A Peer Review System to Enhance Collaborative Learning

Brandon Holt, Luke Komiskey, 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 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 describes the system architecture, database schema, and database implementation of the first iteration of this system. It presents the client-side interface with which the users will interact, which is a Java-based client application (JNLP) that the users of the system (instructors and students) will use to upload assignments, assign reviews, and create reviews. It concludes with our contributions, a summary of what we learned, and our plans to test the system and assess its usability and utility.

Introduction

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. Instructors are often reluctant to use this approach in their classrooms because of the overhead involved in creating and administering these assignments. A computer-based peer review system that automates the administration process seems like a logical solution. A number of online peer review management systems exist, but most lack flexibility in specifying how reviews are assigned and administered, and in specifying review assessment rubrics. The purpose of this project is to develop and evaluate a general purpose peer review system that can support peer reviews in a variety of courses and review configurations, and automatically generate a variety of assessment questions to target specific assignment goals.

This paper describes the development of a prototype system that addresses these requirements. The first section describes our systems development research approach. The next section develops a research framework by describing the peer review process, and reviewing existing peer review systems features. From this review, we identify the requirements for our software prototype. Next, we describe the system architecture, database schema, and database implementation for our prototype, as well as the client-side interface with which the users will interact. The final section describes our contributions, a summary of what we learned, and our plans to test the system and assess its usability and utility.

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.

Conceptual Development and System Requirements

Peer review is widely acknowledged as a beneficial tool for student learning by encompassing critical thinking and active learning (3). Wolfe (4) observes several less obvious advantages of peer review: students have the opportunity to see other students' work, which may be of higher or lower quality than their own and become more aware of where they fit in the overall fabric of the class; students with higher knowledge levels can

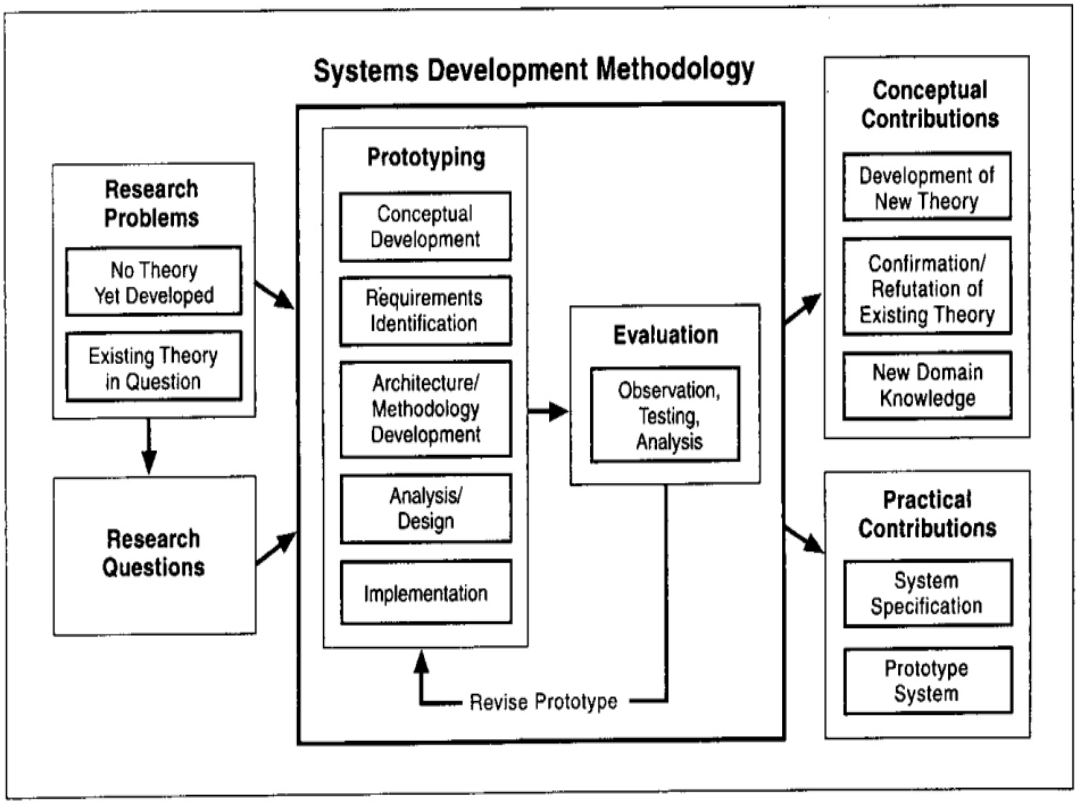

Figure 1: Systems development research methodology

share this with their classmates; and, students were observed to work harder and turn in higher quality work to impress their classmates.

Wang et. al (5) describes peer review as a six-phase process:

1. The author (student) completes his/her assignment program solution files;
2. The author submits the solution files to the instructor;
3. The reviewer performs the review as assigned;
4. The reviewer submits the review and makes it available to the author and the instructor;
5. The author revises the assignment based on the review;
6. The instructor confirms that the author and reviewer perform their work satisfactorily.

In addition, the instructor must specify the following configuration details:

1. Assignment and review due dates;
2. Review assessment rubric or scoring method;
3. Review anonymity level (non-anonymous, single-blind, double-blind);
4. Number of reviewers for each assignment;
5. Reviewer assignments (random or non-random).

The literature describes several existing peer review systems. One is Peer Grader (PG), described in (5). This system is a Web-based application written in Java that allows authors to submit files for review either sequentially, or in a single Zip file. The latter approach allows directory hierarchies to be maintained. The system generates a Web page that contains links to the submitted files. Reviewers can then access files, review them, and provide feedback in the form of a text input field and a letter grade. The system can assign reviewers "pseudo-randomly", or the instructor can assign the reviewers using an external spreadsheet. From this information, we surmise that the student and reviewer information is also contained in an external spreadsheet. This configuration is supports totally anonymous (double-blind) reviews.

A system described by Wolfe (4) supports having students log on to a course Web site, submit the URL of an assignment for review, and then access other submitted assignments and perform reviews. In this configuration, anonymity is one sided: students know who they are reviewing, but do not know who has reviewed them.

Both of these systems do not seem to provide much flexibility in how instructors can target particular aspects of assignments. Read, Review, and Access System (RRAS) (7) is an interesting system that has an administrative interface for maintaining course, instructor, and student information; an instructor interface for creating assignments, assessment rubrics, and accessing submissions and reviews; and a student interface for allowing students to log on, view assignment details, submit assignments, view peer evaluations, and check grades. Of specific interest is the ability of this system to create more detailed assessment questions, such as "Program easy to read and understand; Good use of indentation and white space." However, it is not clear if reviewers can respond to this measure using a quantitative value or a text comment.

In summary, existing systems support the submission and review process, but none support the required configuration options and flexibility. The following section describes the system architecture, interface, and database schema of the system we developed to address these requirements.

System Description

The system is written in Java and uses a Java Web Start launcher, which you place as a link on a Web page. When the user clicks the link, the launcher downloads and runs the Java application on the client machine. This application connects directly with the MySQL database to retrieve the user's data and upload new information. Figure 2 illustrates the system architecture.

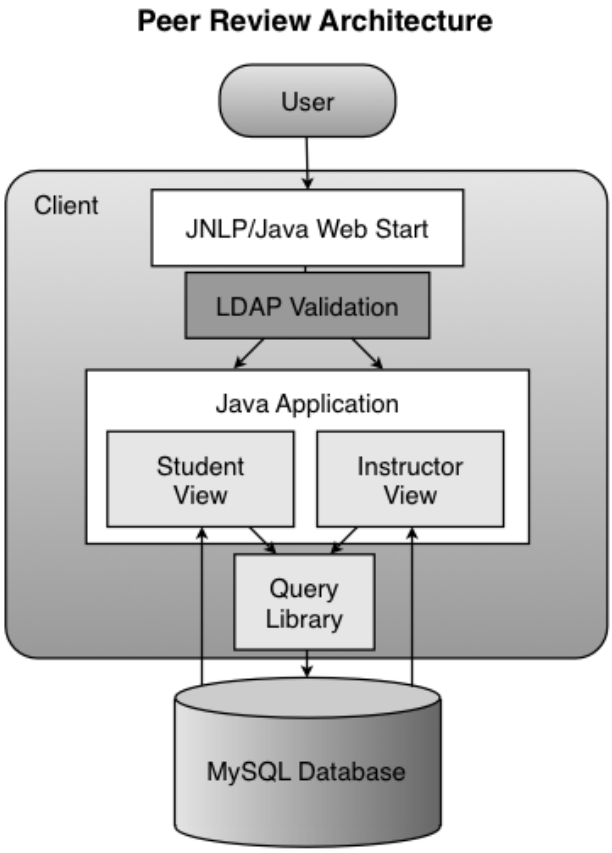

Figure 2: Peer Review system architecture

Java Web Start is a service that allows users to launch Java applications directly from web browsers. Java Web Start requires the Java Runtime Engine (JRE), but prompts the user to download and install the JRE if it is not already installed. Java Web Start differs from Java applets in that its programs don't run inside the web browser. They are fully independent Java applications, so they are able to bypass restrictions that applets encounter, such as allowing hard-drive access (with the user's explicit permission, of course). Java Web Start applications are configured by Java Network Launch Protocol (JNLP) files. These files use an XML schema to define the application's run parameters, security restrictions, and the location of the program's Java Archive (JAR) files.

We chose to use Java for our peer review system for its ability to work across different platforms. Using a JNLP instead of an applet allows the application access to the user's file system, and gives us more flexibility in designing the user interface (such as allowing custom dialogs and all the freedom of the Java Swing library). The first application element that appears is a login screen. We knew that we needed a way to track data of individual users (students and instructors) in our system, so users have individual database entries that links them with all of their review information. This also provides a way to secure each individual's information. Most campuses have some sort of directory service to provide students and instructors with unique usernames and passwords to

access private information. We didn't want our users to have to maintain another password, so we chose to use the existing directory validation system. At the University of Wisconsin-Eau Claire, we have a Microsoft Active Directory server that stores user information. Using the Lightweight Directory Access Protocol (LDAP), we can query the server with a given username and password combination and get a response whether or not it is correct. When a student is added to the peer review system, their University username is associated with their entry in the database. At the login screen, they enter their University username and password and the application connects via LDAP to validate the login information. If it is correct, then the user gains entry to the application. In order to make the system portable to other universities, we plan to add an enhancement that makes the system configurable so it can easily use either LDAP validation or internal security validation.

After the login dialog, the user interface is split into two independent interfaces: one for students and one for instructors. When the login username has been validated, the system searches for the username in the instructor and student tables and brings up the appropriate view. The student view allows students to submit assignments, download and review assignments they've been assigned to review, and view reviews for their assignments.

The instructor view lets the instructor add and edit classes, sections, and assignments, enroll students, then switch between classes and sections and see the assignments, who has submitted and reviewed them, and view the submissions and reviews. (These functions will be described in more detail in the section describing the user interface.)

All system information is stored in a MySQL database. This contains tables for students, instructors, classes, sections, assignments, reviews, questions, and answers. Submitted files are stored in Binary Large Object (BLOB) columns. The Java application accesses the database using a standard Java Database Connection (JDBC) driver for MySQL developers.

User Interface

All interface components were programmed manually in Java, using Java's Swing package. The user interface is divided into two different views: Instructor and Student. After the validation module determines who the user is, the system displays the correct view. The following subsections describe these views.

Student View

Figure 3 illustrates the Submit tab of the Student View interface.

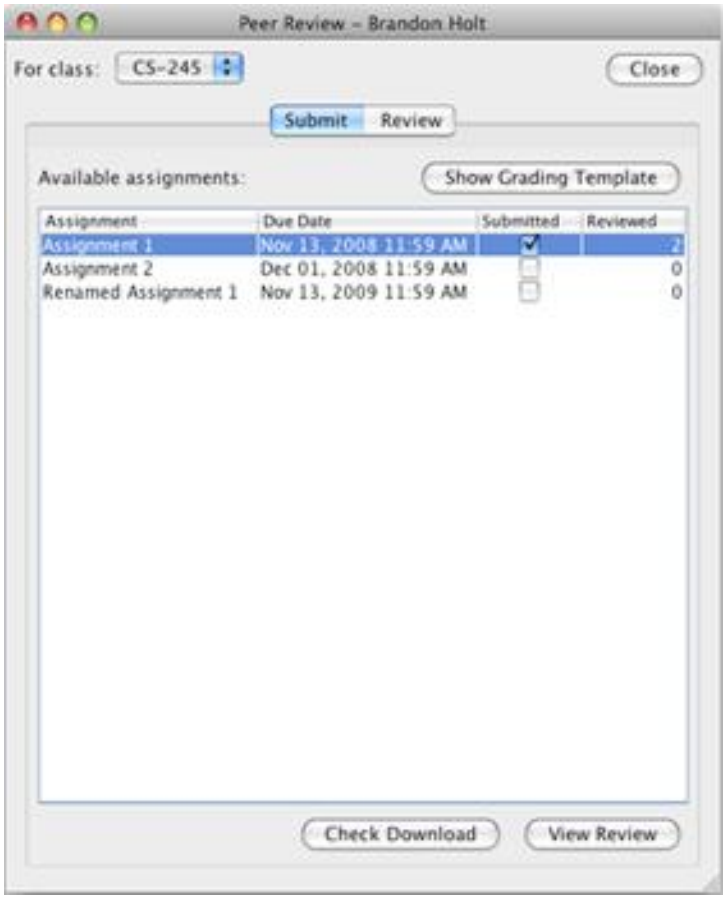

Figure 3: Student View interface (Submit tab)

When a student logs on to the system, the application displays all classes in which the student is currently enrolled. The student can switch between multiple classes using a combo box at the top of the window. The rest of the window is a tabbed pane with a tab that allows them to view their submissions, and a second tab to view their reviews. The multiple tabs and corresponding views were created to avoid confusion over which assignments are their own and which ones they have to review.

The Submit tab allows the user to:

- View a list of all assigned assignments and their due dates;
- Determine whether or not they have uploaded their submissions;
- Find the number of reviews that have been submitted for each submission;
- Submit an assignment;
- Check to make sure they are able to download their own submission;
- Delete their submission if they so choose;
- View reviews that have been submitted;
- Display the grading template that the reviewers will be using to review the assignment.

The user is not allowed to submit an assignment or delete their submission after the assignment's due date is past. This prevents them from altering their submission after seeing others' assignments.

The Review tab (Figure 4) allows the user to:

- View all assignments they are assigned to review;
- View the review due date for each assignment;
- Download the assignment;
- Review other students' submissions, and modify their review if before the review due date;
- View a read-only version of their review (after the review due date);
- Determine whether they have downloaded and/or submitted a review for each assignment.

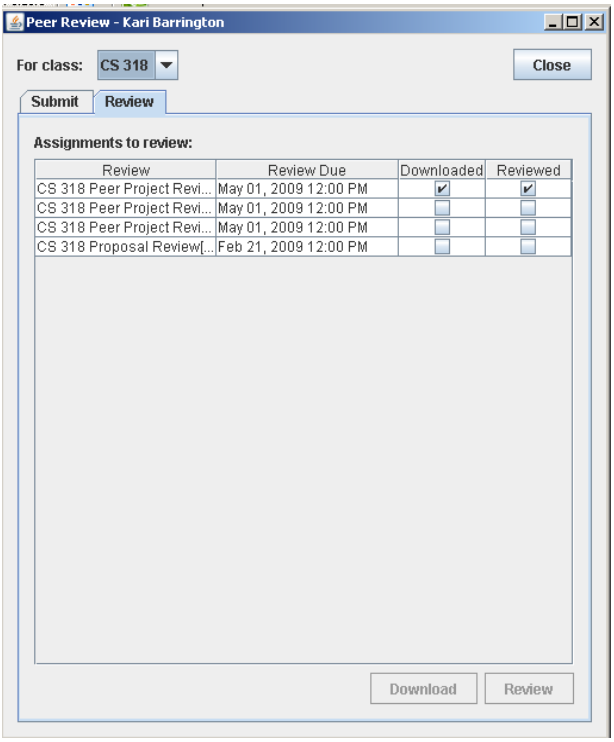

Figure 4: Student View Interface (Review tab)

The Download button in Figure 4 is disabled until after the submission due date so that the student being reviewed has the opportunity to change their submission up to the due date. The Review button is disabled until they have downloaded the submission to encourage them to view the assignment before reviewing it.

Instructor View

The Instructor View (Figure 5) allows users to open a combo box and choose from all of the classes they are teaching or have taught, all the semesters they have taught the selected class, and all of the sections for the given semester of the given class. From those selections, they can manage the classes, sections, and student enrollment from a separate dialog.

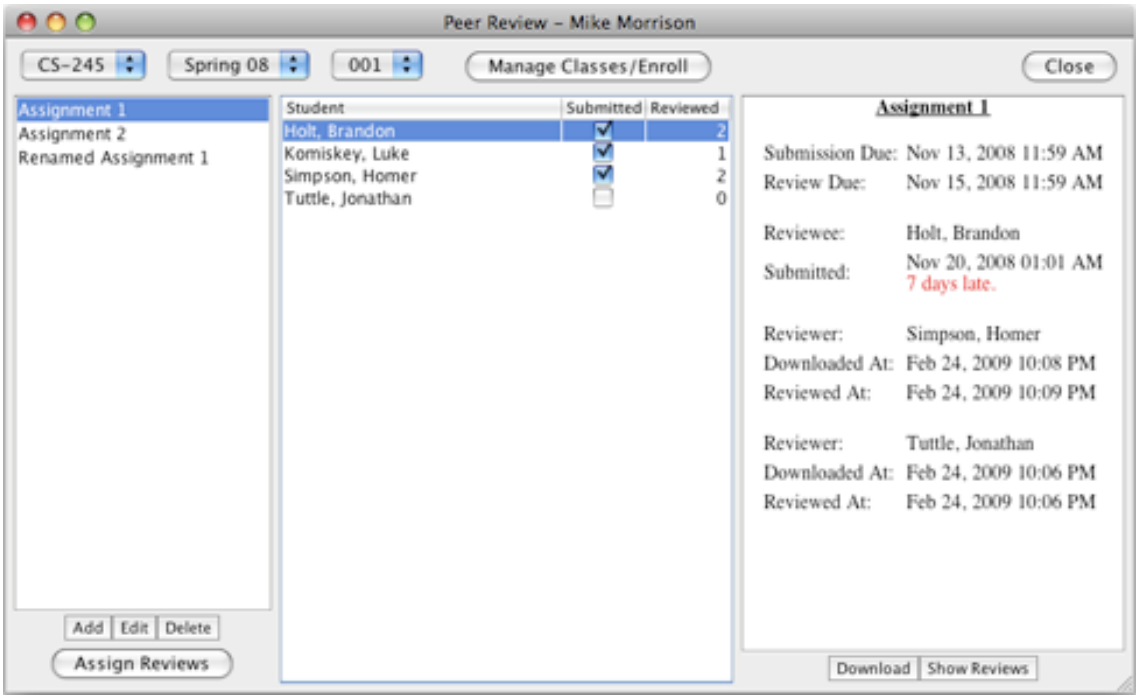

Figure 5: Instructor View

This dialog displays combo boxes allowing the instructor to select classes, associated sections, and associated students. The Manage Classes/Enroll feature allows instructors to add new students to the database, and enroll students in selected sections. In order for the students to be able to log in correctly, their username must match their username in the university directory service that is being used for password validation. At our university, instructors can obtain an auto-generated list of the students in their sections, so there is an option to import a list of students using a specific text format.

The body of the instructor's window consists of three panels: a list of assignments with options to add, edit, delete and assign reviews for assignments; a list of the students in the selected section and the status of their submissions and reviews; and, a panel with information specific to the selected assignment and student. When assigning reviews for an assignment, the instructor can manually select reviewer/reviewee pairings, or have the program randomly assign reviews. The third panel shows the assignment's submission and review due dates, when the student submitted their assignment, who the reviewers are, and when they submitted their reviews. The instructor can download the submission and view the review questions with each reviewer's answers to each question.

When adding and editing an assignment, a separate dialog appears that lets instructors specify the name, due dates, and review questions for an assignment (see Figure 6).

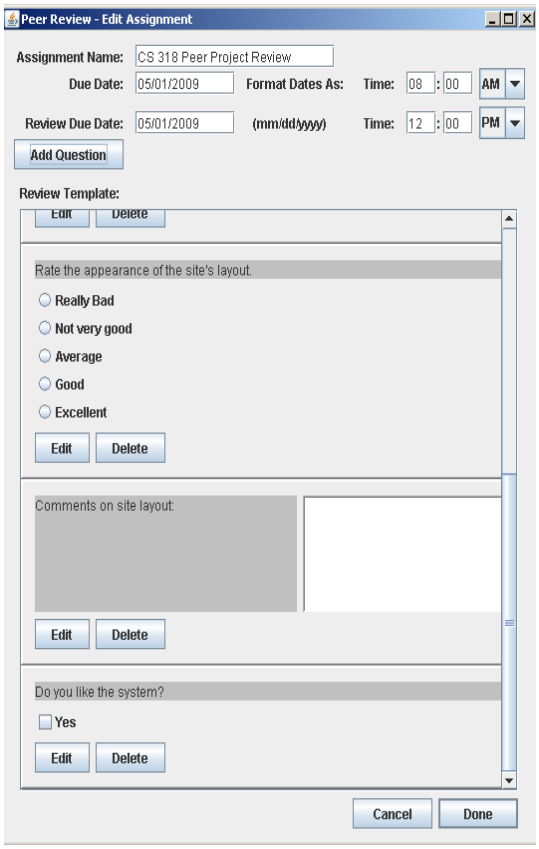

Figure 6: Dialog for editing an assignment and specifying review questions

This interface component allows the instructor to specify the question type (text, radio buttons, or check box), and create review questions and corresponding radio button or check box labels. After the instructor creates the review, he or she can easily revisit the review questions and edit them as needed.

Database Schema

Our system is built on a MySQL database that maintains various tables such as instructor, student, class, assignment, and review to list a few. The design effectively manages the large number of sections, semesters, and students that will eventually be entered into the database by minimizing repeated data. Figure 7 shows the Entity-Relationship model for the system's database.

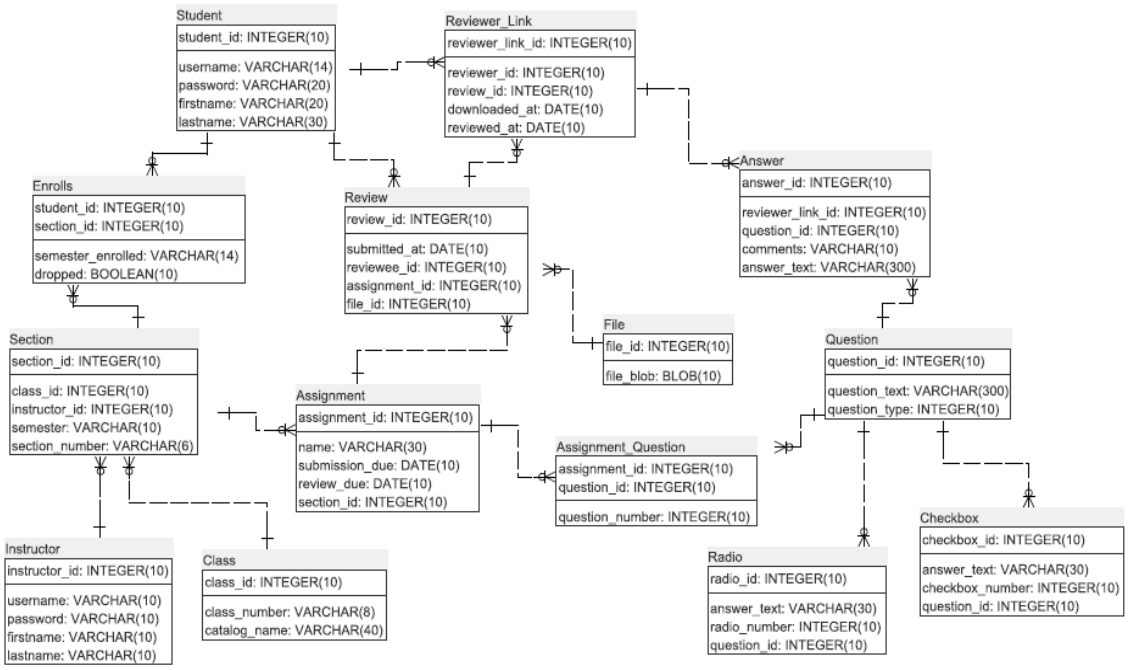

Figure 7: Database ER model

Student and instructor data exists in two separate but similar database tables that connect to a specific course section. Our database design allows us to track students that have also enrolled in a specific course section but have dropped during the semester. This makes it considerably easier for the instructor when assigning random reviewers for assignments. An instructor is allowed to insert assignments into the assignment database table for a specific course section. The assignment table stores details pertaining to the due date of the assignment and the reviews. When an assignment is created, the questions for reviewing are also stored. Our system currently allows text fields, radio buttons, and check boxes to be created and stored in their respective database tables. Once the instructor assigns the reviewers to reviewees, their student primary keys are used to connect their stored files, answers, and submission times appropriately.

Submitted assignment files for review are zipped for simplified storage as Binary Large Objects (BLOB) within the database, which reduces download time and storage. A notable example of how our database retrieves data is when a student requests a completed review done by a peer for a specific assignment. Queries are initially executed to build the specific assignment's review form created by the instructor. Using our

reviewer linking table, the review ID is found, which then links the reviewer and reviewee. The reviewer's comments on the student's work are also inserted. This functionality ensures that anonymity is maintained throughout the reviewing process.

Every interface in our system requires queried information from the MySQL database. A Java class named QueryLibrary manages all of the database queries that need to be made while a user navigates our peer review system. Each query is unique to the specific task that is needed and is stored as a Java prepared statement object that allows us to query and execute our MySQL statements multiple times efficiently.

Conclusions, Reflections, and Future Directions

The peer review system we developed successfully supports all of the basic requirements of a peer review system: submitting assignments, distributing assignments to reviewers, and administering reviews. It adds additional value over existing systems in the way it provides flexibility in managing how reviewer assignments are managed (random vs. non-random, and the number of reviewers per assignment), and how assessment rubrics are structured (using a combination of text fields, radio buttons, and so forth).

While designing the peer review system, we overcame many challenges, particularly on deciding how to design the database and how to work with JNLP technology. We went through many iterations of our database schema as we identified everything that would have to be stored for the peer review process. One issue included connecting enrolled students in a specific class section with an instructor who teaches multiple classes. After designing the class enrollment structure for the database, we had to decide on how students would submit their assignments and store them in the database, ultimately deciding on using BLOBs and a file zipping Java class. The actual review process and the relationships that a single peer review shares with a particular assignment, associated questions and answers, reviewer, and a student's file was the toughest feat to overcome. Our ER model expanded from four database tables to fourteen to manage the complex relationships that the reviews share with every element. Before we could begin the implementation of the actual interface, the ER model needed to be finalized because minor changes later in development could prove troublesome. It was a burden to test every possible query that we would require from our database before actually implementing it, but it saved us a lot of time in the end because we did.

When we started creating the actual application and user interface, we had to decide on our development platform. We wanted our program to work across different operating systems, and with our MySQL database in place already, Java was an obvious choice. We wanted the flexibility a fully fledged client application would allow, so after researching different methods we decided on using Java Web Start (JavaWS) and JNLPs to deliver the application. This proved to be more difficult than just writing a client application, jarring it, and putting it on a website, though. In order for a JavaWS application to access the user's hard-drive, it has to be digitally signed, so we had to learn the whole process of digital signatures and how Java implements it. We also ran into issues getting the JNLP to include the MySQL driver as an external library. Once the system was mostly up and

running on the client, we had to start testing it as a JNLP, which exposed more issues that were specific to applications running via JavaWS. Each of these hurdles forced us to learn a lot because most often when you start looking into a problem, you find out that the issue is far more complex than you initially thought, so we had to learn all about it before we could fix the problem. As a result, we now know the basics as well as subtleties of debugging database designs, basic web security, JavaWS application distribution, and much more we ran into along the way.

The next step is to submit the system to a rigorous testing phase, and then use it 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. This will determine if the system is fulfilling all of the design requirements and especially to see if it is enhancing students' learning. We hope that the system can be used to support a variety of academic classes, such as in English classes to review classmates' papers. If other universities or schools wanted to use the system, it could potentially be expanded and modified to allow various different set-ups so it could be installed on a variety of different campus database/directory systems.

References

1. **Nunamaker, J.F., Chen, M., and Purdin, T.D.M.** Systems development in information systems research. *J. MIS.* Winter, 1990-91, Vol. 7, 3.
2. **Morrison, J. and George, J.F.** Exploring the Software Engineering Component in MIS Research. *Comm. of the ACM.* July, 1995, Vol. 58, 7.
3. *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.
4. **Wolfe, W.J.** Online Student Peer Reviews. SIGITE 04 Proceedings.
5. *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.
6. *Electronic Peer Review and Peer Grading in Computer Science Courses.* **Gehringer, E.F.** Charlotte, NC : ACM, SIGCSE 2001 Proceedings.
7. *Automatic Assignment Managementand Peer Evaluation.* **Trivedi, A., Kar, D.C., and Patterson-McNeill, H.** s.l. : Consortium for Computing in Small Colleges, 2003.