Online vs. Face-to-Face Pedagogical Code Reviews: An Empirical Comparison

Christopher Hundhausen*, Pawan Agarwal*, and Michael Trevisan†
Human-centered Environments for Learning and Programming (HELP) Lab
*School of Electrical Engineering and Computer Science
†College of Education
Washington State University
Pullman, WA 99164
+1 509-335-4590
hundhaus@wsu.edu, pagarwal@eecs.wsu.edu, trevisan@wsu.edu

ABSTRACT
Given the increased importance of communication, teamwork, and critical thinking skills in the computing profession, we have been exploring *studio-based* instructional methods, in which students develop solutions and iteratively refine them through critical review by their peers and instructor. We have developed an adaptation of studio-based instruction for computing education called the *pedagogical code review* (PCR), which is modeled after the code inspection process used in the software industry. Unfortunately, PCRs are time-intensive, making them difficult to implement within a typical computing course. To address this issue, we have developed an online environment that allows PCRs to take place asynchronously outside of class. We conducted an empirical study that compared a CS 1 course with online PCRs against a CS 1 course with face-to-face PCRs. Our study had three key results: (a) in the course with face-to-face PCRs, student attitudes with respect to self-efficacy and peer learning were significantly higher; (b) in the course with face-to-face PCRs, students identified more substantive issues in their reviews; and (c) in the course with face-to-face PCRs, students were generally more positive about the value of PCRs. In light of our findings, we recommend specific ways online PCRs can be better designed.

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 may not adequately prepare students for jobs in the computing profession, which, in addition to technical programming skills, are increasingly requiring so-called "soft" skills, including communication, teamwork, and critical thinking skills.

We have been exploring *studio-based instruction* as a means of providing students with increased opportunities to learn "soft" skills in core computing courses [8]. Adapted from architectural and fine arts education, our iterative 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 to peers and instructors for discussion and feedback.
3. Their peers and instructors critique the solutions and provide comments.
4. Students are given the opportunity to respond to comments and criticisms, and modify their solutions appropriately.

Based upon the formal code inspection process used in industry (see, e.g., [6]), we have developed the *pedagogical code review (PCR)* as an adaptation of the studio-based approach for lower-division computing courses [7]. In a PCR, students come together in teams to identify, discuss and log issues with each other's code. Students are then given the opportunity to resubmit their code solutions based on the feedback they receive in the reviews. In addition to implementing the four key steps of our studio-based approach, the PCR process provides students with opportunities to develop a variety of skills that are important in the computing profession—most importantly, the ability to critically review and discuss code and to improve code solutions based upon feedback.

In previous work [7,9], we studied face-to-face PCRs, in which teams of four to five students, led by a trained moderator (an upper-division undergraduate or graduate student), reviewed each other's code during 170-minute laboratory sessions. In those studies, we gathered empirical evidence that PCRs can positively influence the quality of students' code solutions [7], engage students in meaningful discussions about code [7], and promote positive attitudes with regard to peer learning [9]. However, given the reality that many computing instructors are unable to sacrifice the large amounts of laboratory or lecture time required for face-to-face PCRs, we posed the following research question:

SIGCSE '11, March 9–12, 2011, Dallas, Texas, USA.

RQ1:	*How can we design an online environment to enable PCRs be conducted asynchronously outside of class?*

This led to the following question regarding the potential effectiveness of online PCRs:

RQ2:	*Can PCRs conducted online promote the same benefits as PCRs conducted face-to-face?*

In this paper, we address these questions by presenting (a) an updated version of the Online Studio-Based Learning Environment (OSBLE) [10] designed specifically to support completely online PCRs, and (b) an empirical study that compares a CS 1 course that implemented online PCRs using this version of OSBLE against an identical CS 1 course that implemented face-to-face PCRs using our previous version of OSBLE. Our study had three key results: (a) in the course with face-to-face PCRs, student attitudes with respect to self-efficacy and peer learning were significantly higher; (b) in the course with face-to-face PCRs, students identified more substantive issues in their code reviews; and (c) students in the course with face-to-face PCRs were generally more positive about the educational value of PCRs. In light of these findings, we recommend ways online PCRs can be better designed so that they promote the kinds of benefits that have been observed in face-to-face PCRs.

2. BACKGROUND AND RELATED WORK

Studio-based instructional approaches can be traced to the master-apprentice educational system of the Middle Ages [1]. Educators in architecture and the fine arts adapted this approach in the form of the *design studio*: a shared space where students work iteratively on design projects, periodically presenting their work to their peers and instructor for critical review in so-called *design crits* (design critiques).

Studio-based approaches have been successfully integrated into science and math instruction (e.g., [4]). Computing educators have explored studio-based learning in their courses, with a particular interest in human-computer interaction courses, whose focus on design makes them particularly well-suited to the approach (e.g., [15]). There are even entire computing degree programs built upon the studio model (see, e.g., [3]).

The *pedagogical code review* (PCR) evolved out of our prior work to adapt studio-based instruction for lower-division computing courses [7]. PCRs can be seen as a form of *peer review*, a learning activity that has been embraced by several computing educators in both introductory (e.g., [18]) and upper-division (e.g., [2]) computing courses.

We present in this paper a new version of OSBLE designed to support completely online PCRs. This environment differs slightly from an earlier version of OSBLE [10] that was designed to support face-to-face PCRs. Several other web-based systems have been developed to support online peer reviews in computing education. These include RRAS [17], Peer Grader [5] and, most recently, an environment presented by Reilly et al. [14]. OSBLE has much in common with these existing environments—most importantly, its support for the online submission and review of programming assignments. However, whereas these existing environments have students perform reviews by filling in predefined rubrics, OSBLE is tailored specifically to support line-by-line code reviews. As such, OSBLE allows reviewers to fill out structured review log entries and anchor those entries to specific lines or line ranges of code, thus supporting a form of "artifact-centered discourse" [16].

Many computing educators have gathered anecdotal evidence of the effectiveness of their studio-based approaches; however, few have evaluated their approaches empirically. Only a few studies of which we are aware actually compare the outcomes of students who participate in studio-based learning against students who do not (e.g., [9,14]). The study presented here contributes to this line of research by presenting an empirical comparison of online and face-to-face studio-based learning activities, thus also contributing to the rich legacy of empirical research that compares online and face-to-face learning (see, e.g., [12]).

3. USING OSBLE FOR ONLINE PCRS

In order to address RQ1, we redesigned OSBLE [10] so that it supports completely online PCRs. Below we present a high-level walkthrough of how the new version of OSBLE can be used to facilitate completely online PCRs.

Step 1: Instructor creates assignment and assigns review teams. From OSBLE's instructor dashboard, an instructor clicks on "Create New Assignment," and specifies several important pieces of information through the Create New Assignment page: (a) the number and types of files to be handed in; (b) the due date of the initial submission, (c) the due date for performing the online code review; (d) the due date of the required resubmission of the assignment; and (e) the structure of the line-by-line issue log form. In addition, the instructor must create review teams; OSBLE provides a convenient interface that allows one to construct teams of a specified size using four different assignment methods: *manual*, *random*, *matched-ability*, and *mixed-ability*.

Step 2: Students submit assignment through OSBLE. By the due date, students submit their code solution through OSBLE as one or more code files.

Step 3: Student review teams review each other's solutions. Once the due date for the assignment passes, the review period commences. Students perform line-by-line reviews of their team members' code through the interface presented in Figure 1. Note that the reviews are *not* anonymous; students can see the names of the other members of their review team. However, students are unable to see the issues logged by other team members until *after* the review period is over, thus ensuring that their reviews are independent.

Step 4. Students resubmit code solutions based on feedback from the code review. After the review period is over, students revise their code solutions based upon the feedback they received from their team members. Through OSBLE, students specify whether each issue logged on their code was helpful to them, and whether they intend to address it in their resubmission. Finally, students resubmit their solutions through OSBLE by the specified due date.

4. EMPIRICAL COMPARISON

In order to rigorously compare the educational effectiveness of online PCRs to that of face-to-face PCRs (RQ2), we conducted a between-subjects empirical study with two treatments: *Face-to-Face (FTF)* and *Online*. These treatments corresponded to the spring 2009 and fall 2009 offerings of the CS 1 course at

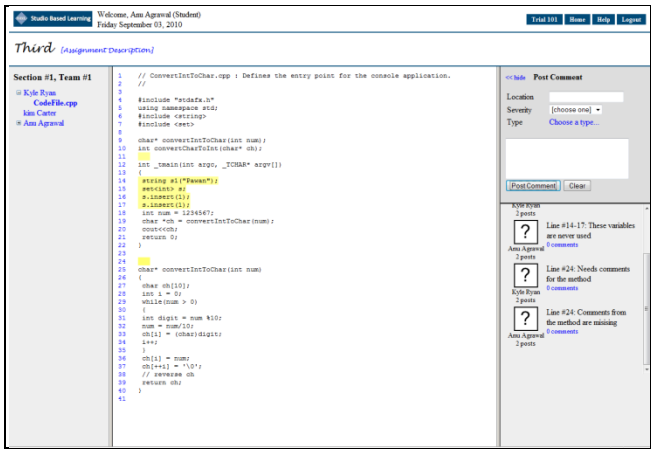

Figure 1. Snapshot of OSBLE's Code Review Interface

Washington State University. These two course offerings were identical in many important ways. Focusing on the C programming language, both had three 50-minute lectures and one 170-minute lab period per week. The spring 2009 (FTF) course was taught by an instructor not involved with this research; the fall 2009 (Online) course was taught by the first author.

The two treatments were defined by the differing ways in which they implemented PCRs. In the FTF treatment, students participated in PCRs of three of the eight course programming assignments. The PCRs were held during the regular lab period in weeks 8, 11, and 13 of the fifteen-week semester. In contrast, in the Online treatment, students participated in completely online PCRs of five of the eight course programming assignments.

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. We used software logs to collect quantitative data regarding the number and types of issues logged in the PCRs of each treatment. Finally, we used exit questionnaires and exit interviews to gather data on students' perceptions of the course and the PCRs.

4.1 Participants

The spring, 2009 course (the FTF treatment) enrolled 87 students, whereas the fall, 2009 course (the Online treatment) enrolled 95 students. In both treatments, the PCRs, pre-test, post-test, and exit questionnaires were required course activities. In contrast, the pre- and post-attitude surveys, along with the exit interviews, were optional. By signing an informed consent form, students could choose to participate in these activities, and to release to this study their data from the required course activities. Those who consented to participate received one percent course extra credit. Because not all students consented to participate, because not all consenting students actually completed the surveys and questionnaires, and because we interviewed only a small number of consenting students, our sample sizes within each treatment differed across study measures (see Table 1).

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

Table 1. Sample sizes of the two treatments by measure

Measure	FTF	Online
Pre-/Post-Test	63	51
Pre-/Post-Survey	19	48
Exit Questionnaire	65	49
Exit Interview	4	6

for each PCR lab in which they worked. No such moderators were involved in the Online treatment.

4.2 Materials and Procedure

The course that defined each treatment took place during a fifteen-week semester. In the first and last week of the semester, study participants took an online survey that included a subset of the Motivated Strategies for Learning Questionnaire (MSLQ) [13] and the Sense of Community Questionnaire (SCQ) [11]. Participants were given one week to complete the survey, which was 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.

Students filled out exit questionnaires either after each PCR (FTF treatment) or once during the final week of the course (Online treatment). The exit questionnaires consisted of a mix of Likert-style scale and open-ended questions designed to elicit students' impressions of the PCRs. In addition, during the final week of both courses, we interviewed a small sample of students in each course. To select students to interview, we partitioned the consenting study participants in each course into three groups (low, medium, and high) based on their course performance. We then randomly chose one to two students from each group for a 15 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 the PCRs, on students' learning; and (c) students' intention to persist in computer science.

Recall that the two treatments were defined by the method they used to implement PCRs. In the FTF treatment, face-to-face 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.

In the lab period following the due date of each assignment that was reviewed, students in the FTF treatment participated in face-to-face PCRs, in which teams of four to five students, led by a

trained moderator, used a previous version of OSBLE [10] to review team members' code against an established checklist of best coding practices [19] 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 [7]. Students were not required to resubmit their solutions following the review.

In contrast, in the Online treatment, teams of four to five students performed online, asynchronous PCRs of five of the eight course assignments. Three of these assignments were the same as the three assignments on which students in the FTF treatment performed PCRs. The other two assignments, which each required roughly 100 lines of code, had students process student records and characters from input files.

As described in Section 3, students in the Online treatment engaged in online PCRs using OSBLE. After each assignment was due, students had five days to perform independent reviews of team members' code. They were provided with an 11-page tutorial that provided guidelines on the process, which involved examining team members' code line-by-line, checking the code against the same checklist [19] used in the FTF treatment, and logging issues. Following the review period, students had five days to resubmit their assignments based on the feedback they received in their reviews. In addition, they were required, through OSBLE, to rate the helpfulness of each issue logged on their code, and to specify whether they implemented the change suggested by the issue. Students' original submissions were worth 70 percent of their assignment grades; their resubmissions were worth 30 percent.

4.3 Results

We organize the presentation of our results around the study's four dependent measures: attitude surveys, programming knowledge tests, PCR issue logs, and exit interviews and questionnaires.

4.3.1 Attitude Survey Results

Items on the MSLQ [13] were scored using a seven point Likert-style scale, with 1 being "not at all true of me" and 7 being "very true of me." The MSLQ defined seven scales used by our study: *intrinsic motivation*, *extrinsic motivation*, *self efficacy*, *critical thinking*, *self-regulation*, and *peer learning*. In contrast, items on the SCQ [11] were scored on a five point Likert-style 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.

Table 2 presents the pre- and post-survey means by treatment. We used a repeated measures analysis of variance (ANOVA) to assess differences in levels of the seven scales from pre- to post-survey by treatment. Given the field-based nature of our study and its use of weak quasi-experimental design, we set the alpha level at 0.10. In order to correct for Type I error, we divided it by 7, giving us a threshold *p*-value of 0.014. We found one statistically significant within-subjects difference: self-efficacy decreased significantly in the Online treatment ($df = 1$, $F = 21.79$, $p < 0.0001$); the effect size was large, $r^2 = 0.85$. No significant decrease was found in the FTF treatment ($df = 1$, $F = 0.72$, $p = 0.4065$).

To measure differences in survey gains between FTF and Online 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 made one notable observation: Students in the FTF treatment made a gain in peer learning, as compared to students in Online treatment, that approached significance ($df = 1$, $F = 3.86$, $p = 0.053$).

Table 2. Pre- and post-survey means by treatment (standard deviations in parentheses)

	FTF (n = 19)		Online (n = 47)	
Scale	**Pre**	**Post**	**Pre**	**Post**
Intrinsic Motivation	4.96 (0.98)	4.65 (1.10)	5.62 (0.91)	5.04 (1.11)
Extrinsic Motivation	5.15 (1.16)	5.10 (0.86)	5.12 (1.10)	5.14 (1.08)
Self Efficacy	5.42 (1.11)	5.19 (1.25)	5.59 (1.17)	4.92 (1.29)
Critical Thinking	3.84 (0.91)	3.88 (1.10)	4.67 (1.05)	4.29 (1.39)
Self Regulation	4.21 (0.68)	4.13 (0.80)	4.53 (0.76)	4.32 (0.85)
Peer Learning	3.08 (1.16)	3.64 (1.28)	3.79 (1.24)	3.54 (1.32)
Sense of Community	2.92 (0.55)	2.95 (0.73)	2.80 (0.45)	2.85 (0.75)

4.3.2 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 (FTF: $df = 1$; $F = 613.83$; $p < 0.0001$; Online: $df = 1$, $F = 437.98$, $p < 0.0001$). However, there was no significant difference in the pre-test- to post-test improvement of the two treatments ($df = 1$, $F = 0.03$, $p = 0.875$).

4.3.3 PCR Issue Log Results

Table 4 presents the number of issues of each type logged in the PCRs carried out in the FTF and Online treatments. In the FTF treatment, the counts and percentages reflect the issues logged by the two *teams* (of size 3 and 5) in which all team members consented to participate in this study, and which stayed intact throughout all three face-to-face PCRs. In contrast, in the Online treatment, the counts and percentages reflect the issues logged by the 19 *individuals* who actually logged issues in all five online PCRs. This difference in units of analysis (team vs. individual) is a consequence of the differing nature of the PCR process in each treatment: Whereas teams *collaboratively* logged issues in the FTF treatment, individual students *independently* logged issues in the Online treatment. While this comparison should be interpreted with caution, the comparison does point out substantial differences in the types of issues logged in the FTF and Online reviews. Indeed, according to a chi-squared test of homogeneity, there existed a significant difference in the distribution of issue types between the two treatments, $\chi^2(6, N = 1138) = 116.0$, $p = < 0.0001$. Upon closer inspection, we can note two key categorical differences. First, students in the Online treatment were far more concerned with issues of documentation and code formatting than students in the FTF treatment (32.6% vs. 7.4%). Second, issues logged in the FTF treatment focused more frequently on defensive

Table 3. Pretest, posttest, and pre- to post-test improvement means by treatment (68 pts possible; std. dev. in parentheses)

Treatment	Pre	Post	Improvement
FTF (n = 63)	9.9 (7.6)	35.8 (7.9)	26.1 (8.4)
Online (n = 51)	16.5 (10.0)	42.8 (7.8)	26.3 (9.1)

Table 4. Categorical breakdown of issues logged by treatment

#	Issue Category	FTF (n = 2 teams)	Online (n = 19 individuals)
1	Meets assignment requirements	19 (20.2%)	363 (29.2%)
2	Structure and design	25 (26.5%)	297 (23.8%)
3	Documentation, standards & formatting	7 (7.4%)	406 (32.6%)
4	Variables & constants	18 (19.1%)	119 (9.6%)
5	Arithmetic operations	1 (1.1%)	9 (0.7%)
6	Loops & branches	6 (6.4%)	25 (2.0%)
7	Defensive programming	18 (19.1%)	25 (2.0%)

programming (19.1%) and variables (19.1%) than in the Online treatment (9.6% and 2.0% respectively). These differences suggest that the PCRs in the FTF treatment tended to focus on more substantive issues than the PCRs in the Online treatment.

4.3.4 Exit Interview and Questionnaire Results

The exit interviews revealed many notable similarities in students' impressions regarding the value of various learning resources in the CS 1 course in which they enrolled. In both treatments, students were overwhelmingly enthusiastic about the value of the programming assignments. As one interviewee in the Online treatment put it, "the programming assignments were hard but [they] helped a lot, because you make a lot of mistakes and [in so doing] you learn a lot." Likewise, student interviewees in both treatments said they had received helpful feedback on their assignments. Whereas students in the FTF treatment singled out PCR moderators, students in the Online treatment most often cited teaching assistants as having provided helpful feedback.

With regard to the PCRs that took place, all students noted that they became comfortable with giving and receiving feedback through the PCR process. However, students in the FTF treatment were more enthusiastic about the PCRs in which they had engaged: whereas all four student interviewees in the FTF treatment said that they had received helpful feedback in the PCRs, just three of the six student interviewees in the Online treatment said they had received helpful feedback. One student in the Online treatment observed that "most students didn't put much effort into the online reviews, so I didn't find them helpful." As another Online participant noted, "I got a lot of opinionated stuff regarding [code] formatting—not too helpful."

The exit questionnaires completed by participants in both treatments provide further insight into their differing perceptions of PCRs. In the FTF condition, 85 percent of respondents found the PCRs helpful or somewhat helpful, in stark contrast to just 31 percent of the respondents in the Online treatment. Based on a reading of respondents' written comments, these differing perceptions appear to relate to two key factors:

Collaborative vs. individual reviews. In the FTF treatment, students appreciated that they sat down as a team to inspect code collaboratively; questionnaire responses tended to underscore the perceived value of this collaborative process. For example, as one respondent in the FTF treatment stated, "This process is very helpful. You are able to get input to make your program better, plus you get to look at others' programs and see different ways a program could be [written]." In contrast, respondents in the Online treatment tended to regard the PCRs as an opportunity to obtain feedback from *individuals*. Further, their perception of these individuals as reviewers tended not to be positive. As one Online respondent put it, "Most reviewers are just as inexperienced as I am and can't see the more serious problems..in [computer code]." Another stated that "not enough students took the reviews seriously and most students didn't know what they were talking about." A third said, "the criticisms" I received often made me feel [resentful] towards other members of my group."

Presence of skilled moderators. In the FTF treatment, student respondents especially noted the value of the expert moderators in the PCR process. In contrast, in the Online treatment, student respondents longed to have received reviews from more expert reviewers. As one FTF respondent bluntly stated, "Even though it would be more difficult to do, I think it would help students more if the [teaching assistants] did the code reviews."

5. DISCUSSION AND IMPLICATIONS

While our study failed to detect any significant learning outcomes differences between the two treatment groups, our study did identify several notable differences with respect to student attitudes and issue logging behavior:

- Unlike students in the FTF treatment, students in the Online treatment experienced a significant decrease in self efficacy.
- Whereas students in the Online treatment experienced a decrease in peer learning from pre-survey to post-survey, students in the FTF treatment experienced an increase. This difference between the two treatments approached significance.
- The categorical distribution of issues logged in the PCRs of each treatment differed significantly, with the Online treatment logging significantly more issues related to documentation and formatting, and the FTF treatment logging more substantive issues related to variables and defensive programming.
- In exit interviews, students in the FTF treatment were more enthusiastic about PCRs than students in the Online treatment.

We question the extent to which these differences were due to our study's independent variable (PCR medium). We believe two key procedural differences in the PCRs may have confounded our results, overshadowing any effects due to PCR medium. First, in the FTF treatment, PCRs were a *collaborative* process: Student teams worked together to identify issues and provide helpful suggestions. In contrast, in the Online treatment, team members performed PCRs *independently*. There was no attempt among team members to coordinate logged issues and arrive at a consensus. Our choice to require team members to work independently in the Online treatment stemmed from an interest in assessing each individual students' ability to critically review code. While many computing instructors certainly share such an interest in individual assessment, it is arguably somewhat at odds with the spirit of the studio-based learning model. Moreover,

requiring students in the Online treatment perform reviews independently may have had unintended consequences unrelated to the fact that students performed the PCRs online. First, it may have diminished the depth of the overall reviews; indeed, issues of documentation and formatting dominated the online PCRs. Second, it may have decreased students' perceptions of their own programming skills (lower self efficacy), and their willingness to learn from each other (lower peer learning).

A second key procedural difference in the PCR process was that, whereas skilled moderators guided the PCR process in the FTF condition, they were not used at all in the Online condition. As we learned in exit interviews in the Online treatment, students longed for feedback from more expert reviewers, who would have been able to provide more substantive feedback and guidance. Moreover, as suggested by past research into online learning [12], we suspect that having an expert moderator oversee and guide the online PCRs of each team would have increased students' motivation and engagement, leading to more positive attitudes about peer learning and more substantive reviews.

We believe our findings have two key implications for computing instructors who cannot spare the time required to implement face-to-face PCRs, and instead would like to implement online PCRs in their courses. First, we recommend that, rather than being concerned with assessing individual reviewing skills, instructors allow teams to perform PCRs *collaboratively*. As our study found, the collaborative process used by teams in the FTF treatment of our study, which was modeled after the successful code inspection process used in industry [19], appeared to promote better student attitudes and also to uncover more substantive coding issues. A more collaborative online PCR process might involve team members being allowed to view issues *as they are logged*. While this could cause team members to slack off, we speculate it might also lead to team members interacting with each other more extensively. As issues are logged, threaded discussions of the issues could take place in an attempt to arrive at a consensus and fine-tune the recommendations. Instructors could provide teams with explicit instructions for engaging in such a process, much as we provided FTF teams with explicit instructions for performing PCRs collaboratively.

Second, based on our findings, we believe it is essential that instructors assign a skilled moderator to each online PCR team. Moderators could be the instructors themselves, volunteer upper-division undergraduate students, or graduate teaching assistants. Whoever they are, moderators should be responsible for keeping the PCR process on track by (a) encouraging discussion around issues that are logged, (b) asking focusing questions, and (c) prompting students to look for deeper (non-superficial) issues. While enlisting skilled moderators in the PCR process will require extra effort on the part of instructors—not to mention a new reviewing interface in OSBLE—we believe our study's results suggest that skilled moderation is so important that it can make or break the success of PCRs conducted online.

6. ACKNOWLEDGMENTS

We thank the instructor who allowed us to run this study in his course. Parts of Section 3 were excerpted from [9], whose study design was similar to the one reported here. This project is funded by the National Science Foundation under grant nos. CNS-0721927 and CNS-093017. Contributions to this work by other members of the research team—N. Hari Narayanan, Dean Hendrix, Martha Crosby, Margaret Ross, and Rita Vick—are gratefully acknowledged.

7. REFERENCES

[1] A History of the Studio-based Learning Model. *http://www.edi.msstate.edu/studio.htm*.

[2] Anewalt, K. 2005. Using peer review as a vehicle for communication skill development and active learning. *J. Computing Sciences in Colleges*. 21, 2 (2005), 148-155.

[3] Docherty, M. et al. 2001. An innovative design and studio-based CS degree. *Proc.32nd SIGCSE Tech. Symposium*. ACM, New York, 233-237.

[4] Faro, S. and Swan, K. 2006. An investigation of the efficacy of the studio model at the high school level. *J. Educational Computing Research*. 35, 1 (2006), 45-59.

[5] Gehringer, E. 2001. Electronic peer review and peer grading in computer science courses. *Proc. 32nd SIGCSE Tech Symposium*. ACM Press, New York. 139-143.

[6] Gilb, T. and Graham, D. 1993. *Software Inspection*. Addison-Wesley.

[7] Hundhausen, C. et al. 2009. Integrating pedagogical code reviews into a CS 1 course: an empirical study. *Proc. 40th SIGCSE Tech Symposium.* ACM, New York 291-295.

[8] Hundhausen, C. et al. 2008. Exploring studio-based instructional models for computing education. *Proc. 39th SIGCSE Tech. Symp.* ACM Press, New York, 392-396.

[9] Hundhausen, C. et al. 2010. Does studio-based instruction work in CS 1? an empirical comparison with a traditional approach. *Proc. 41st SIGCSE Tech. Symposium. mputer* ACM, New York, 500-504.

[10] Hundhausen, C. et al. 2010. The design of an online environment to support pedagogical code reviews. *Proc. 41st SIGCSE Tech. Sympos.* ACM, New York, 182-186.

[11] McKinney, J. et al. The college classroom as a community: Impact on student attitudes and learning. *College Teaching*. 54, 281-284.

[12] Means, B. et al. 2009. *Evaluation of evidence-based practices in online learning: A meta-analysis and review of online learning studies*. U.S. Department of Education, Office of Planning, Evaluation, and Policy Development.

[13] Pintrich, D. et al. 1991. *A manual for the use of the motivated strategies for learning questionnaire*. Technical Report #NCRIPTAL-91-B-004. Nat. Center for Research to Improve Postsecondary Teaching and Learning.

[14] Reily, K. et al. 2009. Two peers are better than one: aggregating peer reviews for computing assignments is surprisingly accurate. *Proc. 2009 ACM GROUP* (Sanibel Island, Florida, USA, 2009), 115-124.

[15] Reimer, Y. and Douglas, S. 2003. Teaching HCI design with the studio approach. *Computer Science Education*. 13, 3 (2003), 191-205.

[16] Suthers, D. and Xu, J. 2002. Kukakuka: an online environment for artifact-centered discourse. *Proc. Eleventh World Wide Web Conference*. WWW2002. 472-480.

[17] Trivedi, A. et al. 2003. Automatic assignment management and peer evaluation. *J. Computing Sciences in Colleges*. 18, 4 (2003), 30-37.

[18] Trytten, D. 2005. A design for team peer code review. *Proc. 36th SIGCSE Tech. Symp*. ACM Press. New York, 455-459.

[19] Wiegers, K. 1995. Improving quality with software inspections. *Software Development*. 3, 4 (1995), 55-64.