Checkbox options: "C," "C++," "Java."

Some students did not answer all of the questions on the survey. The analysis of each research question includes the students with complete data related to that question.

4.3 Effect size

We calculated *Cohen's d* for each offering across each outcome, and will refer to this value as the *effect size*. The effect size is calculated by dividing the difference in population means by the standard deviation for the full sample (*e.g.* the standard deviation for exam 1 in the spring of 2008).

Consider a fictional case where the average score on exam 1 in one offering was 80 points for students with Java experience and 70 points for students without Java experience. The difference between these means is 10 points. If the standard deviation on this exam was 20 points, the effect size would be 10/20 or 0.5. This indicates that students with Java experience scored on average 0.5 standard deviations above students without Java experience.

4.4 Attrition

To test whether or not there were significant differences in attrition patterns across the two course offerings, we compared students who took the final exam with those that took the introductory survey, but did not take the final exam.

For these analyses, attrition serves as the outcome, and we predict it across offerings for each of the three survey questions discussed in section 4.2. Separate log-linear analyses were used for each of the survey questions, forming three 3-way contingency tables. For analyses of gender, for instance, the three-way table is formed with gender, offering, and attrition.

The sample for the attrition analyses is a superset of the sample used for the analyses for performance; the previous includes only students that did not drop the course.

5. RESEARCH QUESTIONS

5.1 Java experience

The majority of students took the Scheme-based CS1 course that introduced object-oriented programming, linked structures, and recursion. It is not expected that students know Java on entry, and approximately the first quarter of the course contains an introduction to loops, conditionals, objects, and arrays in Java (the content of a typical Java-based CS1).

However, some students have had experience with Java before the course, such as from high school. Java is the programming language tested on the Advanced Placement Computer Science (AP CS) Exam and is therefore frequently used in U.S. high schools. Although course time is dedicated to teaching Java, these students who already have experience with Java have an advantage because they are not simultaneously learning Java and the data structures content.

We hypothesized that students without prior Java experience might better be able to keep up with their peers in a collaborative environment where teammates with more advanced Java knowledge could support their learning of Java.

1. Do students with prior Java experience receive higher scores on exams, projects, and total course points?
2. Are differences in scores between students with and without prior Java experience reduced in the highly-collaborative offering?
3. Is attrition reduced among students without prior Java experience in the highly-collaborative offering?

5.2 Gender

A primary source of students' experience with Java before CS2 is a class to prepare for the AP CS exam. This exam has the lowest ratio of female to male test takers of any Advanced Placement exam [3]. In 2010, only 19.2 percent of AP CS test takers were female [3]. This pattern motivates a hypothesis that female students are less likely to have Java experience and as a result will be less successful in CS2.

1. Do male students (who are more likely to have taken AP CS) receive higher scores on exams, projects, and total course points? Are differences eliminated when controlling for prior Java experience?
2. Are differences in scores between male and female students reduced in the highly-collaborative offering?
3. Are patterns of attrition for male and female students different in the highly-collaborative offering?

5.3 Preference for group or individual work

We expected that students who prefer to work alone would perform better on individual assessments than students who preferred to work in a group. The inclusion of additional collaboration would likely favor students who reported a preference for group work and we were curious if this improved their relative scores or alternatively if it increased attrition among students who reported a preference for individual work.

1. Do students who report a preference for individual work score higher on exams?
2. Does the relationship between collaboration preference and exam scores change in the highly-collaborative offering?
3. Does the relationship between collaboration preference and attrition change in the highly-collaborative offering?

6. RESULTS

6.1 Java experience

Students with Java experience outperformed students without Java experience on all outcome measures in both offerings. We calculated the difference between these populations as the average score for students with Java experience minus the average score for students without Java experience. Therefore, this difference was a positive value for each assessment in each offering. Table 2 shows the number of students in each offering with and without Java experience.

Table 2. Sample size included in the analysis of students with and without Java experience. The number in parentheses is the percentage of students in that offering that selected that they did or did not have Java experience, respectively.

	Java experience	*No Java experience*
Less-collaborative offering	69 (43.9%)	88 (56.1%)
Highly-collaborative offering	121 (54.8%)	100 (45.2%)

Table 3 shows the effect size for each assessment in each offering. For each assessment in each offering, we also conducted a linear regression to see if the difference in scores between students' with and without Java experience was statistically significant. These effect sizes are also shown in Figure 1 with ** indicating $p<0.01$ and * indicating $p<0.05$.

Table 3. Differences between students' average performance for students with and without Java experience.

	Assessment	*Effect size*	*Significance*
Less-collaborative offering	Exam 1	0.44	t=3.36, p < 0.001
	Exam 2	0.38	t=2.79, p < 0.01
	Final	0.33	t=2.58 , p < 0.05
	Projects	0.03	t=0.25
	Grade	0.27	t=2.17, p < 0.05
Highly-collaborative offering	Exam 1	0.28	t=2.74 ,p < 0.01
	Exam 2	0.14	t=1.35
	Final	0.13	t=1.36
	Projects	0.09	t=0.95
	Grade	0.12	t=1.55

The largest differences between students with and without Java experience were seen on exam 1, which emphasized loops, conditionals, objects, arrays, and linked lists in Java. In both offerings, there was a statistically significant difference between the performance of students with and without Java experience on exam 1 ($p<0.01$). However, the difference between students with and without Java experience was greater in the less-collaborative offering where students without Java experience averaged scores that were 0.44 standard deviations below their peers as compared to 0.28 standard deviations in the highly-collaborative offering.

The differences in performance between students with and without Java experience were also reduced in the highly-collaborative offering on the other individual assessments. In the less-collaborative offering, the differences in performance between students with and without Java experience were statistically significant for exam 2, final exam, and course grade. On these same assessments, the differences in the highly-collaborative offering were smaller in magnitude and were not statistically significant ($p>0.1$). In the highly-collaborative offering, the difference between students with and without Java experience also decreased between each exam.

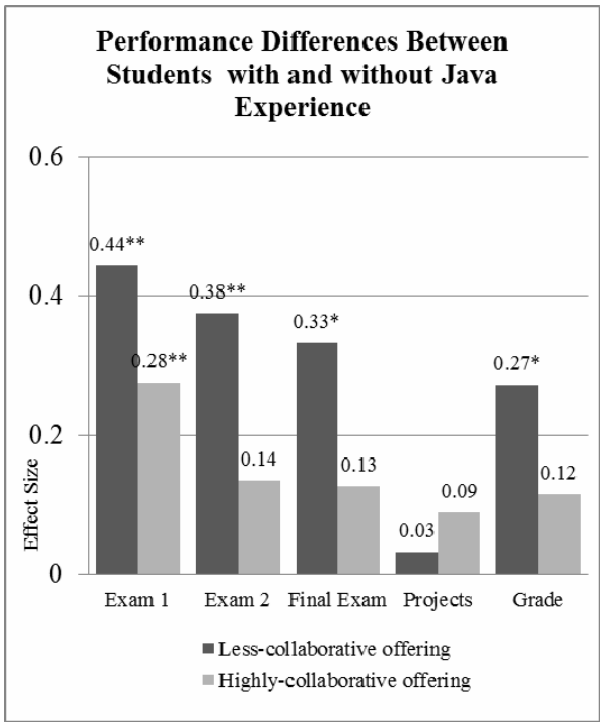

Figure 1. Effect size for difference between students with and without Java experience. ** indicates $p<0.01$ and * indicates $p<0.05$.**

On the projects, the differences between students with and without Java experience were not statistically significant in either offering.

Table 4 shows the attrition rates for students with and without Java experience in each of the two offerings. Rates of attrition were lower in the highly-collaborative offering. The overall attrition rate in the less-collaborative offering was 18 percent and it was only 11 percent in the highly-collaborative offering.

We hypothesized that there would be a greater decrease in attrition between the offerings among students without Java experience, but observed the opposite: there was a greater decrease in the attrition among students who had Java experience. The interaction between offering and Java experience was not statistically significant ($L^2=2.56$, df=1, $p>0.1$), although the main effect of Java experience on attrition approached it ($L^2=3.76$, df=1, $0.1<p<0.05$).

Table 4. Attrition rates for students with and without Java experience.

	Java Experience Attrition	*No Java Experience Attrition*
Less-collaborative offering	14/84 (17%)	20/108 (19%)
Highly-collaborative offering	8/129 (6%)	19/119 (16%)

6.2 Male and female students

Male students outperformed female students on all outcome measures except the projects outcome in the highly-collaborative offering. Across both offerings, a larger percentage of male

students began the course with experience in Java. Fifty-five percent of male students had experience with Java while 43 percent of female students had experience with Java.

Table 5 shows the number of male and female students in each offering.

Table 5. Sample size included in the analysis of male and female students in each offering. The number in parentheses is the percentage of students in that offering that answered this question and selected male or female, respectively.

	Female	*Male*
Less-collaborative offering	11 (7.5%)	135 (92.5%)
Highly-collaborative offering	38 (18%)	171 (82%)

To calculate effect size we chose to represent the difference between male and female students' performance as average male performance minus average female performance. When male students had a higher average performance this difference would be positive, while when female students had a higher average performance this difference would be negative.

Table 6 shows the effect size for each assessment in each offering. For each assessment in each offering, we also conducted a linear regression to see if the difference between male and female student scores were statistically significant. These effect sizes are also shown in Figure 2 with ** indicating $p<0.01$ and * indicating $p<0.05$.

Table 6. Differences between male and female students' performance.

	Assessment	*Effect size*	*Significance*
Less-collaborative offering	Exam 1	0.34	t=–1.29
	Exam 2	0.33	t=–1.25
	Final	0.54	t=–2.12, p<0.05
	Projects	0.30	t=–1.32
	Grade	0.39	t=–1.62
Highly-collaborative offering	Exam 1	0.28	t=–2.10, p<0.05
	Exam 2	0.17	t=–1.30
	Final	0.24	t=–1.96
	Projects	–0.13	t=1.26
	Grade	0.10	t=1.02

Figure 2 shows a graphical representation of the differences in effect size between the less-collaborative and highly-collaborative offerings of the course, which shows a smaller difference in performance between male and female students in the highly-collaborative offering.

Of the ten significance tests that compared male and female students' performance, there were only two differences that were statistically significant. Female students in the less-collaborative offering had an average final score of 0.54 standard deviations below male students ($p<0.5$), which was still statistically significant when controlling for Java experience. Female students in the highly-collaborative offering had an average exam 1 score of 0.28 standard deviations below male students ($p<0.5$), but this was not statistically significant when controlling for students' Java experience. Differences on the remaining outcomes were not statistically significant.

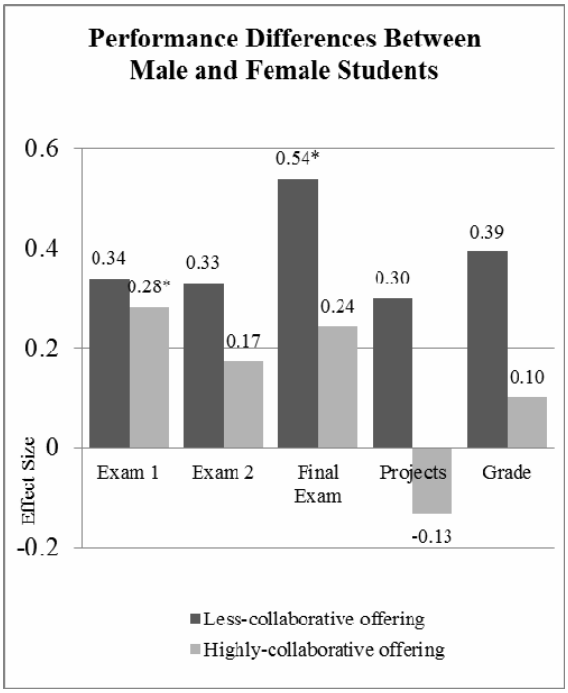

Figure 2. Effect size for difference between male and female students. ** indicates $p<0.01$ and * indicates $p<0.05$.

Table 7 shows the attrition rates for male and female students in each of the two offerings. A higher percentage of female students dropped the less-collaborative offering (37%) than dropped the highly-collaborative offering (5%). This difference approached significance, but was not statistically significant ($L^2=3.45$, df=1, $0.1>p>0.05$). A lack of significance is not surprising given the small numbers of female students dropping the course.

Table 7. Attrition rates for male and female students.

	Female attrition	*Male attrition*
Less-collaborative offering	7/19 (37%)	20/155 (13%)
Highly-collaborative offering	2/40 (5%)	12/193 (7%)

6.3 Preference for group or individual work

There was not a clear pattern of elevated performance either for the students who reported a preference for group work or for the students who reported a preference for individual work. Table 8 shows the total size of each population in each offering.

Table 8. Sample size included in the analysis of students that prefer group and individual work. The number in parentheses is the percentage of students in that offering that answered this question and selected group or individual work, respectively.

	Prefer group work	*Prefer individual work*
Less-collaborative offering	72 (60%)	48 (40%)
Highly-collaborative offering	107 (68%)	50 (32%)

The only statistically significant difference between these populations was on exam 2 in the highly-collaborative offering where students who reported a preference for individual work scored higher than students who reported a preference for group work ($d=0.37$, $p<0.05$). All other p-values were above 0.2 and given the high number of tests performed in this analysis (ten), it is not appropriate to draw conclusions from a single test that shows significance. On projects, students who preferred group work outperformed students who preferred individual work, but only in the highly-collaborative offering were all projects completed in a group.

Given the lack of statistical significance found in differences between collaboration preference and exam scores, we omit the tabular and graphical representation of these data.

Table 9 shows attrition rates for students that reported that they prefer individual versus group work. In both offerings, more students dropped the course that reported a preference for individual work. There is no significant interaction between preference and offering on attrition, and no significant main effect of preference on attrition.

Table 9. Attrition rates for students reported that they prefer individual versus group work.

	Attrition when preferring group work	*Attrition when preferring individual work*
Less-collaborative offering	10/82 (12%)	10/58 (17%)
Highly-collaborative offering	6/113 (5%)	6/56 (11%)

7. DISCUSSION

7.1 Java Experience

We expected that students that did not have Java experience before taking CS2 would be at a disadvantage as they needed to learn both Java and the data structures content. On all assessments in both offerings, students with Java experience outperformed students without Java experience.

In both offerings, the largest difference between these groups appeared on exam 1. This exam was taken by students early in the semester after five weeks and covered loops, conditionals, objects, arrays, and linked lists in Java. We expected differences in Java experience would be pronounced on exam 1.

The differences in performance on exams between students with and without Java experience were greatest in the less-collaborative offering. These differences were statistically significant in the less-collaborative offering, but in the highly-collaborative offering only the differences on exam 1 were statistically significant.

The differences in scores on projects were not statistically significant in either the less-collaborative or highly-collaborative offering. It is reasonable to assume that the projects were less difficult for students fluent in Java, but regardless of students' comfort with Java it might have been possible to successfully complete the assigned task of each project.

In the highly-collaborative offering there were lower attrition rates, but there was not a significant change in the pattern of attrition for students with and without Java experience.

These data show that the increased collaboration may have allowed students without Java experience to catch up with their peers with Java experience. While students without Java experience had not caught up with their peers before the first exam, in the highly-collaborative offering differences in their later exam scores and final grades in the class were not statistically significant ($p>0.1$).

Educational research has documented a number of benefits of collaboration (see [13] for a review). However, if the highly-collaborative offering had a differentially negative effect on students with Java experience we could have observed the same patterns described above. It is possible that students that knew Java learned less because they spent time supporting students with less knowledge of Java. Research has shown a consistently positive benefit only for low-achievers working in heterogeneous groups and not medium- or high-achievers [2], but the collaboration tasks, such as explaining technical details to peers, may improve students' technical communication skills, which is perceived by the instructor as highly valuable. We did not have common assessments across the two offerings and therefore cannot investigate the hypothesis that the highly-collaborative offering had a differentially negative effect on students with Java experience.

7.2 Gender

There were notable differences between the attrition rates for female students between the less-collaborative and highly-collaborative offerings. In the less-collaborative offering 37 percent of female students dropped the course while in the highly-collaborative offering only 5 percent of female students dropped the course. There was also a reduction in attrition among male students from 13 percent in the less-collaborative offering to seven percent in the highly-collaborative offering. Given the small number of students that dropped the class these patterns were not statistically significant, but we believe warrant future research.

Our assumption is that students who drop the course are more likely to be struggling in the course. We expected that the higher attrition among female students in the less-collaborative offering would improve the relative performance of the female students in the course. Therefore in the highly-collaborative offering, when only 5 percent of female students dropped the course, we expected that the average performance of female students would drop in comparison to their male peers. This would predict a relative drop in performance for female students in the highly-collaborative offering, which was not observed.

Male students scored higher on all outcomes excluding the projects in the highly-collaborative offering. However, only two of these ten outcomes were statistically significant and only one was statistically significant when controlling for students' Java experience. We make no claims regarding the performance of male and female students or differential effects for female and male students resulting from collaboration, but remind the reader that the highly-collaborative offering may have included female students that would otherwise have dropped the course.

7.3 Preference for group or individual work

In both offerings, students who preferred collaboration performed slightly better on projects than students who reported a preference for individual work. However, there were no other noteworthy patterns in differences between these populations and the majority of the results were not statistically significant. In future work it may be possible to better triangulate students' collaboration preferences to investigate differential benefits.

8. CONCLUSION

In this paper we compared outcomes in two offerings of CS2. The offerings were taught by the same instructor and used similar assignments, but they differed in the level of collaboration. In the first offering, collaboration was mainly *ad hoc*, while in the second, the instructor adopted course policies and teaching strategies to make collaboration more frequent.

The effect of Java experience on a student's final grade differed between these offerings: in the less-collaborative offering, students with Java experience received final grades that were 0.27 standard deviations above their peers ($p<0.05$), while in the highly-collaborative offering there was no statistically significant difference. We believe that collaboration supported students without Java experience in catching up with their peers who knew Java. This would explain the pattern observed in the highly-collaborative offering that the difference between students with and without Java experience decreased between each exam.

Female students had lower rates of attrition and higher scores in the highly-collaborative offering, but these patterns were not statistically significant. We also did not find statistically significant patterns of performance or attrition for students who reported a preference for collaborative or individual work.

Although the course does not require Java as a prerequisite, students with Java experience performed better than their peers without Java on all outcome measures. We hope that these results encourage instructors to evaluate the accessibility of their courses for students that have only the experience specified in the official prerequisites for their courses.

We hope that this study will encourage other researchers to investigate the effect of collaboration on attrition patterns and will contribute to instructors' motivation to adopt collaborative techniques in their courses. The collaboration techniques in this study are not the only options available and research has shown similar benefits for alternatives to pair programming [6].

9. ACKNOWLEDGMENTS

Work described in this paper was supported by NSF grant DUE-1044106. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation.

10. REFERENCES

[1] Braught, G., MacCormick, J., & Wahls, T. (2010). The benefits of pairing by ability. *Proceedings of the 41st ACM Technical Symposium on Computer Science Education,* 249–253.

[2] Cohen, E. (1994). Restructuring the classroom: Conditions for productive small groups. *Review of Educational Research. 64(1)*, 1.

[3] College Board (2011). Retrieved April 1, 2012, from http://www.collegeboard.org

[4] Felder, R. M. and Brent, R. (1994). Cooperative learning in technical courses: procedures, pitfalls, and payoffs. ERIC Document Reproduction Service Report ED 377038.

[5] Garvin-Doxas, K. & Barker, L. J. (2004). Communication in computer science classrooms: Understanding defensive climates as a means of creating supportive behaviors. *Journal of Educational Research in Computing, 4(1)*, 1-18.

[6] Lewis, C. M. (2011). Is pair programming more effective than other forms of collaboration for young students? *Computer Science Education. 21(2)*. 105-134

[7] McDowell, C., Werner, C., Bullock, H. E., Fernald, J. (2006). Pair programming improves student retention, confidence, and program quality. *Communications of the ACM*. 49, 90-95.

[8] Nagappan, N., Williams, L., Ferzil, M., Wiebe, E., Yang, K., Miller, C., & Balik, S. (2003) Improving the CS1 experience with pair programming. *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, 359-362.

[9] Niederle, M. & Vesterlund, L. (2007). Do women shy away from competition? Do men compete too much? *The Quarterly Journal of Economics. 122(3)*. 1067-1107.

[10] Titterton, N., Lewis, C. M., & Clancy, M. (2010). Benefits of lab-centric instruction. *Computer Science Education* (Ed. Y. Ben-David Kolikant) *20(2).* 79-102. DOI: 10.1080/08993408.2010.486256

[11] Margolis, J., & Fisher, A. (2003). *Unlocking the clubhouse: Women in computing*. Cambridge, MA: The MIT Press.

[12] Parlante, N. Nifty Assignments. http://nifty.stanford.edu/

[13] Springer, L., Stanne, M. E., & Donovan, S. S. (1999). Effects of small-group learning on undergraduates in science, mathematics, engineering, and technology: A meta-analysis. *Review of Educational Research. 69(1)*. 21-51.

[14] Werner, L. L, Hanks, B., & McDowell, C. (2005) Pair programming helps female computer science students. *Journal on Educational Resources in Computing. 4(1)* Article 3.

[15] Williams, L. & Kessler, R. R. (2000). The effects of "pair-pressure" and "pair-learning" on software engineering education, *Proceedings of the 13th Conference on Software Engineering Education & Training*, 59-65.