A Peer Review System to Enhance Collaborative Learning: Testing and Preliminary Evaluation

Luke Komiskey, Daphne Brinkerhoff, Hannah Miller,
Joline Morrison, and Mike Morrison
Department of Computer Science
University of Wisconsin-Eau Claire
Eau Claire, WI 54702
morrisjp@uwec.edu

Abstract

Peer review is a proven learning approach that allows students to observe and critique different solutions to a problem as well as to receive feedback on their own work. The overarching purpose of this research is to develop a computer system to support this learning approach within the classroom environment. This system will allow instructors to create and administer peer review assignments easily and with a variety of configuration options. It will enable students to view the work they are reviewing electronically, submit reviews, and also display peer reviews of their own work.

This paper reports the results of the evaluation phase of the project, and determines if the system we have developed adequately meets the research goals of the original project: to provide a collaborative learning environment that is easy to use for both instructors and students, and provides useful feedback to students to enhance learning.

Introduction

Peer review is an important component of learning in computer science courses, and can also be helpful in other fields of study. By reviewing others' code, students can learn new techniques and approaches as well as improve their analytical ability. Students also benefit from detailed feedback on their own code. Despite the benefits of peer review, it can be difficult and time-consuming for professors to administer. For these reasons, online systems to automate the administration process of peer review have been developed.

The system that is the subject of this research was developed to support peer reviews in a variety of courses and review configurations, and automatically generate a variety of assessment questions to target specific assignment goals. It differs from previous systems in its flexibility in specifying how reviews are assigned and administered, and in specifying review assessment rubrics. The system was also designed specifically to facilitate peer review of computer programming assignments, which often involve submitting multiple computer files that must be stored in a specific folder structure. A complete summary of the system requirements identification and development approach is provided in (8).

This paper describes the development and testing of a prototype system that addresses these requirements. The first section describes our systems development research approach and specifies where the current project fits in this approach. The next section provides an overview of software testing. Next, we describe the testing process that our system has undergone to ensure that it will accommodate a large number of simultaneous users in a classroom setting. The final section describes our contributions and summarizes what we learned.

Research Approach

This research uses a systems development research methodology proposed by Nunamaker and colleagues (e.g., (1), (2)) illustrated in Figure 1. In this methodology, the researcher first identifies research problems and related research questions. He or she then develops and evaluates a software prototype for a new software system using the steps shown. Evaluation results may suggest revision of the prototype concepts, requirements, architecture, design, or implementation. After implementing these revisions, the research repeats the evaluation phase one or more times, with the goal of satisfying the research questions. The prototype itself serves as a system specification or working system to support further research. This paper focuses on the testing and evaluation phase of an existing software prototype.

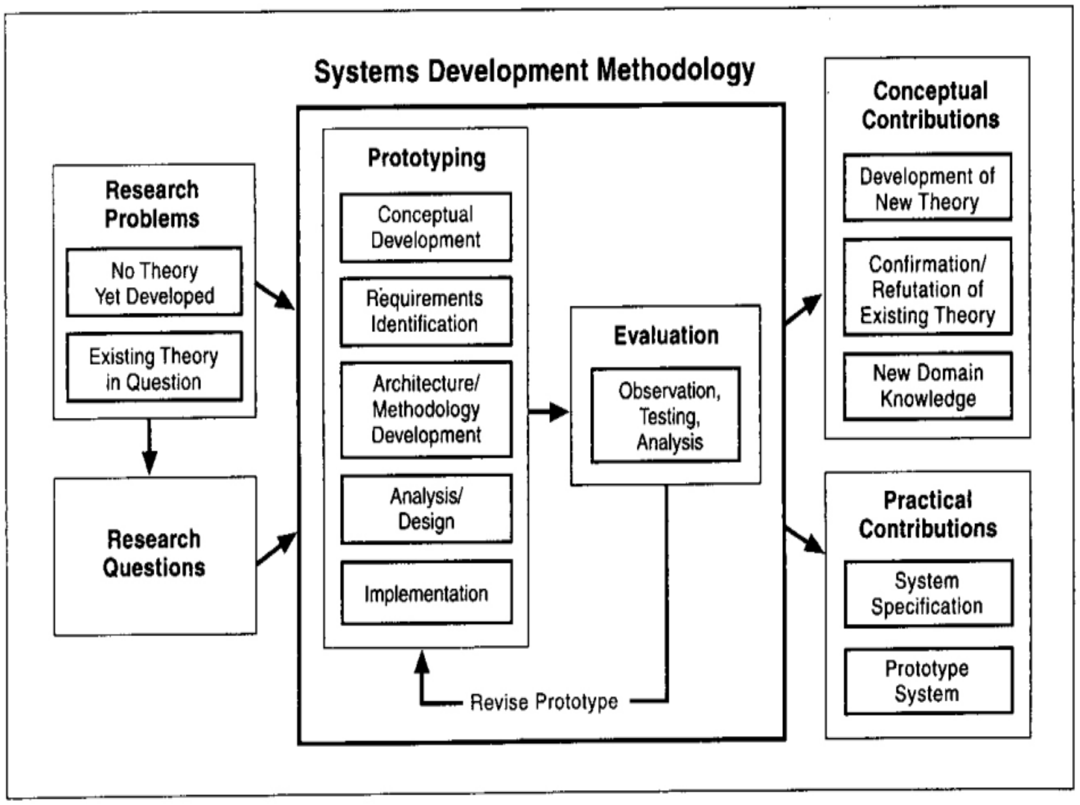


Figure 1: Systems development research methodology

System Description

Figure 2 illustrates the system architecture.

Peer Review Architecture

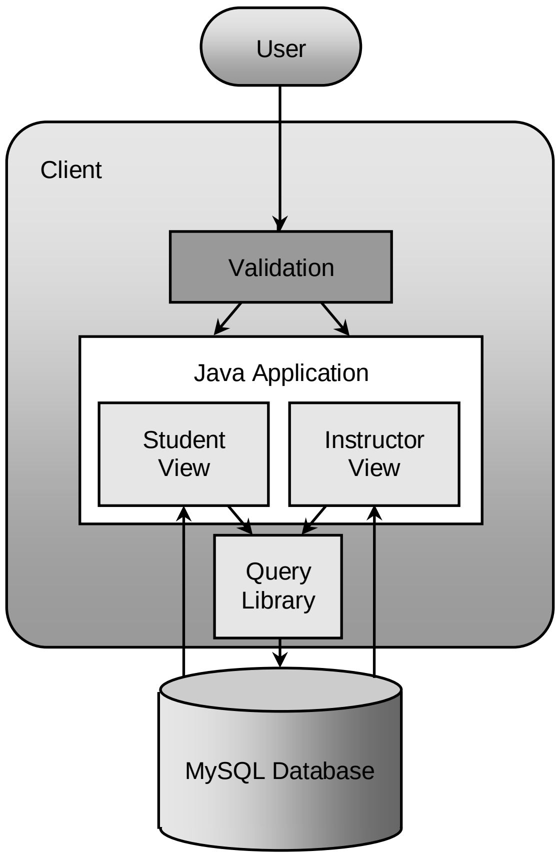


Figure 2: Peer Review system architecture

The first application element that appears is a login screen enables the user to gain entry to the application. After the login dialog, the user interface is split into two independent interfaces: one for students and one for instructors. All system information is stored in a MySQL database that contains tables for students, instructors, classes, sections, assignments, reviews, questions, and answers. Submitted student assignment files are stored in Binary Large Object (BLOB) columns. The application accesses the database using a standard Java Database Connection (JDBC) driver for MySQL developers. A complete description of the system is provided in (8).

Software Testing Basics

Testing is an essential part and is the chief time consumer of all software development. In order to properly test our software, we contracted software testing research from R. E. Fairley's *Software Engineering Concepts* (3). According to Fairley, the goal of software testing is to assess and improve the quality of the work products generated during development and modification of software.

Software testing mainly consists of verification and validation. Verification involves life-cycle verification in which the software is constantly tested throughout its development, and it also involves formal verification in which the source code is intensely tested to verify that it conforms to specifications. Validation is the process of evaluating software at the end of the software development process to determine compliance with the requirements.

A main objective throughout the verification and validation processes to work towards quality assurance—a planned and systematic pattern of all actions necessary to provide adequate confidence that the item or product conforms to established technical requirements. A number of established tests allow for software developers to accomplish this goal. These tests include:

- Function tests, which specify typical operating conditions, input values, and expected results, and are based on requirements that must be quantified and testable;

- Performance tests, which verify response time under varying loads and percent of execution time spent in various segments of the program;

- Stress tests, which intend to purposely overload a system in various ways to determine limitations of a system;

- Structure tests, which traverse a specified number of paths through each routine in the system to establish thoroughness of testing.

Collectively, the entire testing process effectively verifies and validates the software and optimizes quality assurance.

Testing Approach

Based off of R. E. Fairley's software testing research (3) and using J. Henry's test plan outline from *Software Project Management: A Real-World Guide to Success* (4), our team developed a test plan to test our Peer Review System software. Our tests were designed to test if our program functions correctly, but not to prove that it is flawless. In other words, we were open and hopeful to discover errors within our system. Developing a test plan allowed us to test our program structurally so that we could easily discover, identify, analyze, adjust to, and fix errors within our program. We were using the testing stage as an opportunity to find defects within our system and fix them before we actually release it.

Our testing process progressed through a series of milestones in which we tested the program by users with different demographics and for various specific testing purposes. Our first test was to test the system using single, non-simultaneous users in the student role and the instructor role. These users did not have any background knowledge about the system. They were asked to follow specific executable instructions to test for specific functionality and assess the usability of the system.

The next test phase was group testing, in which we tested the program's ability to perform with multiple simultaneous users, which represented a form of stress testing. Then we had a small group of individuals familiar with the system perform tests to exhaustively test the system through additional stress testing, and to test all possible user paths (structural testing). Lastly, we implemented the program into an actual class for field testing.

Our intent was to evaluate each of the different program functions so that every line of our code was executed, and some were executed multiple times using different execution paths. By writing specific test instructions, we could control what was executed, and get specific feedback on which of our functions worked and which were defective. Additionally, by introducing our program into an actual classroom for use by computer-experienced students and an instructor, we subjected our program to testing as if it were being used as a released system. In order to analyze our test results, we collected different types of measurements throughout so that we could analyze our results and use them to better our program.

Test Results

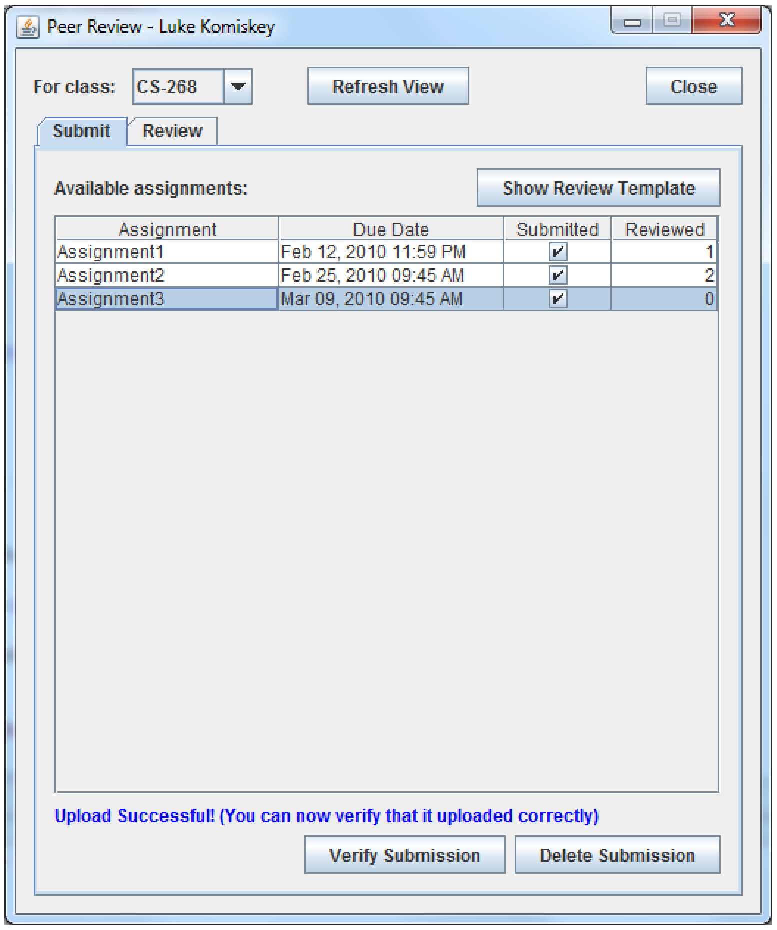
For our first phase of testing, we provided an outlined instruction sheet of steps for an inexperienced student and instructor to complete individually with no additional input or instructions. Both the student and instructor instructions walked through the typical functions that the user would complete. In both scenarios, the test subjects were able to complete all of the steps, so this function test proved successful. However, we did find some usability issues. Users experienced difficulties derived from interface confusion; some of the text used on various buttons was misleading. This outcome spurred us to update our button text and add instructional messages to make our interface more intuitive. Future tests verified the wisdom of this update.

We used the same student instructions from the first function test and distributed them to a collected group of users so we could stress test our system functioning with many users simultaneously. In order to accumulate feedback, we had each test subject fill out a progressive comment sheet documenting their results and findings throughout this testing sequence. Everyone was able to complete all of the steps, verifying that our system could withstand multiple users concurrently.

Reflecting on the feedback that we had collected, our testing revealed even more necessary features to fix or renovate in our system. Prior to group testing, we had not yet changed some of the interface issues from our single-user tests, and the comments reaffirmed the need for these revisions. Some users who were in both the single-user and group test expressed frustration because they felt their previous comments had been disregarded. This was not the case; rather, we felt we couldn't fix everything at once and prioritized the changes. In retrospect, we wished we'd either made all of the changes or not allowed the same testers to be in both test groups.

Additionally, our stress testers found that no size limit existed for the files/folders that could be submitted into the system. For example, users attempted to upload the contents of an entire hard drive (several gigabytes), and the system allowed this action without being able to handle it. In order to fix this, we added a feature that checked for the size of the submission and denied submissions that were too large based on a given size limit of eight megabytes (based on MySQL BLOB limits).

Users also disliked that the interface sometimes showed outdated information; for example, users could not see if something changed while they were in the system, so they had to exit and re-enter the system in order to view the updated data. This prompted us to add a "Refresh View" button (see Figure 3) that retrieves current data from the database and displays an accurate updated view. Overall, this phase of the testing uncovered numerous small problems that caused us to make many small, functionally-beneficial changes to our system. Finally, our comment sheets proved to be a fortunate feature within our group testing because the feedback from the test subjects provided us with a large variety of suggestions to improve our system.



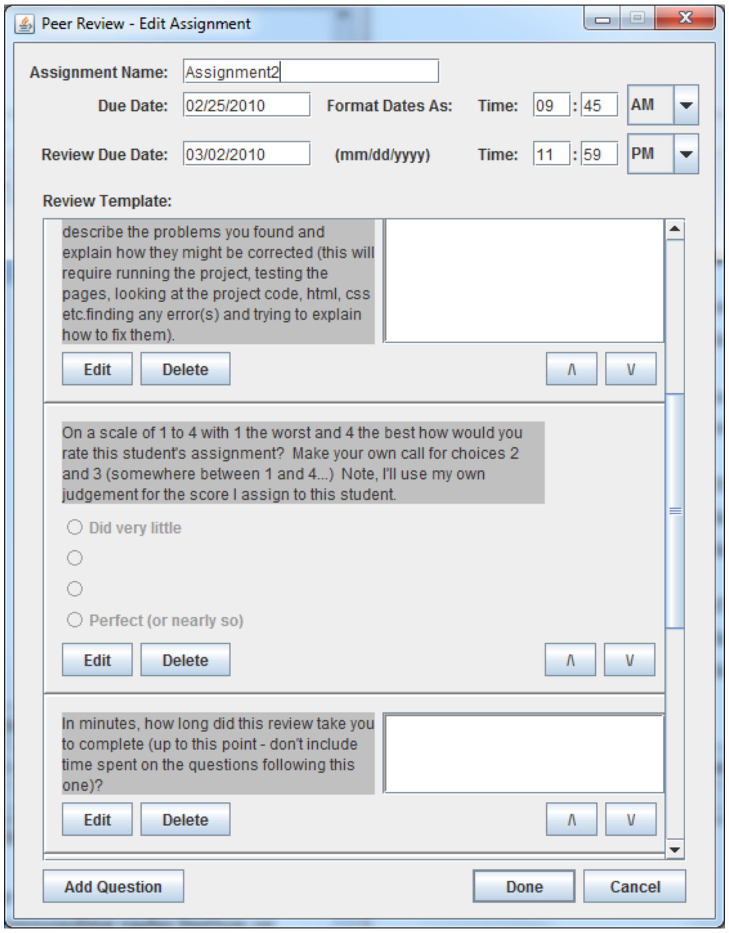
Assignment	Due Date	Submitted	Reviewed
Assignment1	Feb 12, 2010 11:59 PM	✓	1
Assignment2	Feb 25, 2010 09:45 AM	✓	2
Assignment3	Mar 09, 2010 09:45 AM	✓	0

Upload Successful! (You can now verify that it uploaded correctly)

Figure 3: Current Student View displaying added Refresh View button, new text on Submission buttons, and text notifications.

Next, the research team gathered with the goal of structurally testing the system as thoroughly as possible. One tactic was to try and upload many different submissions varying by size and file type. The other tactic was to individually attempt to test every feature in all possible permutations. This allowed us to test the new size limit feature, proving that our system accurately accepted and denied the submissions of the corresponding sizes as well as functioned with a variety of file types. In this phase, we clarified the requirements for the due dates of submissions versus reviews.

Before clarification, the given default due dates and times when creating an assignment were misleading as to what time they were referring to (12 o'clock noon or midnight). Also, an instructor was able to specify a review due date that was before the submission due date. As a result, we disabled that function, and assigned new default due dates and times. This testing also led to many suggestions for desirable features within the instructor view, some of which we implemented, and some of which may be future modifications. The group testing had primarily focused on the student view, so this phase proved very beneficial in that it allowed us to focus much more on the instructor's view and functionality. Some of the added instructor features include the ability to reorder the questions within an assignment and the system's ability to maintain a question in progress while it is being edited (see Figure 4).



Assignment Name: Assignment2

Due Date: 02/25/2010 Format Dates As: Time: 09 : 45 AM

Review Due Date: 03/02/2010 (mm/dd/yyyy) Time: 11 : 59 PM

Review Template:

describe the problems you found and explain how they might be corrected (this will require running the project, testing the pages, looking at the project code, html, css etc.finding any error(s) and trying to explain how to fix them).

On a scale of 1 to 4 with 1 the worst and 4 the best how would you rate this student's assignment? Make your own call for choices 2 and 3 (somewhere between 1 and 4...) Note, I'll use my own judgement for the score I assign to this student.

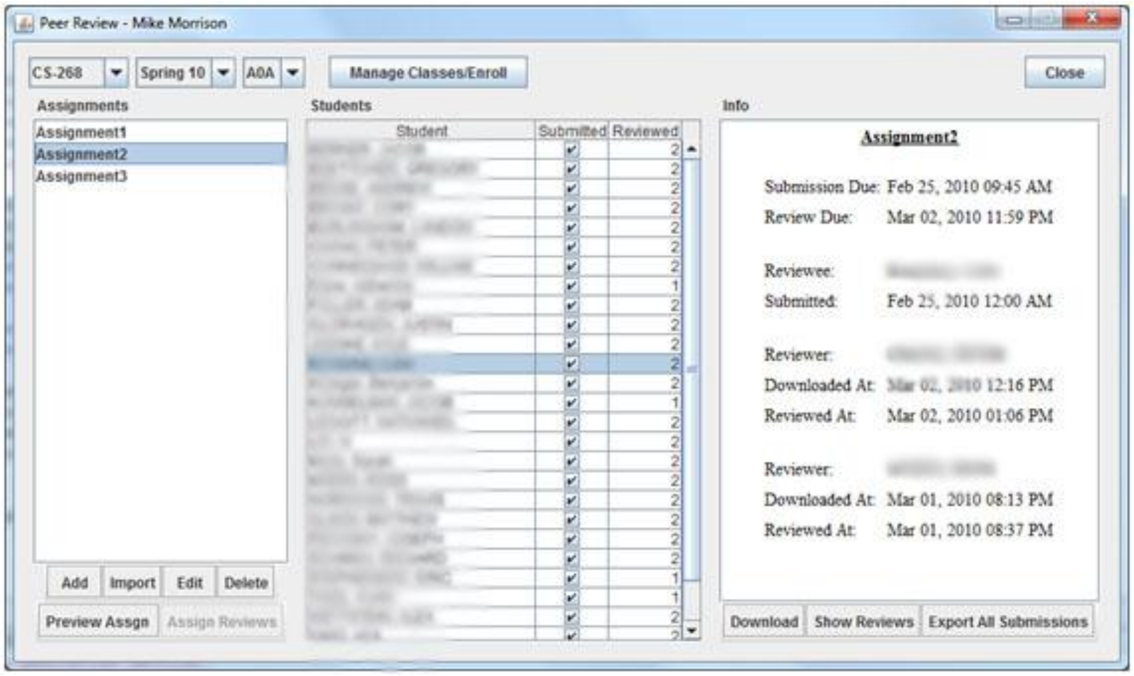
○ Did very little
○
○
○ Perfect (or nearly so)

In minutes, how long did this review take you to complete (up to this point - don't include time spent on the questions following this one)?

Figure 4: Instructor Edit Assignment View using new default times and editing abilities

The final step was to field test our system in a real classroom environment. In preparing for this testing scenario, we realized that we needed some additional features in order to ensure an optimal experience. For the convenience of the instructor, we implemented the ability to import a previously constructed review template. This option simplifies situations where an instructor might have two sections of the same class and might want to give a similar assignment to both. While we already had a feature to download a single student's submission, we recognized that a more efficient approach would be to allow an instructor to download all of the students' submissions for one section in one operation.

Overall, the first round of classroom testing was a success. However, we not only found some more problems to fix in the system, but we also collected some usability feedback and opinions from the users through questions that we composed and added to the review template. The two main problems were uploading difficulties from individual campus hard drives, and slow scrolling and faulty scroll bar defaults. Following the testing, we fixed both of these problems.



CS-268 | Spring 10 | A0A

Assignments

- Assignment1
- Assignment2
- Assignment3

Students

Student	Submitted	Reviewed
[redacted]	✓	2
[redacted]	✓	2
[redacted]	✓	2
[redacted]	✓	2
[redacted]	✓	2
[redacted]	✓	2
[redacted]	✓	2
[redacted]	✓	1
[redacted]	✓	2
[redacted]	✓	2
[redacted]	✓	2
[redacted]	✓	2
[redacted]	✓	2
[redacted]	✓	1
[redacted]	✓	2
[redacted]	✓	2
[redacted]	✓	2
[redacted]	✓	2
[redacted]	✓	2
[redacted]	✓	2
[redacted]	✓	2
[redacted]	✓	2
[redacted]	✓	1
[redacted]	✓	1
[redacted]	✓	2
[redacted]	✓	2

Info

Assignment2

Submission Due: Feb 25, 2010 09:45 AM
Review Due: Mar 02, 2010 11:59 PM

Reviewee: [redacted]
Submitted: Feb 25, 2010 12:00 AM

Reviewer: [redacted]
Downloaded At: [redacted] 12:16 PM
Reviewed At: Mar 02, 2010 01:06 PM

Reviewer: [redacted]
Downloaded At: Mar 01, 2010 08:13 PM
Reviewed At: Mar 01, 2010 08:37 PM

Figure 6: Instructor View now containing the ability to Import already-created assignments, Preview Assignments, and Export All Submissions at once

The most rewarding part of this testing resulted from the user comments collected upon reflection of their experience with the system. As far as usability, 20 of the 26 subjects (77%) found the system to be either "easy" or "very easy" to use, and none of the testers found it to be "difficult." Six of the testers reported that they found usability to be "OK," but their comments revealed that they experienced difficulty in the uploading process, which we fixed upon completion of this round of testing. Besides requesting input on system ease of use, we also inquired as to whether doing the peer review helped the reviewer to better understand the subject material. Fourteen responded that it did help understand the subject material, while ten answered that it did not. The appended explanations were enlightening: the goal of our peer review system was to give students a new way to learn through both seeing other students' techniques and through reflecting on other students' opinions of their work. Several of the students' comments demonstrated success in this goal:

"It's nice to put some ties on the loose ends and questions that I had in my own project by seeing someone else's,"

"I am sure that everyone did the same task different ways and I am not saying that one way is better, but it is nice to learn different techniques,"

"Looking at somebody else's code helped me see what I had done wrong and it allowed me to fix it."

One student even found the peer review process to be so beneficial that he suggested that his class do this after every assignment. Though the system allowed some students to find answers to their questions, learn different techniques, and fix their own mistakes, others understandably did not gain from the experience:

“This student did his/her project almost exactly the way I did mine. I think that that if they had implemented it differently I would have checked yes to this but since there wasn’t anything added by doing the review I selected no,”

“If the assignments were more unique, I think it would be fun. But this assignment allowed for less creativity than other assignments,”

“If it was an assignment that I couldn’t finish, seeing what someone else did would be helpful.”

The process will not always be ideal in all situations, but students will likely benefit from open-ended assignments that implement a general task which lend themselves to differing approaches. Moreover, we found an additional benefit to the system when the instructor reported that reviewing a review had revealed a flaw in the student’s program that he had missed when he actually graded the assignment. Therefore, reviewing students’ reviews also proves advantageous for the instructor because it provides the instructor with potential additional perspective. Overall, all of the involved students voted that they would or possibly would support completing the process again – providing our system with potential future use within classrooms.

Conclusions, Reflections, and Future Directions

The testing process gave us an appreciation for the importance of software testing. When the same people constantly look at their own work, everything seems so intuitive, understandable, and efficient. Testing with a wide range of subjects allowed us to gain insight from many different valuable views. We were able to collect feedback from actual potential users, instructors, and students that each had their own opinions about what the system has to offer. Although we felt that the system provided the required functionality, completing the process allowed us to uncover a variety of necessary enhancements. One example was in the look-and-feel of the system: users had a number of surprisingly specific suggestions about colors, shapes, words, and placement of items in the interface. Before this project, we did not expect such strong or valuable opinions. It is gratifying to see how far our system has evolved as a result of receiving such enthusiastic and useful feedback.

Some future directions for our system lie in some of the enhancements suggested during the various testing phases-- features that we discovered could be advantageous additions to our current system. Currently, our system is administered so that the whole process is anonymous, but a future feature that could be added would be a choice to control the anonymity in case an instructor would like the peer reviewing not to be anonymous. The testing process also revealed that although we already added a Refresh View button for the student view, a similar button would also be useful in the instructor view.

Instructor feedback indicated the need for a notification function within the system: it would be helpful for an instructor to have a built in operation that allows him/her to be able to log in to the system soon before an assignment/review is due and send out an e-mail reminder to whoever has not yet submitted their review or assignment. The instructor also relayed that it is a hindrance to have to log in again after an assignment is due to assign the reviewers; a suggestion was to add functionality so assigning reviews for an assignment could be set up ahead of time. Lastly, the instructor provided input that, while the system currently tracks and displays the number of people that have reviewed a specified student, it would be beneficial to know also how many reviews that student has completed.

Testing within the team exposed supplementary enhancement possibilities such as protecting against the ability to change the file extension while saving when downloading a file, and providing a more aesthetic way to display review information. Finally, discussion of possible use cases led us to the idea of team use; we would like to implement a function within the system that allows teams to use the system versus individual students. Our team believes that adding or bettering each of these features will lead to a strongly evolved prototype of our system that will bring us much closer to a final product.

Currently, we have just begun to use the system in an actual classroom setting to determine its viability in helping students see other ways of solving a programming problem and get the benefits of reviewing others' work. We are in the process of determining if the system is fulfilling all of the design requirements and enhancing student learning. We believe that in the future the system can be used to support other academic disciplines. Eventually, if other universities or schools want to use the system, it could potentially be expanded and modified to allow different set-ups with other campus database/directory systems. Ultimately, we would like the system to become open-source software so that classrooms everywhere can benefit.

References

1. **Morrison, J. and George, J.F.** Exploring the Software Engineering Component in MIS Research. *Comm. of the ACM.* July, 1995, Vol. 58, 7.
2. **Nunamaker, J.F., Chen, M., and Purdin, T.D.M.** Systems development in information systems research. *J. MIS.* Winter, 1990-91, Vol. 7, 3.
3. **Fairley, R. E.** *Software Engineering Concepts (McGraw-Hill Series in Software Engineering and Technology).* New York : McGraw Hill Companies, 1985. ISBN:0-07-019902-7.
4. **Henry, J.** *Software Project Management: A Real-World Guide to Success.* Toronto : Addison Wesley, 2003. ISBN-10: 0201758652 .

5. *Lightweight Preliminary Peer Review: Does In-Class Peer Review Make Sense?* **Denning, T., Kelly, M., Lindquist, D., Malani, R, Griswold, W.G. and Simon, B.** Portland, OR : ACM, SIGCSE 2007 Proceedings.
6. **Wolfe, W.J.** Online Student Peer Reviews. SIGITE 04 Proceedings.
7. *Process Improvement of Peer Code Review and Behavior Analysis of its Participants.* **Wang, Y., Li, Y., Collins, M., and Liu, P.** Portland, OR : ACM, SIGCSE 08 Proceedings.
8. **Holt, Brandon, Komiskey, Luke, Morrison, J., Morrison, C. M.** A Peer Review System to Enhance Collaborative Learning. *MICS 2009 Proceedings.* [Online] 2009. [Cited: March 15, 2010.] http://mics.sdsmt.edu/proceedings/Paper/mics2009_submission_5.pdf.
9. **Shnedlerman, Ben, and Leavitt, Michael.** Research-Based Web Design & Usability Guidelines. *Usability.gov.* [Online] 2006. [Cited: September 29, 2009.] http://www.usability.gov/pdfs/guidelines.html#2. ISBN 0-16-076270-7.
10. *Electronic Peer Review and Peer Grading in Computer Science Courses.* **Gehringer, E.F.** Charlotte, NC : ACM, SIGCSE 2001 Proceedings.
11. **Infopoll.** Survey Tips: How to write a good survey questionnaire. [Online] March 15, 1998. [Cited: September 29, 2009.] http://www.accesswave.ca/~infopoll/tips.htm.
12. **QuestionPro Survey Software.** Survey Questions - Online Surveys. [Online] 2009. [Cited: September 29, 2009.] http://www.questionpro.com/survey-questions.html.
13. *Automatic Assignment Management and Peer Evaluation.* **Trivedi, A., Kar, D.C., and Patterson-McNeill, H.** s.l. : Consortium for Computing in Small Colleges, 2003.