Preprint version of the publication. Use following information as referencing information:
Harri Hämäläinen, Ville Hyyrynen, Jouni Ikonen and Jari Porras , APPLYING PEER-REVIEW FOR PROGRAMMING ASSIGNMENTS, International Journal on Information Technologies & Security, No 1, 2011, pp. 3-17, ISSN 1313-8251

APPLYING PEER-REVIEW FOR PROGRAMMING ASSIGNMENTS

Harri Hämäläinen, Ville Hyyrynen, Jouni Ikonen and Jari Porras

Lappeenranta University of Technology; P.O.Box 20, FIN-53850 Lappeenranta
e-mails: harri.hamalainen@lut.fi, ville.hyyrynen@gmail.com, jouni.ikonen@lut.fi, jari.porras@lut.fi
Finland

Abstract: Peer-review can be used to provide more accurate feedback for students without straining teachers more on courses with large number of students. We implemented a web-based peer-review system for software assignments and released it as open-source. The usability of the system and peer-review process were examined in a Web programming course. Recent studies describe methods of analyzing peer-review data to produce better accuracy and reliability. We carried out a questionnaire concerning students' experiences and produced statistics for analysis of student assessments. Students found the system usable and the feedback received via the peer-review process useful.

Key words: peer-review, student assessment, programming.

1. INTRODUCTION

The principles of peer-reviewing are simple and easily adaptable for various contexts. Use of student peer-review has been studied in numerous educational institutions but according to Garousi [1] it has not been used on many design-oriented engineering courses. Motives for its use derive from the observations that peer-reviewing can, for example, boost learning outcomes and provide accurate results in estimating the quality of submissions. In this paper we present analysis of the system and experiences on a Web programming course.

The beneficiaries of peer-reviewing can be divided in two parts: a) the students who learn from others' solutions and received feedback and b) the teachers who can get assistance for the evaluation. This should motivate the teachers to use peer-review especially on popular courses. Large number of students in a single course typically creates a problem that teacher(s) have time for evaluation only, not giving proper feedback of the solutions. In this case other students can be utilized to help each other. While using peer-review the increasing number of students does not affect to the workload of the students as they review a fixed number of tasks.

To follow through with the peer-review process in a course, information systems can be effective. However, using software applications the usability has to be carefully considered. Problems with the tools and methods for conducting the process can sometimes surpass the gained benefits. It is unreasonable to expect people to spend much time to learn system functionality when that time and effort could be allotted to actual work. In order to have a positive effect on students' learning experience the system providing the framework for the process should a) minimize the requirements needed to learn and use the system, b) perform well under load and volumes of data and c) be simple to use to minimize the impact of having to learn a new system. In other words, the overhead caused by the system in performing the required use cases should be minimized. Majority of users may use the system only once or twice during their studies so their time should be spent efficiently. Therefore usability and especially learnability play an important role when using this type of application. To motivate the students to complete their peer-review tasks meaningfully the process itself has to be efficient and support their learning.

This paper presents introduction to related work in the following chapter and then introduces the application we have developed for carrying out peer-review process in chapter 3. In chapter 4 we discuss on issues using peer-review on a web programming course. Experiences of deployment on a web programming course both from system usability and peer-reviewing process approach are presented in chapter 5. The last chapter provides a conclusion of our paper.

2. RELATED WORK

In case of student assessment in programming courses the peers are exposed to various styles of coding and asked to give critical evaluation of others' work. This can give students an opportunity to improve their own proficiency as a programmer, and deepen their understanding of used languages and techniques. Also, the time spent on studying the solutions in practice is always beneficial when teaching new concepts.

Seeing different approaches is particularly insightful when handling a common problem that has "solutions that meet the requirements" instead of only a "single correct answer." The teacher can present a few examples of such solutions, but it is essential for the students to realize the multitude of approaches. Sometimes, an absolute measurement of a solution's suitability is hard to determine – its suitability can depend on factors like compatibility, reliability, optimized execution time, scalability etc. Often in practice, finding the best possible solution is not even desired, but instead, producing a working solution under the given parameters in time is all that counts.

Studies note that students find the opportunity to inspect peers' solutions interesting because it can e.g. give them new ideas and a chance to learn analytical abilities from varying styles in solving problems [2]. Increased feedback about the assignment is usually appreciated especially if the teachers are only capable of giving out limited reviews. Also, peer-reviewing can incite peer interactivity within the group and provide ground for development of a community of active learning. Students stuck with a problem can seek the help from group in form of suggestions and critical evaluation of their solutions. The group can then confirm good approaches and point out flawed ones, for example.

From the peer-reviewing point of view growing numbers of students can be turned into an advantage. For example Bouzidi & Jaillet [3] have reported that students can provide reliable and accurate results when group sizes are large (242 students in their case) giving a chance to assign enough peer-reviews for each. However, while they applied the process in a very controlled way on the assessments of mathematics exam solutions, the process we describe is used on evaluating student submissions on a programming course. It still applies, though, that the process will scale without burdening the individuals; a student's workload along with the accuracy of student assessments depends on the ratio of authors per reviews. More students mean more work to the staff, but adding a peer-review process into a content of a course barely adds any work.

Studying the validity of peer-review can be straightforward. At its simplest it can be done by taking the combined results of a group's evaluation of a given peer solution, and then comparing it to the teachers' evaluation. The process can be fine-tuned to produce even more accurate results by using different algorithms. Hamer et al. [4] have introduced an automatic calibration algorithm where the calibration factor is calculated based on the difference between the grades assigned by the reviewer and averaged grades thus emphasizing the consensus of the class. Loll et al. [5] have proposed counting weights to reviews and also compared the reviews to teachers' reviews to estimate their knowledge of the subject. Increasing the number of reviewers and grouping reviewers evenly by the measured quality of their reviews may also be used to improve the quality and motivation of peer-reviewing [6, 7].

The problem with the skill level differences among the students can substantiate in several ways. It can be argued that novices benefit the most from the process, whereas advanced students are only contributing while not getting much from their less adept peers. However, for the best learning outcomes, students should be exposed to good solutions, as it can be true that reviews of poor solutions mostly benefit the receiving author. Ranking the students by performance and arranging them in an optimal way is one way of fighting the problem of uneven reviewer groups.

To get valuable reviews motivation of the students is important. On students' perspective, making the most of the process often depends on personal attitudes and opinions. At a glance, peer-reviewing can seem as extra burden for a student who feels his input mostly benefits everyone else, namely, the staff and the peers – not the reviewer himself. In case the student is committed to improve his skills only to pass the course requirements the claim can be valid regardless of the arguments for its use. It is hardly ever a good idea to simply force people to adopt a process and expect them to contribute in a meaningful and productive way. Instead, the key is to inform the participants of the purpose of the system, emphasize the benefits, assess the concerns involved with the process, and using tools that enable the process as unobtrusively as possible. To motivate the students to put effort into their reviews, the reviews can also be rated by the receiver, the teacher or even by some other student as an evaluator, as proposed in [8]. In fact, evaluating the quality of reviews can be a key factor in motivating students to give proper assessments.

Peer-review can also be used to improve course material. Reusable learning objects have been considered to be used to create reviewed material to support and to be used in improving teaching and examples shown for the students in the course on following years as proposed e.g. on [9]. The reviewable content has not to be limited to a written text, but programming code can be re-used and elaborated as well. An issue to be considered to serve also students' needs

Preprint version of the publication. Use following information as referencing information:
Harri Hämäläinen, Ville Hyyrynen, Jouni Ikonen and Jari Porras , APPLYING PEER-REVIEW FOR PROGRAMMING ASSIGNMENTS, International Journal on Information Technologies & Security, No 1, 2011, pp. 3-17, ISSN 1313-8251

is if they could somehow use the given feedback, not just the improved objects, for further purposes such as advertising their know-how based on favorable feedback. This could also be one way to motivate the students for better quality peer-reviewing.

It is not always clear how to label solutions good or poor. For example, a perfectly working solution can have hopelessly obfuscated source code while an apparently well-formatted and syntactically correct code can be riddled with logical mistakes and unacceptable features in its design (wasteful use of resources, blatant disregard of security and so on). Clearly, identifying and assessing problems like these can be completely out of reach for a novice. One method of dealing with this is to allow students to respond on their own level and use fine grained criteria in the reviews. Comments and ratings on different aspects of the submission can help them to analyze the solution with at least some success. Becker [10] has found rubrics useful in assuring reliable and consistent grading when using multiple teaching assistants for grading same projects. The same approach would ease and improve the grading when the evaluation and grading is completed by students in peer-reviewing. Reily et al. [11] have broken down the reviews in code formatting, conventions, documentation, interface design, and modularity.

Negative attitudes toward peer-reviewing can emerge from other reasons as well, such as feeling that the received feedback is of little use. People receiving only superficial or neutral comments may not find them helpful in improving their work, especially if it appears to be dishonest and undeserved. One source of this problem can be peers' reluctance to give critical feedback on others work. Reasons for this can be, for example, fear of retaliatory reviews, unwillingness to make critical judgments-based on lack of expertise, collusion among students, or plain laziness. The effects of "rogue reviews" are discussed in more detail by Reily et al. [11].

Students' tendency to overlook problematic parts in their peers' work can be addressed by several ways such as training, and showing examples of good reviews. The performance of the group can be calibrated by having students to do preliminary reviews on example solutions prior to starting actual exercises. Studies discussing biased reviews also mention the importance of double-blindness in combating the effect of personal factors [12].

2.1. Applications for code peer review

When we analyzed the features that a peer-reviewing system for programming courses should fulfill, several requirements were listed. First of all it should be an open-source web-based standalone system which should be usable with the most common browsers and it should not require any extra plug-ins to be installed. The submissions and students should be able to remain anonymous through the whole process while teachers need to be able to find out real identities and need to have a chance to control who is reviewing whose task. Since programming assignments may require submitting several separate files, multiple file upload for a single assignment has to be supported. The peer-review has to be completed in the system and teachers need to have a tool for creating evaluation form, which supports different types of input fields, both textual and numerical. Statistics about the reviews, such as averages of evaluations and the time used for reviews should be available for the teacher. To support and control the peer-review process, the system has to support pre-defined deadlines for submission and evaluation. Based on these requirements we begun our survey on existing applications.

There are currently several systems designed for managing user submissions for the purpose of student assessment. Apparently the first peer review system using Web for both submission and review was Peer Grader (PG) system [13]. This Web-based Java application has later been used as a part for submitting and reviewing purposes in the Expertiza platform [8] used for producing reusable learning objects.

PeerWise [14] is a peer-review system administrated by the University of Auckland. The system allows students to submit answers to questionnaires created by their peers on given topics. The platform supports discussion and evaluation of the student-created questionnaires. Students are expected to be self-reliant in using the system – after initial setup the staff's involvement with the process is minimal. Using PeerWise is free but it requires contacting the administrators.

The University of Melbourne utilizes web-based PRAZE in their teaching in a variety of learning environments. It is currently being used and developed only at Melbourne, and studies [15, 7] have been published regarding the system and the use of peer-review. The system enables students to submit their work in form of an essay, a report, multimedia content, or any other type of file into the system for peer-review. Students can work as individuals or groups, and they can also rate the reviews according to given criteria. The lecturer can allow "group self assessment" to let the peers assess the contribution of the other members in the group.

iPeer [16, 17] is an open-source peer-review system aimed for filling evaluations. It however lacks the possibility to upload files for evaluation and was therefore left outside of our study. Another open-source system called Aropä [18] is preferably offered as a hosted service and seems to contain most of the features we require, but we were not able to test it in practice.

Instead of using a standalone application for peer-reviewing, some more comprehensive Course or Learning Management Systems (CMS, LMS) also include peer-reviewing activities. Moodle [19] is one of the open-source CMS targeted for educational use. Lykourentzou et al. [20] describe its use as a platform for a peer-review process in "an introductory level e-learning course on Web Design". They found the presented system user-friendly and providing the necessary functionality to deliver the procedure of peer assessments. Using the modules for peer reviewing requires installation of the whole Moodle platform and was therefore rejected in our case.

Each of the studied systems had some disadvantages from code peer review not suiting extremely well for our requirements and evaluation of programming assignments. Due to these problems we implemented a new open-source peer-review system, MyPeerReview.

3. INTRODUCTION OF MYPEERREVIEW SYSTEM

MyPeerReview is a custom built module to Drupal to provide special functionality for peer-reviewing, and to alter the behavior of the other modules. The source code is published as open-source and it is available to download at [21]. More detailed description of the system can be found also in [22]. The system was designed to enable the use cases involved with running the process of peer-review and it has been tested on the LAMP stack using the latest versions of common browsers. The system consists of five distinct content elements: course, exercise, solution, review form, and review. The elements map to the underlying Drupal framework so that they can be handled as native data structures. However, the peer-review process requires bookkeeping of special relationships of elements. Therefore those are stored and handled as records in their own tables.

Although the underlying framework allows virtually any modifications there are some restrictions on how courses and exercises can be set up and run. In MyPeerReview several courses can be set up, each having its own group of students completing exercises. A student can take part in any number of courses. All exercises that use peer-review contain a reference to a designated review form which is used to submit the reviews. Thus, a new, blank form must be created for each such exercise. Figure 1 presents some of typical review form components. Using the administrative tools the teacher can assign review tasks for any combination of the course participants regardless of whether or not they submit their own solution.

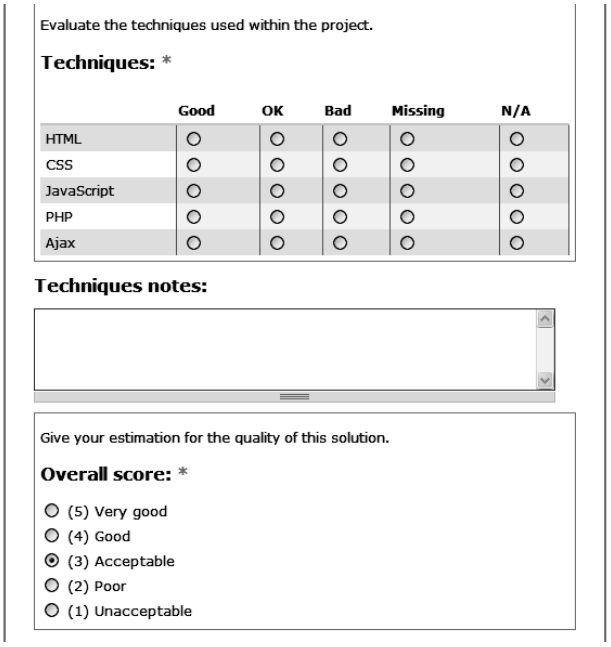

Fig. 1. An example review form that contains a radio-button matrix, free text field, and a selection list

Preprint version of the publication. Use following information as referencing information:
Harri Hämäläinen, Ville Hyyrynen, Jouni Ikonen and Jari Porras , APPLYING PEER-REVIEW FOR PROGRAMMING ASSIGNMENTS, International Journal on Information Technologies & Security, No 1, 2011, pp. 3-17, ISSN 1313-8251

The system has a relatively simple scheme for student navigation. We attempted to minimize the number of views to the system and gather all the central links and resources in one page, the student home view. A student's review task in progress is presented in figure 2. The view displays a personal submission (Exercise 1), anonymous submissions of other students to be reviewed (Reviews) and a possibility to submit the next personal task (Exercise 2).

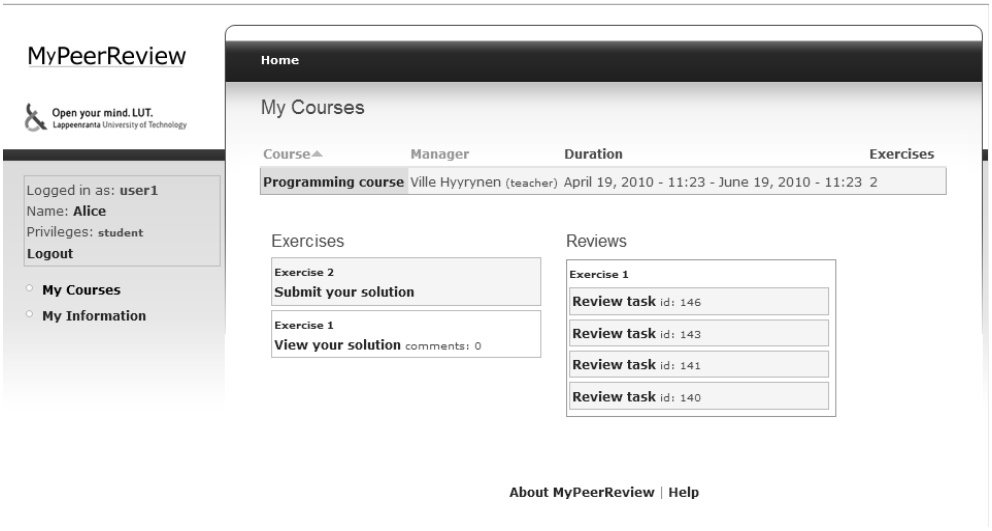

Fig. 2. Partial view of a student review task in progress

While reviewing the anonymous submissions of other students the student is expected to inspect each submission, fill in the form accordingly, and then submit the form. To reduce the required steps in small assignments for reviewing, the system is capable of displaying highlighted source code in a pop-up window. When he has completed the review he redirects back to the main view and receives a notification of a completed review. The details of the process are listed below.

1. Teacher opens the exercise for solutions
2. Students submit their solutions
3. Teacher sets the exercise on hold, and
   3.1. examines and accepts the submissions
   3.2. assigns review tasks
4. Teacher opens the exercise for reviews
5. For every assigned review task, each student
   5.1. fetches the solution
   5.2. evaluates the solution
   5.3. submits the review
6. Teacher sets the exercise on hold, and
   6.1. examines and accepts the submitted reviews
7. Teacher completes the exercise
8. Students access the reviews they received

4. REVIEW PROCESS IN WEB PROGRAMMING COURSE

Deployment of the MyPeerReview-system was conducted as a part of bachelor-level Web programming course (worth 3 ECTS) that concentrated on the basics of HTML, CSS, JavaScript, PHP and Ajax techniques. The students were expected to have at least some experience in programming and adequate skills in using web-based systems, but the preliminary knowledge of the students varied a lot. In addition to the weekly practical assignments the course included a personal project on which we applied the peer-review process. The topic of the project was freely choosable by student but it had to be approved by the teacher. The guidelines were loose and offered a chance to use

any suitable methods, as long as the project a) was done using web programming techniques, b) used a database, c) implemented user management in some form, and d) was set up online accessible from the university's network.

Participation to peer-review process was required in order to pass the course, but the students were informed it would not affect the final grades. The hours spent in peer-reviewing were, however, taken into account in the course workload. The submission of projects began immediately after the project deadline and because of the course schedule and the oncoming end of semester, the phase was very short, lasting only a day. In this phase the students were instructed to 1) register an account in the MyPeerReview system and 2) submit their project there. In the end, total of 24 students managed to complete the steps in time. The submissions included uploaded archive file of all of the source codes, project report having five pages or less, and a URL to the application with possible username and password combination(s) for testing purposes.

The review phase started moments after the submission phase closed. The entries were inspected only superficially by teachers just to confirm no files or URLs were missing. First, each of the 24 authors was assigned five randomly chosen peers for review. In their reviews the students were expected to read the project report, test each application online, download the files and assess various aspects of the submission. Since the projects contained several files and hundreds of lines of code on average, the reviewers were instructed to spend half an hour for each review and not to perform a line-by-line inspection, but to study the code just to get a sufficient view on its structure and design.

The students were given nine days to complete the reviews. The review form that was filled for each submission can be divided in 3 major sections: program analysis, used techniques and overall score. The evaluation was constructed out of multiple-choice questions (mandatory) and open feedback as text fields (optional) so the students could explain the rationale behind their ratings. Numerical scales were labeled Very good (5), Good (4), Acceptable (3), Poor (2), and Unacceptable (1). An option (N/A) was included for unexpected cases in which reviews could not be performed, so the overall results would not be distorted by technical problems experienced by individuals. The overall score had no option for problematic submissions so the students were instructed to rate those Unacceptable (1).

After the deadline for peer-reviewing the system was closed for any changes. The results were collected and authors were able to access the reviews of their work. By the end all the reviews of the 24 students were completed resulting 120 peer-reviews. In addition, we had four other students who had submitted their tasks late. They participated in peer-reviewing process as their own group, but their reviews were not taken into account when analyzing the results in this paper.

5. RESULTS

After the peer-review process was completed, we analyzed the results by going through the written comments and comparing the numerical evaluations to our evaluations. We asked for feedback about the interface and other aspects of the peer-review process in a post-test survey. Comments, opinions, and ratings were collected regarding 1) usability and given instructions, 2) review questions, 3) anonymity, and 4) peer-review as a teaching method.

5.1. Evaluations of received peer-reviews

While analyzing the overall grades students had given to each other we had one major challenge. As previous studies have proven, students seemed to evaluate the works gently thus decreasing the grade variance between the evaluations. More than half (58%) of the students got the second best grade when counting the average of received peer-reviews, and there were no lowest grades given at all. Figure 3 presents the comparison between the grades given by teachers and students.

Preprint version of the publication. Use following information as referencing information:
Harri Hämäläinen, Ville Hyyrynen, Jouni Ikonen and Jari Porras , APPLYING PEER-REVIEW FOR PROGRAMMING ASSIGNMENTS, International Journal on Information Technologies & Security, No 1, 2011, pp. 3-17, ISSN 1313-8251

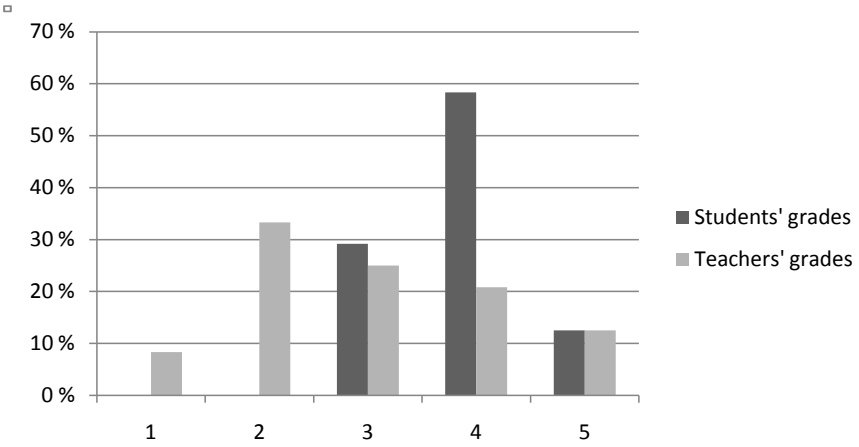

Fig. 3. Grades of the assignment

Instead of normalizing the values we put the assignments in order based on average of given grades in peer-review and compared these results to teachers' ranking. In addition, we also calculated the average grades using weighted base ratings. Instead of using the teachers' evaluation as a quality measure giving more weight to the students who have completed their task better as introduced in [5], we counted this factor using the grades received from the peer-reviews. Our comparison pointed out that there was no remarkable difference in factors whether we were using either of these as the quality measure. Weighted evaluation slightly improved the results by reducing the difference of average between teachers' and students' evaluation. We believe that peer-reviewing can be used and improved for evaluation without teacher's participation when the evaluation pattern is well defined. When comparing the rating based on teachers' and students' overall evaluation of the assignments presented in table 1, the results follow each other. However, there are a couple of significant differences (students #7 and #17 in table 1) in rating between students and teachers. Based on re-examination of these assignments and comparing the overall grades to the technology-based evaluation, it seems that these differences are a consequence of application layout and design, either good or bad, which were not listed as a part of rating arguments.

Table 1.Differences between teachers' and students' rankings of the submissions

Student ID	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Teacher rankings	1	1	1	4	4	4	4	8	9	10	10	10
Student rankings	1	2	3	7	8	8	15	11	11	3	3	8
Weighted student rankings	1	2	3	4	8	10	15	13	11	5	7	9
Student ID	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
Teacher rankings	13	14	15	15	17	17	17	17	17	17	23	23
Student rankings	18	16	18	18	3	11	11	16	22	23	21	23
Weighted student rankings	19	16	20	18	4	14	12	17	22	23	21	24

Double-blindness was not completely achieved in this test. Although we told the students to remove names from their peer-review submissions it turned out only a few of them did so. We believe the main reasons for this were that

a) they did not bother to prepare two versions of the project (for the teacher and for the peer-reviews), b) many had set up their project on a personal web space (containing revealing addresses), and c) the applications were anyway publicly presented in the class a few days before the peer-reviews. However, many of the students still gave support to anonymity although all the respondents felt that students' identity had not had an effect to the given or received grade. The reviews in this test were anonymous.

5.2. Student feedback

We gave two weeks of time for students to fill out a questionnaire concerning the peer-review process on the course. 16 students (57% of the students who completed the practical assignment) replied with comments and we analyzed their responses. The student feedback about the MyPeerReview system was promising and indicated that the design of the system was working short of the spotted bugs. Having some issues with forms and the interface design in the earlier test, we saw no failures in the student input this time. Most of the complaints concerned site navigation and some students had struggled with the interface and had to "click around" to find what they were looking for. Due to this, 25% of the respondents estimated the site navigation "below neutral". Evidently, this did not prevent them from completing the tasks correctly. Although we experienced some problems due to bugs in the system, which we managed to fix on the fly, all participants were capable of registering and uploading their work through the web-based interface as instructed.

The teachers' assessment of the system suitability was definitely accepting. With appropriate modifications and additions, the process can be adapted in programming courses for the purposes of teaching and analyzing student performance.

More than 75% of the respondents considered the received reviews helpful. According to the survey the most useful aspect to the students in peer-reviews is textual responses with reasoned comments. Discrete analysis of features may be helpful too, but mainly as a method of quick technical assessment of the submission. In terms of learning outcomes the preference of properly explained textual assessments is clear. Even a handful of succinct comments can help when combined with other reviews. Even though the students had no choice but to take part of the peer-reviews, they reported they appreciated the opportunity to have a look at the variety of different approaches and receiving feedback. On the other hand the complaints also indicate that students prefer to receive written explanations from their peers. The positive feedback on the review process indicates that the majority of students did not consider the peer-review too burdening.

6. CONCLUSION

Peer-review systems have not usually been used in context of programming and since we found no open-source systems which would have fulfilled our requirements, we implemented such a system. Our experiments show that the system that was implemented can be used to run and administer programming review process. This paper presented a short description and an initial evaluation of the system.

Students found the use of peer-reviewing system for programming assignments useful. Feedback indicates that review of other's code and received comments are helpful. However, numerical feedback given by the students and teachers were clearly on different scales, which leads us to think that evaluation criteria must be defined more clearly and evaluations should also be rated. More attention must also be paid to provide written feedback which is considered the most valuable part by the students. This can be promoted by giving a change to evaluate the received review and by requiring the peer-reviewers to include certain aspects to their reviews.

To get evaluation that is usable for teachers evaluation arguments have to be split into pieces and asked separately from students, or the arguments have to be reported punctually in other means. Single overall score, which is easily based on feeling or emotion, is not precise enough, especially with inexperienced students.

REFERENCES

[1] Garousi, V. Applying Peer Reviews in Software Engineering Education: An Experiment and Lessons Learned. *IEEE Transactions on Education*, Volume 53, Issue 2, May 2010, pp.182-193, ISSN 0018-9359.

[2] Diefes-Dux, H.A., Verleger, M.A. Student reflections on peer reviewing solutions to Model-Eliciting Activities. *39th IEEE Frontiers in Education Conference*, San Antonio, TX, USA, October 18-21, 2009, pp.674-680, ISSN 0190-5848.

[3] Bouzidi, L., Jaillet, A. Can Online Peer Assessment be Trusted? *Educational Technology & Society*, Volume 12, 2009, pp.257-268.

[4] Hamer, J., Ma, K. T. & Kwong, H. H. A method of automatic grade calibration in peer assessment. *Australasian Computer Society Education Conference'*, 2005, pp.67–72.

[5] Loll, F., Pinkwart, N. Using Collaborative Filtering Algorithms as eLearning Tools. *Proceedings of the 42nd Hawaii International Conference on System Sciences*, Waikoloa, Big Island, Hawaii, Jan 5-8, 2009, pp.1-10.

[6] Hundhausen, C., Agrawal, A., Fairbrother, D., Trevisan, M. Integrating Pedagogical Code Reviews into a CS 1 Course: An Empirical Study. *ACM SIGCSE Bulletin*, Volume 31, Issue 1, pp.291-295, ISSN 0097-8418.

[7] Søndergaard, H. 2009. Learning from and with Peers: The Different Roles of Student Peer Reviewing. *Proceedings of the 14th annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education*, *ITiCSE '09*, Paris, France, July 6–9, 2009, pp.31-35. ISSN 0097-8418.

[8] Gehringer, E. F., Ehresman, L. M., and Skrien, D. J. Expertiza: Students helping to write an OOD text. *Conference on Object Oriented Programming Systems Languages and Applications*, Portland, Oregon, USA, 2006, pp.901-906, ISBN 1-59593-491-X.

[9] Gehringer, E., Ehresman, L., Conger, Susan G., and Wagle, P. Reusable Learning Objects Through Peer Review: The Expertiza Approach. *Innovate: Journal of Online Education* 3 (5), June/July 2007.

[10] Becker, K., Grading Programming Assignment using Rubrics, in *Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education*, June 30- July 2, 2003, Thessaloniki, Greece, pp. 253.

[11] Reily, K., Finnerty, P. L., Terveen, L. Two Peers are Better Than One: Aggregating Peer Reviews for Computing Assignments is Surprisingly Accurate. *Proceedings of the ACM 2009 international conference on Supporting group work*, Sanibel Island, Florida, USA, May 10-13, 2009, pp.115-124, ISBN 978-1-60558-500-0.

[12] Sitthiworachart, J., Joy, M. Effective Peer Assessment for Learning Computer Programming. *ITiCSE '04*, 2004. pp.122-126, ISBN 1-58113-836-9.

[13] Gehringer, E. F. Electronic peer review and peer grading in computer-science courses. *ACM SIGCSE Bulletin*, Volume 33, Issue 1, March 2001, pp.139-143, ISSN 0097-8418.

[14] Denny, P., Luxton-Reilly, A., and Hamer, J. The PeerWise system of student contributed assessment questions. *Proceedings of the Tenth Conference on Australasian computing education*, Wollongong, NSW, Australia, January 01, 2008, pp.69-74.

[15] Pearce, J., Mulder, R. and Baik, C. Involving students in peer review: Case studies and practical strategies for university teaching. The University of Melbourne. 2009.

[16] iPeer. [Online] http://burrito.olt.ubc.ca/trac.

[17] Spiridonoff, S. iPeer Software: Online Rubric-Based Peer Evaluation. *8th Annual WebCT User Conference*. Chicago, IL, USA, July 10-14, 2006.

[18] Hamer, J., Kell, C., and Spence, F. Peer Assessment Using Aropä. *Proceedings of the ninth Australasian conference on Computing education*. Ballarat, Victoria, Australia, 2007, pp.43-54.

[19] Moodle.org: open-source community-based tools for learning. [Online] http://moodle.org.

[20] Lykourentzou, I., et al. An Online Peer Assessment Platform for e-Learning Environments. *Proceedings of The 5th E-learning Conference on Learning and the Knowledge Society*, Berlin, Germany, August 31–September 1, 2009. e-Learning'09.

[21] MyPeerReview project at Sourceforge. [online] http://sourceforge.net/projects/mypeerreview/.

[22] Hyyrynen, V., Hämäläinen, H., Ikonen J. MyPeerReview: An Online Peer-Reviewing System for Programming Courses. *10th Koli Calling International Conference on Computing Education Research*, Koli National Park, Finland, October 28-31, 2010.

Information about the authors:

Harri Hämäläinen - M.Sc, Lappeenranta University of Technology, +358 5 621 2828, harri.hamalainen@lut.fi. Harri Hämäläinen is a Ph.D. student. He has worked with wireless networks, RFID and IT-solutions for construction industry. His current research interests include technology enhancing learning tools and e-learning.

Ville Hyyrynen - M.Sc, ville.hyyrynen@gmail.com. Ville Hyyrynen is currently working at Mfabrik Production Oy as a software engineer. He has been doing research and client projects for the mobile web. His interests include user interface design and usability, content management frameworks, and future web technologies.

Jouni Ikonen – DSc. (Tech.), Lappeenranta University of Technology, +358 50 555 0665, jouni.ikonen@lut.fi. Jouni Ikonen is an Adjunct Professor of Wireless Services and Open Access Networks. He has worked with image processing, distributed simulation, municipal networks and wired and wireless networks. His current research interests include technology enhancing learning tools, network protocols, open access networks and network services.

Jari Porras – DSc. (Tech.), Lappeenranta University of Technology, +358 400 555 427, jari.porras@lut.fi. Jari Porras is a Professor of Distributed Systems and Wireless Communications. His interests include wireless networks, ad hoc networking, peer-to-peer computing, aspects of grid computing, as well as distributed computing and environments. He acts in the Wireless World Research Forum (WWRF) and in the eMobility consortium.