The Design of an Online Environment to Support Pedagogical Code Reviews

Christopher Hundhausen, Anukrati Agrawal, and Kyle Ryan
Visualization and End User Programming Laboratory
School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164
+1 509-335-4590
hundhaus@wsu.edu, aagrawal@eecs.wsu.edu, kdryan@wsu.edu

ABSTRACT
Inspired by the formal code inspection process commonly used in the software industry, we have been exploring the use of pedagogical code reviews (PCRs), in which a team of three to four students, led by a trained moderator, (a) walk through segments of each other’s programming assignments, (b) check the code against a list of best coding practices, and (c) discuss and log issues that arise. We have found that PCRs not only improve the quality of students’ code, but also positively impact students’ communication and sense of community. However, implementing PCRs also presents a key logistical challenge: how to make code solutions and review results accessible to team members before, during, and after the team reviews? To address this challenge, we are developing an online environment specifically tailored to support PCRs. Our environment enables students to submit their code solutions; to review team members’ code solutions on-line prior to PCRs; to carry out PCRs; and to resubmit their solutions based upon the results of the PCRs. In an empirical evaluation of the environment in a CS1 course, we found that it not only eased the logistics of implementing PCRs, but also improved the organization and efficiency of the PCR process.

Categories and Subject Descriptors
K.3.1 [Computer Uses in Education]: Collaborative Learning, Computer-Assisted Instruction; 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, CS1, online peer review environment

1. INTRODUCTION
New jobs in the computing field are increasingly requiring design, critical thinking, communication, and teamwork skills in addition to programming skills. We believe that traditional approaches to teaching computing, which aim to develop programming skills through a series of individual programming assignments, inadequately prepare students for such jobs. In order to better prepare students for jobs in the computing profession, we believe computing courses need to emphasize design, communication, and collaboration skills as well as programming skills.

To that end, we have been exploring studio-based instruction as a means of engaging students more actively in the learning process and helping them to develop design, communication, and collaboration skills [6]. Adapted from architectural and fine arts education, our studio-based instructional approach emphasizes learning activities in which students construct solutions to assigned computing problems, and then present their solutions to their instructors and peers for review and discussion.

One possible adaptation of studio-based instruction for lower-division computing courses is the pedagogical code review (PCR) [5]. Based on the formal code inspection process (see, e.g., [4]) commonly used in the software industry, PCRs have small teams of students first review each other’s code individually, and then come together as a team both to log issues (defects and improvements) that they encountered, and to identify additional issues as a team. Students can then be given an opportunity to resubmit their assignments based upon the results of the review.

In previous work, we implemented PCRs in a CS1 course at Washington State University in the Spring Semester of 2008 [5]. Our empirical evaluation produced evidence that PCRs (a) positively impacted the quality of students’ code; (b) stimulated discussions about increasingly sophisticated programming issues, including issues of design; and (c) inspired students to seek help from other students and work together outside of the PCRs.

In addition to documenting these educational benefits of PCRs, our empirical evaluation identified two key logistical problems with implementing PCRs purely with paper-based materials:

1. Distribution of student code solutions was difficult. Running a PCR requires that all members of a review team have a copy of each team member’s code. We found it both logistically challenging and wasteful to print out all of this code. Moreover, we were unable to distribute code printouts to students such that they had sufficient lead time to review each other’s code prior to the face-to-face reviews—one of the keys to the success of code reviews [10].

2. Distribution of code review results was difficult. A PCR yields an issue log, which documents in detail the issues that were found. We required the code author to fill in and retain a paper-based issue log of the issues that arose within the review of his or her code. However, we did not copy and distribute these issue logs to other students, as it would have required more effort and paper than we were willing to sacrifice. Thus,
students missed out on a potentially valuable educational activity: that of reviewing each other’s issue logs in order to verify their accuracy and identify commonalities between their and others’ issues.

To address these problems, we present here a preliminary design and evaluation of OSBLE (Online Studio-Based Learning Environment), an online environment specifically tailored to support PCRs. The Instructor interface allows instructors to set up a course, add students and review teams, and to specify programming assignments. The Student interface allows students to upload assignment solutions, and to review and log issues on team members’ solutions online. The Moderator interface allows a moderator to assign team roles, and to run a PCR. In an empirical evaluation, we found that OSBLE not only succeeded in overcoming the logistical obstacles identified above, but also improved the organization and efficiency of the PCR process.

2. RELATED WORK
The on-line environment presented here (OSBLE) supports both asynchronous reviews performed by individual students, and face-to-face PCRs performed by teams. Educators in many fields have become increasingly interested in incorporating online components into their courses. In addition to supporting both asynchronous and synchronous discussions, widely-used course management systems such as Blackboard [1] allow the instructor to post course announcements, materials, and assignments; students can hand in assignments and even take quizzes online through these systems. While not designed to support general course management, OSBLE is similar to these systems in that it does support some basic course management features. For example, it allows instructors to post course announcements, provide links to course materials, and configure course assignments for online submission.

Several computing educators have developed web-based systems specifically to facilitate asynchronous online peer review. For example, RRAS [9] enables instructors to develop online assessment rubrics for programming assignments, and to assign student reviewers to assignments. After submitting their assignments online through RRAS, students can anonymously review the assignments to which they were assigned by clicking in and submitting the assessment rubric.

In a similar vein, Peer Grader [2] allows students to submit an assignment for review by their peers. Reviewers can anonymously provide both written feedback and a grade to authors. Peer Grader allows authors to revise and resubmit assignments, in order gradually to converge upon a more polished product. In addition, to encourage reviewers to perform high-quality reviews, Peer Grader requires a set of students to assess and grade each review. Review grades are then factored in to students’ final assignment grades.

Peer Grader is used as a foundation for Expertiza [3], which uses peer review as a basis for building up repositories of reusable learning objects from which students in subsequent courses can benefit.

More recently, Reily et al. [7] describe an environment designed specifically to support online peer review of student programming assignments in introductory computing courses. Like RRAS, the Reily et al. environment requires reviewers to submit reviews based on a rubric that is specified by the instructor. Like Peer Grader, the Reily et al. environment also allows peer reviews to be rated; however, the review ratings may be provided only by the author who receives the reviews—not by others, as is the case for Peer Grader. Unlike both RRAS and Peer Grader, the Reily et al. environment requires reviewers to specify, as part of their reviews, specific test cases and the results of executing those test cases. The test case results, along with the rubric-based assessment, are combined into a peer review score. In an empirical evaluation of their environment, Reily et al. found that (a) the peer review scores of students well matched those of teaching assistants; (b) students generally preferred the reviews of students who had less programming experience than they had; and (c) students who wrote peer reviews performed significantly higher in the course than students who did not.

OSBLE has much in common with the environments just reviewed—most notably, its support for the on-line submission and review of programming assignments. However, because OSBLE is specifically tailored to support the PCR process, it has two key differences: (a) it allows reviewers to anchor structured review log entries to specific lines or line ranges (thus supporting a form of “artifact-centered discourse”; see [3]); and (b) it supports both asynchronous code reviews performed by individuals, and face-to-face code reviews performed by teams.

3. USING OSBLE TO SUPPORT PCRs
Through an iterative, user-centered design process, we developed OSBLE in C# using ASP.NET 3.5. OSBLE is designed specifically to support the following usage scenario, which involves five key steps:

1. Instructor creates an assignment. The instructor logs in and creates a course assignment. This entails specifying several assignment details, including the due date and the precise files that students must submit as part of their solutions (see Figure 1).

2. Students submit solutions. Prior to the assignment due date, students log in and submit the specific deliverables required for the assignment, "locking" them once they have reached their final form. Once a student’s solution has been locked, the student can no longer modify it, and the code solutions of the student’s team members become available to the student for review.

3. Students perform online reviews. Once the due date passes, those students who have submitted assignment solutions are charged with individually reviewing the solutions of their team members. To review the code solution of a team member, a student locates the team member in a list, and then selects a specific component of the solution to view. To log an issue that pertains to a specific line of code, the student can either click on the line or on the "Log an issue" button (Figure 2), both of which bring up a dialog box that can be filled in with the details of the issue (Figure 3). Once an issue is logged, the line(s) to which it pertains are highlighted in yellow; highlighted lines can be clicked on to view the issue(s) associated with them.

4. Students perform face-to-face PCRs with moderator. PCRs are typically held in a regularly-scheduled lab period that takes place after the assignment is due. At the beginning of each PCR, the team moderator assigns the team members to specific PCR roles (author, reader, inspector, recorder), and then clicks the "Start Inspection" button to commence an in-lab review (see Figure 2). OSBLE makes it easy, within this review, for teams to review the issues logged by individual members of the teams, and to transfer those issues upon which they agree to the team review log. To do this, the team can choose the "View Issue Log" button, which brings up a list of the
issues logged by team members ahead of time (see Figure 4). Next to each issue is a link entitled “To Lab”; if clicked, this link adds the corresponding issue to the team review log, and changes the highlighting of the issue from yellow to orange to indicate that it was logged by the team (see Figure 5).

5. Students resubmit solution based on PCR. After a PCR is completed, students are given an opportunity to review the issue logs for their assignment solutions, and to resubmit a solution based upon those logs.

4. EMPIRICAL EVALUATION

The original motivation for our development of OSBLE was to make the PCR process easier to implement and more effective for students. In particular, we were interested in exploring the following three research questions:

RQ1. As compared to implementing PCRs with paper, will implementing PCRs with OSBLE improve access to student code and issue logs?

RQ2. As compared to implementing PCRs with paper, will implementing PCRs with OSBLE improve the organization and efficiency of the PCR process?

RQ3. Will students’ ability to use OSBLE to review team members’ code solutions prior to face-to-face PCRs enable teams to log more issues than they can with paper, which does not easily support “ahead-of-time” reviews?

To address these research questions, we ran an empirical study in the spring, 2009 offering of CptS 121 (“Program Design and Development”), the CS 1 course at Washington State University. Our study (henceforth referred to as the “OSBLE Study”) nearly replicated the one we ran in the Spring, 2008 offering of CptS 121 [5] (henceforth referred to as the “Paper Study”). The key difference was that, instead of using paper to implement the PCRs that took place in the course, we used the version of OSBLE presented in the previous section. Running this follow-up study allowed us to make direct comparisons between a paper-based and OSBLE-based implementation of PCRs.

The CptS 121 courses in which both studies took place were nearly identical: they had the same instructor (who was not involved in this research), textbook, assignments, schedule, and grading scheme. In addition, both CptS 121 courses required students to participate in three PCRs that took place in course labs in the eighth, eleventh, and thirteenth weeks of the fifteen week semester. The dates of the code review labs immediately followed (by two to five days) the due dates of three (of the eight total) course assignments. Below, we summarize key details of the studies, and report our main findings.

4.1 Participants

The Paper Study focused on three review teams consisting of 10 students enrolled in one section of the spring 2008 offering of CptS 121. That section enrolled a total of 22 students. In contrast, the OSBLE Study focused on six code review teams consisting of 26 students enrolled in two sections of the spring 2009 offering of CptS 121. These two sections had a total of 40 students.1

1 The numbers of participants in each study reflect the fact that only a subset of students signed an informed consent form to release their data for analysis in our study.
In addition, we hired five computer science graduate students to serve as moderators in the Paper Study, and six computer science graduate and upper-division undergraduate students to serve as moderators in the OSBLE Study. Three of these moderators were common across both studies. We required the moderators to attend a one hour training session prior to the first PCR. We paid moderators $40 to $50 for each code review lab they worked.

4.2 Formation of Code Review Teams
Students were divided into code review teams of three to four. To form code review teams, we created an ordered list of students based on their course grades as they stood the day before the first scheduled PCR. We then assigned students to teams in a round-robin fashion, so that the teams were roughly equivalent in terms of the abilities of their members.

4.3 PCR Materials and Procedure
Students in both studies completed all course assignments in the C programming language, the instructional language for the course. The three course assignments that were the focus of the PCRs in both studies varied in complexity and length. The first assignment had students process numbers in a file and solve equations in different forms; it required roughly 150 lines of code. The second assignment, the longest and most complex (roughly 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 Paper Study, students did not have access to each other’s solutions prior to the PCRs. In contrast, in the OSBLE Study, students were asked to review their team members’ assignment solutions ahead of time in OSBLE.

In the OSBLE study, both moderators and students were provided with training on OSBLE. Prior to the first PCR, moderators and students were given a brief (fifteen-minute) demonstration of how to use OSBLE to perform PCRs. In addition, a nine-page tutorial on how to use OSBLE was available online by clicking on the “Help” button in OSBLE.

4.4 Data Collection Methods
In both studies, we employed three primary data collection methods. First, we attended all of the PCR sessions, and noted key observations. Second, we videotaped one code review performed by each team in each of the three PCR sessions. Third, we collected and analyzed all code review logs generated by the teams. In addition, in the OSBLE study, we instrumented OSBLE to collect user activity and usage data.

4.5 Results
We organize the presentation of our results around the three research questions posed for this study. For each research question, we present and compare relevant data and observations collected in the Paper and OSBLE Studies.

RQ1: As compared to implementing PCRs with paper, will implementing PCRs with OSBLE improve access to student code and issue logs?

In the Paper Study, the code solutions of other students were not available prior to the PCRs, thus preventing students from reviewing each other’s code ahead of time. Moreover, after a team reviewed a given student’s code solution, only that student (the code author) retained a copy of the team issue log; other team members could not review the issue log at a later time.

In contrast, in the OSBLE Study, students were able both to review the code solutions of others online, and to log issues online, immediately after the assignment due date passed. All team members were able to view issues so logged. Further, since teams used OSBLE to log issues during the PCRs, all team members could access the issue logs generated in the PCRs at any time after the PCR took place.

RQ2: As compared to implementing PCRs with paper, will implementing PCRs with OSBLE improve the organization and efficiency of the PCR process?

In the Paper Study, we required students to bring four printouts of their code solution to each PCR lab. However, students frequently forgot to bring the printouts. As a result, we observed that the team moderators, in collaboration with students, generally spent the first 15 to 30 minutes of each PCR lab making the printouts necessary to perform the code reviews. In addition, in the Paper Study, the course instructor had to print out and distribute two copies of the issue log to each student: one that was filled out and retained by code author, and a second that was filled out by the team recorder and retained by us for research purposes.

In contrast, in the OSBLE Study, students accessed each other’s code solutions during the PCR labs using their own laptop computers, or one of those available in the lab. While logging in to OSBLE and accessing the code solutions of authors incurred a minimal amount of delay, we observed that the process went much more smoothly than in the Paper Study. Moreover, since issue logs were filled out maintained online through OSBLE, the instructor did not have to print out paper issue logs, nor did students have to retain (and potentially lose) paper copies.

To determine whether OSBLE might enable teams to log issues more quickly, we wanted to use our video data to compare the amount of time teams needed to log issues in both studies. Unfortunately, the video recordings from the Paper Study were of such a poor quality that we were unable to obtain timing data from them. However, we did obtain high quality recordings of all the review sessions of two teams that were enrolled in the Spring, 2009 offering of CptS 121, but that were in a different section from the teams that participated in the OSBLE study. These teams used OSBLE to view each other’s code, but logged all issues (both ahead of time and within the PCR) on paper. Under the circumstances, the video data of these teams provided us with a reasonable basis for comparing the efficiency with which teams can log issues with paper versus OSBLE.

Based on 27 issues logged, the two teams that used paper took an average of 66.4 seconds ($sd = 23.6$ seconds) to log an issue. In contrast, based on 52 issues logged, four OSBLE teams took an average of 42.0 seconds ($sd = 24.4$ seconds) to log an issue. (We failed to gather complete video data on two of the OSBLE teams, so their data were not included in this analysis.) According to an independent samples $t$-test, this difference is statistically significant ($df = 77$, $T = 4.31$, $p < 0.0001$). In practical terms, OSBLE enabled teams to log issues 37 percent faster than paper, resulting in a more efficient PCR process.

RQ3: Will students’ ability to use OSBLE to review team members’ code solutions prior to face-to-face PCRs enable teams
We invite interested educators to join us in this research. OSBLE is under active development, and is freely available at osble.sourceforge.net under the GPL open source license. Educators who are interested in tailoring OSBLE to their own courses are encouraged to contact the authors for assistance.

<table>
<thead>
<tr>
<th>Condition</th>
<th>n</th>
<th>PCR 1</th>
<th>PCR 2</th>
<th>PCR 3</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paper</td>
<td>3</td>
<td>13.7 (10.5)</td>
<td>16.0 (7.2)</td>
<td>9.3 (1.5)</td>
<td>39.0 (2.6)</td>
</tr>
<tr>
<td>OSBLE</td>
<td>6</td>
<td>20.5 (3.5)</td>
<td>12.3 (7.2)</td>
<td>15.0 (6.0)</td>
<td>47.8 (18.3)</td>
</tr>
</tbody>
</table>

6. ACKNOWLEDGMENTS

We are grateful to the anonymous instructor who allowed us to implement and study PCRs in his CptS 121 course. This project is funded by a National Science Foundation CPATH Award (CNS-0721927). Contributions to this work by other members of the research team—Mike Trevisan, Dana Fairbrother, N. Hari Narayanan, Martha Crosby, Margaret Ross, Rita Vick—are gratefully acknowledged.

7. REFERENCES