FACILITATING PEER REVIEW IN AN ONLINE COLLABORATIVE LEARNING ENVIRONMENT FOR COMPUTER SCIENCE STUDENTS

By

Michael Allen Chasteen II

A thesis submitted in partial fulfillment

of the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

Middle Tennessee State University

May 2012

ACKNOWLEDGEMENTS

I want to thank Dr. Li for all of her help and guidance while writing my thesis, and I sincerely appreciate being given the opportunity to work on such a great project. I would also like to thank Dr. Dong and Dr. Untch for all of the knowledge and insight they shared with me throughout my work on the project, which made for a wonderful learning experience. Furthermore, I want to express my gratitude to Dr. Pettey for her efforts in establishing a recognized thesis format for the Computer Science Department. Finally, I would like to thank all of the computer science professors for their commitment to education and the knowledge they shared as well as my family and friends for their support and understanding during this time.

ABSTRACT

High student drop out and failure rates in entry-level computer science (CS) courses have contributed to a lower number of qualified CS graduates nationwide. Among various underlying causes that lead to this phenomenon, several unsatisfactory behavioral traits have been identified in CS students, including lack of motivation and combativeness towards the opinions of peers. CS educators have attempted to address these problems by developing learning tools to increase students' motivation through engaging educational activities. A better possible solution is to combine learning and social activities to create an engaging learning environment that fosters collaboration among student peers. We explore this idea by developing a peer review system as an integral component of an online social network that contains collaborative features found in popular websites, such as Facebook. Results from the controlled experiment and student feedback indicate that both student performance and sense of community increases with use of the system.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

c-account – Course Account

CMS – Content Management System

CS – Computer Science

CSV – Comma-Separated Values

GUID – Globally Unique ID

GUS – Grading Using Subversion

SVN – Subversion

CHAPTER I

INTRODUCTION

Introductory computer science (CS) courses are suffering nationwide from a high rate of students failing and dropping out [[18]]. These unfortunate trends can be attributed to a lack of motivation, persistence, and passion towards course material, procrastination on assignments, unwillingness to support or aid others, and students' combativeness towards the opinions of peers [[35]]. Educational research provides evidence that some methods, for example forming learning community and performing peer assessment, can be effective in tackling many of these problems while enhancing the learning process and making it more enjoyable for students [[24], [31]]. With the advances in computing technologies, it is of great interest to many computer science educators to expand on successful pedagogies by designing new teaching and learning tools with the purpose of better engaging students. This work develops a modernized version of the traditional learning tool, peer review, and studies its effects in improving student learning in computer programming and in enhancing students' sense of learning community.

There tends to be a strong correlation between a student's poor learning performance and his unwillingness to accept constructive criticism or inability to evaluate the work of others due to lack of knowledge. Also, students with a low sense of community are less willing to help others with their work. These trends are undesirable and will have a negative impact on students throughout their academic and practical careers. Peer review addresses these concerns and has been proven successful in many disciplines, including CS.

Peer review has produced successful results in several fields of study, for example biology, psychology, nursing, and CS. It presents students with the opportunity to learn through evaluation of peer projects and by reflection on their work [[32]], which is a skill found in a complex subdivision of the cognitive learning domain. In other words, the use of peer review strengthens the overall cognitive thinking capacity in students.

In a typical peer review process, each student reviews and receives feedback from other students. This allows the students to view alternative solutions to a problem. In the context of CS, this helps students improve their programming style and teaches the importance of proper documentation in source code. This also promotes the use of critical analysis skills and helps students learn to accept constructive criticism, as well as suggestions, from their peers. With that in mind, it has been observed that students tend to be more willing to accept criticism from their peers than from instructors. Therefore, the quality of students' programs can be improved more effectively by having them learn from their peers' programs and suggestions. Additionally, this process generates a greater sense of community because students are helping each other evolve into a more mature and skilled programmer.

It is our conjecture that combining the ideas of a proven learning tool, i.e., peer review, with a peer-friendly, collaborative learning environment can be effective in improving student learning and providing a constructive atmosphere for students to develop a sense of community. The peer review portion of the system will allow students to review their classmates' assignments, which will reinforce good coding practices and assist students in both providing and accepting constructive criticism. The environment aspect that supports the peer review process is an online social network based environment called PeerSpace [[7], [20]].

PeerSpace is an innovative online collaborative learning network developed for CS education with the purpose of bringing real-world activities into a virtual environment. One of the major benefits of this is the elimination of transaction costs, the time and effort wasted in organizing an event because of scheduling or other conflicts, found in real-life. PeerSpace's asynchronous, online environment creates a more efficient work flow, which allows students to participate in tasks such as a discussion or group assignment without the instructor or other students being present at the same time.

PeerSpace is made up of individual components, called modules, which support innovative learning methods and stimulate the development of long-term social connections for CS students. Because of these two goals, components are classified as either learning or social related modules. The learning modules provide functionality for multiple learning tools, such as a collaborative editor, test preparation, and course discussions, while the social modules support features found in popular social networking sites such as Facebook and Twitter. These features include profiles, statuses, blogs, forums, chats, and even games.

The selection of features available in PeerSpace is based on a model aimed at driving both social actions and learning activity throughout the site. This is done by promoting the use of a secondary module in conjunction with the primary module being used. Figure 1 demonstrates an example of how this concept works.

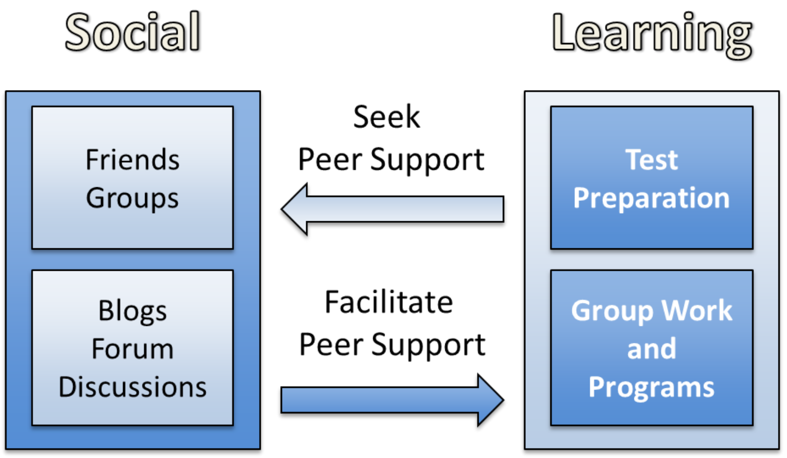


Figure 1. Social and learning modules interacting in PeerSpace

Using this model, students can seek peer support through social features when they need help with a learning activity. This kind of activity helps build peer support networks and strengthens understanding of material for all parties involved.

The following is an outline of work presented in this thesis. Chapter two presents a background of peer review. This includes a discussion on the advocated effects of peer review, the various design decisions involved in creating a peer review system, and a survey of past peer review implementations. Chapter three discusses the features and implementation of the peer review system developed in this study. The details are broken down into a description of the overall system architecture, the handling of student assignments, and the peer review process itself. Chapter four discusses the set of controlled experiments performed to evaluate the effectiveness of the peer review tool in improving student learning and students' sense of community. Finally, the thesis work is concluded and summarized in chapter five, where there is a brief discussion of the relevant and influential findings, as well as a listing of future work.

CHAPTER II

BACKGROUND

Peer Review

Peer review is a proven educational tool that stimulates greater cognitive thinking by involving students in a growing, peer collaborative learning environment in which constructive feedback and a positive attitude toward accepting and understanding ideas found in peer criticism are encouraged [[32]]. Students who participate in peer review are shown to be better readers and have developed stronger self-editing skills from the experience gained while evaluating other students' work. During this process, students also learn how to perform reviews that are helpful rather than destructive and harsh, which benefits students receiving feedback and cultivates their ability to accept criticism from others. Students who participate in peer review attain supplemental knowledge and understanding of the course material.

Peer review provides students with the opportunity to learn through evaluation of peer projects and by reflection on his or her own work [[32]] by integrating an essential, complex subdivision of the cognitive learning domain found in Bloom's taxonomy [[4]]. The ability to evaluate work involves making judgments based on a set of criteria that is internal or external to the evaluator and requires knowledge, comprehension, application, and analysis skills. The use of these skills strengthens the overall cognitive thinking capacity of the student. The most common use of peer review is found in English or literature courses where peers swap drafts or essays with each other and leave feedback in the form of corrections and suggestions to the author. However, peer review has been applied to many other fields of study such as biology, psychology, nursing, and computer science with successful results [[32]].

Peer Review in Computer Science

Peer review has been used throughout the CS curriculum with reasonably successful outcomes. The most common occurrence of peer review appears in introductory CS courses [[16], [17], [25], [29], [33], [34], [36]]. These courses typically appear in the first year and introduce programming in a language such as C++ or Java. Results in this area are very good and range from an increased sense of community [[16]] to better understanding of the material presented [[25]]. This trend has been carried over to more advanced courses such as operating systems [[11]], computer architecture [[12]], and web technologies [[14]] with equally satisfactory results. Another course peer review is frequently used in is software engineering because of its connection with the real world. This is useful for demonstrating software development processes [[37]] and encouraging students to learn from other projects [[2]]. This is also observed in a game design course [[26]] that lets students continually improve their projects based on the feedback they receive, and in a scientific writing course [[3]] where students review peers' papers. Peer review has also been implemented to supplement courses where it would not be obvious to do so. One successful example would be a compiler course [[30]] where groups can borrow code from other students to use as a framework for the final portion of the semester-long project if the group's own framework didn't turn out as well as expected. Finally, peer review can even be used in a master's level course such as parallel processing [[13]] in order to help students see alternative methods to the same problem since there are often multiple, complex ways to solve a problem in parallel.

In order to construct a successful peer review system, the design must be carefully formed to either fit specific situations and needs, or be flexible to make it useful in more situations. Regardless, a good design is needed so that the process can be effective.

Designing a Peer Review System

When designing a peer review system, there are several decisions that need to be made concerning what features the system will have:

- Manual or electronic system?
- How to assign reviewers to authors?
- Should reviews be anonymous?
- Does the reviewer have guidelines?
- How to maintain high feedback quality?

The first decision is on having a manual or electronic based system. A manual system will use the traditional method of paper and pencil while an electronic system will require the use of a computer. If an electronic system is chosen then another choice will have to be made to use an existing system or start one from scratch. This selection will depend on the resources available.

Once a system is selected, the review process needs to be defined. Will the author or reviewer know who they are paired with or will it be a completely anonymous process? Can students review each other individually [[5], [12], [13], [25], [30], [33], [37]], in groups [[2], [6], [28], [34]], or both [[3], [16], [17], [29]]? Should certain students be matched with others to generate higher quality feedback for authors? Finally, how will the reviewer assess the assignment? Some options available are a simple checklist [[34]], a guided rubric as a web form [[14]], or a free format text response box for comments [[36]]. It’s an important choice to make because most students have no experience being a reviewer.

It is also important to consider how students will react to the form of review they are assigned to complete because there will be a lack of motivation in certain cases. In other words, what will persuade students to take the review process seriously? A grade could be assigned for completion of a review, but the quality may still suffer. Should the reviewer’s feedback affect the author’s grade? Some more possibilities would be for the instructor to analyze each review individually or have the author rate the review he or she received.

Procedure of Peer Review

The entire peer review procedure must be well laid out in advance so that the implementation can match it. This means everything must be planned from the initial assignment of a project to the submission of a completed review or the last action required. A possible layout of the procedure in stages could look like this:

- Stage 1. Create Assignment
- Stage 2. Distribute Assignment
- Stage 3. Assignment Deadline Passes
- Stage 4. Create Review Assignment
- Stage 5. Distribute Review Assignment
- Stage 6. Review Deadline Passes
- Stage 7. Grade Assignment and Reviews
- Stage 8. Return Grades to Students

While these may be typical steps found in the peer review process, they can vary greatly depending on what is desired. Some systems may need to support an additional stage allowing students to submit revised work. On the other hand, if a system needs to support individual review followed by group review then the entire structure may change [[16], [17], [34]].

Measuring Results

After all of the decisions regarding the system and procedure have been made, one needs to plan on how to evaluate the system in order to understand how it has affected the students. This is done by examining the objectives of peer review and monitoring the social and learning effects in the students and community. For example, when studying whether peer review enhances understanding of course material, the results can be gathered by observing grades and comparing them with a control group. Other benefits such as feeling of community may need to be assessed qualitatively, which might require the use of surveys or interviews [[2], [3], [5], [12], [16], [25], [28], [29] [30]].

The next section covers the design decisions of various peer review methods and their implementations. First, an overview of the basic methods taken by many systems is given. Then, a closer look at four systems of particular interest is presented.

Examine Peer Review Methods

Peer review has been implemented in many different ways. The traditional method of pencil and paper has been the most common, and it has produced positive results [[2], [16], [34]]. However, this method comes with many disadvantages for practical use. Paper based peer review takes a lot of time to set up and is not always reliable. One of the most common approaches requires students to bring enough copies of their source code to class to be reviewed. If a student forgets to print off his or her code then valuable time is wasted while the student goes to print the code [[17]] or experience is lost because the student cannot participate. Even alternative methods where the instructor distributes the source code can cause a problem if there are students absent. It is often not practical for the instructor to wait for a time where every student shows up prepared before assigning a peer review. Moreover, the steps in the review process are difficult to manage, and the review quality is harder to control.

Instructors need to evaluate the reviews to make sure they were not only completed but done so appropriately before a grade can be assigned and returned to the author. Since students usually have little or no experience in being a reviewer or do not understand the material themselves, they are often unsure of their ability or feel less confident when reviewing another student's work. This would greatly reduce the effect on how much the reviewer and the author learn from this process. However, even if all of the students are expert reviewers and the peer review process is not graded or checked by the instructor, the time spent performing a peer review in class would still take up time the instructor could be lecturing.

The process gets more complicated when a double-blind peer review is desired. The students' code can no longer easily be collected and distributed back out because there would be no names to identify the reviewed code when the instructor needs to grade or hand them back. While there are ways around this, the extra effort needed to match students together and create an anonymous review process is an added cost in time and course management. In summary, manual peer review may be a viable option in smaller classes as the complications involved are minimized, but in most cases a better technique for peer review is needed in order to suit larger [[34]] and even normal size classes. In the last decade there has been a drive to move the peer review process into a virtual environment [[3], [5], [6], [12], [14], [17], [25], [26], [28], [29], [37]] that allows each step to be completed electronically with the intent to minimize or even fully eliminate the drawbacks found in manual peer review and increase the benefits for students. Many attempts at online peer review make use of existing technology in order to reduce the amount of work that has to be done to both save time and ensure the possibility for others to implement.

One approach involves using email to submit source code and return completed reviews [[36]]. This small improvement certainly alleviates some of the problems with manual peer review because there are no physical copies of source code to handle. But it still has most of the original problems. In particular, time management is very important to the instructor since computer science courses can require a lot of work and preparation behind the scenes. Tracking down emails and organizing them in a way that peer review assignments can be sent out to the correct students takes a fair amount of time and effort and cannot be done easily. In fact, it may only be a minor advancement from dealing with the physical hardcopies themselves as it does solve the problem of students receiving their peer review assignment at the same time.

An additional effort to virtualize the peer review process includes the use of web servers to host assignments as web pages [[11], [12], [37]]. This puts the responsibility back into the students’ hands to upload their work in an HTML format. This method requires knowledge of technologies such as HTML and FTP. While it is fairly common for computer science students to know these, it is not guaranteed. This leads toward much of the same problems associated with paper and pencil approaches.

The next section takes a closer look at four systems designed for peer review. They are chosen as representative examples of approaches using the traditional method, custom implementations, or an implementation based on pre-existing software.

Review of Existing Systems

Implementing peer review with paper and pencil can be successful under the right circumstances. Trytten's traditional method is group based and involves a double-blind review process guided by a rubric [[34]]. Students review as a group during the starting portion of their lab time, monitored by the teaching assistant(s). Trytten mentioned groups argued over review decisions initially even though the rubric was a checklist in the form of true/false questions. This was solved by letting groups throw a question out if it couldn't be answered unanimously. In this design, members of a group review the same work, a student submission selected by the instructor and possibly modified for the purpose of peer review. Although results were not published for this method, Trytten mentioned that even though it was a small exercise affecting a small part of the overall grade, it is possible that the students may have a change in attitude toward the course's activities [[34]].

Gehringer's Peer Grader (PG) system was the first fully electronic system to be used for submission and review of assignments [[11]]. Students submit assignments as web pages to PG. The system features a double-blind review process in which instructors can purposefully or randomly pair students together. Reviewers leave feedback for the author based on a rubric, and students later have a chance to rate the completed reviews

[[12]]. Reviews and ratings are used as grades to ensure that the peer review process is taken seriously. This is understandable, but the method is also criticized for not being helpful because it is peer grading, not peer review [[30]]. Gehringer also admits the set-up of and steps in peer review are not easy to get right [[11]]. Among the list of challenges are course and student management, keeping work anonymous, and handling deadlines. However, even with the challenges present in the early versions of PG, the results were still positive.

In an effort to improve an already verified method, Hundhausen et al. moved their paper based peer review to an online studio-based learning environment (OSBLE) [[17]]. With this in mind, the system itself was named OSBLE. The system supports and uses both individual and group review techniques. Reviews are done individually before the group comes together to create one review out of all the members' reviews. Students can place comments on specific lines or line ranges, along with marking the severity of the problem, and then drag them over to the group review if the group likes the assessment that was made. The students can do individual reviews from any computer, but the group reviews are done during lab time under a moderator. After the review process, students can view their feedback and resubmit their work. It is important to note that OBSLE does not support any kind of anonymous reviewing and does not use a rubric like most implementations. While this is in contradiction with most methods [[6]], results from the system seem to have been positive. Overall, Hundhausen et al. report that the transition to an online environment sped up the review process and made reviewing work a more efficient process for students [[17]]. However, OSBLE is more than just a peer review tool as it supports basic course management. Future work entails adding more social features to OBSLE such as profiles and giving authors a way to rate the feedback they received. Even though this version of OBSLE doesn’t support anonymous review, future versions of it will.

Harri Hämäläinen et al. originally experimented with an open-source peer review system called MyReview [[22]] for a master’s level course [[13]]. They concluded that advanced students could see a benefit in looking at each other’s code. But the use of a system not designed with code review in mind was too distracting. Therefore, Ville Hyyrynen et al. (including Hämäläinen) decided to create a custom implementation of a peer review system called MyPeerReview [[21]]. This design is different from the previous examples because it is built off of an existing content management system (CMS) called Drupal [[8]] that already has basic features including users, profiles, and navigation tools. Because of this, only a module was needed to add support for assignment submission and peer review. The system itself does not support anonymous peer review, but the students were asked, not forced, to remove identifying information from their files before making a submission. Reviews are done individually and guided by a rubric constructed inside Drupal. In reference to the procedure, instructors have to manually “open” an assignment for students to submit work, and put it on “hold” when the deadline is reached. There were not a lot of features implemented in MyPeerReview initially, but the results were positive. Students felt like peer review was a worthwhile assignment, and the process of submitting and reviewing assignments went smoothly. However, several improvements were suggested, including adding a rating system, automated timing for deadlines, and using a rubric more similar to what the instructor uses.

Design Ideas

The knowledge gained from examining the existing systems is a valued resource when making decisions to develop a new peer review system in this work. Since electronic peer review is proven to be a more efficient process than paper and pencil, the options are narrowed down to using an existing system or designing a custom implementation. Realizing that using an existing system would have the benefit of saving time, the idea is more feasible than building one from scratch. At the same time, custom-built systems are more likely to support more desired features because they are created for a specific purpose. To get the advantages from both sides, the system can use the existing code for the foundation of the system and new code for the peer review process itself. Similar to the MyPeerReview system [[21]], a CMS will be used to provide the system with ways to handle users, groups, and other objects created for the purpose of peer review. The CMS chosen is an online social network engine, called Elgg [[9]]. Peer reviews will be done individually to support asynchronous participation and generate more feedback for students. When reviewing an assignment, students should have access to a rubric to guide them and keep reviews objective [[1]]. This rubric will contain the same criteria the instructor or teaching assistant uses to grade the work [[14], [17]]. Students should be able to assess and review work by grading and commenting on this criteria. The assessment portion serves as a quantitative means of peer review, and gives the students experience in evaluating an assignment’s overall worth. The comment boxes act as a medium for qualitative feedback in the form of a personalized message to the author regarding the perceived quality of the work, which allows the reviewers to improve constructive feedback skills and authors to accept criticism from others. Even though students are “grading” each other, the given grades will have minimum effect on the actual grade received for the assignment. Therefore, a rating process is needed after the review to ensure the reviews are taken seriously. The author will be able to rate all received feedback in a constructive manner. This serves as quality control for the instructors and allows the students to build a reputation as a reviewer. Both the reviewing and rating process are double-blind to prevent any biased results. Instructors will be able to create and distribute both the assignments and the peer review assignments to students through the same system, and deadlines will be handled automatically by the system.

Pairing students for peer review is to be done randomly so that it ensures fairness for both the author and the reviewer. Even if an instructor has time to pair students manually, it is challenging to decide the best ways to pair students with work submitted. This is because student performance can fluctuate through the course as the topic changes. Furthermore, if a good student is consistently forced to review work that is below par then he or she will miss out on the aspect of peer review that allows the reviewer to see alternative, working solutions. Likewise, a student who always reviews good work will not improve his or her skills in providing constructive criticism. Therefore, a random assignment for each project will allow students to see work of different levels in quality.

Instructors should be able to monitor the entire peer review process. As soon as the peer review assignment has been distributed to the students, the instructor needs to be able to see who each student is reviewing. Similarly, when a review is completed it should immediately be available for the instructor to view. The instructor should have access to several other administrative features including a way to extend or override reviews and export the peer review assignment to an external system. These additional features will be covered in more detail in the next chapter.

Implementation Discussion

The peer review procedure should have a standard yet flexible layout. In a typical scenario, an instructor will create an assignment for students to submit, and after the deadline a peer review assignment will be assigned. After that, students can perform reviews and rate feedback while the instructor monitors the progress. This process is shown in Figure 2. Notice that creating a peer review assignment can be done at any point in time between creating the original assignment and assigning the peer review assignment.

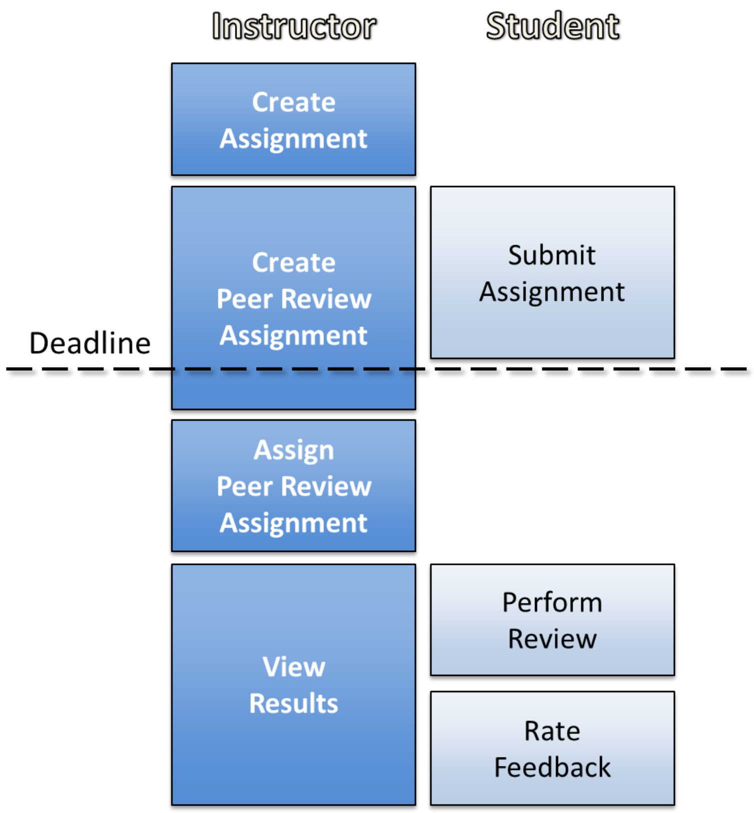


Figure 2. Standard peer review procedure

There are additional events that can happen and paths that the peer review procedure can take. The assignment of review categories that govern which students will be used in the review process and how they participate can be done at any time before a peer review assignment is created. This will be explained in more detail in the next chapter. Also, there may not be a deadline for an assignment, which means students can continue submitting files after the review process. This can be used to allow students a chance to resubmit their work. For all of this to work, the data has to be managed properly in the Elgg CMS.

Elgg

The Elgg CMS runs on a unified data model that is built on atomic units [[10]]. These units are called entities. There are four main specializations of entities to represent sites, users, groups, and objects. All specialization types inherit an entity structure that includes a globally unique ID (GUID) and owner ID, as seen in Figure 3.

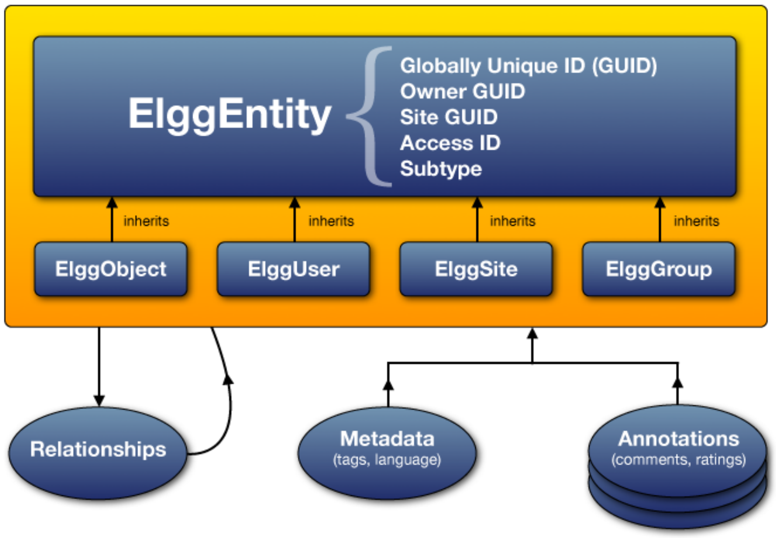


Figure 3. Elgg data model

The GUID is important because it is the entity’s key, and the owner ID specifies who can make changes to that entity. It is also worthwhile to mention that entities have an access ID which prevents unauthorized access to viewing or handling data that is protected in any way. The subtype of an entity is used by the site and custom modules to specify what kind of data the entity holds. An entity with the subtype “message” would indicate the entity contains a message, possibly from one user to another.

Entities can also have relationships with other entities. For example, a user can be a member of a group. Furthermore, entities can have multiple relationships of the same or different type. This allows a complex association of entities to be formed that supports a high level of customization and increased extensibility for developers.

Entities also have the ability to support metadata and annotations. Metadata can be attached to any entity to describe it in more detail with the purpose of making it easier to find or to store sensitive information. An entity that contains a message might have a piece of metadata attached to it that holds the subject of the message. Annotations are slightly different in the sense that they usually contain additional information related the entity. A possible annotation to a message might be the recipients given categorization of the message content which allows organization of his or her inbox. The categorization is not necessarily part of the message, as the recipient could have labeled it as important or not important depending on his or her perception of the content.

Elgg provides an entire library of methods that manage entities. These library methods make development easier and greatly speed up the development process. New modules can be implemented without the need for additional resources. Therefore, Elgg is selected as the underlying framework for the peer review system.

PeerSpace

The peer review design will be part of a larger system called PeerSpace, which is an online collaborative learning environment with many tools that support the development of social networks and peer learning. However, in this thesis, only the peer review portion will be presented. Therefore, references to PeerSpace essentially refer to the Elgg CMS that contains the peer review system. More information about PeerSpace can be found in [[7]], [[19]], and [[20]].

The next chapter presents the features of the peer review system and discusses the implementation details. First, an overview of the system architecture and the overall structure of PeerSpace are discussed. Then, the interrelation of the two main components in the peer review process is covered. Finally, the features for the instructors and students will be presented along with the implementation details in both the Assignment and Peer Review module.

CHAPTER III

METHODOLOGY

Peer review is an educational tool that may be used to enhance student learning and students' sense of community. The peer review system developed in this study is designed to give students the opportunity to learn both through evaluating peer projects and by reflecting on their own work inside a peer-friendly environment.

The layout of this chapter is organized in the following manner. First, an overview of the peer review system is presented. Next, the set of features provided to instructors and students is discussed. Finally, the implementation details are given.

<u>Overview of the Peer Review System</u>

System Architecture

The overall system architecture includes the PeerSpace server and a separate server that stores students' programs. The PeerSpace server contains a series of modules divided into social, learning, and administrative categories. The overall design of the system architecture is shown in Figure 4.

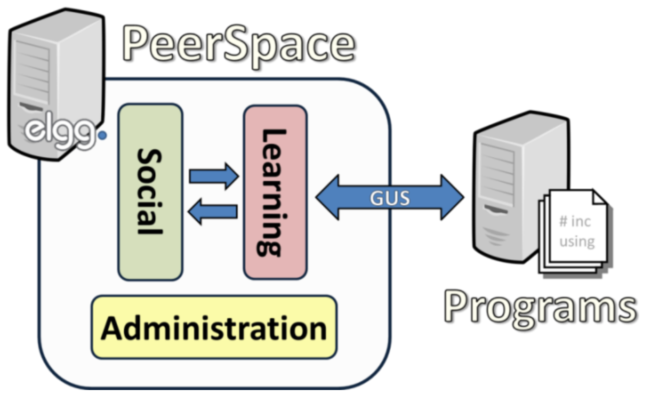


Figure 4. Overview of the system architecture

The two modules developed for peer review, the Assignment and the Peer Review modules, belong to the learning category. Both of these modules need access to the external server that holds students' programs so they can submit and retrieve assignments for the peer review process. This server runs a version control system called subversion (SVN) which accepts assignments and allows them to be checked back out, thus making it a crucial part in the peer review process.

System Component Interaction

Figure 5 illustrates the overall design of the peer review system and how the components in the system interrelate. To handle the different needs of the two types of users for this system, i.e., the instructors and the students, two sets of features were developed in each of the two modules, as shown under the "Student side" and the "Instructor side" columns in the figure. Both the Assignment module and the Peer Review module interact with the SVN server through the Grading Using Subversion (GUS) system. More details of GUS will be discussed in the implementation section.

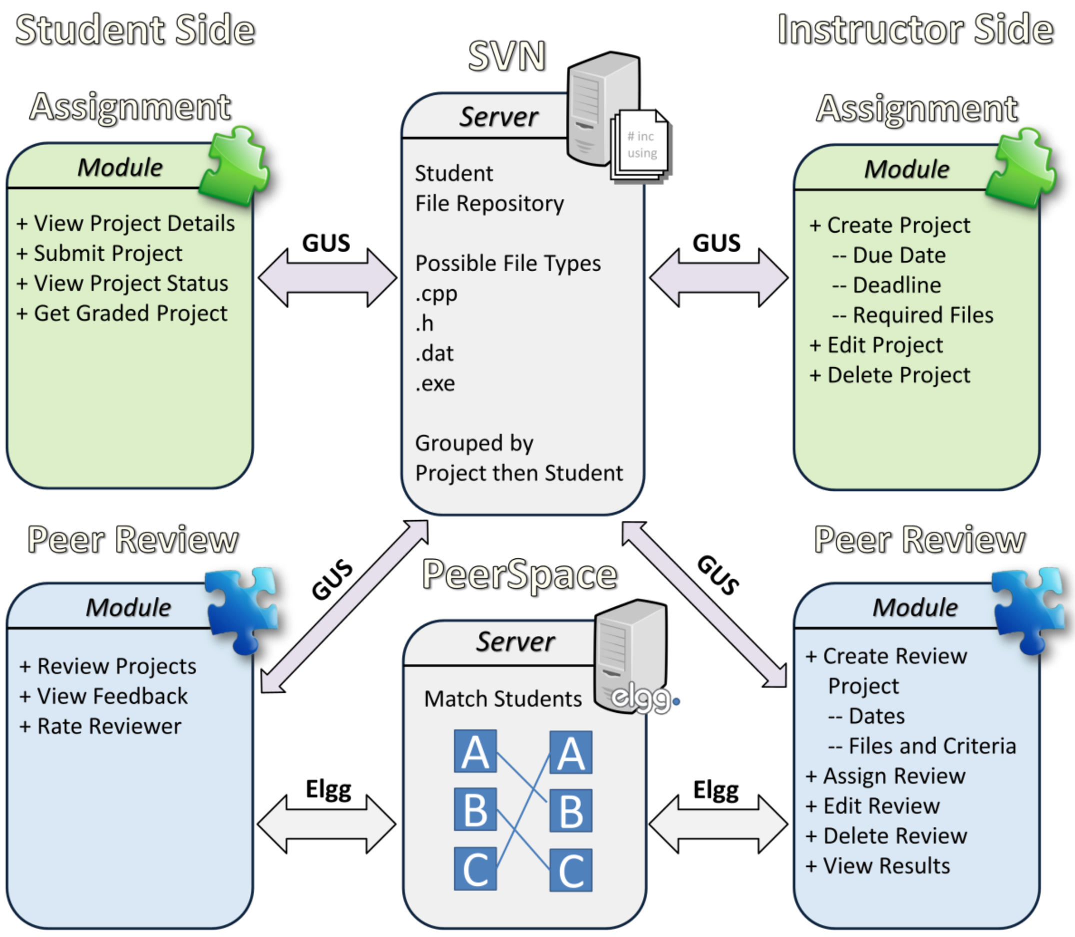


Figure 5. Interrelation of system components

The Assignment module handles the process of creating and submitting assignments. Instructors can create assignments through the instructor side of the Assignment module which will allow the SVN server to accept student submissions for those assignments. After this, the student side of the Assignment module will poll the SVN server in order to get a list of assignments the student can submit. A student can then submit one or more of those assignments to the SVN server where they are stored for the instructors to grade. Additionally, students can retrieve graded assignments on the SVN server through the Assignment module.

The Peer Review module allows the students to review their classmates' assignments. First, an instructor must create a peer review assignment using tools provided for the instructor in the peer review module. After the peer review assignment has been created, it can be assigned to the students. When the assignment process takes place, the peer review module uses GUS to retrieve the submitted assignments from the SVN server. Copies of these assignments are stored on the PeerSpace server for the reviewers to view or download during the peer review process. All completed peer reviews are stored in the Elgg database.

System Design Details

Assignment Module and Features

To facilitate peer review, PeerSpace needs to provide the students with access to each other's program after they are submitted. It is ideal for the students to submit their programs and perform reviews through the same system, and the Assignment module is designed for this purpose. It provides a user-friendly web interface for the students to submit their programs according to the specifications set by the instructors, and the assignment repositories used in the submission system allow for convenient and easy retrieval of student assignments in the peer review process.

Students are able to log into PeerSpace from any computer and submit their assignments through a web browser. Instead of performing program submission using a separate system, the Assignment module brings another aspect of the work inside PeerSpace. This allows the students to perform multiple tasks in one location rather than having to visit multiple locations for different tasks. Also, since students have to get on

PeerSpace to submit their programs, the Assignment module also serves as a magnet tool giving the students more opportunities to participate in PeerSpace activities.

Features Provided For the Instructors

The instructor side of the Assignment module consists of the four basic functions of persistent storage used in computer science. Instructors may create, retrieve, update, and delete assignments for any of his or her courses. The most important function is the ability to create an assignment because once an assignment is created it is automatically assigned to the students.

Creating Assignments

For each assignment, an instructor can provide basic information such as the title and description of the assignment as well as details concerning the submission of the assignment. An Instructor can specify number of files, type of files, and names of the files to be submitted for an assignment. This ensures that all submissions have the same required files. In addition to the required files, extra files can be submitted without the need for the instructor to specify them explicitly. An example would be when students choose to supplement their submission with additional files for extra credit.

Instructors can also specify a due date and deadline for the assignment. The due date informs the student of when the instructor expects the assignment to be completed and submitted. The deadline specifies the latest the instructor is willing to take late submissions of the assignment. Since many instructors have their own preferences in accepting assignments, the due date and deadline are not required when creating an assignment. If neither is provided, then the assignment will remain open until the end of the semester. Alternatively, only the due date can be provided. This would automatically set the deadline to the same as the due date. However, if an instructor is willing to accept late submissions, then a deadline should be provided that is later than the due date.

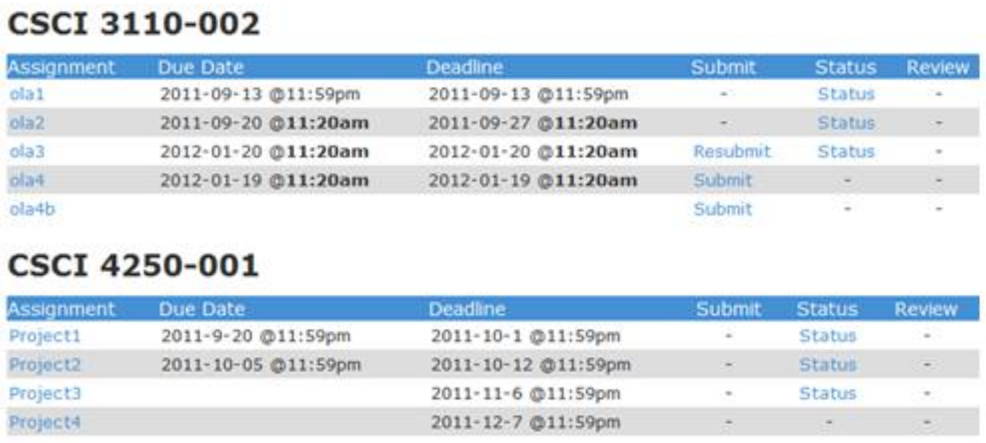
Editing and Deleting Assignments

When an assignment needs to be updated or corrected, the instructor can edit it to make the needed adjustments. For example, if an instructor wants to extend the due date of an assignment, then he or she can simply change the time for the due date. These changes, once submitted, will take place immediately. Therefore, assignments can be updated at any time. Moreover, if an assignment needs to be deleted, it can be easily removed by erasing the given assignment entry.

Features Provided For the Students

Viewing Assignments

The student side of the Assignment module allows students to submit their assignments. Students can view all of their assignments organized by course on one screen. Each assignment entry displays the title, due date, and deadline. The title of the assignment also functions as a hyperlink to the assignment’s description if it was provided by the instructor. Students may submit their work to any assignment, multiple times, as long as the deadline has not passed. Meaningful hyperlinks appear where appropriate to specify student actions. The submission hyperlink is one of these cases. If a student has made no previous submissions to an assignment, then the action word for the hyperlink will be “Submit”. But if the student is making another submission then it will appear as “Resubmit”. Figure 6 shows a view for a student who is enrolled in two courses and has assignments from both.



CSCI 3110-002

Assignment	Due Date	Deadline	Submit	Status	Review
ola1	2011-09-13 @11:59pm	2011-09-13 @11:59pm	-	Status	-
ola2	2011-09-20 **@11:20am**	2011-09-27 **@11:20am**	-	Status	-
ola3	2012-01-20 **@11:20am**	2012-01-20 **@11:20am**	Resubmit	Status	-
ola4	2012-01-19 **@11:20am**	2012-01-19 **@11:20am**	Submit	-	-
ola4b			Submit	-	-

CSCI 4250-001

Assignment	Due Date	Deadline	Submit	Status	Review
Project1	2011-9-20 @11:59pm	2011-10-1 @11:59pm	-	Status	-
Project2	2011-10-05 @11:59pm	2011-10-12 @11:59pm	-	Status	-
Project3		2011-11-6 @11:59pm	-	Status	-
Project4		2011-12-7 @11:59pm	-	-	-

Figure 6. Student view of the assignment page

Submitting Assignments

The submit action provides an easy to follow interface for students to turn in their work. There is a clearly labeled submit field provided for each individual file required by the instructor (Figure 7). These fields have an auto check feature to make sure the file the student selected is the file that belongs in that field. This ensures that all the correct files are provided to make a successful submission. Additional file submit fields are also provided so students can submit extra files.

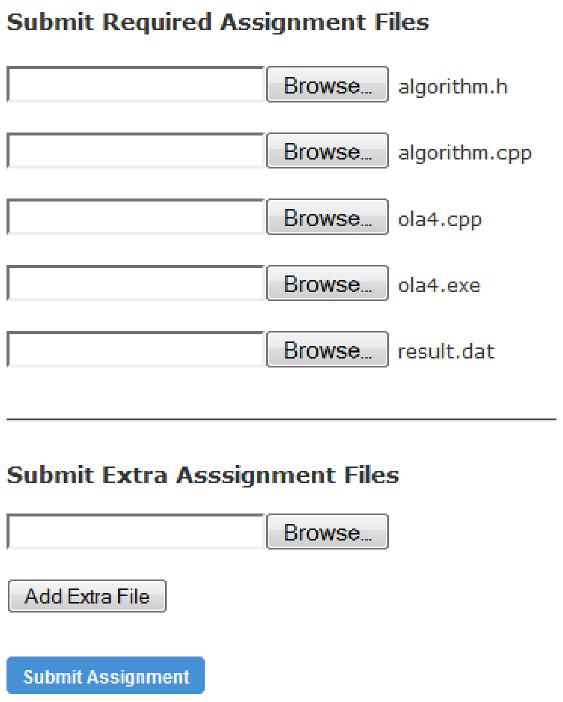
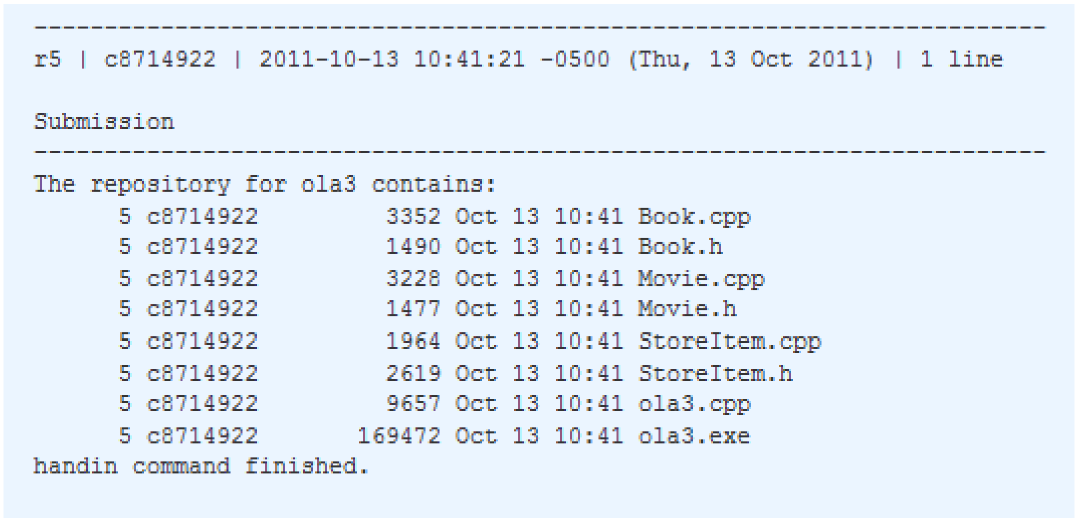


Figure 7. Student view of an assignment submit form

Assignment Status

After a student makes a submission, a status screen appears to inform the student if the files were accepted successfully or if there was a problem. The status of an assignment can be accessed at any time once a submission has been made. An example of the status screen can be seen in Figure 8.



```
------------------------------------------------------------------------
r5 | c8714922 | 2011-10-13 10:41:21 -0500 (Thu, 13 Oct 2011) | 1 line

Submission
------------------------------------------------------------------------
The repository for ola3 contains:
      5 c8714922         3352 Oct 13 10:41 Book.cpp
      5 c8714922         1490 Oct 13 10:41 Book.h
      5 c8714922         3228 Oct 13 10:41 Movie.cpp
      5 c8714922         1477 Oct 13 10:41 Movie.h
      5 c8714922         1964 Oct 13 10:41 StoreItem.cpp
      5 c8714922         2619 Oct 13 10:41 StoreItem.h
      5 c8714922         9657 Oct 13 10:41 ola3.cpp
      5 c8714922       169472 Oct 13 10:41 ola3.exe
handin command finished.
```

Figure 8. Assignment submission confirmation screen

Assignment Feedback

Students can also retrieve graded assignments through the Assignment module. This feature allows instructors to provide feedback electronically. The feedback can be in any format, and the student can download it to their computer through the web browser he or she is using.

Assignment Module Implementation

GUS and SVN Server

The Assignment module works by using GUS to communicate with the SVN server. Inside the SVN server, each instructor has a directory structure, which contains all of his or her current courses. Inside each course directory are the students' repositories. Every student in the course has a repository which is used to hold submitted assignments. Students enrolled in multiple courses will have a repository for each course, all password protected to prevent unauthorized access. The SVN accounts used for the repositories are based off course accounts (c-accounts) created in the CS account management utility. Every student is assigned a c-account which comes with an account number and password that is associated with their CSCI course. When submitting an assignment using GUS, this account information must be provided so that the submission is committed to the correct student’s repository.

GUS features two extended SVN commands that allow assignments to be submitted and retrieved in a controlled manner. GUS’s “handin” command provides the functionality to submit files into the appropriate repository. This command requires arguments for the location of the appropriate repository, the username and password, the assignment name, and the files to be submitted in order to successfully store the files in the student’s repository. GUS also has a “handback” command that retrieves files from a repository. The arguments are similar but do not require a list of files.

GUS is originally a command line application, and the “handin” and “handback” commands are available for students to turn in and retrieve work from the command line. It was incorporated into PeerSpace so all of the assignments, regardless if they were from a control or experiment group, could be stored in the same grading repository for the instructor. In addition, GUS has already proven to be a reliable system and can be counted on as a backup if needed. It also provides instructors with a means of accepting assignments before PeerSpace accounts are created and given to the students.

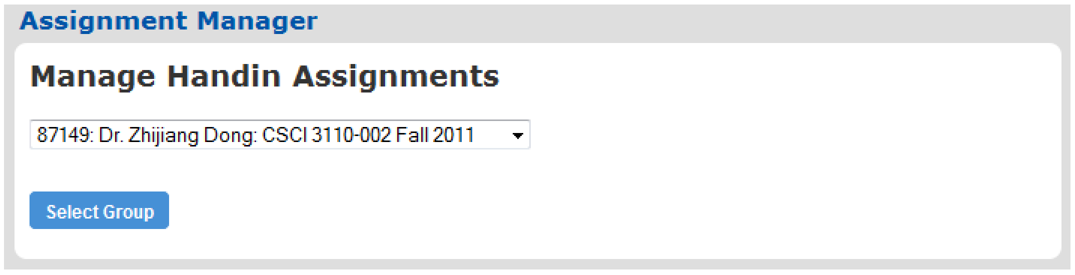
Instructor Side

Course Selection

The instructor side of the assignment manager has a minimal interface. When accessing the assignment manager, the instructor must select a course to manage from a list containing all his or her active CSCI courses. These courses are gathered by an Elgg function call which grabs all of the instructor's active course groups. The code required to perform this action is shown in Figure 9, and the display of the course selection page can be viewed in Figure 10. The next section details the assignment editor which the instructor has access to after selecting a course.

```
Fetch Instructor's Active Course Groups:

$owner = (isadminloggedin()) ? NULL : $user;

$metadata = array(array('name'  => 'group_type',
                        'value' => 'crn'),
                  array('name'  => 'status',
                        'value' => 'active'));

$options = array('types' => 'group',
                 'owner_guids' => $owner,
                 'metadata_name_value_pairs' => $metadata,
                 'limit' => NULL);

$user_groups = elgg_get_entities_from_metadata($options);
```

Figure 9. Retrieval of instructor's active courses


Assignment Manager

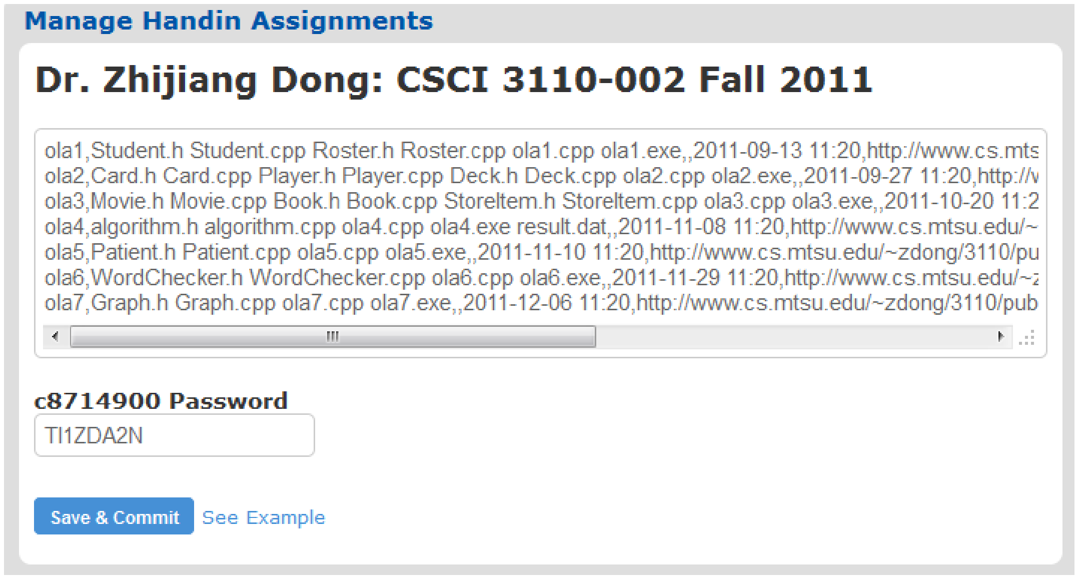
Manage Handin Assignments

87149: Dr. Zhijiang Dong: CSCI 3110-002 Fall 2011

Figure 10. Course selection form

Retrieving Assignments

In order to view or edit assignments for a selected course, the assignments must be retrieved from the instructor's repository. Therefore, an SVN checkout command is performed to retrieve the assignments file. This file is then loaded into the assignment manager to allow instructors to make changes through the editor, implemented as an HTML form containing a text-area field. The instructor can use this input field just like a basic text editor to make any needed changes to the assignment file. The file itself is formatted as a comma-separated values (CSV) file so that GUS can easily parse the details of the assignment. An example assignment file, as it would appear when loaded into the editor, is shown in Figure 11.


Manage Handin Assignments

Dr. Zhijiang Dong: CSCI 3110-002 Fall 2011

```
ola1,Student.h Student.cpp Roster.h Roster.cpp ola1.cpp ola1.exe,,2011-09-13 11:20,http://www.cs.mts
ola2,Card.h Card.cpp Player.h Player.cpp Deck.h Deck.cpp ola2.cpp ola2.exe,,2011-09-27 11:20,http://v
ola3,Movie.h Movie.cpp Book.h Book.cpp StoreItem.h StoreItem.cpp ola3.cpp ola3.exe,,2011-10-20 11:2
ola4,algorithm.h algorithm.cpp ola4.cpp ola4.exe result.dat,,2011-11-08 11:20,http://www.cs.mtsu.edu/~
ola5,Patient.h Patient.cpp ola5.cpp ola5.exe,,2011-11-10 11:20,http://www.cs.mtsu.edu/~zdong/3110/pu
ola6,WordChecker.h WordChecker.cpp ola6.cpp ola6.exe,,2011-11-29 11:20,http://www.cs.mtsu.edu/~z
ola7,Graph.h Graph.cpp ola7.cpp ola7.exe,,2011-12-06 11:20,http://www.cs.mtsu.edu/~zdong/3110/pub
```

c8714900 Password: TI1ZDA2N

Figure 11. Assignment editor

Managing Assignments

To create a new assignment, the instructor can simply add the details for the assignment on an empty line. Editing assignments only requires making adjustments to the parts that need to be changed. Similar to creating an assignment, the instructor can delete it by removing the line containing the assignment from the file. The instructor can also comment out the line containing the assignment by placing the special character ‘#’ at the start of the line. This can be used to temporarily disable an assignment or leave notes in the assignment file. Instructors who use the same assignments may prefer to use the same file from year to year, and this allows them to do so without recreating it every time, even though some editing may be required to adjust dates.

All changes to the assignment file must be submitted back to the instructor’s repository in order for it to take effect. Therefore, the instructor must provide the password for his or her account in order to submit the changes through GUS. However, the Assignment module will remember the password so it only needs to be entered once.

Student Side

Displaying Assignments

The student side of the Assignment module retrieves all of the assignments that are assigned to a particular student, and allows that student to submit his or her work. In order to do this, the most recent revision of the assignment file for each of the student’s courses must be retrieved from the SVN server. Therefore, all the active courses a student belongs to must be fetched using the same method found in Figure 9, but with different metadata constraints. Once all of the assignment files are retrieved through GUS, each one is parsed with the purpose of extracting the details of each assignment. This includes the title, description hyperlink, due date, deadline, and required files. Since the assignments are organized in a CSV file, it is easy to collect all of this information. Figure 12 shows the section of code used to parse the assignment file.

```
Parse Assignment File:

for( $file as $line )
{
     if( $line[0] != '#' )
     {
          $details = explode(',', $line);
          $name = $details[0];
          $files = explode(' ', $details[1]);
          $due_date = $details[2];
          $deadline = $details[3];
          $url = $details[4];

          ...
     }
}
```

Figure 12. Parsing of assignment file

To determine if an assignment has been graded, a separate call to the SVN server using “handin” is used. The output of this command provides a list of assignments with an indicator if they have been graded. Once this is analyzed, the Assignment module has everything it needs to display the student’s assignments.

The assignments displayed are organized by course and ordered by their position in the assignment file. Each course the student is enrolled in has its own table of assignments, and every assignment for that course has its own entry in the table. An assignment entry contains the title of the assignment, which links to the description of the assignment, as well as the due date and deadline if they exist. Assignments that do not have either time specified will have nothing displayed. If only the due date or deadline is provided then that time will be used for both dates. The submit functionality is only enabled on assignment entries that have no deadline or have a deadline that has not passed. This is determined upon closer inspection of the dates in the assignment file after it has been parsed. The status of an assignment is only displayed and accessible once a successful submission has been completed for the assignment so that students do not assume the assignment has already been turned in. Additionally, the hyperlink that allows a student to retrieve feedback for an assignment only appears on the assignment entry if the gathered information indicated the assignment has been graded.

Submitting Assignments

When students attempt to submit their work for an assignment, the Assignment module enforces the requirements found in the assignment file. The hyperlink that allows students to submit assignments only shows up if the student is turning in his or her work on time, and the files being submitted must match the files the instructor has specified. Since the Assignment module is web-based and security is a concern, all of the enforcements in the Assignment module are also enforced by GUS.

The status of an assignment submission is displayed after a student attempts to submit their work and can be accessed at any time after a successful submission is made. The status of an assignment is displayed for the student after a submission is attempted by returning the output of the “handin” command. This will inform the student whether the submission went through successfully or not. A student can check the status of any assignment that has been successfully submitted in order to view the details of that submission. This allows the student to verify the correct version of their work was turned in.

Retrieving Assignment Feedback

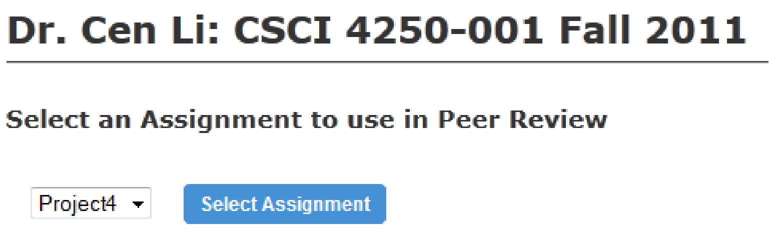
Students can retrieve feedback on their assignments via the review hyperlink on an assignment entry. The hyperlink only appears if the assignment has been graded by the instructor and feedback has been provided. Any instructor who chooses to provide feedback for their students can do so in any format of their choosing. If GUS indicates feedback has been given and the student uses the available hyperlink in order to review that feedback then the “handback” command is invoked to retrieve the file containing feedback. The file is then moved to a temporary location on the server that allows the student to access it through the browser for viewing or download.

Peer Review Module and Features

Features Provided For Instructors

Creating Peer Review Assignments

The instructor side of the Peer Review module allows instructors to create and manage peer review assignments for their active courses. Instructors can create peer review assignments based on assignments created in the Assignment module, as shown in Figure 13. Peer review assignments have an assign time and a deadline. The assign time refers to the time when peer review assignments are distributed to students and the students can start reviewing, while the deadline marks the end of the allotted review time.

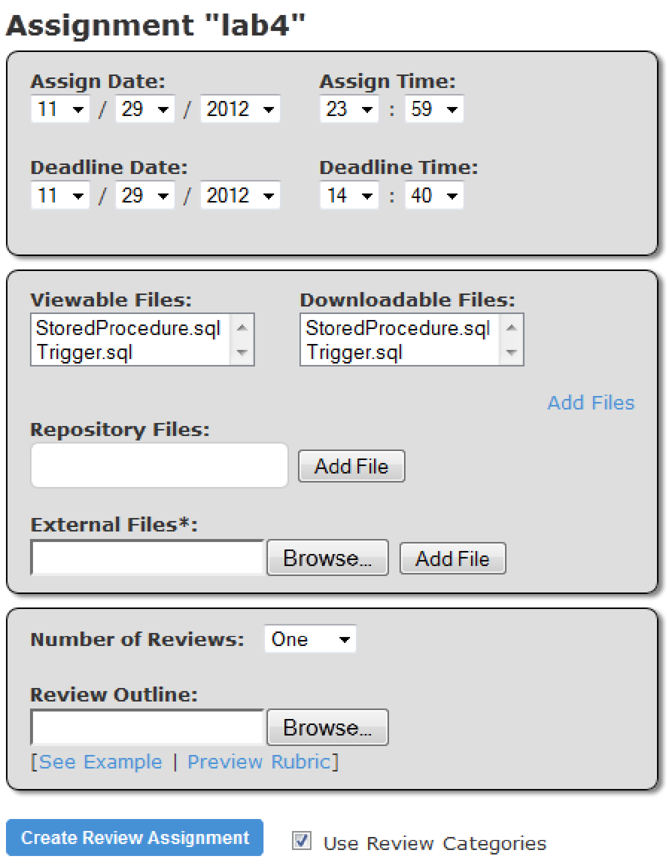

Dr. Cen Li: CSCI 4250-001 Fall 2011

Select an Assignment to use in Peer Review

Figure 13. Assignment selection for peer review

For CS projects, a student typically needs to submit a number of files, including source code, data files, and executable files. For peer review, instructors can specify which of the files and any extra files students should have for reviewing. Additionally, instructors can specify the files as viewable files, downloadable files, or both. Files marked as viewable are converted into images and viewable by students. Since the peer review process is anonymous, all names are removed from viewable files during conversion. Students can retrieve downloadable files. Furthermore, instructors can specify files to be used in the review process that were not listed in the required files for the assignment. This allows instructors to do things such as upload special test cases from their personal computer or select a file that was added to the students' repositories after the submission deadline.

Students may be assigned to review one, two, or three programs for each review assignment. This allows students a better chance to learn through the peer review process because they can see more than one additional response to the assignment and receive more feedback about their solutions. The form to create peer review assignments can be seen in Figure 14.


Assignment "lab4"

Assign Date: 11 / 29 / 2012
Assign Time: 23 : 59

Deadline Date: 11 / 29 / 2012
Deadline Time: 14 : 40

Viewable Files:
StoredProcedure.sql
Trigger.sql

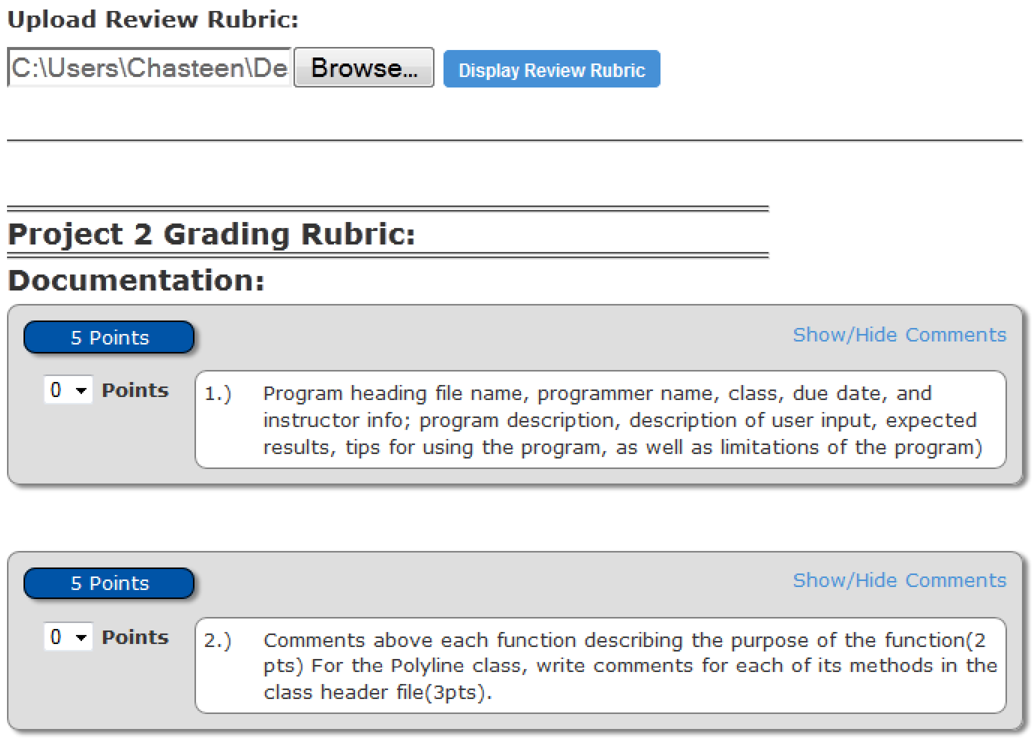
Downloadable Files:
StoredProcedure.sql
Trigger.sql

Number of Reviews: One

[✓] Use Review Categories

Figure 14. Form for creating a peer review

Instructors can guide the peer review process by providing a review rubric for students to follow when performing a review. This allows customization of peer review assignments to fit the assignment being reviewed. The rubric allows instructors to create custom point values for criteria provided in the rubric as well as section headers to control the flow of the review and comments to clarify how to assess specific parts of the assignment. Instructors can preview how their rubric will be displayed to the students by using the preview feature, shown in Figure 15.



Project 2 Grading Rubric:

Documentation:

5 Points

0 Points

1.) Program heading file name, programmer name, class, due date, and instructor info; program description, description of user input, expected results, tips for using the program, as well as limitations of the program)

5 Points

0 Points

2.) Comments above each function describing the purpose of the function(2 pts) For the Polyline class, write comments for each of its methods in the class header file(3pts).

Figure 15. Display of a partial peer review rubric

Assigning Peer Review Assignments

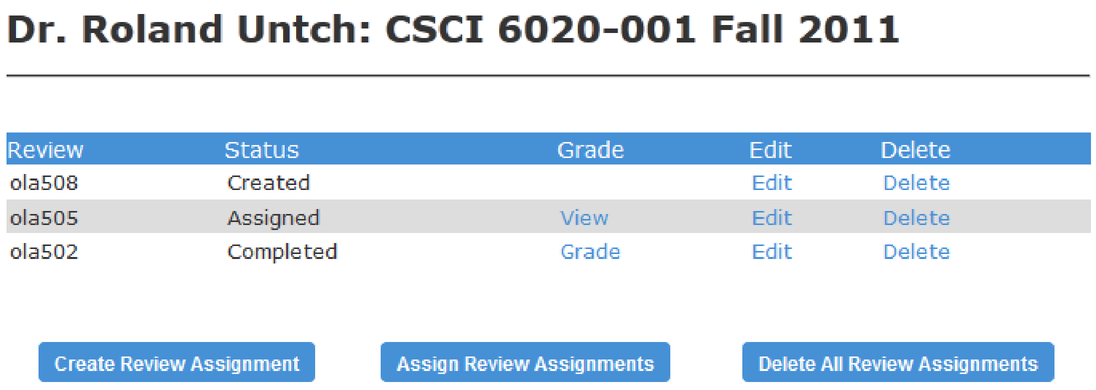
Once a peer review assignment has been created, it can be assigned to the students manually or automatically. If the instructor wants to assign it immediately after the original assignment is due, say midnight, but doesn't want to get on the computer at that time to assign it, then an automatic assignment based on the assign time would be appropriate. However, if the instructor wants to assign the peer review assignment immediately after creating it, then he or she can do so manually.

Editing Peer Review Assignments

Once a peer review assignment has been created, it can be edited if small changes need to be made. If the peer review assignment has yet to be assigned, then instructors can alter the assign date and time. Therefore, if the instructor gives students more time to complete an assignment, the assignment of the peer review can be delayed. At any point in time the instructor can also change the deadline of the peer review assignment. If there is anything else that needs to be changed in the peer review assignment, it can be done by first deleting the assignment and then recreating it.

Deleting Peer Review Assignments

Peer Review assignments can be deleted at any time. Deleting a peer review assignment before it was assigned will simply delete the template used to assign the peer reviews to the student. This is done to recreate an assignment that was created incorrectly or if an instructor decided not to assign peer reviews for the assignment after it has already been created. If the peer review assignment is deleted after it has been assigned then the entire peer review assignment will be purged from the system, including all of the students' files and peer reviews. Peer review assignments can be deleted one at a time or collectively. The collective delete operation can be used at the end of a semester to clear all student assignments from PeerSpace. The peer review menu is shown in Figure 16.

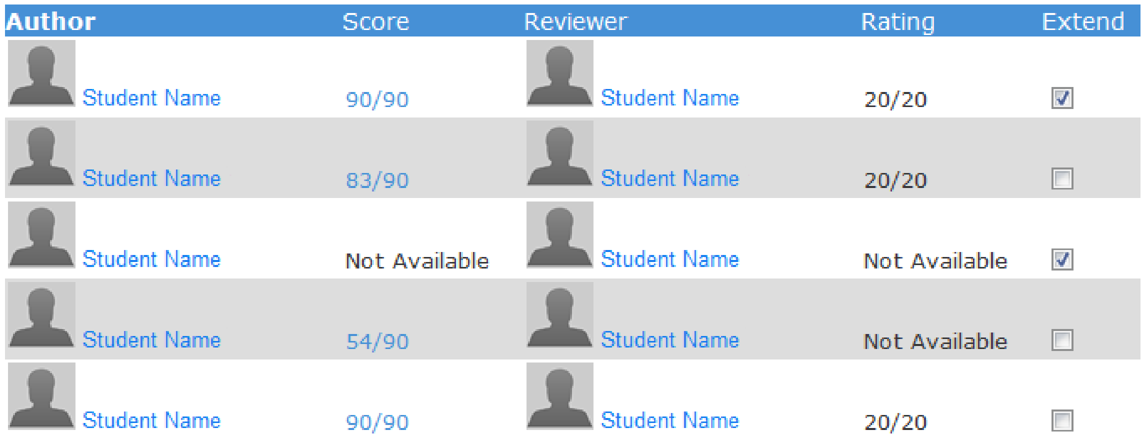

Dr. Roland Untch: CSCI 6020-001 Fall 2011

Review	Status	Grade	Edit	Delete
ola508	Created		Edit	Delete
ola505	Assigned	View	Edit	Delete
ola502	Completed	Grade	Edit	Delete

Figure 16. Peer review menu

Monitoring Peer Review Assignments

After a peer review assignment has been assigned, instructors can view the progress of the class as well as the individuals through the “Grade” view. A list of all the assigned peer reviews for a particular assignment is available in the Peer Review module for the instructor to see which reviews have been completed. A fully completed review listing contains the author’s name, picture, and score the author received as well as the reviewer’s name, picture, and rating the reviewer received. The list of peer reviews can be sorted by author, score, reviewer, or rating to make it easier to view the progress and performance of the review assignments. The instructor view of the students’ individual peer review assignments can be seen in Figure 17.

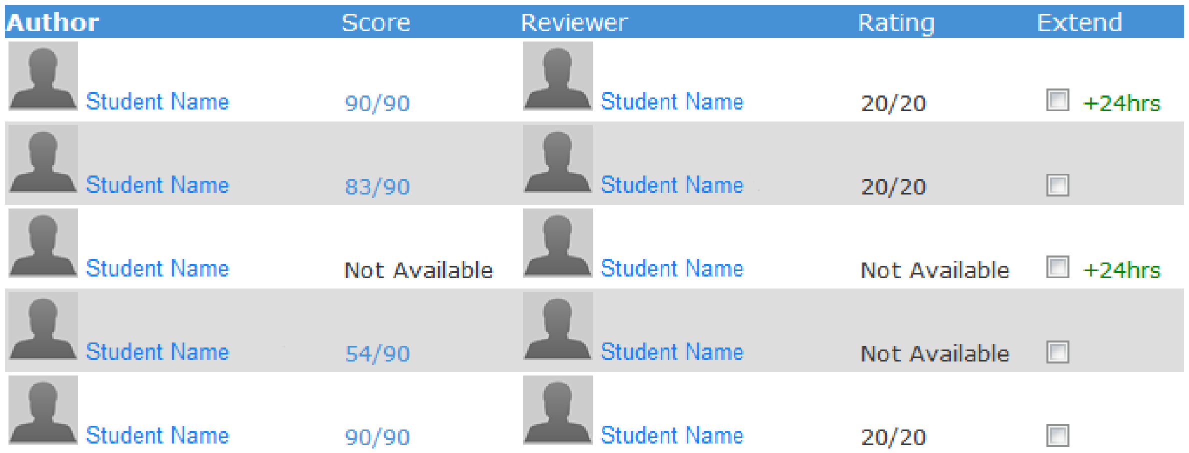


Author	Score	Reviewer	Rating	Extend
Student Name	90/90	Student Name	20/20	✓
Student Name	83/90	Student Name	20/20	☐
Student Name	Not Available	Student Name	Not Available	✓
Student Name	54/90	Student Name	Not Available	☐
Student Name	90/90	Student Name	20/20	☐

Figure 17. Instructor view of peer review assignments

Extending Peer Review Assignments

From this page, instructors can extend the deadline of a submission for an individual student or set of students. An extension will also override a previous submission to allow corrections to be made to a review. The length of an extension is provided in hours and can be seen next to the reviews that were extended. Figure 18 displays two reviews that were extended by one day.



Author	Score	Reviewer	Rating	Extend
Student Name	90/90	Student Name	20/20	☐ +24hrs
Student Name	83/90	Student Name	20/20	☐
Student Name	Not Available	Student Name	Not Available	☐ +24hrs
Student Name	54/90	Student Name	Not Available	☐
Student Name	90/90	Student Name	20/20	☐

Figure 18. Extended reviews

Exporting Peer Review Assignments

Peer review assignments can be exported from PeerSpace to an external system. This allows the instructor to keep a digital copy of all the peer review assignments rather than losing them when a peer review assignment is deleted from the system. These copies can be printed out and handed back to students, or kept as a record. The instructor can also export a separate file containing all of the scores and ratings for a peer review assignment. This serves as an additional record and can aid the grading process.

Viewing Peer Review Assignments

The instructor can view any individual peer review assignment as soon as it has been reviewed. The instructor view contains all of the author’s work, the reviewer’s assessment and comments, as well as the score and rating. After the review, the author has a chance to rate the feedback he or she received. The rating comes from the author’s perceived quality of the review. The author does this by evaluating the reviewer's accuracy, helpfulness, knowledge, and fairness. Each category used in the rating process can be given a score of one at the lowest and five at the highest. Therefore, the lowest possible rating is a four and the highest is a twenty. Both the score and the rating are shown to the instructor when an individual peer review assignment is viewed. An example can be seen in Figure 19.

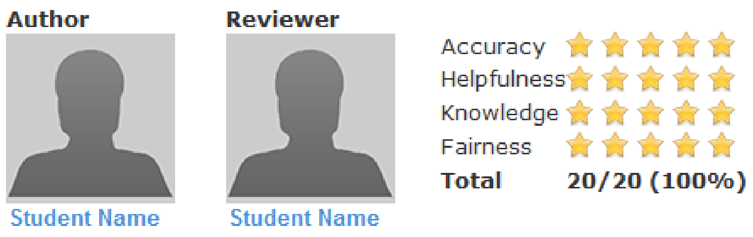


Figure 19. Results of an individual peer review assignment

When viewed by the instructor, a submission graph is provided for individual peer reviews, showing how many of the students turned in the assignment. This is supplemented by performance graphs, which are also available for the author, for individual review criterion and for the entire peer review assignment. The performance graphs show a distribution of scores across all students whose work was reviewed, with an indicator marking where the individual student stands in comparison with his or her peers. The performance graphs and completed peer review assignments will be discussed in more detail in the student feature section. Figure 20 shows a submission graph for a peer review assignment.

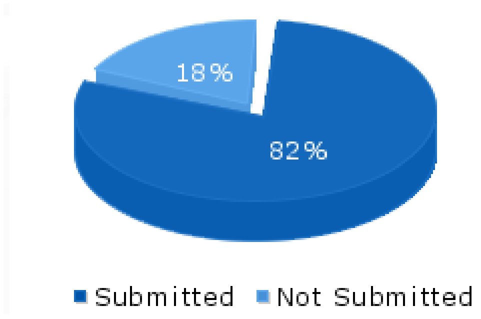
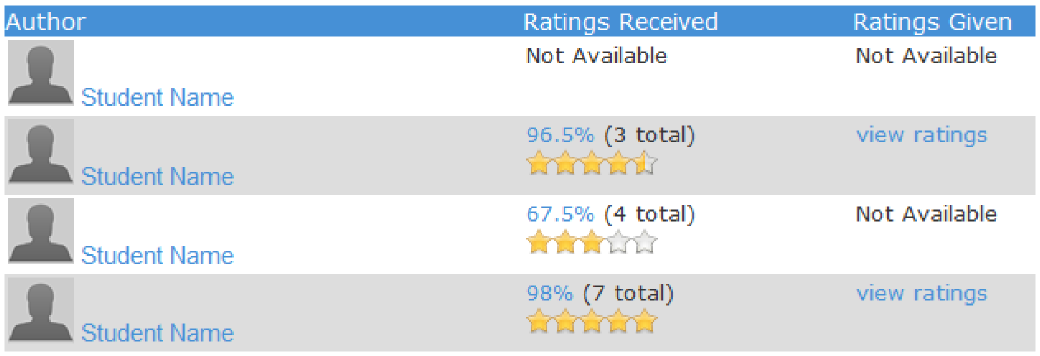


Figure 20. Submission graph for peer review assignment

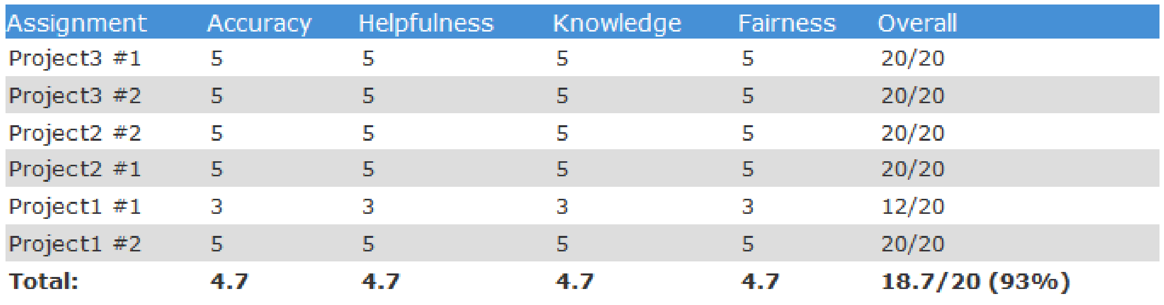
Reviewer Ratings

The instructor side of the Peer Review module also contains a feature to display student ratings. An overall rating is available for each student, which shows an average of all the ratings he or she has received for a given course (Figure 21). Viewing a student's overall rating gives the instructor an idea of how well the student performs as a reviewer. If the instructor wishes to see the details of the ratings, then they can be organized by assignment and displayed with the rating for each category, as shown in Figure 22. Additionally, the ratings a student has given can be viewed to ensure fair rating.



Author	Ratings Received	Ratings Given
Student Name	Not Available	Not Available
Student Name	96.5% (3 total)	view ratings
Student Name	67.5% (4 total)	Not Available
Student Name	98% (7 total)	view ratings

Figure 21. Reviewer's overall rating



Assignment	Accuracy	Helpfulness	Knowledge	Fairness	Overall
Project3 #1	5	5	5	5	20/20
Project3 #2	5	5	5	5	20/20
Project2 #2	5	5	5	5	20/20
Project2 #1	5	5	5	5	20/20
Project1 #1	3	3	3	3	12/20
Project1 #2	5	5	5	5	20/20
Total:	**4.7**	**4.7**	**4.7**	**4.7**	**18.7/20 (93%)**

Figure 22. Reviewer's ratings for a course

Peer Review Categories

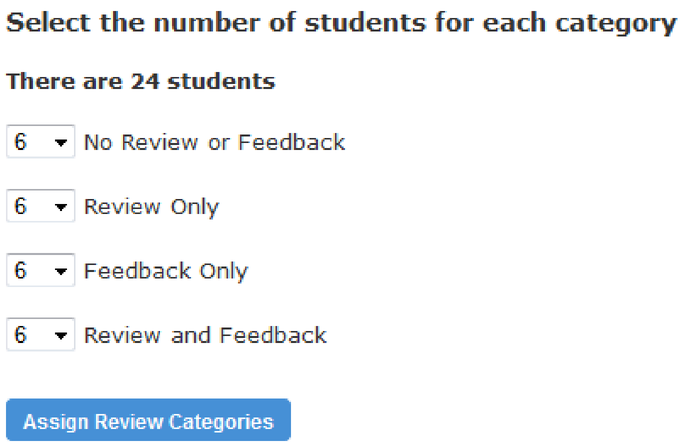
Using the Peer Review module, the instructor can specify the distribution of students into four peer review categories:

- Category 1. No Participation
- Category 2. Only Review Assignments
- Category 3. Only Receive Feedback on Assignments
- Category 4. Review and Receive Feedback on Assignments

Students in category 1 will not participate in the peer review process at all, which means they will not review other assignments and peers will not review their assignments. Students in category 2 will only review assignments and receive no feedback. Students in the third category will only receive feedback from their peers, and students in the final category will participate in both reviewing and receiving feedback on assignments.

These categories are provided to give flexibility to the system in cases when there is a need to research the effectiveness of various roles available in the peer review process. They can also be used to customize a course for peer review. However, if the instructor would like everyone to fully participate, i.e., participate both as an author and a reviewer on all the review assignments, then there is no need to use the peer review categories.

To remain impartial when selecting students for categories and to facilitate randomized trials for research purposes, the Peer Review module randomly selects students to be placed into review categories according to the student distribution provided by the instructor (Figure 23).


Select the number of students for each category

There are 24 students

6 No Review or Feedback

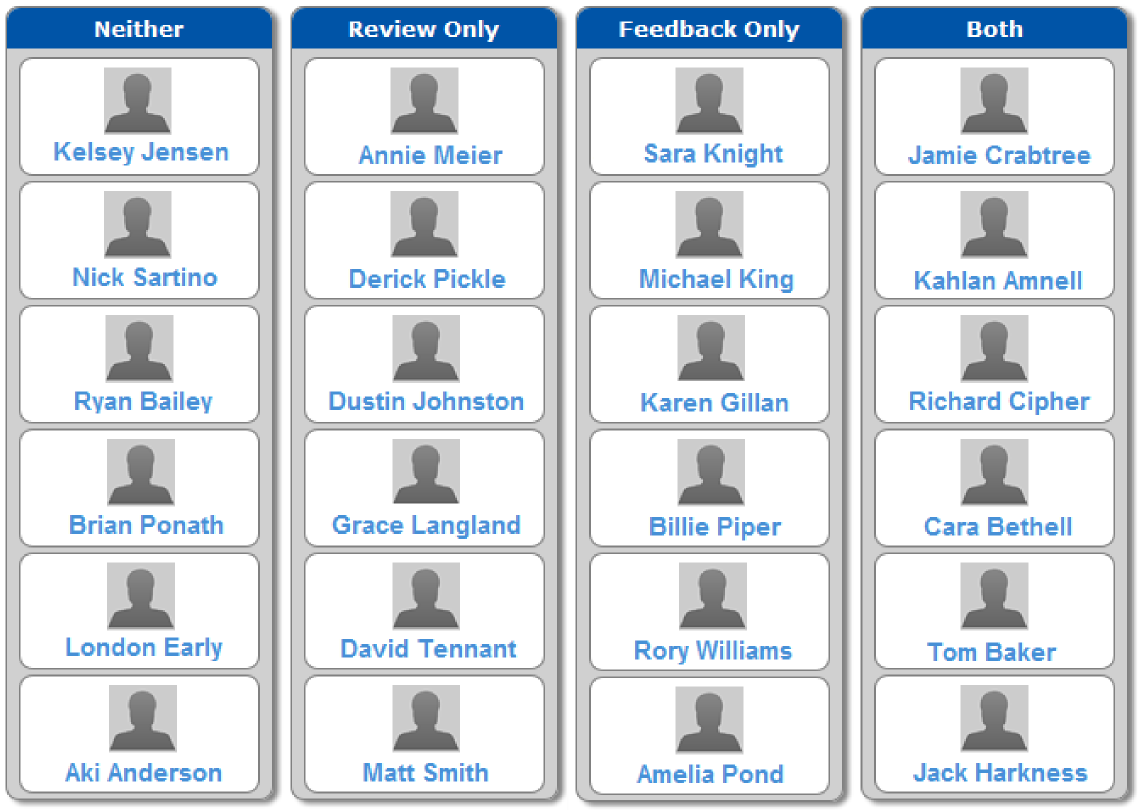
6 Review Only

6 Feedback Only

6 Review and Feedback

Figure 23. Student distribution for review categories

If there is a problem with the generated selection then the instructor can clear the categories and assign again. This is needed for extenuating circumstances that prevent a student from being able to participate effectively. Additionally, the instructor can view a list of students in each category at any time through the Peer Review module in order to see which students are in what categories. Figure 24 shows a random partition of a class of 24 students into 4 categories.



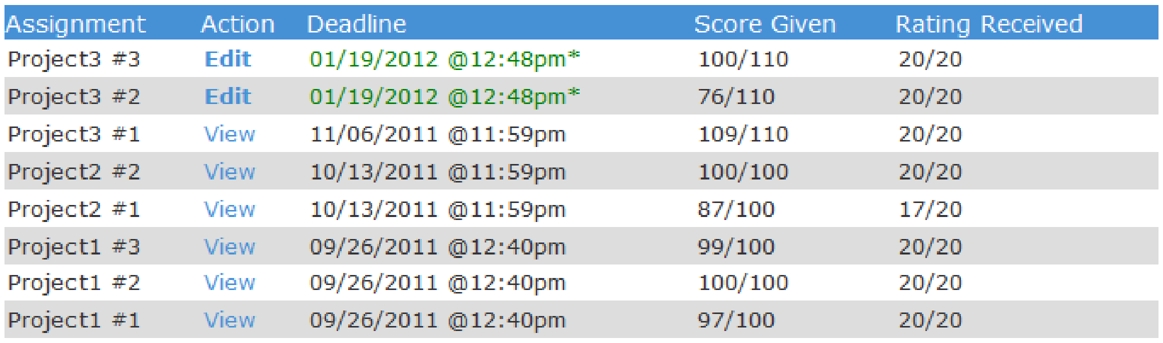
Neither	Review Only	Feedback Only	Both
Kelsey Jensen	Annie Meier	Sara Knight	Jamie Crabtree
Nick Sartino	Derick Pickle	Michael King	Kahlan Amnell
Ryan Bailey	Dustin Johnston	Karen Gillan	Richard Cipher
Brian Ponath	Grace Langland	Billie Piper	Cara Bethell
London Early	David Tennant	Rory Williams	Tom Baker
Aki Anderson	Matt Smith	Amelia Pond	Jack Harkness

Figure 24. Students in review categories

Features Provided For Students

Accessing Peer Review Assignments

The student side of the Peer Review module provides students with a way to complete peer reviews, review feedback on their assignments, and rate their reviewers. Students can see a list of all the peer reviews they have been assigned, separated by course, as well as a list of all the peer feedback they have received on their assignments. Since peer review assignments have deadlines, students can only review assignments if it is done before the deadline. The only exception to this is if the instructor has extended a peer review assignment for the student. Once an assignment has made it through the entire peer review process, it will have the score the reviewer gave the author and the author’s rating of the reviewer listed by the peer review assignment, as shown in Figure 25.

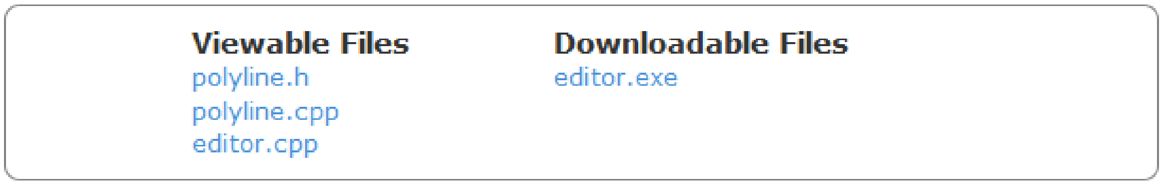


Assignment	Action	Deadline	Score Given	Rating Received
Project3 #3	Edit	01/19/2012 @12:48pm*	100/110	20/20
Project3 #2	Edit	01/19/2012 @12:48pm*	76/110	20/20
Project3 #1	View	11/06/2011 @11:59pm	109/110	20/20
Project2 #2	View	10/13/2011 @11:59pm	100/100	20/20
Project2 #1	View	10/13/2011 @11:59pm	87/100	17/20
Project1 #3	View	09/26/2011 @12:40pm	99/100	20/20
Project1 #2	View	09/26/2011 @12:40pm	100/100	20/20
Project1 #1	View	09/26/2011 @12:40pm	97/100	20/20

Figure 25. Peer review assignment listing

Reviewing Assignments

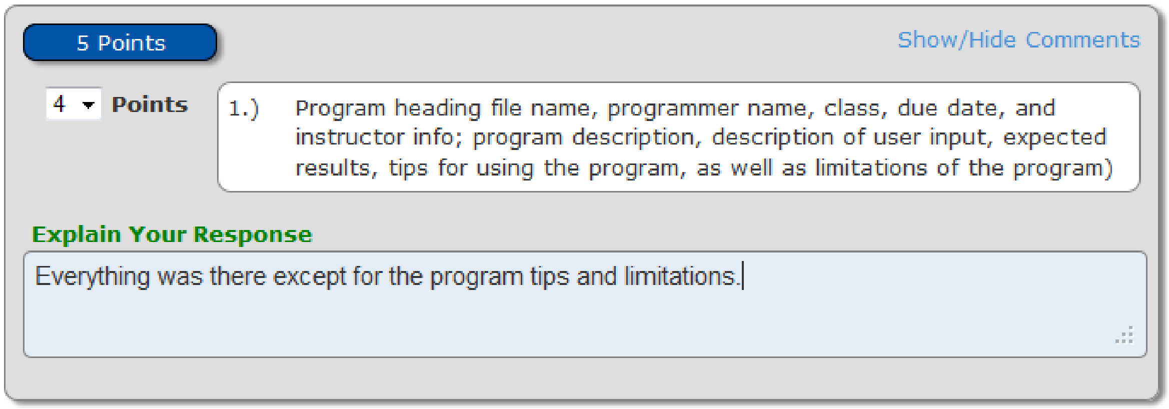
When reviewing another student’s work, the reviewer can access files from the peer review assignment that have been designated viewable or downloadable by the instructor. They will appear in two different lists. One list contains viewable files that the student can see as a picture through the browser, and the other list contains downloadable files that the student can save to his or her system. Figure 26 shows a snapshot of the two file lists available in a review assignment.



Viewable Files	Downloadable Files
polyline.h	editor.exe
polyline.cpp	
editor.cpp	

Figure 26. Files to be reviewed

The review of the assignment is guided by the instructor's rubric. The student will assign point values for each criterion according to that rubric, and if the student does not give full credit to any part of the review, then the student is presented with the opportunity to explain their decision in the form of a comment to the author, as shown in Figure 27. It is worthwhile to mention that the reviewer can also leave positive comments on criteria with a perfect score. Comments are strongly encouraged but not required to complete the review.



1.) Program heading file name, programmer name, class, due date, and instructor info; program description, description of user input, expected results, tips for using the program, as well as limitations of the program)

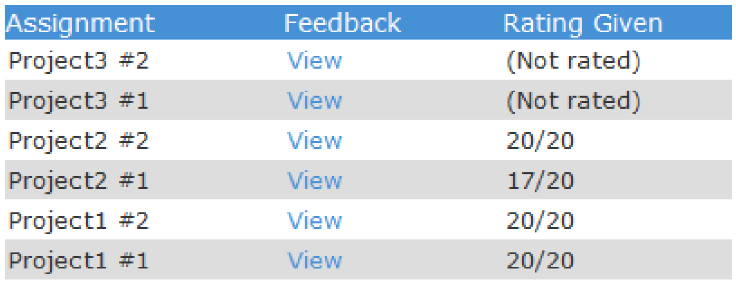
Everything was there except for the program tips and limitations.

Figure 27. Criterion review

If a student does not have time to complete a review in one session, there is an option to save the review as a draft so the student can come back later and finish the review without losing any work. Once a peer review is submitted, it is final. Therefore, students are warned before making a submission to double check the review. If an instructor has given the student an extension or performs an override on a previous submission, then the student will see a corrected deadline or message informing him or her that another submission is allowed.

Receiving Feedback

When a student receives peer feedback on his or her assignment, he or she is notified via email and site message. The feedback is viewable to the student as long as the instructor has not deleted the peer review assignment from the system. Once the author rates the reviewer's feedback then the rating will show up next to it. Figure 28 shows a list of review feedback received by a student.

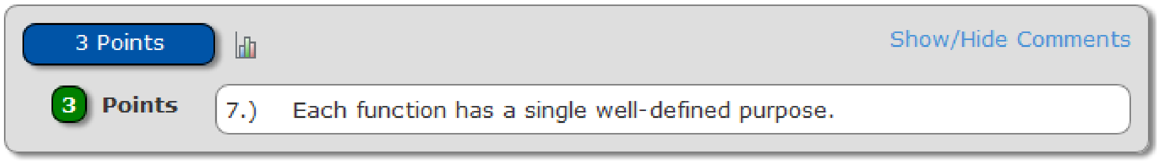


Assignment	Feedback	Rating Given
Project3 #2	View	(Not rated)
Project3 #1	View	(Not rated)
Project2 #2	View	20/20
Project2 #1	View	17/20
Project1 #2	View	20/20
Project1 #1	View	20/20

Figure 28. Peer review feedback listing

Viewing Feedback

The feedback contains the instructor's rubric and the reviewer's evaluation of the work according to the rubric. The score for each criterion is displayed over a green background if all the points were given to the author and over a yellow background if any points were deducted, see Figure 29. This allows the author to easily identify where points were taken off. In addition, each criterion contains a graph that displays how well the student did on that specific criterion compared to all of the submitted peer reviews in the class.



7.) Each function has a single well-defined purpose.

Figure 29. Criterion feedback

At the bottom of the page, there is another graph to show the student's overall performance on the assignment compared to the class. Figure 30 shows an example graph of scores obtained by all of the students in the class for a given project. The column appearing as a different color indicates the score range of the current student. The details of the assignment score for that student can be seen in the graph's legend. These graphs are used to provide instant feedback to the students on how they rank with their peers.

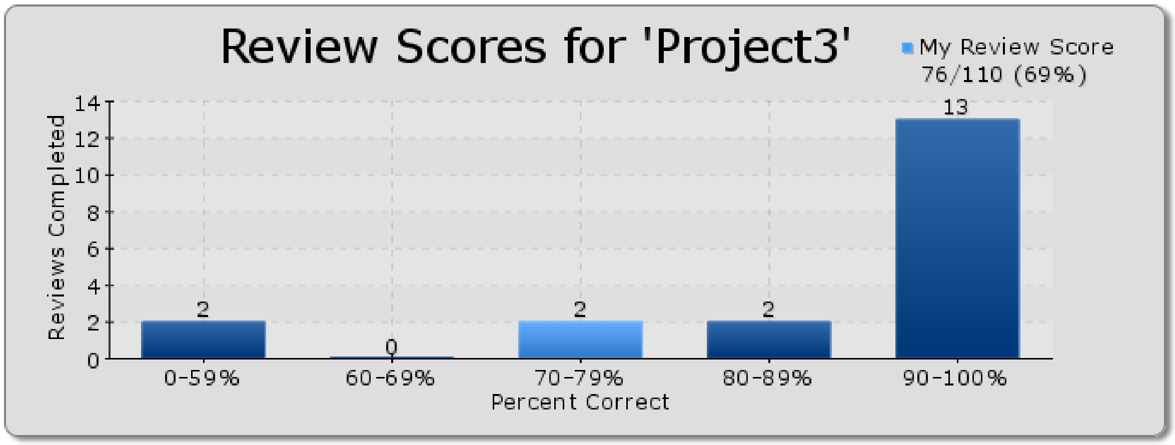
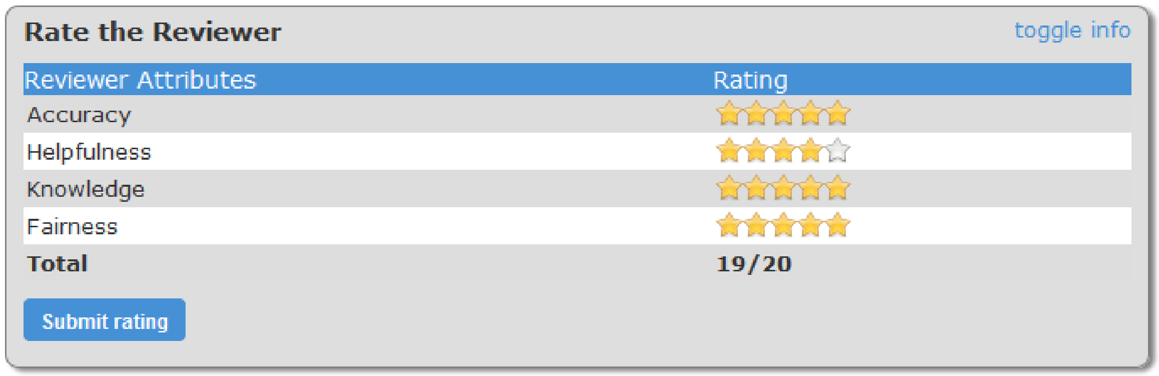


Figure 30. Feedback graph

Rating the Reviewer

After reading through all of the feedback, the student can rate the review based on the reviewer's accuracy, helpfulness, knowledge, and fairness (Figure 31). This gives students a chance to respond to feedback, and allows the reviewer to establish a reputation based on their efforts. This also serves as a quality assurance mechanism that encourages students to take the peer review process seriously.



Rate the Reviewer

toggle info

Reviewer Attributes	Rating
Accuracy	★★★★★
Helpfulness	★★★★☆
Knowledge	★★★★★
Fairness	★★★★★
Total	**19/20**

Figure 31. Rating the reviewer.

Reviewer Badge

Every student who is a reviewer and receives at least one rating will have a reviewer badge displayed in the Peer Review module. The badge averages all of the received ratings and condenses them into a five star rating. If a student has an average of 4 stars or higher, he or she is ranked as a "Super Reviewer!"

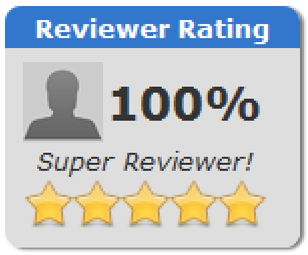


Figure 32. Reviewer badge

Peer Review Module Implementation

Instructor Side

Selecting an Assignment for Peer Review

The Peer Review module is designed to work with the Assignment module to create a seamless peer review management tool for instructors. Since peer review assignments are based on course assignments, the creation of a new peer review assignment begins by using GUS to retrieve the assignment file containing all of the course's assignments. Only assignments that appear in the assignment file and do not have peer review assignments already created for them will be available for instructors to create new peer review assignment from.

After an assignment is selected as the subject for the peer review process, all of the details for that assignment are read from the assignment file and used to auto-populate the input fields for the peer review assignment being created. If possible, the assign date is set to the deadline of the original assignment so that the peer review process can be continuous. However, the instructor can change this field and any other field to fit his or her situation.

Instructors can select which files are available to students when reviewing an assignment. These files are automatically pulled from the required files in the assignment file and added to the viewable and downloadable files list. The instructor can add extra files to the viewable and downloadable lists from the students' repository or from an external source such as the instructor's computer. If the extra file comes from the student repositories, then it will automatically be retrieved during the assignment of peer reviews.

On the other hand, if a file is uploaded from the instructor's computer, it must be stored on the PeerSpace server at the time the peer review assignment is created.

Creating a Review Rubric

The review rubric provided by the instructor is the most critical part in creating a peer review assignment. Since the rubric will guide the students while they review each other's assignments, it has to be well formatted and easy to understand. Therefore, the contents of the rubric may include headers, notes, criteria, and comments. Headers are sectional markers to be used in the rubric to indicate a new area that has a different focus, while notes allow the instructor to further explain sections and criteria to students. Each criterion appearing in the rubric can have a series of point values with conditions for attaining them explained using notes. Comments are statements that will not be shown to the student during the review process and only exist to aid the instructor. The rubric itself is contained in a plain text file with special characters assigned to a set of possible content types. The content types and their associated characters are listed in Table 1, and an example of the instructor's rubric is shown in Figure 33. Additionally, the characters used for the content types can be changed inside the function used to parse the rubric, which can be found in a Peer Review module function library.

Table 1. Rubric symbols

Symbol	**Description**
!	Header
*	Note
#	Comment (Redacted)
=	Criterion

```
# Questions use the following format:
# Possible_Points Question
= 10 The program was well documented.

# Questions can be continued on multiple lines
= 25 Uses the quicksort algorithm discussed in class
     and not a bubble sort.

# Break down points for a question
! Section on formatting
*    +2 each line of the output lined up
*    +3 each statement started on a new line
     =5 The output was well formatted
```

Figure 33. An example review rubric

Assigning Peer Reviews

When a peer review assignment is created, a template is made to store the details the instructor provided. This template is used to create all of the individual peer reviews. The creation and distribution of the individual peer reviews will only happen once the assign date and time specified in the template have passed. In order for this to happen in a timely manner, a function is called every hour to check if any peer review assignments are waiting to be assigned. Any peer review assignments found to have an assign date and time less than the current date and time will be assigned and distributed to students. An instructor may also trigger this manually if he or she wishes to assign the peer review assignment immediately.

Before the assignment of peer reviews can begin, all of the student submissions must be retrieved. Since the Peer Review module has no reliable way of knowing which students have submitted their assignment, it attempts to retrieve the assignment in question through GUS for every student in the course. If nothing was submitted for that student, then nothing will be returned. After all of the student submissions have been gathered, the Peer Review module selects the appropriate assignment algorithm to use and distributes the peer review assignment to students.

The Peer Review module has two different algorithms in order to accommodate the use of peer review categories. The simpler and more general-purpose algorithm, which does not use review categories, pairs every student who successfully submitted the assignment to the instructor’s specified number of peers in the course. A student will only be paired up with other students who have also successfully submitted the assignment. Since the review process needs to be random, the students who qualify for the peer review assignment are randomly shuffled and placed into a circular vector. From here, any given student will review N students to the right (or clock-wise), where N is the number of reviews specified by the instructor. In order for this to work, there has to be enough students who qualify to provide an anonymous review. For example, in a rare case where only two students have submitted the assignment, the review assignment process fails to preserve anonymity. Whenever a review assignment fails, the review assignment’s template is deleted to prevent the system from trying to assign it later, and the instructor is alerted to why it failed. Figure 34 provides an example of how the circular vector is used, and Figure 35 contains the algorithm for peer review assignment.

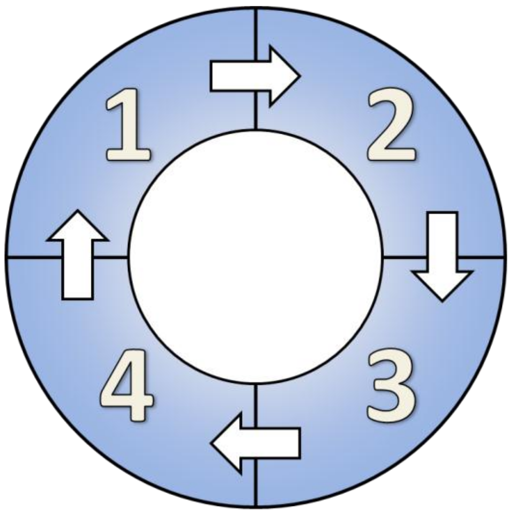


Figure 34. Circular vector used in assignment algorithm

```
Peer review program assignment:

Retrieve requirements for peer review assignment
N = number of reviews each student is required to review
Retrieve student programs with GUS
M = number of student submission retrieved
Student[1..M] = students performing review (random)

if(M <= 2) alert instructor and fail
     for i = 1 → M:
          for j = 1 → N:
               assign Student[i] to review Student[i+j]
```

Figure 35. Pseudo code for assignment algorithm

The assignment of peer reviews using categories is slightly more complicated because there are four different categories that students might be in. Rather than using these categories directly in the process, students are sorted into two groups, students who perform review and students who receive feedback. A student can be in both of the groups if they are required to fully participate or in neither of the groups if they are not participating at all. The goal is to randomly assign the students in the review group to the students in the feedback group. However, it must be done carefully because a student could end up reviewing him or herself if they are in both groups. In the case where this situation cannot be avoided or there are not enough students in either group, then the review assignment process will fail.

In order to prevent a failure because each group does not have the exact same number of students, a threshold was added to allow students up to one extra feedback response or assignment to review. Adding this functionality made the review assignment algorithm more flexible at an acceptable cost to the student workload for a peer review assignment.

When pairing students together, the students in the review group are processed first. Each reviewer is matched up with the next available student in the feedback group as long as they have not already been assigned together for this assignment and the student is not the same person as the reviewer. This continues until the entire review group has been assigned the required amount of work to review. If there are more students in the reviewer group than the feedback group, then some authors will receive more feedback on their assignment.

After the review group has been assigned to the appropriate amount of authors, the feedback group is checked to make sure everyone will receive the expected amount of feedback. Because the two groups can vary in size, it is possible for some authors to not be assigned to enough reviewers. This happens when there are a larger amount of students in the feedback group. When this happens, anyone in the feedback group who is still missing the required number of reviewers is paired up with another reviewer. Note this is what causes reviewers to sometimes receive an extra assignment to review. Pseudo code for this algorithm can be seen in Figure 36.

```
Peer review program assignment:

Retrieve requirements for peer review assignment
N = number of reviews each student is required to review
Retrieve student programs with GUS
M1 = number of students to perform peer review
M2 = number of students to receive feedback
ReviewGroup[1..M1]: students to perform review (random)
FeedbackGroup[1..M2]: students receiving feedback (random)

FeedbackIndex = 1
for ReviewIndex = 1 → M1:
R = ReviewGroup[ReviewIndex]
     for i = 1 → N:
          do until conditionsMet(R, F):
               F = FeedbackGroup[FeedbackIndex]
               FeedbackIndex = (FeedbackIndex + 1) % M2
          assignReview(R,F)

ReviewIndex = 1
for FeedbackIndex = 1 → M2:
     F = FeedbackGroup[FeedbackIndex]
     FeedbackNeeded = N – feedbackReceived(F)
     for i = 1 → FeedbackNeeded:
          do until conditionsMet(R, F):
               R = ReviewGroup[ReviewIndex]
               ReviewIndex = (ReviewIndex + 1) % M1
          assignReview(R,F)
```

Figure 36. Pseudo code for assignment algorithm using categories

During the assign process, every file is processed for review. The author’s downloadable files are copied to a directory where they can be retrieved through the browser, and the author’s viewable files are converted into images. Viewable files must be checked for student names and formatted correctly before being turned into an image. In order to make the removal of student names easier, students are asked to document their work using a special format that indicates what should be removed. When four forward slashes (“////”) are seen then the entire line is redacted.

When converting a file to an image, the following precautions are taken to ensure the conversion does not affect the readability:

- All non-ASCII characters are removed,
- Tabs are converted into spaces,
- Line numbers are added,
- Appropriate spacing and padding are added, and
- A fixed-width font is used.

The removal of all non-ASCII characters prevents unknown characters from being incorrectly converted to blank tiles or symbols that might confuse the reviewer. Likewise, tabs are converted to a set amount of spaces so that style and indentation can be assessed correctly. The line numbers are added as a convenience to the reviewer who can then reference specific parts of the file when leaving comments. Enough padding is used to make the image appear as if it were a photocopy of the original file so that the text is not pressed against the margins or other lines of itself. The most important part of the conversion is the use of a monospaced font to keep all of the characters in their intended location.

When accessing another student’s files, the GUID of that student is needed. Since student GUIDs can be found by anyone and all file names are the same, it would be possible to access everyone’s assignment. In order to prevent this and keep work anonymous, each peer review assignment is given a security key that becomes part of the directory containing the given student’s files. This key is stored in the peer review entity when it is created.

Every review has its own peer review entity associated with it. These entities are tied to the instructor’s peer review template and contain all of the data relevant to the review, including information about the author and reviewer. Because this information is sensitive, it is stored as metadata for each entity. The same is done for the reviewer’s comments and score. However, the entities are owned by the instructor, and the students do not have permission to alter data stored in the entity. To allow the students access to the entities an exception is added into the Peer Review module. This exception permits students to add and change data on the entity only when it is appropriate to do so.

Editing Peer Review Assignments

There are only two parts of a peer review assignment that can be edited. The assign date can be edited if the peer review assignment has not been assigned, and the deadline can be edited at any time. Both of these pieces of information reside in the metadata of the peer review template. The assign time is only important up until when the peer review assignment is assigned. However, the deadline is important throughout the whole process. When the deadline is updated, it is only changed on the peer review template. This is done instead of updating the individual peer review assignment entities created for the students in order to simplify the procedure of updating and checking the deadline. This is further explained in the student implementation section.

Deleting Peer Review Assignments

Deleting a peer review assignment involves purging all references to the assignment from the system. If a peer review assignment has not yet been assigned to the students, then the delete action will delete the template that was created and any files uploaded by the instructor to support the review. However, if the delete action is triggered after a peer review assignment has been assigned, then it needs to delete all of the student peer review entities from Elgg as well as all of the files that were used and created in the process. The “Delete All” function performs this process for all of the peer review assignments owned by the instructor. This is useful for clearing out peer review assignments at the end of the semester because it deletes all of the student files as well.

Monitoring Peer Review Assignments

When an instructor views the page containing all of the individual peer reviews for a peer review assignment, all of the Elgg entities that were created during the assign process are retrieved and displayed whether they are complete or not. This allows the instructor to see the results of the review assignment process, i.e., who will be reviewing whose work, at the beginning of the review process; as well as which peer review assignments have been completed.

Extending Peer Review Assignments

When an extension is given to a student, extra information has to be added to the peer review assignment entity. The new deadline is added to a special piece of metadata that overrides the original deadline and allows the student to make a submission. The instructor can alter the new deadline at any time or give multiple extensions to a student if needed. The time left on the extension is calculated by subtracting the current date and time from the most recent deadline given and displayed to the instructor on the page next to the individual peer review assignment entry.

Exporting Peer Review Assignments

Every time an instructor goes to the page containing the listing of all individual peer reviews for an assignment, two downloadable files are created. The first file is a CSV file containing each individual peer review's author, reviewer, score, and rating. The second file is a text file containing all of the individual peer reviews and their details. This includes the entire review for every entity, each appearing on a separate page, in addition to the results of the review. These are generated by iterating through all of the peer review assignment entities and copying the necessary information to file.

Viewing Peer Review Assignments

When viewing individual peer review assignments, all of the metadata attached to the entity is displayed to the instructor in a web-friendly format very similar to what the students see. The only difference is the instructor can see the identity of the two students who participated in the review. The implementation detail is given in the student implementation section.

Reviewer Ratings

Instructors can view the ratings of students in their courses through the Peer Review module. After selecting an active course from the menu, a list of students in that course is displayed along with the overall rating they have received. These ratings are calculated by retrieving all of the course's peer review assignment entities for each student and averaging all the ratings they received. When the instructor chooses to display a student's ratings in more detail, each peer review assignment entity is entered into a table with the assignment title and the details of the review. This allows instructors to view the category ratings as well.

Peer Review Categories

Peer review categories define how students participate in the peer review process. When assigning peer review categories to a course group, instructors are presented with the option to distribute any number of students into a selected category. For instructors to be able to do this, the group is first examined to count how many members of the group have the metadata role of “Student”. Once the number of students for each category has been selected, the students are randomly assigned to the categories. This information is stored as part of the course group metadata because a student can be in multiple courses that use peer review categories. This keeps courses from conflicting with each other.

After peer review categories have been assigned, they can be viewed at any time. If a student drops the course after peer review categories have been assigned, he or she will be removed from his or her category so that an accurate listing of students is maintained and displayed.

Peer review categories can be reassigned. When this happens, the metadata attached to the group is deleted so that the instructor can redistribute the students into categories. It is not necessary to reset the peer review categories if the instructor decides to do a peer review assignment without them. They can be turned off for a peer review assignment when it is created.

Student Side

Retrieving Peer Review Assignments

The student side of the Peer Review module allows students to complete peer review assignments and view feedback they have received. When a student uses the Peer Review module to review work, all of his or her assigned peer review assignments, completed or not, are displayed in a table. If the student is in more than one course, then there will be multiple tables. To retrieve the peer review assignments, all of the student's active courses are gathered. Then, the peer review assignment entities are retrieved for each course where the student is the reviewer. Finally, the peer review entities are displayed in the table for the student to view with all of the pertinent information.

A peer review assignment may be in one of two possible states, incomplete and complete. If a peer review assignment is incomplete and the deadline has not passed, then the student can perform the review on the given assignment. However, if the peer review assignment is complete or the deadline has passed then the student can only view the assignment. In order to determine what state a peer review assignment is in, a flag is checked in the peer review assignment entity's metadata. If the flag is marked complete then the peer review assignment has been completed. Also, the deadline is checked on the peer review template entity to see if the deadline has passed. Again, this is done to allow the instructor to easily make changes to the deadline. However, the peer review template deadline can be overridden if the instructor extended the deadline for that specific peer review assignment. In that case, the deadline attached to the peer review assignment entity will be used instead. Also, to alert the student this is not the original deadline, the peer review assignment will be displayed in a different color with an asterisk mark to indicate it has been extended.

Reviewing Peer Review Assignments

When a student opens a peer review assignment, he or she is granted access to all of the author's viewable and downloadable files using the security key embedded in the peer review assignment entity's metadata. The reviewer will then review these files with the help of the rubric provided by the instructor.

A copy of the rubric has been attached to the peer review assignment entity and its contents are displayed to the reviewer by looping through every line in the rubric. The rubric is formatted and displayed according to the content type of each line specified in the rubric file. Headlines appear as larger text with a horizontal line over them, while notes appear in a normal sized, monospaced font. Each criterion is displayed in a box along with its total possible points. Under the points, a drop down menu is placed containing the full range of point values for the reviewer to select.

When assessing the provided work with the rubric, if the reviewer selects a number less than the max point value for any given criterion then a text field will appear under the description. Additionally, instructions will appear in red text requesting that the reviewer explain his or her response. After a set amount of characters have been typed into the text field then the instructions will turn green to indicate the reviewer has entered a satisfactory length response for this field. This is to encourage the students to provide an explanation for why points were deducted. To support positive feedback, the reviewer can leave a comment even if no points are deducted. Furthermore, an area for additional comments is provided at the bottom of the peer review where the reviewer can give feedback that did not fit in with the instructor’s rubric.

Since students are often assigned multiple peer reviews and some assignments are quite long to review, there is a feature to allow the students to save their review as a draft. This feature can be used any time a student decides to temporarily save their review before final submission. This saves the reviewer’s work to the peer review assignment entity, but the metadata flag that marks a peer review assignment complete will not be set. This prevents students from receiving incomplete reviews, which can be easily mistaken as bad review.

When a student submits a peer review, it is considered a final submission. Therefore, students are warned each time they click on the submit button to make sure they have filled everything out correctly. Once a peer review has been submitted the reviewer will no longer be able to edit the review, and the score will be final. Because the reviewer does not own the entity that holds the review, a special flag must be set to allow changes to the entity. This flag is only set temporarily when the review is being submitted. This means that it is impossible to make changes to this entity from anywhere else. Figure 37 shows a simplified code segment that adds this functionality to the Peer Review module.

```
Add Function to Override Metadata Permissions:

function review_metadata_can_edit($parameters)
{
     global $override_review_metadata_flag;
     $type = $parameters['entity']->getSubtype();

     if( $override_review_metadata_flag == true &&
         $type == 'review_assignment' )
          return true;
     else
          return false;
}
```

Figure 37. Permission check for Peer Review module

Collecting Statistics

Statistics are collected and stored in a peer review statistic entity every time a peer review assignment is completed. This entity exists for every peer review assignment that an instructor creates and is responsible for storing all of the scores given by the reviewers. It stores not only the overall score of the peer review assignment but also the individual scores for every criterion in the rubric. The existence of this entity allows easy access to information about completed peer review assignment without having to analyze multiple entities.

Retrieving Feedback

After a peer review assignment is completed, the author receives a notification that new feedback can be viewed for the given assignment. The Peer Review module can then be used to display a complete list of all the feedback a student has received on his or her assignments. A process similar to retrieving peer review assignments is used to retrieve the feedback as well. After finding all of the student's active courses, the peer review assignment entities for each course are retrieved where the student is the author and the review is complete. The results are displayed in a table that contains the title of the assignment and the rating given to the reviewer. However, new feedback will not yet contain a rating. It is important to note that if a reviewer does not complete an assigned peer review assignment before the deadline then the author will never receive feedback from that student.

Viewing Feedback

The feedback students receive is displayed the same way it appears to the reviewer, and the author can reference the same files and rubric the reviewer had access to. This means that each comment left by the reviewer is available for the author to read. However, several efforts have been made to make the feedback easier to analyze and more beneficial to students. In addition to changing the background color of the received points' box to indicate if points were deducted, a series of performance graphs have been implemented to give students a better understanding of how they rank amongst their peers.

The performance graphs are dynamically created using the peer review statistic entity each time feedback is viewed. This is necessary because peer review assignments are not completed at the same time. Therefore, the statistics used in the performance graph will change as peer review assignments are being completed. Still, for a truly asynchronous process, students should be able to view the statistics based on available data when they receive feedback.

A graphing library called pChart [[23]] is used to draw and render the performance graphs. Since pChart has an existing bar graph layout to use for the performance graphs, the only step needed is to enter the data into the graph. Each performance graph has the same layout which features five groupings: 00 – 59%, 60 – 69%, 70 – 79%, 80 – 89%, and 90 – 100%, based on score. These groupings are similar to most grading practices and provide students with a good reference to their standings.

The data from the peer review statistic entity is distributed to these groupings appropriately, and the grouping that contains the author's score is rendered with a unique color. Once the performance graphs are rendered, they are stored in a predetermined location so that they can be displayed with the feedback. Figure 38 demonstrates how to store scores into a basic bar graph using pChart.

The performance graphs are hidden by default to reduce the space needed to display a review. A graph icon is placed next to each criterion instead of showing the full graph. When students click on the graph icon, the full image of the performance graph is shown along with the description of the selected criterion. This is done using a JavaScript media viewer called Highslide JS [[15]].

Distribute Scores into a Bar Graph:

```
foreach( $scores as $score )
{
        if( $score > 89 ) $A++;
        else if( $score > 79 ) $B++;
        else if( $score > 69 ) $C++;
        else if( $score > 59 ) $D++;
        else $F++;
}

$myData = new pData();
$myData->addPoints(array($A,$B,$C,$D,$F),"Scores");

$myPicture = new pImage(700,230,$myData);
$myPicture->setGraphArea(50,50,675,190);

$myPicture->drawScale();
$myPicture->drawBarChart();

$myPicture->stroke();
```

Figure 38. Creating a bar graph with pChart

Highslide's gallery feature positions the performance graph on a layer above the rubric. This is done to support a more efficient method of viewing the performance graphs. The student can use the arrow keys to move from one performance graph to another, and each one is labeled so the student is aware of what is being viewed. These graphs are swapped out without changing the position of the main page so when a student is done viewing the graphs he or she can exit Highslide and return to the same location. The only exception is the overall performance graph which is always displayed at the bottom next to the author's score. This ensures that the student will look through the review because this is the only place the author's score is displayed.

Rating the Reviewer

Once the author has finished viewing the feedback he or she received, the reviewer can be rated. This is done by evaluating the reviewer’s accuracy, knowledge, helpfulness, and fairness. When this information is submitted, it is added to the peer review assignment entity, using the special permission flag, and incorporated into the reviewer’s badge. Furthermore, a notification is sent to the reviewer to inform him or her of the rating. The Elgg function used to send notifications in the Peer Review module is show in Figure 39. The first argument is the recipient and the second argument is the sender. Notice that the message is sent from the instructor in this case to preserve anonymity. The last argument can be left out to default to the user’s preferences in notifications, but to guarantee every student is informed about academic material the message is specifically sent to the student’s PeerSpace account and school email address.

Send Message to a Reviewer from Instructor:

```
    notify_user( $review->author,
                 $review->owner_id,
                 $subject,
                 $message,
                 NULL,
                 array(0=>'site','1'=>'email') );
```

Figure 39. Elgg function to notify user

Reviewer Badge

Reviewer badges are generated from the ratings students receive and are displayed in the Peer Review module. The rating that appears on the badge is an average of all the ratings a student has received. Because ratings are tied to the peer review assignment entities and are often deleted at the end of the course, every rating a reviewer receives is averaged into an overall reviewer rating stored in the user metadata. Consequently, the rating is easy to acquire and is independent from the peer review assignment entities.

The next chapter covers the experiments performed to study the effectiveness of peer review. Details of the experiment setup, approach to the analysis, as well as results from the experiments are examined.

CHAPTER IV

EXPERIMENTS AND ANALYSIS

Experiment Setup

Three CS courses, with two sections each, took part in the peer review study. The three courses were computer science 1 (CS1), computer science 2 (CS2), and Advanced Data Structures (ADS). Each course had access to PeerSpace and both sections of each course had the same instructor, textbook, assignments, and tests. Every course had a control and experimental section. The only difference between them was the use of peer review. The experimental sections participated in peer review while the control sections did not. The distribution of students in the course sections can be seen in Table 2.

Table 2. Student distribution in course sections

Number of Students	CS1	CS2	ADS	Total
Control Section	28	27	22	77
Experimental Section	29	27	16	72

The analysis of the experiment was organized by grouping students in both the control and experimental sections into categories. These categories divide students according to their seniority-status and by their major, respectively. This allows the effect of peer review on lower division students to be compared with upper division students, and for the effect on CS majors to be compared against non-CS majors. The distribution of students into the appropriate categories can be seen in Table 3 for CS1, Table 4 for CS2, and Table 5 for ADS.

Table 3. Student distribution into categories for CS1

CS1 Sections	Lower Division	Upper Division	CS Major	Non-CS Major
Control Section	17	11	5	23
Experimental Section	22	7	9	20

Table 4. Student distribution into categories for CS2

CS2 Sections	Lower Division	Upper Division	CS Major	Non-CS Major
Control Section	15	13	18	10
Experimental Section	15	11	17	9

Table 5. Student distribution into categories for ADS

ADS Sections	Lower Division	Upper Division	CS Major	Non-CS Major
Control Section	4	18	18	4
Experimental Section	1	15	14	2

Analysis of Experiment Results

All of the participating courses have been analyzed quantitatively and qualitatively to produce a more detailed report on the effects of peer review. Each analysis compares student grades between the control and experimental groups in order to study the benefits of student learning produced by peer review. Furthermore, students in both sections were given a survey to measure their outlook on groups at the end of the course. The survey used was the modified Group Environment Questionnaire (MGEQ) [[27]] that features four categories that consider the individual attraction to group activities and group integration aspects of both social actions and learning tasks. These categories are made from groupings of the eighteen questions found in the MGEG. Table 6, Table 7, Table 8, and Table 9 list all the questions under their respective categories.

Table 6. MGEQ: Individual Attraction to Group (Social)

#	Questions
1	I do not enjoy interacting with members of this class.*
3	I am not going to miss the members of this class when the semester ends.*
5	Some of my best college acquaintances/friends are from this class.
7	I enjoy the atmosphere in other classes more than this class.*

Table 7. MGEQ: Individual Attraction to Group (Learning)

#	Questions
2	I’m unhappy that this class is not challenging enough.*
4	I am unhappy with the amount of effort my fellow classmates devote to this class.*
6	This class does not give me enough opportunities to improve my programming skills.*
8	I like the way group activities enhance the class.
9	This class is one of my most important college classes.

Table 8. MGEQ: Group Integration (Social)

#	Questions
11	Members of our class would rather work alone than work in groups.*
13	Class members rarely interact with one another.*
15	Members of this class would enjoy spending time together outside of class.
17	Members of this class do not interact outside of lectures and labs.*

Table 9. MGEQ: Group Integration (Learning)

#	Questions
10	Class members help one another.
12	Instead of blaming someone else, members of our class generally take personal responsibility if they are unable to complete an assignment.
14	Class members have conflicting aspirations for what they want out of the class.*
16	If members of this class have problems with the material, other members of the class express concern and help them with these problems.
18	Class members are reluctant to communicate with one another.*

All questions use a Likert scale with “strongly agree” having a value of 1 and “strongly disagree” having a value of 5. For questions 1, 2, 3, 4, 6, 7, 11, 13, 14, 17, and 18 (the questions marked with an asterisk), “strongly disagree” is considered a positive result so a higher score is better. For the remaining questions, a lower score is better.

Each category has an overall value calculated by adding the results from each question in the category together. Questions where “strongly agree” is considered a positive result are flipped so that all questions are uniform in their worth. This allows comparisons to be done for each category. Both grades and survey results were analyzed using a two-tailed t-test with a threshold of 0.1.

The following five analyses were performed:

- Analysis A: Control Sections vs. Experimental Sections
- Analysis B: Lower Division vs. Upper Division Students
- Analysis C: CS Major vs. Non-Major Students
- Analysis D: CS2 Control Section vs. Experimental Section
- Analysis E: Students' Peer Review Experiences

Analysis A: Control Sections vs. Experimental Sections

This analysis compares all of the control sections against the experimental sections to test the differences between the students who used peer review and those who did not. The final grades for the control and experimental sections, across all three courses, were averaged together to assess student performance. The experimental group had a higher average than the control, but the result was not statistically significant. Table 10 shows the final grade average for each group.

Table 10. Final grade averages

	Control	**Experimental**	**T-Test**
	Avg. (Std.)	Avg. (Std.)	
Final Grades	72.72 (23.27)	78.97 (22.59)	0.16

The student responses to the MGEQ were also compared. The experimental group had consistently higher markings across all four categories, and both the social and learning aspects of group integration had a statistically significant difference over the control group. The average markings for each group are displayed in Table 11. The next two analyses examine the grade averages and MGEQ results in finer detail to determine where peer review was most effective.

Table 11. MGEQ category results

	Control	**Experimental**	**T-Test**
	Avg. (Std.)	Avg. (Std.)	
Individual Attraction to Group (Social)	12.62 (2.76)	13.52 (2.71)	0.139
Individual Attraction to Group (Learning)	19.41 (3.12)	19.79 (2.76)	0.567
Group Integration (Social)	12.03 (3.73)	14.07 (3.17)	***0.009***
Group Integration (Learning)	17.33 (3.47)	18.62 (2.95)	***0.076***

Analysis B: Lower Division vs. Upper Division Students

The purpose of this analysis was to study the effects peer review had on lower division students vs. upper division students. In order to do this, the students from every course section were divided into four groups:

- Lower division students in the control sections (38 students),
- Upper division students in the control sections (35 students),
- Lower division students in the experimental sections (37 students), and
- Upper division students in the experimental sections (40 students).

The course average of the lower division students and the upper division students is compared for both the control and experimental sections.

The results of both comparisons reveal the final course average for the experimental group to be higher than the control group. The lower division student averages have a higher difference than the upper division student averages. However, neither has proven to be statistically significant from the t-test. Table 12 contains the average, standard deviation, and t-test results for each group and division.

Table 12. Lower division vs. upper division final grade averages

Final Grades	**Control** Avg. (Std.)	**Experimental** Avg. (Std.)	**T-Test**
Lower Division	72.49 (26.62)	82.07 (19.97)	0.15
Upper Division	72.98 (19.38)	76.37 (24.60)	0.58

The students' response to the MGEQ showed that lower division students benefit more than upper division students from peer review in terms of their perspective on a group environment. The average scores from the control group were lower than those of the experimental group in all four categories. The differences were also statistically significant according to the t-test. Table 13 and Table 14 show the results for each category of the MGEQ in terms of lower and upper division students.

Table 13. Lower division MGEQ category results

Lower Division Students	**Control** Avg. (Std.)	**Experimental** Avg. (Std.)	**T-Test**
Individual Attraction to Group (Social)	11.74 (2.38)	13.64 (3.05)	***0.02***
Individual Attraction to Group (Learning)	18.43 (2.55)	20.04 (3.06)	***0.06***
Group Integration (Social)	11.08 (3.51)	14.40 (3.23)	***0.002***
Group Integration (Learning)	16.70 (3.65)	18.50 (2.84)	***0.07***

Table 14. Upper division MGEQ category results

Upper Division Students	Control Avg. (Std.)	Experimental Avg. (Std.)	T-Test
Individual Attraction to Group (Social)	13.88 (2.85)	13.40 (2.35)	0.59
Individual Attraction to Group (Learning)	20.81 (3.39)	19.50 (2.44)	0.19
Group Integration (Social)	13.38 (3.72)	13.70 (3.13)	0.78
Group Integration (Learning)	18.25 (3.09)	18.75 (3.14)	0.64

Analysis C: CS Major vs. Non-Major Students

In this analysis, the focus was on determining if peer review affected CS majors the same as non-CS majors. In other words, which group benefits more from using the Peer Review module in PeerSpace? Again, students were organized into four groups:

- Students with a CS major in the control sections (41 students),
- Students with a non-CS major in the control sections (37 students),
- Students with a CS major in the experimental sections (41 students), and
- Students with a non-CS major in the experimental sections (31 students).

It turns out the CS majors in the experimental group had a higher course average than the ones in the control group by a statistically significant amount. This was not true for the non-CS majors, even though the experimental group did have a higher course average. Table 15 contains the course averages of all the groups in this analysis.

Table 15. CS major vs. non-CS major final grade averages

Final Grades	Control Avg. (Std.)	Experimental Avg. (Std.)	T-Test
CS Major	74.39 (22.68)	84.72 (13.33)	***0.028***
Non-CS Major	69.27 (24.82)	71.07 (29.72)	0.842

Student responses to the MGEQ show peer review leads to a more positive collaborative learning environment for CS majors when compared to non-CS majors. When examining CS majors, the experimental group average was higher in every category, and statistically significant in both aspects of group integration. This was expected since students tend to take courses more seriously if they are part of their major. Still, even the non-CS major group had a higher average with the use of peer review, but none of the results were statistically significant. Table 16 and Table 17 show the control and experimental group responses to the MGEQ when organized by CS and non-CS majors.

Table 16. CS major MGEQ category results

CS Major Students	**Control** Avg. (Std.)	**Experimental** Avg. (Std.)	**T-Test**
Individual Attraction to Group (Social)	11.75 (2.95)	12.12 (1.95)	0.56
Individual Attraction to Group (Learning)	18.85 (3.78)	18.87 (3.69)	0.98
Group Integration (Social)	12.50 (2.71)	13.88 (2.01)	***0.027***
Group Integration (Learning)	16.35 (2.65)	17.45 (2.25)	***0.086***

Table 17. Non-CS major MGEQ category results

Non-CS Major Students	**Control** Avg. (Std.)	**Experimental** Avg. (Std.)	**T-Test**
Individual Attraction to Group (Social)	11.50 (1.58)	11.80 (2.50)	0.64
Individual Attraction to Group (Learning)	17.65 (2.62)	18.96 (3.14)	0.14
Group Integration (Social)	12.05 (2.70)	13.12 (2.30)	0.16
Group Integration (Learning)	16.30 (2.64)	17.60 (2.80)	0.12

Analysis D: CS2 Control Section vs. Experimental Section

The data collected from the CS2 course was analyzed to provide detailed results of a single course that participated in the peer review study. In addition to the final grades, student programs and tests were averaged for the control and experimental sections and compared with each other. The experimental section had a higher average in all areas, with each area being statistically significant. Table 18 shows the averages for each area and group.

Table 18. CS2 course work averages

CS2	**Control** Avg. (Std.)	**Experimental** Avg. (Std.)	**T-Test**
Programs	73.05 (21.84)	88.93 (12.37)	***0.013***
Tests	65.08 (26.76)	82.29 (17.73)	***0.033***
Overall	67.63 (25.03)	87.09 (11.88)	***0.006***

The programming assignments (PA) were examined individually to see if peer review had an effect from assignment-to-assignment. Because the experimental section had a higher average on every PA, the progression and details of each PA were reviewed. Table 19 contains the average of each PA for the control and experimental sections. The first peer review assignment was over PA2. Therefore, it was to be expected that the first two PAs show little difference between the control and experimental section. More importantly, PA3, PA7, PA8, and PA9 are all statistically significant when compared to the control section.

Table 19. CS2 programming assignment averages

CS2 **PAs**	**Control** Avg. (Std.)	**Experimental** Avg. (Std.)	**Difference**	**T-Test**
PA 1	92.22 (17.41)	94.78 (19.58)	02.56	0.691
PA 2	92.38 (08.70)	93.78 (10.82)	01.40	0.682
PA 3	89.31 (11.24)	95.72 (09.30)	06.41	***0.078***
PA 4	73.31 (32.96)	82.00 (34.84)	08.69	0.462
PA 5	71.38 (37.45)	85.94 (25.48)	14.57	0.190
PA 6	72.94 (37.59)	85.00 (23.46)	12.06	0.264
PA 7	58.19 (41.94)	92.39 (09.91)	34.20	***0.002***
PA 8	57.38 (47.78)	90.06 (24.14)	32.68	***0.015***
PA 9	50.31 (44.94)	80.67 (31.71)	30.35	***0.028***

Figure 40 graphs the PA averages of individual assignments for the control and experimental section of CS2. This provides a visual representation of the experimental section in relation to the control section. As more peer review assignments were completed by the experimental section, the difference between the two sections grew larger.

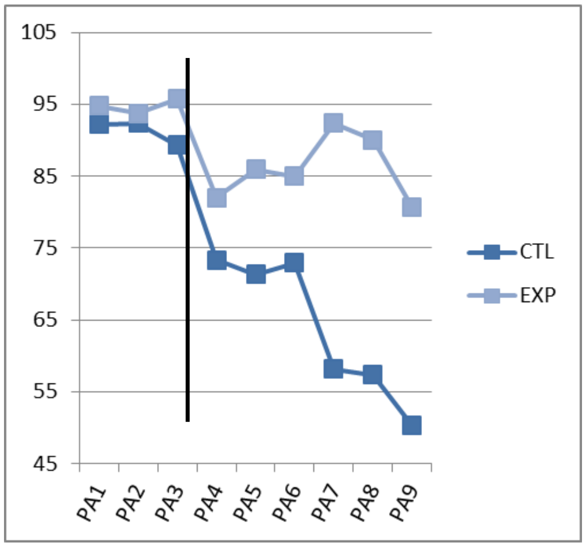


Figure 40. Graph of CS2 programming assignment averages

The difference between the experimental and control section's PA averages listed in Table 19 has been graphed in Figure 41. A best-fit line has been plotted for PA2 through PA9 to show the trend of the differences in PA averages after students started peer review. The difference averages out to about five grade points per assignment.

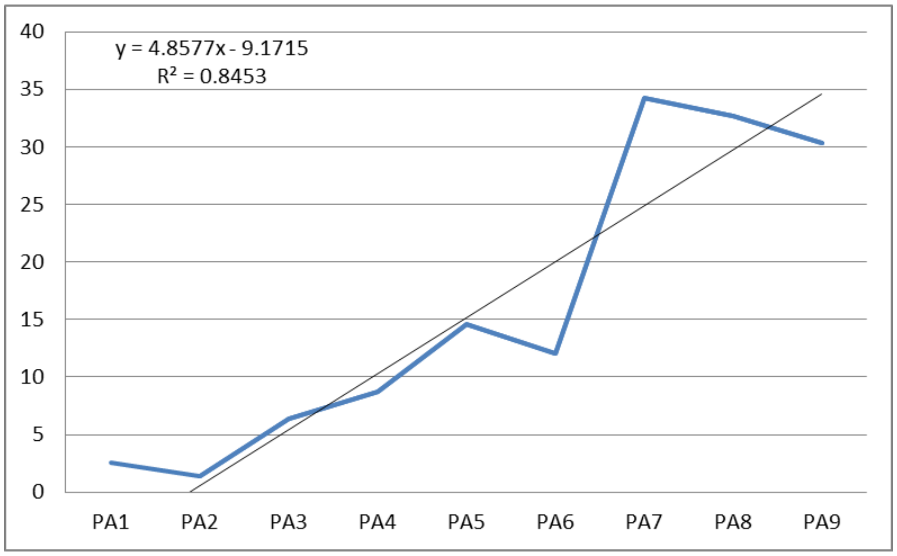


Figure 41. Graph of CS2 programming assignment average differences

The experimental section also formed a better collaborative learning environment according to student responses to the MGEQ. All four categories of the MGEQ had a higher average than the control section. The results were statistically significant in the group attraction category concerning social actions, as well as the group integration category for both social and learning, as shown in Table 20.

Table 20. CS2 MGEQ category results

CS2	**Control** Avg. (Std.)	**Experimental** Avg. (Std.)	**T-Test**
Individual Attraction to Group (Social)	11.77 (2.12)	13.94 (2.79)	***0.0290***
Individual Attraction to Group (Learning)	18.23 (2.83)	19.63 (2.98)	0.2117
Group Integration (Social)	10.00 (4.18)	15.00 (2.97)	***0.0008***
Group Integration (Learning)	16.30 (3.75)	18.88 (2.55)	***0.0375***

Student responses from the experimental section were considerably better than those from the control section along the following questions:

- I do not enjoy interacting with members of this class,
- I am unhappy with the amount of effort my fellow classmates devote to this class,
- Class members help one another,
- Members of our class would rather work alone than work in groups,
- Class members rarely interact with one another,
- Members of this class would enjoy spending time together outside of class,
- If members of this class have problems with the material, other members of the class express concern and help them with these problems, and
- Members of this class do not interact outside of lecture and labs.

This variance indicates that students in the experimental section interact more with classmates and are more willing to work together to help each other, which makes for a more supportive learning community.

Student responses from the CS2 course were also analyzed to study the effectiveness on peer review in terms of improving students' sense of community for both connectedness and learning. The connectedness of the community was determined by adding the results from both social categories of the MGEQ (Table 6 and Table 8), and the learning in the community comes from adding the results from both learning categories (Table 7 and Table 9). The Students in the experimental section of CS2 had a significantly higher sense of belonging to a learning community than the control section. Yet, the control section had a slightly higher rating concerning the student's feeling of connectedness among students, but it was not statistically significant. The overall sense of community was still higher for the experimental section. Table 21 contains the averages for the sense of community in both CS2 sections.

Table 21. CS2 sense of community

CS2	**Control** Avg. (Std.)	**Experimental** Avg. (Std.)	**T-Test**
Connectedness in Community	32.60 (6.9)	31.90 (2.3)	0.72
Learning in Community	29.80 (5.8)	33.10 (3.3)	***0.06***
Overall Sense of Community	62.40 (9.5)	65.00 (4.0)	0.32

Analysis E: Students' Peer Review Experiences

Students from all three experimental sections in CS1, CS2, and ADS completed a survey concerning the use of peer review at the end of the semester. All questions used the Likert scale with "strongly agree" having a value of 1 and "strongly disagree" having a value of 5, except for the last one which refers to an actual number value. The questions in the survey were the following:

(Q1) I learned something from other students' programs.

(Q2) I will use something I learned from other students' programs in writing my own code.

(Q3) I realize that proper program documentation is very important after reviewing other students' code.

(Q4) The review feedback from other students is helpful.

(Q5) The review feedback from other students is discouraging.

(Q6) I feel very confident about reviewing other students' programs.

(Q7) I feel peer review is not worth the effort.

(Q8) I don't want to do peer review in the future.

(Q9) I prefer to only look at other students' programs rather than reviewing them.

(Q10) If there were fewer questions in the review, I will be more likely to give detailed comments.

(Q11) How many programs would you prefer to review for each assignment?

Table 22 gives the average response from the students for each course and all the experimental sections combined. Students agree, for the most part, that they have learned something from reviewing other students' programs (Q1), and they will use it in their own programs (Q2). Also, the importance of documentation has been made clear to them (Q3), especially to CS2 students. The feedback students receive was generally considered helpful (Q4) and not too harsh (Q5). Students also feel confident in reviewing other students' programs (Q6). Most students do not feel that peer review was a waste of time (Q7) and would do it again in the future (Q8). Even though students admitted to finding feedback useful to them, they would rather be able to look at other students' code without reviewing it (Q9). Some students may prefer a shorter rubric to perform peer review with, but the general response of students indicated they were fine with the way it was done during the study (Q10). CS2 rubrics may have been longer than the other courses. Students generally only want to review one other program (Q11), but CS1 students had a higher response specifying they wanted to review two programs. This most likely has to do with the time it takes to review an assignments in different courses, since CS2 and ADS programs typically contain more complex work.

Table 22. Peer review survey results

Questions	**CS1**	**CS2**	**ADS**	**Total**
	Avg. (Std.)	Avg. (Std.)	Avg. (Std.)	Avg. (Std.)
Q1	2.2 (1.0)	2.7 (1.2)	2.7 (1.0)	2.5 (1.1)
Q2	2.0 (1.1)	3.0 (1.3)	2.6 (1.1)	2.6 (1.2)
Q3	2.2 (1.0)	1.3 (0.5)	2.0 (1.2)	1.8 (1.0)
Q4	2.1 (1.1)	2.9(1.6)	2.5 (1.1)	2.5 (1.3)
Q5	4.1 (0.9)	3.5 (1.0)	3.9 (1.0)	3.8 (0.9)
Q6	2.4 (1.2)	2.4 (1.2)	2.2 (0.8)	2.3 (1.1)
Q7	3.8 (0.9)	3.0 (1.3)	3.5 (1.1)	3.4 (1.1)
Q8	3.5 (1.1)	2.9 (1.3)	3.6 (1.2)	3.3 (1.3)
Q9	3.2 (1.6)	2.5 (1.2)	2.7 (1.1)	2.8 (1.3)
Q10	3.2 (1.4)	2.6 (1.3)	3.6 (0.8)	3.1 (1.2)
Q11	2.4 (0.9)	1.2 (0.9)	1.0 (0.6)	1.5 (1.0)

CHAPTER V

CONCLUSION

Peer review is a proven educational tool that has been adapted for CS education through the use of several different electronic systems. These systems range from using email based communication for review to developing an entirely new and customized framework for a specific peer review design. Customized systems have been the most successful in terms of ease of use and incorporating peer review into CS courses because of the automated process they provide. But the time it takes to create a system from scratch and maintain it is not practical. For this reason, a customizable system based on an existing CMS was considered in developing a new peer review system in this work.

Our peer review system consists of two main components: the Assignment module and the Peer Review module. These modules are plugins that provide peer review functionality for the Elgg CMS called PeerSpace. The Assignment module handles the management of assignments and storing student submissions, while the Peer Review module handles everything related to the peer review activity. This includes management on the instructor side that allows new peer review assignments to be created and assigned to students, as well as supporting the peer review process for students and delivering completed feedback. The two modules work seamlessly together to create a continuous process for peer review.

Three CS courses participated in a controlled study for the effectiveness of peer review. Each course had a control and experimental section. Only the experimental section used peer review and all other aspects of the course were identical. All of the control sections were compared to the experimental sections by quantitative and qualitative means. This involved analyzing grades and group environment surveys. The sections were further divided into groups identified by lower/upper division and groups pertaining to the students' major. These groups were also analyzed to produce a more detailed result on the effectiveness of peer review in regards to general student attributes.

The results of the analyses performed on the experiment shows peer review benefits students both in learning and in supporting a peer-friendly collaborative environment. The general study, which incorporated all of the control and experimental sections, produced results that back up the claimed benefits of peer review. Students who participate in peer review must evaluate other students' work. This provides an opportunity to develop a greater understanding of the material. Also, the chance to see more solutions to the same programming problem helped students build their skillset for more complex assignments and do better on tests. Results from the CS2 course reinforce this because the experimental section had significantly higher grades in all types of assignments as well as a better sense of community. The growth in sense of community may be attributed to the fact that peer review is a collaborative learning activity among the students. It fosters constructive criticism that pulls students closer together.

Additional detailed results were revealed in the other individual analyses. When comparing lower division students and upper division students, peer review benefited the lower division students more than it did for the upper division students in both areas. This is most likely because lower division students need additional support in courses and have not yet had the opportunity to develop strong social connections with other students. Furthermore, lower division students benefit more from the extra, personalized feedback on assignments because they tend to have more questions or misunderstandings about the material which the instructor's feedback alone cannot always answer. Also, upper division students have already had the chance to meet other students and improve their own social network. Therefore, peer review does not affect upper division students as much as lower division students in regards to social connections. The same result can be seen when comparing the CS majors against the non-CS majors. The non-CS majors benefit less. In general, CS majors benefit more from peer review because they participate in PeerSpace activities more than non-CS majors. This is because students tend to take courses required for their major more seriously than others. Consequently, non-CS majors do not see as many positive changes in performance and community as the CS majors.

The peer review system and aforementioned analyses offer distinctive contributions to the CS and educational fields. These contributions include:

- Developing a peer review system with a complete suite of instructor tools which provide a wider range of control in the peer review process and allow instructors to customize peer review assignments to fit their needs,
- Combining peer review with a social network to create a centralized place for students to complete work and contribute to the community, allowing social and learning components to feed each other and drive students,
- Create a peer review process that is seamless and provides both instructors and students with a continuous feel, and
- The analyses performed in this study provide a better understanding of the effectiveness of peer review on student learning and on promoting collaborative learning than in previous studies.

Future development on this system will give instructors even more control over peer review assignments. Plans include:

- Adding manual pairing of students in peer review so the instructor can assign certain students to review each other if desired, and
- Adding manual assignment of students to peer review categories in order for instructors to select which students should participate in peer review.

More research will be performed on peer review with the aim of validating the results presented here. This will be done by:

- Experimenting with more CS courses to see if they lead to the same conclusions,
- Using different instructors for the same CS courses used in this study to provide additional data, and
- Creating control and experimental groups within the same course to further reduce differences between the two groups.

BIBLIOGRAPHY

[1] Ahoniemi, T., Lahtinen, E., and Reinikainen, T. 2008. Improving pedagogical feedback and objective grading. *SIGCSE Bull*. 40, 1 (March 2008), 72-76. DOI= http://doi.acm.org/10.1145/1352322.1352162.

[2] Anewalt, K. 2005. Using peer review as a vehicle for communication skill development and active learning. *J. Comput. Small Coll*. 21, 2 (December 2005), 148-155.

[3] Bauer, C., Figl, K., Derntl, M., Beran, P., and Kabicher, S. 2009. The student view on online peer reviews. *SIGCSE Bull.* 41, 3 (July 2009), 26-30. DOI= http://doi.acm.org/10.1145/1595496.1562892.

[4] Bloom, B. *Taxonomy of Educational Objectives Book I: Cognitive Domain*. David McKay Co., 1956.

[5] Denning, T., Kelley, M., Lindquiest, R., Griswold, W., and Simon, B. 2007. Lightweight preliminary peer review: does in-class peer review make sense?. *In Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education* (SIGCSE '07). ACM, New York, NY, USA, 266-270. DOI= http://doi.acm.org/10.1145/1227310.1227406.

[6] Dietrich, S., Haag, S., and Folkestad, L. 2008. Quality-based assessment of papers and projects in computer science. *J. Comput. Sci. Coll*. 23, 3 (January 2008), 16-22.

[7] Dong, Z., Li, C., and Untch, R. 2011. Build peer support network for CS2 students. *In Proceedings of the 49th Annual Southeast Regional Conference* (ACM-SE '11). ACM, New York, NY, USA, 42-47. DOI= http://doi.acm.org/10.1145/2016039.2016058.

[8] Drupal. Retrieved February 29, 2012, from Drupal: http://drupal.org.

[9] Elgg. Retrieved February 29, 2012, from Curverider Limited: http://elgg.org.

[10] Elgg Wiki. Retrieved February 29, 2012, from Curverider Limited: http://docs.elgg.org/wiki/.

[11] Gehringer, E. 2001. Electronic peer review and peer grading in computer-science courses. *SIGCSE Bull*. 33, 1 (February 2001), 139-143. DOI= http://doi.acm.org/10.1145/366413.364564.

[12] Gehringer, E. 2003. Building resources for teaching computer architecture through electronic peer review. *In Proceedings of the 2003 Workshop on Computer Architecture Education: Held in Conjunction with the 30th International Symposium on Computer Architecture* (WCAE '03). ACM, New York, NY, USA, Article 9. DOI= http://doi.acm.org/10.1145/1275521.1275534.

[13] Hämäläinen , H., Tarkkonen, J., Heikkinen, K., Ikonen, J., and Porras, J. 2009. Use of peer-review system for enhancing learning of programming. *Ninth IEEE International Conference on Advanced Learning Technologies* (ICALT ’09). 658-660.

[14] Hämäläinen, H., Hyyrynen, V., Ikonen, J., and Porras, J. 2010. MyPeerReview: an online peer-reviewing system for programming courses. *In Proceedings of the 10th Koli Calling International Conference on Computing Education Research* (Koli Calling '10). ACM, New York, NY, USA, 94-99. DOI= http://doi.acm.org/10.1145/1930464.1930481.

[15] Highslide JS. Retrieved February 29, 2012, from Highslide JS: http://highslide.com/.

[16] Hundhausen, C., Agrawal, A., Fairbrother, D., and Trevisan, M. 2009. Integrating pedagogical code reviews into a CS 1 course: an empirical study. *In Proceedings of the 40th ACM Technical Symposium on Computer Science Education* (SIGCSE '09). ACM, New York, NY, USA, 291-295. DOI= http://doi.acm.org/10.1145/1508865.1508972.

[17] Hundhausen, C., Angrawal, A., and Ryan, K. 2010. The design of an online environment to support pedagogical code reviews. *In Proceedings of the 41st ACM Technical Symposium on Computer Science Education* (SIGCSE '10). ACM, New York, NY, USA, 182-186. DOI= http://doi.acm.org/10.1145/1734263.1734324.

[18] Kinnunen, P., and Malmi, L. 2006. Why students drop out CS1 course?. *In Proceedings of the 2nd International Workshop on Computing Education Research*. 97-108.

[19] Li, C., Dong, Z., Untch, R., and Jagadeesh, D. 2011. Preparation station: a practice tool for CS1 and CS2 students in peerspace. *In Proceedings of the 49th Annual Southeast Regional Conference* (ACM-SE '11). ACM, New York, NY, USA, 322-323. DOI= http://doi.acm.org/10.1145/2016039.2016128.

[20] Li, C., Dong, Z., Untch, R., Chasteen, M., and Reale, N. 2011. PeerSpace - an online collaborative learning environment for computer science students. *IEEE 11th International Conference on Advanced Learning Technologies* (ICALT ‘11). 409-411.

[21] MyPeerReview. Retrieved February 29, 2012, from SourceForge: http://sourceforge.net/projects/mypeerreview/.

[22] MyReview. Retrieved February 29, 2012, from SourceForge: http://myreview.sourceforge.net.

[23] pChart. Retrieved February 29, 2012, from SourceForge: http://pchart.sourceforge.net/.

[24] Perez-Prado, A., and Thirunarayanan, M. 2002. A qualitative comparison of online and classroom-based sections of a course: Exploring student perspectives. *Educational Media International*. 39, 2, 195-202.

[25] Reily, K., Finnerty, P., and Terveen, L. 2009. Two peers are better than one: aggregating peer reviews for computing assignments is surprisingly accurate. *In Proceedings of the ACM 2009 International Conference on Supporting Group Work* (GROUP '09). ACM, New York, NY, USA, 115-124. DOI= http://doi.acm.org/10.1145/1531674.1531692.

[26] Repenning, A., Basawapatna, Ashok., and Koh, Kyu H. 2009. Making university education more like middle school computer club: facilitating the flow of inspiration. *In Proceedings of the 14th Western Canadian Conference on Computing Education* (WCCCE '09), Roelof Brouwer, Diana Cukierman, and George Tsiknis (Eds.). ACM, New York, NY, USA, 9-16. DOI= http://doi.acm.org/10.1145/1536274.1536281.

[27] Rovai, Alfred P. 2002. Development of an instrument to measure classroom community. *The Internet and Higher Education*. 5:197-211.

[28] Silva, E. and Moreira, D. 2003. WebCoM: a tool to use peer review to improve student interaction. *J. Educ. Resour. Comput*. 3, 1, Article 3 (March 2003). DOI= http://doi.acm.org/10.1145/958795.958798.

[29] Sitthiworachart, J. and Joy, M. 2004. Effective peer assessment for learning computer programming. *In Proceedings of the 9th annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (ITiCSE '04). ACM, New York, NY, USA, 122-126. DOI= http://doi.acm.org/10.1145/1007996.1008030.

[30] Sondergaard, H. 2009. Learning from and with peers: the different roles of student peer reviewing. *In Proceedings of the 14th annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education* (ITiCSE '09). ACM, New York, NY, USA, 31-35. DOI= http://doi.acm.org/10.1145/1562877.1562893.

[31] Swan, K. 2002. Building learning communities in online courses: the importance of interaction. *Education Communication and Information*. 2, 1, 23-49.

[32] Topping, K. 1998. Peer assessment between students in colleges and universities. *Review of Educational Research,* 68, 3, 249-276.

[33] Trivedi, A., Kar, Dulal C., and Patterson-McNeill, H. 2003. Automatic assignment management and peer evaluation. *J. Comput. Small Coll.* 18, 4 (April 2003), 30-37.

[34] Trytten, D. 2005. A design for team peer code review. *In Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education* (SIGCSE '05). ACM, New York, NY, USA, 455-459. DOI= http://doi.acm.org/10.1145/1047344.1047492.

[35] Waite, W. M., and Leonardi, P. M. 2004. Student culture vs. group work in computer science. *In Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*. 12-16.

[36] Wang, Y., Yijun, L., Collins, M., and Liu, P. 2008. Process improvement of peer code review and behavior analysis of its participants. *SIGCSE Bull*. 40, 1 (March 2008), 107-111. DOI= http://doi.acm.org/10.1145/1352322.1352171.

[37] Wolfe, W. 2004. Online student peer reviews. *In Proceedings of the 5th Conference on Information Technology Education* (CITC5 '04). ACM, New York, NY, USA, 33-37. DOI= http://doi.acm.org/10.1145/1029533.1029543.