An Experimental Study of Cooperative Learning in CS1

Leland L. Beck
Department of Computer Science
San Diego State University
San Diego, CA 92182-7720
(619) 594-6807
beck@cs.sdsu.edu

Alexander W. Chizhik
School of Teacher Education
San Diego State University
San Diego, CA 92182-1153
(619) 594-1222
achizhik@mail.sdsu.edu

ABSTRACT

An experiment was conducted to study the effectiveness of the cooperative learning approach to teaching CS1. The cooperative learning exercises, which used specific roles to focus students' attention on key concepts, were designed so they could be used in a variety of educational settings. Experimental results show that the benefits of cooperative learning clearly outweighed any possible losses due to reduced lecture time. These benefits were enjoyed by both male and female students, and by students from a variety of majors. Majority and minority students performed at approximately the same overall level when using the cooperative learning approach. There are indications that the educational benefits continue when students who had a cooperative learning experience in CS1 go on to CS2.

Categories and Subject Descriptors

K.3.2 [**Computers & Education**]: Computer & Information Science Education – *computer science education, curriculum.*

General Terms

Experimentation, Human Factors

Keywords

Cooperative learning, Classroom management, Pedagogy, CS1.

1. INTRODUCTION

This paper reports results from an educational experiment that compared a cooperative learning approach to CS1 with a traditional lecture-based approach. The application of cooperative learning in teaching Computer Science is not a new idea. However, our approach and experimental design have some innovative features, which are described in the following sections.

Section 2 of this paper briefly reviews some of the previously published work on cooperative learning, and gives the motivation for the current study. The following sections describe our cooperative learning materials and the design and results of our experiment. Future work will include testing these cooperative learning materials in other educational settings; please contact us if you are interested in participating.

SIGCSE'08, March 12–15, 2008, Portland, Oregon, USA.

2. PREVIOUS WORK

Cooperative learning techniques have been applied with a wide variety of subject matter and a broad spectrum of populations. Good discussions of cooperative learning methods and research can be found in [5, 13, 14]. Accumulated evidence suggests that cooperative learning results in higher student achievement, more positive attitudes toward the subject, improved student retention, and a variety of other benefits [7, 13, 14]. It appears that the cooperative learning approach may be especially beneficial for women and members of under-represented minority groups [10, 14, 17, 20].

A number of educators have studied the use of cooperative learning techniques in Computer Science courses [2, 3, 4, 8, 12, 15, 16, 19]. Anecdotal evidence indicates that instructors and students value cooperative learning experiences, and are pleased with the results. However, it has often been difficult to perform objective evaluations of the outcomes. Some of the difficulties arise from small course sizes, differences in student and instructor preparation, and possible changes in course content and standards over a period of years. (See, for example, [2] and [16].)

Many previous efforts to apply cooperative learning in Computer Science courses involved dividing students into small groups to work on programming problems or laboratory exercises. Instructors using this approach expect that students will learn from each other as they work cooperatively on an assignment. However, the problems or exercises being used can often be completed by individual students, without actually working together as a group in a structured way. A similar observation can be made concerning pair programming [9, 18], in which roles are generic rather than being related to the specific problem to be solved. Most theories of cooperative learning hold that the learning process is improved when cooperative learning tasks are designed to involve an interdependent group of students, with a structured division of group responsibilities into roles [6, 13, 14].

In our previous work [1], we designed exercises that assigned specific roles to group members. We conducted an experiment in which students met in a traditional lecture setting for two class periods per week. During the third class meeting, students in the experimental group did cooperative learning exercises while students in the control group did whole-class discussion and question/answer activities. The results of this experiment indicated that there were substantial educational benefits for the students who used the cooperative learning approach. There were also indications that cooperative learning was especially beneficial for women and non-white students.

However, our previous project had a number of limitations. Students in the experimental and control groups spent the same amount of time in lectures. In a more realistic situation, spending class time on cooperative learning activities would require spending less time in lectures. It is important to examine whether or not this trade-off of class time is beneficial. In addition, the cooperative learning activities in our previous experiment were led by the principal investigator for the project. This raises the possibility of instructor-related effects, as well as the possibility that students might have been responding to receiving special attention. Finally, the exercises were designed to be used in one specific way (a separate class meeting of one hour per week), with the course material being presented in one specific sequence. Because of this relatively rigid structure, it might have been difficult to implement the exercises in a different type of course.

The cooperative learning exercises and educational experiment described in this paper were designed to address some of these limitations in our previous work.

3. DESIGN OF THE EXERCISES

Our cooperative learning exercises are similar in some ways to those that were used in our previous study [1]. For example, in many exercises specific roles are assigned to students in order to focus attention on the most important concepts being studied. However, the set of exercises has been expanded and revised so that they can be used in a variety of different sequences and educational settings. The preliminary exercises provide an introduction to the most fundamental concepts of programming. Another group of exercises focuses on the syntax and semantics of statements in the Java language. There are also exercises that introduce important approaches to problem solving, such as selecting appropriate data and control structures, designing and implementing methods and classes, and using pseudocode. Testing and debugging are significant parts of almost every exercise. However, there is also a separate group of exercises that focus on techniques such as separate testing of methods and generation of test cases based solely on the problem specification.

Another significant modification is the use of two recurring problem contexts throughout the set of exercises: the "Karel the Robot" world [11] and a ticket-selling application. These problem contexts are used to introduce most of the fundamental concepts to be studied, and to allow students to practice using newly-learned concepts in a familiar setting before they go on to apply them in other environments.

For example, the first exercise presents a problem in the Robot world, and a program to solve that problem. (Because students do not know any programming languages yet, the program is expressed in a simple graphical form.) Students work cooperatively to execute this program. As they do this, one person simulates the operation of the robot, receiving messages that command the robot to move or ask questions about its environment. Other students read various parts of the program, one step at a time, passing control to each other and sending messages to the student who is simulating the robot. After students have successfully executed the program and understand how it works, they are shown the same program in Java syntax. Even though they have not yet learned to write Java, they find that they can read and make sense of the Java statements.

The program in this exercise works correctly when students execute it with the starting situation that is specified. However, there are other situations in which the program does not work properly. Students are challenged to work cooperatively to find these situations, to think about the process they used to find them, and to suggest changes to fix the errors in the program. During this phase of the exercise, most groups have insightful discussions about strategies for looking for errors.

This first exercise introduces a variety of central topics that students will revisit later in the course: step-by-step program execution, decision making, loops, objects, methods, and debugging — all during the first week of class. Even more important, the students gain experience working together to solve a non-trivial problem that requires creative thought.

In later exercises, students return to the simulated Robot world to practice using the new Java concepts they have learned, before going on to apply them in less familiar settings. The ticket-selling problem context is used in a similar way. (Many students already have some experience using online ticketing systems, and enjoy getting some insight into how such systems might work.)

4. METHODOLOGY

Our educational experiment involved two different sections of our CS1 course in the Spring Semester, 2007. In the experimental section, students worked on cooperative learning exercises for 30 to 45 minutes during almost every 75-minute class meeting; the remainder of class time consisted of mini-lectures, administrative topics, and general class discussions. The comparison section was taught in a traditional lecture-based format, with some time spent on questions and general class discussions. The instructor for the comparison section had regularly taught CS1 in this lecture-based format for the past nine years. The instructor for the experimental section had taught CS1 in a lecture-based format for two semesters; she also taught using a similar version of the cooperative learning materials during one previous semester.

Students were not randomly assigned to the experimental and comparison sections; they enrolled for the section that best fit their schedules. However, the two sections were similar on general measures of academic performance such as GPA and SAT scores. Table 1 shows the distribution of students in the sections

Table 1. Distribution of students by major.

Category	Majors included	Number of students enrolled: Experimental	Comparison
C	Computer Science, Computer Engineering	9	8
E	Other engineering	5	2
M	Mathematics, Statistics	7	7
S	Other laboratory sciences	1	3
B	Business Administration	3	5
G	Geography	1	7
O	Other	5	4
U	Undeclared	3	1
	Total	34	37

by major; the categories indicated will be used in the discussion of results later in this paper. (Most of the Geography majors were students who were enrolled in a certificate program in Geographic Information Sciences.) Similarly, Table 2 shows the distributions of students by gender and by ethnicity (as reported by the students during registration).

Table 2. Distribution of students by gender and ethnicity.

		Number of students enrolled			
		Experimental		Comparison	
Category	Ethnicities included	Male	Female	Male	Female
B	Black, African American	0	0	1	3
F	Filipino	3	1	2	1
M	Mexican American, Mexican, Chicano	1	2	4	1
W	White	11	7	14	2
O	Other	3	2	2	3
X	No response or declined to state	4	0	3	1
	Total	22	12	26	11

The experimental and comparison sections met at different times, and had different (although similar) lab assignments and midterm exams. Our analysis in this experiment focused on the final exam for the course, which was taken by all students from both sections at the same time. The instructors for the two sections collaborated in writing and refining the questions on this final exam. Care was taken to ensure that the exam did not give students from either section any special advantage. For example, none of the exam questions were similar to problems that had been discussed in one section but not in the other. None of the questions referred to the standard problem contexts that were used in the cooperative learning exercises. The students from both sections received the same sample exams and review materials in preparation for the final exam. In order to avoid any possible unconscious bias in grading, we obscured the students' names and mixed the exam papers from the two sections together before scoring them.

The final exam consisted of 8 multiple-choice questions and 10 free-response questions, which are described in more detail in the following section of this paper. In order to improve uniformity of scoring, we developed a rubric for assigning credit to answers on the free-response questions on a scale of 0 to 9 points. Scores on the rubric were defined according to key elements of each question — for example, whether an appropriate control structure was selected or whether a method header was correctly constructed.

5. EXPERIMENTAL RESULTS

Figure 1 shows the distribution of final exam scores in the experimental and comparison sections. Each multiple choice question was worth one point for a correct answer, and each free-response question was worth from 0 to 9 points, using the rubric previously described. Thus the maximum possible score was 107 points. The mean exam score was 58.9 for the comparison section and 73.5 for the experimental section. This difference in means is statistically significant ($t = -3.75$, $p < .01$).

There were three students in each section who did not take the final exam; one other student from the experimental section is not represented in Figure 1 because a defective exam paper prevented meaningful scoring on one of the questions.

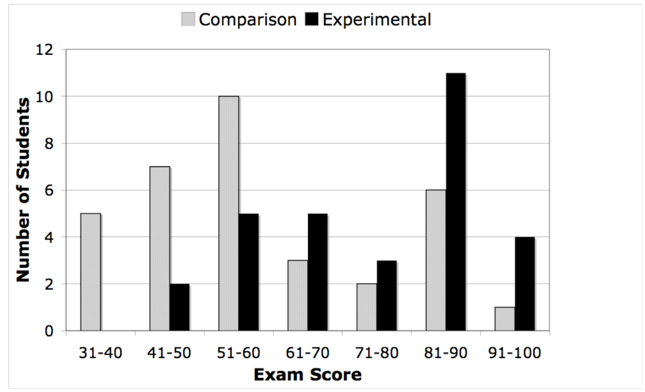

Figure 1. Distribution of final exam scores.

In an effort to better understand the differences in total score, we examined the students' scores on individual exam questions. For analysis purposes, we divided the free-response questions into three categories, as shown in Table 3. Code Tracing questions asked the student to show the results that would be produced by executing a given Java code segment or program. Code Writing questions asked students to write Java code according to a given specification, to fill in blanks to complete partially written code, or to modify given code to perform a different function. There was also one Test Planning question, which asked students to devise a set of input values that would thoroughly test a given piece of code relative to its specifications. We used the Mann-Whitney test for statistical significance, because the distribution of scores on many of the individual questions did not satisfy the assumptions for using a t-test.

In general, students from the two sections performed about equally well on the simpler types of questions (Code Tracing and Test Planning). However, students from the experimental section

Table 3. Analysis of scores by type of question.

Question		Mean score for section		Mann-Whitney	
Type	Number	Comparison	Experimental	U	p
Multiple Choice	1–8 (total)	5.5	6.5	734.0	.003
Code Tracing	9	7.5	8.2	601.5	.17
	10	8.1	8.8	587.5	.21
	12	7.7	7.3	580.0	.25
	17	4.1	6.0	720.5	.005
Test Planning	11	6.9	6.5	585.0	.23
Code Writing	13	3.7	6.0	800.5	.0001
	14	3.1	5.5	771.0	.0006
	15	3.5	5.8	757.0	.001
	16	4.4	6.5	697.5	.01
	18	4.3	5.7	629.0	.06

performed better than the comparison group on the most difficult of these questions (Question 17), which involved tracing the execution of a main program and a relatively complex invoked method. The experimental section also did significantly better on the Code Writing questions. The single exception was Question 18, for which the difference between the two sections was not statistically significant. This may have been because Question 18 was quite similar to a question on the practice exam, and was a member of a class of problems that students had practiced extensively during the review sessions.

As Figure 2 shows, students from almost all majors appear to have benefited from the cooperative learning approach. The greatest benefits seem to have been for students majoring in Computer Science, Engineering, and Mathematics (please see Table 1 for definitions of the categories). The differences between experimental and comparison sections are not statistically significant for the individual categories shown, because of the small numbers involved. However, the difference is significant for major categories C, E, and M combined ($t = -3.31$, $p < .01$).

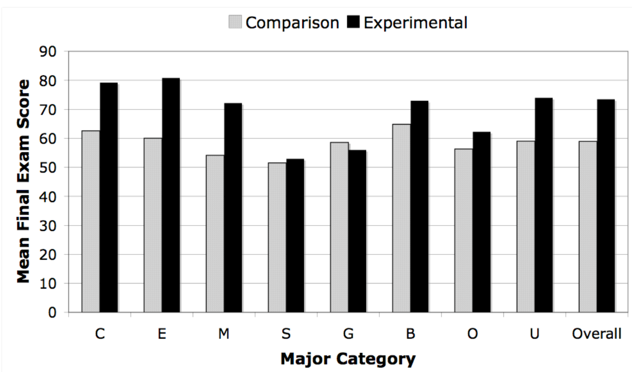

Figure 2. Final exam scores by major category (see Table 1).

Both male and female students achieved higher mean exam scores using the cooperative learning approach (Figure 3). The differences between experimental and comparison sections are statistically significant ($t = -2.65$, $p = .01$ for males and $t = -3.12$, $p < .01$ for females).

As Figure 4 shows, students from all ethnicities achieved higher mean scores in the experimental section. (Please see Table 2 for the definition of the ethnicity categories.) The difference between the experimental and comparison sections approaches statistical significance for white students ($t = -1.90$, $p = .07$). The difference

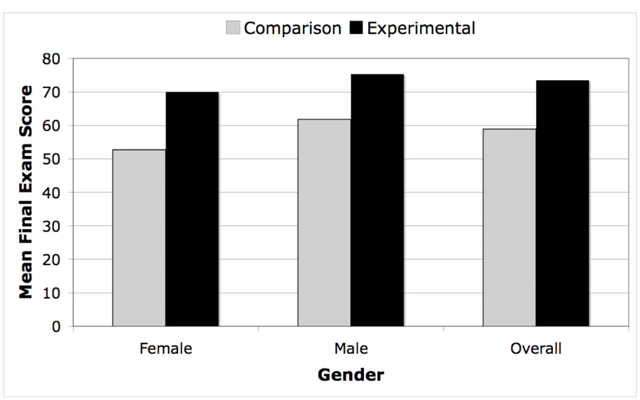

Figure 3. Final exam scores by gender.

is not significant for the other categories (some of which contain very small numbers of students); however, the consistent improvement across categories is worth noting. The difference between sections is significant for all non-white students combined (category NW: $t = -2.85$, $p < .01$).

As the last three columns in Figure 4 show, white and non-white students in the experimental section achieved approximately the same average exam scores. The difference between white and non-white students approaches statistical significance for the comparison section ($t = -2.00$, $p = .05$), but is not significant for the experimental section ($t = -0.28$). This must be interpreted with caution, because the overall ethnic composition of the two sections was different and because an ethnicity by section ANOVA did not detect significantly different benefits for white and non-white students. However, it is an interesting and potentially important result that deserves further investigation.

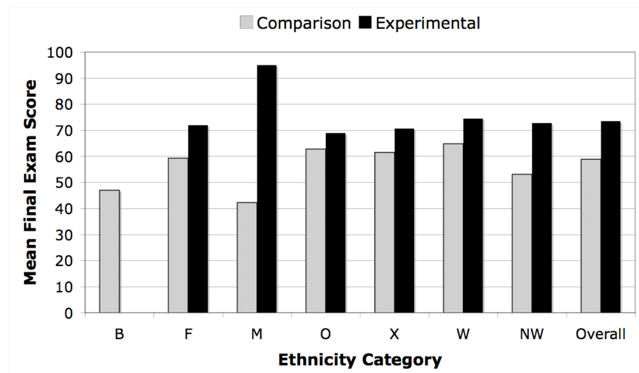

Figure 4. Final exam scores by ethnicity category (see Table 2).

One critical question is whether or not students who use the cooperative learning approach in CS1 continue to perform at a higher level in subsequent courses. Preliminary results suggest that the benefits from cooperative learning do continue, at least in CS2. Table 4 shows the mean scores on CS2 midterm exams in Spring 2007 for students who took cooperative-learning and lecture-based versions of CS1 in Fall 2006. One reason for the difference in means is that there were very few low exam scores in the CS1 cooperative-learning group. For example, all 12 students from this group scored 90 or above on Midterm 2. In contrast, 11 students from the lecture-based group scored below 90, including 4 scores that were below 70. This difference is statistically significant ($\chi^2 = 7.14$, $p < .01$). In addition, all of the students in this CS2 course who had taken the cooperative-learning version of CS1 passed CS2; in contrast, 3 of the 29 students who had taken the lecture-based CS1 did not pass CS2.

Table 4. CS2 exam scores by type of CS1 course.

Fall 2006 CS1 course type	Number of students	Mean score on CS2 exams Midterm 1	Midterm 2
Lecture-based	29	73.1	86.0
Cooperative-learning	12	79.1	95.2

6. CONCLUSIONS AND FUTURE WORK

In our previous study [1], students met in a traditional lecture setting for two class periods per week. During the third class meeting, students in the experimental group did cooperative learning exercises while students in the control group did whole-class discussion and question/answer activities. Thus both groups of students received the same amount of lecture time during the course. The students who did the cooperative learning exercises scored, on average, 13% higher on the final exam than did the students from the control group.

In the experiment reported in this paper, the experimental section spent much less time than the comparison group in lectures. Cooperative learning exercises were the main focus of the experimental section. The benefits for the cooperative learning students were clearly greater using this approach: their average final exam score was 25% higher than the average for the comparison section. This strongly indicates that the cooperative learning approach more than made up for any losses that might have been due to the reduction in lecture time.

Our plans for the future include replication of the experiment with another group of students, and a more detailed analysis of results on exam questions. We plan to use a scoring rubric that measures important factors in the solution process — for example, whether students recognize key steps that are required to solve a problem. We will continue to track students' performance in CS2 and later courses, examining their scores on various types of exam questions and programming assignments in order to see whether there are continuing benefits from the cooperative learning experience in CS1.

The cooperative learning sections in our two experiments were taught by different instructors. This suggests that the improved exam performance of the cooperative learning students was due to the approach itself, not to the instructor. However, we note that the type of CS1 course and the underlying student population was the same in both of our experiments. It is very important to validate these results by studying the cooperative learning approach and materials when they are used by instructors with different populations of students, and in different educational environments. These studies should include larger numbers of women and minority students, in order to further investigate our indications of special benefits to students in these categories.

We are now seeking faculty members who would like to take part in this follow-up project. If you are interested in participating, please contact the first author of this paper.

7. ACKNOWLEDGMENTS

We gratefully acknowledge the support of the National Science Foundation (CCLI program) under grant DUE-0442121.

8. REFERENCES

[1] Beck, L. L., Chizhik, A. W., and McElroy, A. C., "Cooperative Learning Techniques in CS1: Design and Experimental Evaluation," *Proceedings, 2005 SIGCSE Technical Symposium on Computer Science Education,* pp. 470–474.

[2] Chase, J. and Okie, E., "Combining Cooperative Learning and Peer Instruction in Introductory Computer Science," *Proceedings, 2000 SIGCSE Technical Symposium on Computer Science Education,* pp. 372–376.

[3] Finkel, D. and Wills, C. E., "Computer Supported Peer Learning in an Introductory Computer Science Course," *SIGCSE Bulletin,* Special Issue, 1996, pp. 55–56.

[4] Gonzalez, G., "A Systematic Approach to Active and Cooperative Learning in CS1 and Its Effects on CS2," *Proceedings, 2006 SIGCSE Technical Symposium on Computer Science Education,* pp. 133–137.

[5] Johnson, D. W. and Johnson, F. P., *Joining Together: Group Theory and Group Skills,* Prentice-Hall, 1975.

[6] Johnson, D. W. and Johnson, R. T., *Learning Together and Alone* (4th edition), Allyn and Bacon, 1994.

[7] Johnson, D. W., Johnson, R. T., and Smith, K. A., *Active Learning: Cooperation in the College Classroom,* Interaction Book Company, 1991.

[8] Keeler, C. and Anson, R., "An Assessment of Cooperative Learning Used for Basic Computer Skills Instruction in the College Classroom," *Journal of Educational Computing Research,* 1995, pp. 379–393.

[9] McDowell, C. and Werner, L., "The Effects of Pair-Programming on Performance in an Introductory Programming Course," *Proceedings, 2002 SIGCSE Technical Symposium on Computer Science Education,* pp. 38–42.

[10] Nelson, C. E., "Student Diversity Requires Different Approaches to College Teaching, Even in Math and Science," *American Behavioral Scientist,* 1996, pp. 165–175.

[11] Pattis, R. E., *Karel The Robot: A Gentle Introduction to the Art of Programming* (2nd ed.), John Wiley & Sons, 1994.

[12] Priebe, R., "The Effects of Cooperative Learning in a Second-Semester University Computer Science Course," National Association for Research in Science Teaching, March 1997 (available as ERIC document ED406189).

[13] Sharan, S., *Handbook of Cooperative Learning Methods,* Greenwood Press, 1994.

[14] Slavin, R. E., *Cooperative Learning: Theory, Research, and Practice* (2nd edition), Prentice Hall, 1995.

[15] Troeger, D., "Formal Methods, Design, and Collaborative Learning in the First Computer Science Course," *New Directions for Teaching and Learning,* 1995, pp. 55–66.

[16] Walker, H. M., "Collaborative Learning: A Case Study for CS1 at Grinnell College and UT–Austin," *Proceedings, 1997 SIGCSE Technical Symposium on Computer Science Education,* pp. 209–213.

[17] Williams, L., Layman, L., Slaten, K., Berenson, S., and Seaman, C., "On the Impact of a Collaborative Pedagogy on African American Millennial Students in Software Engineering, Proceedings of the 29th International Conference on Software Engineering, 2007, pp. 677-687.

[18] Williams, L., Wiebe, E., Yang, K., Ferzli, M. and Miller, C., "In Support of Pair Programming in the Introductory Computer Science Course," *Computer Science Education,* September 2002, pp. 197–212.

[19] Willis, C. E., Finkel, D., Gennert, M. A., and Ward, M. O., "Peer Learning in an Introductory Computer Science Course," *Proceedings, 1994 SIGCSE Technical Symposium on Computer Science Education,* pp. 309–313.

[20] Yerion, K. A. and Rinehart, J. A., "Guidelines for Collaborative Learning in Computer Science," *SIGCSE Bulletin,* December 1995, pp. 29–34.