Does Studio-Based Instruction Work in CS 1? 
An Empirical Comparison with a Traditional Approach

Christopher Hundhausen*, Anukrati Agrawal*, Dana Fairbrother†, and Michael Trevisan†

*School of Electrical Engineering and Computer Science
†College of Education
Washington State University
Pullman, WA 99164
+1 509-335-4590

hundhaus@wsu.edu, aagrawal@eecs.wsu.edu, danafairbrother@wsu.edu, trevisan@wsu.edu

ABSTRACT
Given the increasing importance of communication, teamwork, and critical thinking skills in the computing profession, we believe there is good reason to provide students with increased opportunities to learn and practice those skills in undergraduate computing courses. Toward that end, we have been exploring studio-based instructional methods, which have been successfully employed in architecture and fine arts education for over a century. We have developed an adaptation of studio-based instruction for computing education called the pedagogical code review, which is modeled after the code inspection process used in the software industry. To evaluate its effectiveness, we carried out a quasi-experimental comparison of a “studio-based” CS 1 course with pedagogical code reviews and an identical “traditional” CS 1 course without pedagogical code reviews. We found no learning outcome differences between the two courses; however, we did observe two interesting attitudinal trends: (a) self-efficacy decreased more in the traditional course than in the studio-based course; and (b) peer learning decreased in the traditional course, but increased in the studio-based course. Additional questionnaire and interview data provide further evidence of the positive impact of studio-based instruction.

Categories and Subject Descriptors
K.3.2 [Computer and Information Science Education]: Computer science education, Curriculum.

General Terms
Design, Experimentation, Human Factors.

Keywords
Studio-based learning and instruction, pedagogical code review, code inspection, peer review, CS1

1. INTRODUCTION
Computer science instruction has traditionally emphasized individual problem solving and programming skills, especially at the introductory level. We believe that this traditional approach is deficient, because it may not adequately prepare students for jobs in the computing profession, which are increasingly requiring communication, collaboration and critical thinking skills in addition to programming skills.

To address this deficiency, we have been exploring studio-based instruction as a means of providing students with increased opportunities to learn communication, collaboration, and critical thinking skills in core computing courses [9]. Adapted from architectural and fine arts education, our general studio-based approach includes four key steps:

1. Students are given complex and meaningful problems for which they have to construct computational solutions.
2. Students present their solutions and justifications to the entire class for discussion and feedback.
3. Their peers critique the solutions and provide comments.
4. Students are given the opportunity to respond to comments and criticisms, and modify their solutions appropriately.

How might this approach be adapted for lower-division computing courses, which traditionally revolve around a series of individual programming assignments? Based upon the formal code inspection process used in industry (see, e.g., [7]), we have developed the pedagogical code review (PCR) [8], in which students first review each other’s code solutions individually, and then come together in teams to identify, discuss and log issues with the code (defects and improvements). Students are then given the opportunity to resubmit their code solutions based on the feedback they receive in the reviews. In addition to including the four key steps of our general studio-based approach, the PCR process provides students with opportunities to develop a variety of skills that are important in the computing profession, including computer programming, communication, collaboration, and critical thinking.

In a previous empirical study [8], we gathered preliminary evidence that PCRs can positively influence the quality of students’ code solutions, engage them in meaningful discussions, and promote a sense of community. In this paper, we present a more rigorous empirical evaluation of the educational effectiveness of PCRs. Our quasi-experimental study compared student learning outcomes and attitudes in two CS 1 courses: a “studio-based” course that implemented PCRs, and an identical “traditional” course without PCRs. While we found no learning outcome differences between the two courses, we did observe two interesting trends in our attitudinal data: (a) students’ self-efficacy decreased more in the traditional course than in the studio-based course; and (b) students’ peer learning decreased in the traditional course, but increased in the
studio-based course. We speculate why this might have been the case, and identify directions for future research into studio-based instruction for computing education.

2. BACKGROUND AND RELATED WORK

Studio-based instructional approaches evolved out of the master-apprentice educational system of the Middle Ages [10]. Architecture and fine arts educators adapted this approach in the form of the design studio: a shared space where students work iteratively on assigned design projects, periodically presenting their work to their peers and instructor for critical review and feedback in so-called design crits (design critiques).

Studio-based approaches have been successfully integrated into science and math instruction at both the high school [4] and undergraduate (e.g., [11]) levels. Several computing educators have explored the use of studio-based learning in their courses (see, e.g., [14,17]). There are even entire computing degree programs built upon the studio model (see, e.g., [3,12]).

In a similar vein, as part of a multi-institutional research project to adapt studio-based instruction for core computing courses [9], we have developed the pedagogical code review (PCR) [8], a form of studio-based instruction rooted in peer reviews of students’ code solutions. Recognizing the pedagogical value of peer reviews in promoting critical thinking, collaboration, and teamwork skills (e.g., [1,16,19]), several computing educators have explored the use of peer reviews in both introductory (e.g., [19]) and upper-division (e.g., [1]) computing courses. In addition, several web-based systems have been developed to support asynchronous online peer reviews (e.g., [5,16,18]).

While several computing educators have gathered anecdotal evidence of the effectiveness of their studio-based approaches, few have evaluated their approaches empirically. Moreover, only one study (to our knowledge) actually compares the learning outcomes of students who participate in studio-based learning against students who do not [16]. The study presented here contributes to this line of research by providing a rigorous quasi-experimental comparison of a studio-based and a traditional course based on both learning and attitudinal outcomes.

3. QUASI-EXPERIMENTAL STUDY

We hypothesized that the process of participating in PCRs would be educationally beneficial in two respects:

\[ H1: \text{Students who participate in PCRs will learn computer programming significantly better than students who do not.} \]

\[ H2: \text{Students who participate in PCRs will experience positive shifts in their attitudes toward learning that are significantly higher than those who do not.} \]

To test these hypotheses, we conducted a between-subjects quasi-experimental study with two treatments: Traditional and Studio. In the Traditional treatment, which corresponded with the Fall 2008 offering of the CS 1 course at Washington State University, students participated in a traditional CS 1 curriculum that focused on the C programming language. The course included a weekly 170-minute lab in which students worked together on programming problems under a teaching assistant’s supervision. In contrast, in the Studio treatment, which corresponded with the Spring 2009 offering of the same CS 1 course at Washington State University, students participated in PCRs instead of regular lab activities for three of the labs: those that took place in weeks 8, 11, and 13 of the fifteen-week semester.

The courses that defined each treatment were identical in nearly every respect except for the presence or absence of PCRs. The same instructor (not involved in this research) taught both courses. Moreover, the same textbook, lecture slides, assignments, and labs were used in both courses. Only the weekly quizzes, along with two midterm exams and one final exam, differed slightly—the instructor's modest attempt to make cheating more difficult.

Participants’ learning outcomes were measured using pre-test to post-test improvement on a test of target programming knowledge. Participants' attitudinal changes were measured using pre-survey to post-survey differences. In addition, we gathered qualitative data on student attitudes by conducting exit interviews with a small sample of students in each treatment, and we gathered qualitative data on students’ perceptions of the PCRs in the Studio treatment by administering exit questionnaires at the end of each of the three PCR sessions.

3.1 Participants

The fall, 2008 offering of the CS 1 course at Washington State University (the Traditional treatment) enrolled 89 students, whereas the spring, 2009 offering (the Studio treatment) enrolled 87 students. In both treatments, the pre-test and post-test were a required part of the course. In the Studio Treatment, students were also required to participate in the PCRs and fill out the PCR exit questionnaires. Students were given the option of signing an informed consent form to release their data from these required parts of the course for analysis in the study.

In contrast, participation in the pre- and post-surveys and exit interviews were optional course activities. Students could agree to engage in these activities, for which they received course extra credit, by signing the informed consent form. Because not all students signed the informed consent form, the sample sizes considered in this study were smaller than the enrollments in each course. Moreover, because not all consenting students actually completed both the pre- and post-survey and exit interview, and because we chose to interview only a small number of consenting participants, our sample sizes within each treatment differed across our study measures (see Table 1).

In addition to the students who participated in this study, we hired six computer science graduate and upper-division undergraduate students to serve as moderators for the PCR teams in the Studio treatment. We required the moderators to attend a one hour training session prior to the first PCR. We paid moderators $40 for each PCR lab in which they worked.

3.2 Materials and Procedure

The course that defined each treatment took place during a fifteen-week semester.

Table 1. Sample sizes of the two treatments by measure

<table>
<thead>
<tr>
<th>Measure</th>
<th>Traditional</th>
<th>Studio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-/Post-Test</td>
<td>26</td>
<td>63</td>
</tr>
<tr>
<td>Pre-/Post-Survey</td>
<td>23</td>
<td>19</td>
</tr>
<tr>
<td>PCR Questionnaire</td>
<td>—</td>
<td>65</td>
</tr>
<tr>
<td>Exit Interview</td>
<td>8</td>
<td>4</td>
</tr>
</tbody>
</table>
In the first and last week of the semester, study participants took a survey that included both the 83-question Motivated Strategies for Learning Questionnaire (MSLQ) [15], and the Sense of Community Questionnaire (SCQ) [13]. Participants were given one week to complete the survey, and could save partial progress and come back to the survey as often as they wished within that week. Both these questionnaires were designed to get at students’ attitudes regarding their own learning, motivation, and sense of community.

Likewise, during the first week of the course, and again as part of the course final exam, students were required to complete a 30-question test of the programming knowledge covered in the course. The test included (a) 20 multiple-choice questions that required students to trace and identify the elements of code snippets of at most 20 lines; and (b) 10 short-answer questions that required students to trace code snippets and write short pieces of code. Students were given up to 45 minutes to answer the pre-test questions, and two hours to complete the comprehensive final exam in which the same post-test questions were embedded.

During the final week of both courses in the study, we interviewed a small sample of students in each course. To select students to interview, we partitioned the study participants in each course into three groups (low, medium, and high) based on their course performance thus far. We then randomly chose one to two students from each group for a 15 to 30 minute interview. Interviews consisted of 16 open-ended questions that focused on three broad issues: (a) students’ sense of community and the extent to which they interacted with others in the course; (b) the perceived impact of various course features, including studio-based activities in the studio course, on students’ learning; and (c) students’ intention to persist in computer science.

Recall that the course activity that differentiated the two treatments was the presence or absence of PCRs. In the Traditional treatment, students were given a series of programming problems to solve in each lab. They were asked to work through the problems during the lab with a partner. A graduate teaching assistant was present both to answer questions, and to ensure that students made progress on the tasks.

In contrast, in the Studio treatment, PCRs were substituted for the regular labs in weeks 8, 11, and 13 of the semester. The PCRs focused on three of the eight course assignments. The first assignment required students to perform computations on a file of numbers, and to solve equations in different forms; its solution required roughly 150 lines of code. The second assignment, the longest and most complex (over 500 lines of code), had students implement the game of battleship. The final assignment required students to implement a nine-function string library; it required roughly 250 lines of code.

Five days after each assignment was due, students in the Studio treatment came together for the PCRs, in which teams of three to four students, led by a trained moderator, reviewed their members’ code against an established checklist of coding best practices [20] augmented with a list of requirements for the specific programming solution being reviewed. The checklists included questions in seven general categories, including “structure and design,” “loops and branches,” and “defensive programming.” Each individual code review was limited to 30 minutes, and followed a well-defined procedure we have detailed elsewhere [8]. At the conclusion of each PCR session, students filled out an exit questionnaire that elicited their perceptions of the activity.

### 3.3 Results
We organize the presentation of our results around the study’s four dependent measures: attitude surveys, programming knowledge tests, PCR exit questionnaires, and exit interviews.

#### 3.3.1 Attitude Survey Results
Items on the MSLQ were scored using a seven point Likert-type scale with 1 being “not at all true of me” and 7 being “very true of me.” The MSLQ defined seven scales of interest to our study: intrinsic motivation, extrinsic motivation, self efficacy, critical thinking, self-regulation, and peer learning. In contrast, items on the SCQ, which gave us a measure of sense of community, were scored on a five point Likert-type scale with 1 being “strongly agree” and 5 being “strongly disagree.” Thus, the scale of the SCQ was the opposite of that of the MSLQ: lower scores indicated a higher scale level.

Table 2 presents the pre- and post-survey means by treatment. We used a repeated measures analysis of variance (ANOVA) to assess differences in means of the seven scales from pre-survey to post-survey by treatment. Given the field-based nature of our study and its use of quasi-experimental design, we set the alpha level at 0.10. In order to correct for Type I error, we then divided it by 7, giving us a threshold p-value of 0.014. We found no statistically significant within-subjects differences, although we can note one difference that trended toward significance: in the Traditional treatment, self-efficacy decreased from pre-survey to post-survey ($df = 1, F = 5.68, p = 0.0262$). No such significant decrease occurred in the Studio treatment ($df = 1, F = 0.72, p = 0.4065$).

To measure differences in survey gains between students enrolled in the Traditional and Studio treatments, we employed a between-subjects ANOVA. Again, to correct for Type I error, we divided the alpha level by 7 to obtain a threshold p-value of 0.014. Although we were unable to detect any significant between-subjects differences, we can report one notable observation: Students in the Studio treatment made a gain in peer learning, as compared to students in Traditional treatment, that trended toward significance ($df = 1, F = 3.78, p = 0.059$). It was also notable that students in the Studio treatment had a significantly higher mean post-survey peer learning score ($df = 1, F = 10.38, p = 0.0025$).

<table>
<thead>
<tr>
<th>Scale</th>
<th>Intrinsic Motivation</th>
<th>Extrinsic Motivation</th>
<th>Self Efficacy</th>
<th>Critical Thinking</th>
<th>Self Regulation</th>
<th>Peer Learning</th>
<th>Sense of Community</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pre</td>
<td>Post</td>
<td>Pre</td>
<td>Post</td>
<td>Pre</td>
<td>Post</td>
<td>Pre</td>
</tr>
<tr>
<td>Intrinsic Motivation</td>
<td>5.25 (1.06)</td>
<td>4.84 (1.16)</td>
<td>4.96 (0.98)</td>
<td>4.65 (1.10)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extrinsic Motivation</td>
<td>4.97 (1.38)</td>
<td>5.01 (1.05)</td>
<td>5.15 (1.16)</td>
<td>5.10 (0.86)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Self Efficacy</td>
<td>5.78 (0.66)</td>
<td>5.26 (1.06)</td>
<td>5.42 (1.11)</td>
<td>5.19 (1.25)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Critical Thinking</td>
<td>3.72 (0.98)</td>
<td>3.68 (1.43)</td>
<td>3.84 (0.91)</td>
<td>3.88 (1.10)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Self Regulation</td>
<td>3.90 (0.44)</td>
<td>3.91 (0.54)</td>
<td>4.21 (0.68)</td>
<td>4.13 (0.80)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Peer Learning</td>
<td>2.72 (1.17)</td>
<td>2.52 (0.98)</td>
<td>3.08 (1.16)</td>
<td>3.64 (1.28)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sense of Community</td>
<td>2.81 (0.49)</td>
<td>2.78 (0.63)</td>
<td>2.92 (0.55)</td>
<td>2.95 (0.73)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Pre- and post-survey means by treatment (std. dev. in parentheses)
3.3.2 Programming Knowledge Test Results
A total of 68 points were possible on the programming knowledge test. Table 3 presents the mean pre- and post-test scores by treatment. Both treatments experienced a significant gain in test performance from pre-test to post-test (Traditional: \(df = 1, F = 203.69, p < 0.0001\); Studio: \(df = 1, F = 613.83; p < 0.0001\)). However, there was no significant difference in the pre-test- to post-test improvement of the two treatments \((df = 1, F = 1.02, p = 0.3162)\).

3.3.3 PCR Exit Questionnaire Results
The first question on the PCR exit questionnaire asked students whether they found the code review of their own assignment solution to be helpful. Answers to this question were generally positive, with 77 percent of the responses expressing that the reviews were helpful, and another 8 percent expressing that the reviews were somewhat helpful. For example, one student said that “I found the code review of my assignment solution helpful because it was much easier to identify errors in the group,” while another student said that “it was a good way to go over everyone else’s code and help each other out, [as] others might see an error that I overlooked.”

In response to the second question, which asked students about the lessons they learned from the review of their own code, one student stated that “having someone else read my code out loud showed me some simple problems I didn’t notice.” Another student remarked that “I learned to appreciate criticism.”

Students also appreciated the opportunity to review other students’ code. Eighty-one percent of responses to the question “Did you find the code review of your team members’ assignment solutions helpful?” were in the affirmative, with another 6.5 percent expressing that reviews of peers’ code were somewhat helpful. For instance, one student remarked that “I learned to appreciate criticism.”

In response to a question that asked whether students had a sense of shared experience, whereas only half of the students in the Traditional treatment said that they had a sense of shared experience, whereas only half of the students in the Traditional treatment said so. Finally, in response to a question that asked students whether their experiences in the course had helped them feel more comfortable with interacting with others about computer programming, all students in the Studio course answered “yes,” whereas students in the traditional course seemed less sure, with only five of the eight students answering “yes.”

4. DISCUSSION AND FUTURE WORK
Our study results would appear to furnish limited empirical evidence for one of our two hypotheses. While we found no evidence that PCRs improved student learning outcomes as compared to traditional course activities (H1), we did find two pieces of evidence that PCRs may have positively influenced student attitudes about their own learning (H2).

The first piece of evidence relates to students’ self-efficacy, which can be seen as a measure of their self-confidence in computer programming. While self-efficacy diminished in the Traditional treatment at a level that approached significance, it did not decrease significantly in the Studio treatment. Why might this have been the case? Bandura’s theory of self-efficacy [2] holds that enactive experiences, in which learners have opportunities to assess their self-efficacy accurately, have the most influence on their perception of self-efficacy. We speculate that PCRs were powerful enactive experiences in the Studio treatment. In PCRs, students received feedback from other students on their own progress, and also provided feedback to others on their progress. Thus, PCRs provided students with powerful resources for judging their own progress relative to others in the course. Perhaps when students saw that their peers, too, struggled with the same things they did, computer

<table>
<thead>
<tr>
<th>Treatment</th>
<th>Pre</th>
<th>Post</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traditional</td>
<td>14.3 (8.0)</td>
<td>38.3 (8.7)</td>
<td>24.1 (8.6)</td>
</tr>
<tr>
<td>Studio</td>
<td>9.9 (7.6)</td>
<td>35.8 (7.9)</td>
<td>26.1 (8.4)</td>
</tr>
</tbody>
</table>

Table 3. Pretest, posttest, and pre- to post-test improvement means by treatment (std. dev. in parentheses)
programming seemed less intimidating, and their self efficacy did not decrease as much as it might otherwise have without PCRs.

The second piece of evidence relates to students’ attitudes toward peer learning—a measure of students’ ability to use each other effectively as learning resources. We found a statistical trend in favor of the Studio treatment with respect to its gain in peer learning. In addition, the Studio treatment had a significantly higher post-survey peer learning value. To explain this, we first note that PCRs are clearly a form of peer learning; they provide students with opportunities to improve their assignment solutions based on the feedback of their peers. As our PCR exit questionnaire data suggest, PCRs may have had positive effects beyond the three PCR sessions. Indeed, 96 percent of students said that they would consider working with another student on a future assignment. Thus, it appears that PCRs instilled in students a positive attitude toward peer learning. The Traditional treatment provided fewer opportunities for peer learning, and hence did not promote such a large shift in attitudes toward peer learning.

In interpreting our results, we believe it is important to keep in mind that, in the Studio treatment, PCRs constituted a relatively small part of the course. Indeed, students reviewed each other’s code for only three of the eight programming assignments in the course. We wonder whether a fuller implementation of studio-based instruction—for example, one that required students to engage in PCRs for all course assignments—might have had a more profound impact on both student learning outcomes and attitudinal changes. In future work, we will implement studio-based instruction more fully in a CS 1 course by requiring nearly every assignment to be peer reviewed. By collecting the same data as in the study presented here, we will be able to determine whether more studio work really is better.

In addition, in future work funded by a new multi-institutional NSF CPATH Class II Award, we will develop studio-based instructional methods for a greater variety of courses, and study the impact of studio-based instruction across many courses and institutions. Presently, more than 15 colleges and universities have committed to participating in this new line of research. We invite interested readers to contact the first author if they are interested in collaborating with us in this research.

5. ACKNOWLEDGMENTS

We are grateful to the anonymous instructor who allowed us to carry out this study in two of his courses. This project is funded by a National Science Foundation CPATH Award (CNS-0721927). Contributions to this work by other members of the research team—N. Hari Narayanan, Martha Crosby, Margaret Ross, and Rita Vick—are gratefully acknowledged.

6. REFERENCES