Je-LKS
Journal of e-Learning and Knowledge Society
The Italian e-Learning Association Journal
PEER REVIEWED PAPERS
Vol. 13, n.3, 2017
ISSN: 1826-6223 | eISSN: 1971-8829

PEER ASSESSMENT TO PROMOTE DEEP LEARNING AND TO REDUCE A GENDER GAP IN THE TRADITIONAL INTRODUCTORY PROGRAMMING COURSE

Antonella Carbonaro
Mirko Ravaioli

Dipartimento di Informatica - Scienza e Ingegneria
University of Bologna, Italy
antonella.carbonaro@unibo.it

Keywords: Peer assesment, Gender gap, Programming skills.

Peer assessment mechanism is becoming increasingly important in different computer programming courses to give students opportunities to learn from each other, to improve the learning experience and to reach efficient learning outcomes. Nevertheless, new methodological approaches developed to help students in computer programming courses have had to deal with a thorough study of the aspects related to the gender difference in this context.

The paper presents a web-based system, which improved students' program skills by reviewing peers' source codes and delivering feedback to peers. The purpose was to provide evidence that peer assessment in computer programming has a positive instructive effect. Moreover, we want to investigate if there is a gender gap in the traditional introductory programming course and how can we use peer assessment principles to reduce it.

for citations:
Carbonaro A., Ravaioli M. (2017), *Peer assessment to promote Deep Learning and to reduce a Gender Gap in the Traditional Introductory Programming Course,* Journal of e-Learning and Knowledge Society, v.13, n.3, 121-129. ISSN: 1826-6223, e-ISSN:1971-8829
DOI: 10.20368/1971-8829/1398

1 Introduction

Programming skills are becoming ever more important, quickly turning into the core competency for all kinds of 21st Century workers. Therefore, programming skills have become a core competence especially for engineering and computer science students; that inescapable fact is leading individuals to seek out new ways of learning to code. However, learning to code is usually challenging and difficult for most students, because it involves comprehension and a lot of practice about a range of theoretical background, semantic and syntactic knowledge, coding and algorithmic skills (Yang *et al.*, 2015). To cope with such difficulties, various teaching strategies and learning activities have been applied to support programming courses, including mechanisms to ensure continuous assessment during a programming course to guarantee enough practice as well as give feedback on the quality of student's solutions. But providing quality assessment manually for even a small class requires time and when the class size grows, the amount of assessed work has to be cut down or rationalized in some other way.

Automatic assessment of programming assignments is one major task in programming classes used to evaluate and mark student's programming exercises. Relying on computer assistance allows for instant feedback without the need to reduce exercises. Research in the context of automatic programming assessment has a long history. It has been of interest to computer science educators since the 1960s and has continued to gain vast attention until present.

Students learn programming skills when they review code, write and read comments and see how other students tackle the same problems. These practices/skills are necessary also to work in a team environment in their future careers.

Peer assessment has been used successfully in different computer programming courses for many years to get students opportunities to learn from each other. In those situations, each student acts as both an author and a reviewer. Positive feedbacks report better learning experience and efficient learning outcomes (Wang *et al.*, 2012); in fact, when students evaluate each others' work they think more deeply, learn to criticize constructively, and display important cognitive skills such as critical thinking.

Several authors have found that female students are much less confident in their programming abilities than male students. So, if we want to involve more women in computing, the pedagogy of introductory programming courses needs to change. Although there have been several studies about gender differences in introductory programming, to our knowledge no one study compared the effect of the peer assessment.

In this paper, we propose a web-based programming-assisted system, which improved students' programming skills by reviewing peers' codes and delivering feedback to peers. The purposes were to investigate the extent that peer assessment in a programming course promotes deep learning, to assess the accuracy of students' judgements during a peer assessment exercise and to provide evidence that peer assessment in computer programming has a positive instructive effect. Moreover, our aim in this study is to answer the following research questions: is there a gender gap in the traditional introductory programming course? And, if the answer is affirmative, can we use peer assessment principles to reduce it?

The paper is organized as follows: the next Section introduces the literature related to automatic programming assessment in the context of programming assignments and explores research efforts related to the gender difference in this context. Section three describes our implemented web-based programming-assisted. Section four proposes the conducted questionnaire survey to evaluate the system. Finally, Section five provides some considerations on case study and experimental results.

2 Literature review

In the context of programming assignments, several approaches to automatic programming assessment can be found in related literature; they are typically based on either static analysis or dynamic testing. This refers to whether a program needs to be executed while it is being assessed and focus on which features of programming assignments are automatically assessed. Dynamic analysis (assessment based on executing the program) is often used to assess functionality, efficiency, and testing skills, while static checks that analyze the program without executing it are used to provide feedback on style, programming errors, software metrics, and even design. Tools that cover both static and dynamic testing are also well presented in the survey (Ala-Mutka, 2005).

On top of that, the assessment process can be done by looking into a code structure (white-box) or simply based on a functional behavior of a program (black-box) (Gupta & Dubey, 2012). Output comparison is the traditional approach used by many of the systems we found (Ihantola *et al.*, 2010; Ala-Mutka, 2005) already reported several variations of output comparison including running the model solution and student's code side by side and the use of regular expressions to match the output

In the peer assessment process, students are involved both in the learning and assessment processes. Peer assessment plays an extremely important role in helping students see work from an assessor's perspective, with potential additional benefits (Kulkarni *et al.*, 2015). For example, it exposes students to solutions, strategies, and insights that they otherwise would likely not see. Evaluating peers' work also helps students reflect on gaps in their understanding, making them more resourceful, confident, and higher achievers. Peer assessment has been used for many different kinds of assignments, including design, programming (Chinn, 2005; Wang *et al.*, 2015) and essays (Kulkarni *et al.*, 2015).

We want to use peer assessment to promote learning in programming courses rather than for summative assessment (Sitthiworachart & Joy, 2003) remark that "peer assessment is not only a tool to provide a peer with constructive feedback which is understood by the peer. Above all, peer assessment is a tool for the learner himself."

There are many successful examples of new methodological approaches developed to help students in computer programming courses due to growing interest in learning and teaching programming, however, they rarely deal with the aspects related to the gender difference in this context. In fact, the number of women involved in computer science is surprisingly low (Rubio *et al.*, 2015). In the United States only 0.4 percent of girls entering college intended to major in computer science in 2013 and they made up 14 percent of all computer science graduates, down from 37% in the mid-80s (Patitsas *et al.*, 2014; Tiku, 2014).

Other studies show similarly disproportionate ratios of participation between male and female students in computer science programs (Stoilescu & Gunawardena, 2010). This problem is global: a study conducted on the use of computers and the Internet among fifteen-year olds showed that boys report using computers more often than girls in the vast majority of the 40 countries under investigation (Rubio *et al.*, *op. cit.*).

3 System framework and functions

Practice is very important in acquiring programming skills and there should be room for mistakes and learning from them. Automatic assessment can help as it can give feedback despite the limited human resources. Peer assessment has been successfully used to get students opportunities to learn from each other.

Besides, one of the biggest difficulties for most programming courses is to get and keep the focus and commitment of the students right from the beginning and constantly throughout its development. We propose a web-based system that automatically manage reviewing peers' codes and delivering feedback to peers in a process that favors the incremental learning of the concepts throughout the duration of the course.

The authors of this paper have many years of experience in modeling and implementation of teaching support systems in also in mobile settings (Riccucci *et al.*, 2005; Andronico *et al.,* 2003; 2004; Carbonaro, 2010). We now conducted experiments in the winter semesters from 2014 to 2016 at the University of Bologna, Italy, with around 240 first year computer science students within the Introduction to Algorithms and C programming language course which is offered during the first semester. The course is six traditional lecture hours and four practical class hours per week, for a total of 13 weeks. The purpose of the course is to introduce C language tools with lots of programming examples and to gain algorithmic thought. Each week a new C language element is covered and practiced by writing several sample programs. The coursework consists of weekly incremental programming assignments of increasing difficulty.

At the end of the course, we conducted a questionnaire survey to evaluate the system and the multi-peer assessment model.

Unlike automatically graded quizzes and programming assignments, peer assessments require a good-faith effort on the part of each student not only to submit their original work, but also to then anonymously evaluate the work of others attentively and constructively. Therefore, for each assignment submitted in a course, students are generally then asked to evaluate the work of up to 8 or 10 peers. That is not a negligible amount of work or time, especially in a course requiring weekly peer-assessed assignments and when the students' prior programming experience varies significantly.

Each student that participates in weekly peer assessment mechanism acts as an author when she/he writes the weekly assigned program, as a reviewer when she/he reviews a program written by another student and as a reviser when she/he revise her/his program as suggested by reviewer's comments. The teacher gives weekly assignment, grading and quality assurance

The system manages different type of documents: code submission that is the source code of a program written by an author (a program passed steps of compiling, building and testing) and submitted within deadline. Authors should complete submission of their codes before a given deadline but they may submit several source codes within the deadline; in this case, the system will consider the last one, corresponding to the final, and presumably best solution. Review comments, that is the suggestions and criticism that a reviewer gives to an author, mainly including coding standards, fatal defects, design logic, redundant code and non-functional requirements. Revision code, that is a new edition of program submitted by a reviser based on the review comments.

To ensure the review equality and exclude personal relationship factors, the implemented system does not use a fixed designation strategy to accomplish blind review. That is, it does not implement a mutual review by two students or a ring review by three or more students (Wang *et al.*<, 2011).

Our system uses a random designation strategy as the strategy of reviewer designation; that is when designating reviewers, the system generates correspondences randomly thus no student is aware of who will review her/his program. The designation algorithm randomly assigns the total N source codes of one class associated to weekly coursework to at least 10 students of the class. On a weekly basis, the system randomly generates new correspondences without creating group of students; in this way a student does not review another student's code from the same group but she/he reviews the code of students from other groups.

4 Questionnaires and analytic results

At the end of programming course the research team conducted a questionnaire survey to evaluate the system and the assessment approach proposing a set of multiple-choice questions. The analytic results of the questionnaire can be classified into the following categories: degree of student satisfaction, rationality of review process, acceptance of real-time assessment, rationality of online peer evaluation, impacts on students and time management capability.

1. Degree of student satisfaction. The students were generally satisfied with the proposed assessment approach. About 90% ("satisfied" plus "very much satisfied") of the students positively considered implemented peer assessment approach while only 10% of the students consider themselves not satisfied with the experience. Analyzing male vs female students' opinion we gather a difference satisfaction levels; in particular, 100% of female are satisfied against 89% of male students. This result corresponds to the overall perception of differences between male and female in their perceived ease of programming. We will come back to this point in section XYZ.
2. Rationality of review process. The survey results indicated that 75% of the students considered the blind review a rational process. They held the view that this mechanism could eliminate the factors of personal emotion. The peer assessment could produce a more objective attitude of making suggestions and accepting criticism and encourage students to share their opinions more directly. Gender differences in rationality of review process are much smaller and none is statistically significant.
3. Acceptance of real-time assessment. Most students (89%) applauded real-time assessment. When the deadline of an assignment had passed, the system assigned students' work to a pool of other students so that they could get real-time information about the assignment evaluation within a week of their submission. As a result, students could remain

aligned with respect to the development of the lesson content working on their assignments. Gender differences in acceptance of real-time assessment of review process are much smaller and none is statistically significant.

4. Rationality of online peer evaluation. The survey results of this item have reasonable levels of satisfaction. 65% of the students consider the whole blind peer review mechanism to be rational. In this case male and female students answered slightly differently: 75% female versus 63% of male. Once again, the data confirmed the increased demand of the female students to participate in the peer review mechanism to improve their programming skills. In both cases, some students were confused by evaluations that were not described with enough detail and hoped to identify specific error types to facilitate more accurate reviews and better revision.
5. Impacts on students. Student programming competence was improved significantly in the code review stage. As shown in the questionnaire responses, 65% students agreed that this process improves their programming skills because they are able to enhance his/her programming skills and learn different programming techniques from reviewing programs written by other students. However, in this case, gender differences in impacts of review process on students are significant: 83% of female students reported that they are improved their programming competences due to peer review process while only 61% of male students are agree with this statement. In particular, only 40% of female students consider the peer review process too long to run, while this percentage increases up to 78% considering male students. Moreover, only 16% of interviewed female students consider peer review process a tedious mechanism, while this percentage increases up to 83% for male students. Finally, 60% of the students were pleased to support the class with the review process and in this case, gender differences aren’t statistically significant.
6. Time management capability. Rigorous process control helps to build students’ time management capabilities. The teacher set two deadlines for each assignment in the assessment system. Uploading source code, submitting reviews and uploading revision code should be completed by student on time. If a student fails to complete a certain step on time, the system will automatically take some points off from the step. The system data showed that almost all students submitted their assignments and completed review process on time. At the end of the semester, most of the students reported that they had developed solid time management skills.

Conclusions

With the aim of improving deep learning in a programming course, we have developed a novel web-based peer assessment tool. It has advantages over ordinary automatic assessment in several aspects. Peer review can stimulate student interests in learning and enhance their awareness of active learning. When compared with traditional assessment processes, this process helps to improve the actual programming ability adding more fairness to assessment and helping to achieve learning objectives more efficiently.

In order to positively influence students learning outcomes it is necessary to take into account several approaches to enhance the traditional teaching methodology in teaching computer programming. These approaches comprise peer evaluation mechanisms. We demonstrated that student programming competence was improved significantly in the code review stage because this process improves their programming skills enhancing his/her programming skills and learning different programming techniques from reviewing programs written by other students. Furthermore, peer assessment process provides encouragement of students' deep learning skills in programming by making judgements and providing feedback on other student's work. It provides opportunities to compare and discuss what constituted a good or bad piece of work, which helps students to improve their programming style and think more deeply about the quality of work.

Our first question was: is there a gender gap in computer programming course? The answer is affirmative. We have found differences in perception between men and women; the perceived complexity of programming and the intention to actively participate to a guided group development mechanism to improve acquisition of program skills were significantly higher in females than in males. Female might find easier to overcome their lack of confidence thanks to the assessment mechanisms.

One of the most remarkable results from our experience was that students reported that assessing others' work was an extremely valuable learning activity.

REFERENCES

Yang, T, Chen S.Y., Hwang G. (2015), *The influences of a two-tier test strategy on student learning: A lag sequential analysis approach*. Computers & Education 82 (2015): 366-377.

Wang, Y., *et al.* (2012), *Assessment of programming language learning based on peer code review model: Implementation and experience report*. Computers & Education 59.2 (2012): 412-422.

Ala-Mutka, Kirsti M. (2005), "*A survey of automated assessment approaches for programming assignments*. Computer science education 15.2 (2005): 83-102.

Gupta, S., Dubey, S.D. (2012), *Automatic assessment of programming assignment*. Computer Science & Engineering 2.1 (2012): 67.

Ihantola, P., *et al.* (2010), *Review of recent systems for automatic assessment of programming assignments*. Proceedings of the 10th Koli Calling International Conference on Computing Education Research. ACM.

Kulkarni, C., *et al.* (2015), *Peer and self assessment in massive online classes*. Design thinking research. Springer International Publishing. 131-168.

Chinn, D. (2005), *Peer assessment in the algorithms course*. ACM SIGCSE Bulletin 37.3 (2005): 69-73.

Wang, Y., *et al.* (2015), *A multi-peer assessment platform for programming language learning: considering group non-consensus and personal radicalness*. Interactive Learning Environments (2015): 1-20.

Sitthiworachart, J., Joy, M. (2003), *Deepening computer programming skills by using web-based peer assessment*. Proceedings of the 4th Annual Conference of the LTSN Centre for Information and Computer Sciences. LTSN Centre for Information and Computer Sciences.

Rubio, M.A., *et al.* (2015), *Closing the gender gap in an introductory programming course.* Computers & Education 82 (2015): 409-420.

Patitsas, E., Craig, M., Easterbrook, S. (2014), *A historical examination of the social factors affecting female participation in computing*. Proceedings of the 2014 conference on Innovation & technology in computer science education. ACM.

Tiku, N. (2014), *How to get girls into coding*. New York Times.

Stoilescu, D., Gunawardena, E. (2010), *Gender differences in the use of computers, programming, and peer interactions in computer science classrooms*. Computer Science Education 20.4 (2010): 283-300.

Riccucci, S., Carbonaro, A., Casadei, G., (2005), *An Architecture for Knowledge Management in Intelligent Tutoring System*. CELDA.

Andronico, A., *et al*. (2003), *Designing models and services for learning management systems in mobile settings.* Workshop on Mobile and Ubiquitous Information Access. Springer Berlin Heidelberg.

Andronico, A., *et al.* (2004), *Personalisation services for learning management systems in mobile settings*. International Journal of Continuing Engineering Education and Life Long Learning 14.4-5 (2004): 353-369.

Carbonaro, A. (2010), *Towards an automatic forum summarization to support tutoring*. Technology Enhanced Learning. Quality of Teaching and Educational Reform. Springer Berlin Heidelberg, 2010. 141-147.

Wang, T, *et al.* (2011), *Ability-training-oriented automated assessment in introductory programming course.* Computers & Education 56.1 (2011): 220-226.