Interactive Learning Environments

ISSN: 1049-4820 (Print) 1744-5191 (Online) Journal homepage: http://www.tandfonline.com/loi/nile20

Good practices to influence engagement and learning outcomes on a traditional introductory programming course

Antonella Carbonaro

To cite this article: Antonella Carbonaro (2018): Good practices to influence engagement and learning outcomes on a traditional introductory programming course, Interactive Learning Environments, DOI: 10.1080/10494820.2018.1504307

To link to this article: https://doi.org/10.1080/10494820.2018.1504307

Published online: 26 Jul 2018.

INTERACTIVE LEARNING ENVIRONMENTS
https://doi.org/10.1080/10494820.2018.1504307

Routledge
Taylor & Francis Group

Good practices to influence engagement and learning outcomes on a traditional introductory programming course

Antonella Carbonaro

Department of Computer Science and Engineering, University of Bologna, Bologna, Italy

ABSTRACT

There have been many successful examples of new methodological approaches developed to help students in computer programming courses. Of these approaches, the peer assessment mechanism could be useful in providing students with opportunities to learn from one another, improve their learning experience and reach efficient learning outcomes. The paper presents and analyses an improved system based on the received evaluations of a previously developed web-based programming-assisted environment. This system automatically manages the peer code review process and delivers feedback to peers in a manner that favours the incremental learning of the concepts presented throughout the course. The experimental results are focussed on the impact of this system on students' programming competence, time management capabilities and student engagement.

ARTICLE HISTORY
Received 22 November 2017
Accepted 3 July 2018

KEYWORDS
Peer-assessment; computer programming courses; time management capabilities; student engagement

Introduction

Programming skills have become a core competency, particularly for engineering and computer science students; therefore, teachers are pursuing new methods of teaching coding. However, acquiring these skills is usually challenging and difficult for most students as it involves a background understanding of a range of theories, semantic and syntactic knowledge, coding and algorithmic skills, as well as a great deal of practice (Yang, Chen, & Hwang, 2015). In order to cope with these difficulties, a variety of teaching strategies and learning activities have been applied in computer programming courses, including assessment mechanisms to guarantee that students gain sufficient practice, as well as providing feedback on the quality of students' solutions. Manually assessing each student takes time even for small classes; when the class size increases, the amount of assessed work has to be cut down or rationalised in some other way. For these reasons, one major approach being used in computer science classes to evaluate and mark students' programming exercises is the automatic assessment of programming assignments. Relying on computer assistance allows for instant feedback without the need to reduce the number of exercises. Research into automatic programming assessment has a long history. It has been of interest to computer science educators since the 1960s, and continues to gain vast attention in the present day.

In order to provide students with opportunities to learn from one another, improve their learning experience and reach efficient learning outcomes, we have to give them the opportunity to review code, write and read comments and see how other students tackle the same problems. These practices/skills are also necessary for them to be able to successfully work in a team environment in their future careers.

CONTACT Antonella Carbonaro antonella.carbonaro@unibo.it Department of Computer Science and Engineering, University of Bologna, Mura Anteo Zamboni, Bologna 7 – 40126, Italy

Peer assessment has been being used successfully in different computer programming courses for many years to provide students with opportunities to learn from one another. In peer assessment, each student acts as both author and reviewer. Positive feedback leads to a better learning experience and more efficient learning outcomes (Wang, Li, Feng, Jiang, & Liu, 2012); in fact, when students evaluate one another's work, they think more deeply, learn to criticise constructively and display important cognitive skills, such as critical thinking. For example, in a study on undergraduate peer assessments, students reported positive experiences with peer assessment and recommended its use in college courses (Vickerman, 2009). Furthermore, peer assessment may reduce the workload of faculties and increase learning outcomes (Murakami, Valvona, & Broudy, 2012).

The usefulness of peer assessment could be very attractive to teachers, but they do not adopt this methodology until they believe that it can be trusted and that its results are reliable. Sometimes, teachers have argued that peer assessment is a rather complex and unreliable practice, because students may not have enough content knowledge and skill to evaluate a peer's work. To address this problem, we developed a web-based programming-assisted system, which improved students' programming skills through the practice of peer code review and the delivery of feedback to peers. One of the most remarkable results from this previous experiment was that students reported that assessing the work of others was an extremely valuable learning activity (Carbonaro & Ravaioli, 2017). However, the previous work also highlighted certain limitations and suggested some improvements.

In this paper, we propose and analyse an improved system based on the evaluations we received of a web-based programming-assisted system, which supports students' programming skills through peer code review and the delivery of feedback to peers. Its purpose was to investigate the extent to which peer assessment in a programming course promotes deep learning and favours the incremental learning of the concepts presented throughout the course. In the current study, we want to improve the previously obtained student satisfaction by focussing on the impact of the proposed system on students' programming competence, time management capabilities and student engagement.

The paper is organised as follows. The next Section introduces the automatic assessment of programming assignments and explores research efforts related to the student engagement mechanism in this field. Section 3 describes the architecture and functionalities of our implemented, improved system. Section 4 proposes the peer approach. Section 5 describes the questionnaire survey conducted to evaluate the system, focussing on its impact on students' programming competence, time management capabilities and student engagement. Finally, Section 6 provides some thoughts on the case study and the experimental results.

Related works

There have been many successful examples of new methodological approaches developed to help students in computer programming courses as a result of the growing interest in learning and teaching programming; some of these include the automatic assessment of programming assignments.

In the peer assessment process, students are involved in both the learning and assessment processes. Peer assessment plays an extremely important role, exposing students to solutions, strategies and insights that they would otherwise be unlikely to access (Kulkarni et al., 2015). Evaluating their peers' working also helps students to reflect on any 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, Liang, Liu, & Liu, 2015) and essays (Kulkarni et al., 2015).

We want to use peer assessment to promote learning in programming courses, rather than as a tool for summative assessment (Sitthiworachart & Joy, 2003). It should be noted 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". To prevent students from dropping

out of our course, which incorporates peer assessment in computer programming due to lack of time, it is important to evaluate the workload accompanying these assessment activities in order to ensure that students' total workload remains balanced.

When conducted over time, peer code review can support the development of confidence in students' emergent competence (Pon-Barry, Packard, & St. John, 2017).

Peer code review facilitates active learning; in fact, students are invited to participate in their own learning process as part of a community of learners (Søndergaard & Mulder, 2012). Furthermore, we intend to offer the opportunity for mini-feedback sessions focussing on a specific concept that may be most helpful to the learner at that time, rather than flagging every potential error. This should be especially beneficial for introductory courses.

Features introduced in the system framework

Practice is very important in acquiring programming skills; there should be room to make mistakes and to learn from them. Teachers could help their students as they practice, but automatic assessment can help to provide feedback in many different situations due to the limitations of human resources. We propose the use of automatic peer assessment to provide students opportunities to successfully learn from one another.

The author of this paper has many years of experience in the modelling and implementation of teaching support systems in mobile settings (Carbonaro, 2009, 2010; Riccucci, Carbonaro, & Casadei, 2005). We recently proposed a web-based programming-assisted system, which improved students' programming skills through peer code review and the delivery of feedback to peers. Its purpose was to investigate the extent to which peer assessment in programming courses promotes deep learning in order to assess the accuracy of students' judgements during a peer assessment exercise and provide evidence that peer assessment in computer programming has a positive instructive effect (Carbonaro & Ferrini). The Introduction to Algorithms and C Programming Language course is offered during the first semester at the University of Bologna, Italy, to first-year computer science students. The class consists of about 100 students in each academic year. The course comprises six traditional lecture hours and four practical class hours per week, for a total of 12 weeks. The purpose of the course is to introduce C language tools with copious programming examples and to encourage algorithmic thought. Each week, a new C language element is covered and practised by writing several sample programmes. The coursework consists of weekly incremental programming assignments of increasing difficulty. The author of this paper maintains a website dedicated to the course, available at www.programmazione.info On the site, in the ''Corso/Testo Esercizi'' section, one can find course content, learning outcomes, reading lists, teaching tools, assessment methods, previous examination papers and the weekly assignments for each academic year. Figure 1 shows the developed system back end for the management of programming assignments related to the seventh week of the course, during which students are invited to deal with topics related to static and dynamic structures. Each student who participates in a weekly peer assessment mechanism acts as an author on account of having to write the weekly assigned programm, as a reviewer when she/he reviews a programme written by another student or as a reviser when she/he revises her/his programme as suggested by reviewers' comments. During the course, authors must submit their codes before a given weekly deadline, as well as submit their revision of other students' programmes within a given deadline, which is set one week later.

Based on experiments conducted to evaluate the system and the assessment approach, we found reasonable levels of satisfaction with the system. Sixty-five percent of students considered the whole blind peer review mechanism to be rational. Nevertheless, some students were confused by evaluations that were not described in sufficient detail, and hoped to identify specific types of error that would facilitate more accurate reviews and better revision. We also evaluated the degree of student satisfaction, the rationality of the review process, the acceptance of real-time assessment, the rationality of online peer evaluation and the impact on students and their time management

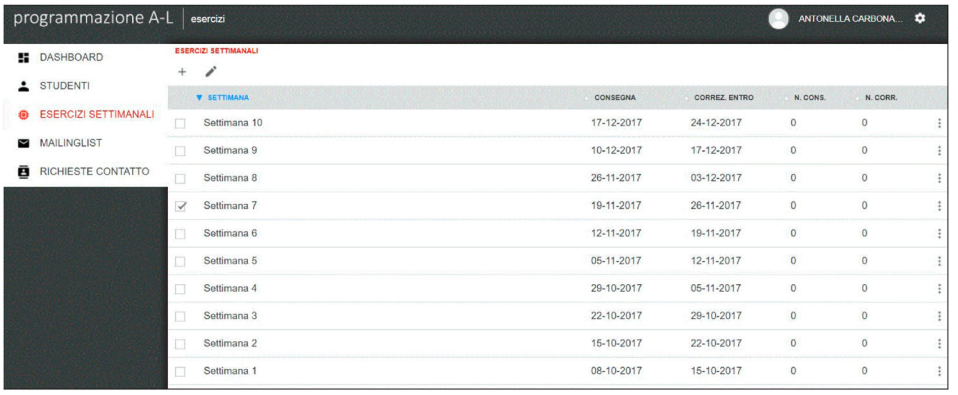

Figure 1. Screenshots of the system back end, which show programming assignments from the seventh week of the course and related review comments.

capabilities. Based on the evaluations received from last year's investigations, we now propose an improved system to increase students' programming skills and experience.

Our updated system introduces a ranking-based reviewer designation strategy. Students are considered as belonging to two different categories: higher-level or standard-level based on their programming skills. In order to decide to which category a student belongs, the teacher evaluated the first four weekly incremental programming assignments. Initially, the implemented designation strategy means that higher-ranking students review the programmes written by other high-ranking students. So as not to introduce new problems, we carefully planned the implementation of this strategy and decided to add it to the previous ones. By so doing, our system uses an overall strategy to randomly generate correspondences; thus no student is aware of who is reviewing his/her programme. This general designation algorithm randomly assigns the total N source codes to at least five students in the class, and randomly generates new correspondences on a weekly basis. In addition, higher-level students receive further revisions to do on programmes produced by other higher-level students. This assures the avoidance of student polarisation and enables the most skilled students to receive useful comments on their advanced programmes. In this way, we mitigate a previously detected problem whereby reviewers were evaluating a programme written by a highly competent student and giving it lower scores because they did not properly understand it.

In order to contribute instructively and assist students in producing their revised code, i.e. a new edition of a programme submitted by a reviser based on the review comments, we introduced the possibility of writing comments and observations into the system. In this way, a reviewer can give the author his/her suggestions and criticisms, mainly related to coding standards, fatal defects, design logic, redundant code and non-functional requirements.

The most important aspect we introduced when developing the new system was a closed set of review comments for each assignment. During the course design period, the teacher prepares weekly incremental programming assignments of increasing difficulty, as well as a set of questions for each assignment, which assess competence in the skills necessary to correctly solve the problem in hand. In Figure 1, five review comments on the first assignment given out during the seventh week can be seen, as well as four review comments concerning the second assignment. In this way, the teacher can precisely focus students' attention on the relevant aspects of the solution process, and guide them towards the identification of specific error types during their evaluations. We expect this to result in a higher degree of student satisfaction in their evaluations and a more informed opinion of the rationality of the review process. Moreover, we expect more homogeneity in this process among different students.

Peer assessment approach

Peer assessments require a good faith effort on the part of each student not only to submit their own original work, but also to then anonymously evaluate the work of others responsively and constructively. Therefore, for each assignment that they submit, students are generally then asked to evaluate the work of up to five of their peers. This is not a negligible amount of work or time, especially in a course that requires weekly peer-assessed assignments and when the students' prior programming experience varies significantly.

We decided to not use the peer assessment process to evaluate students in their final exam. First, the aim of the peer review process is to increase students' engagement over the duration of the course. This is a crucial aspect of a 12-week strength programme dedicated to introducing C language tools with lots of programming examples and to fostering algorithmic thought processes. Furthermore, student engagement in educational activities during their first year of academic studies is especially useful, and has a positive, statistically significant effect on persistence and academic achievement (Kuh, Cruce, Shoup, Kinzie, & Gonyea, 2008). Secondly, skills acquired during the demanding progression of the course will be more deeply assimilated during the suspension of lessons before exams. Students will have more than one month to make progress in their programming knowledge, including more complex topics such as pointer, recursion and data structure representation. Again, the implemented peer review system does not include software to assist in detecting possible instances of plagiarism. This is because it can be very difficult to detect plagiarism when the number of students is so high. Moreover, in spite of the temptation among some students to copy and modify the work of others due to the simplicity of copy and edit computer programmes, they understand the relevance of practice and appreciate that the opportunity to practice is fundamental in acquiring programming skills. We are aware that the number of plagiarism cases dropped drastically as the semester wore on, but we do not want to explicitly evaluate it as part of the peer review process. Finally, we want to empower our academic students to recognise useful activities to enhance their intellectual commitment and encourage them to think about their own value and future plans, regardless of their marks.

Thanks to the results obtained from a questionnaire survey conducted at the end of the course to evaluate the new multi-peer assessment model, we found that, on average, assessment activities accounted for 10% of students' total workload, although this varied greatly from week to week. It is important to the evaluate workload that accompanies assessment activities to ensure that students' total workload remains balanced and avoid decreased dedication to other activities. Additionally, students reported that assessment activities were often associated with more time spent on using computers to analyse, compile and execute source codes. This could help to ensure a higher pass rate than learning activities with a lower relative frequency of assessment measures.

Experimental report

This section reports on the results and gives some consideration to a questionnaire survey to evaluate the system and the multi-peer assessment model. We wish to evaluate the challenge of providing regular, timely and specific feedback to introductory students, given that the first semester is when students' sense of their potential competence begins to emerge. In particular, we aim to focus on the following characteristics: the impact of the system on students' programming competence, time management capabilities and student engagement.

In this research study, we were faced with the challenge of reaching the largest but defined population of course students. Therefore, a web-based survey with direct contact during last two lessons of the course was chosen for time- and cost-efficiency purposes. The questionnaire runs to around 17 questions in length and takes around 30 minutes to complete The questionnaire administered included questions such as: "On average, how many hours do you use to complete delivery (of

the exercise of the week)?'', ''On average, how many hours do you use to complete a review (of one exercise of the week)?'', ''Evaluate your experience after considering all the sources related to an exercise (express your degree of agreement from 1, the weakest, to 5, the strongest)'' [Boring because of the repetitions] [I learned from another] [Happy to have helped the class], ''As a reviewer, how many sources do you need to learn problem-solving skills from others?'', ''As a student, evaluate the degree of satisfaction in having participated in the peer review process along with the weekly administration of the programme assignments'', and ''As a student, evaluate the degree of satisfaction in having periodically obtained the evaluations of your exercises [not satisfied] [satisfied] [very satisfied]''. 87 students answered the questionnaire.

Impact on students' programming competence. As shown by the questionnaire responses, 82% of the students agreed that being involved in the assessment process helped them to improve their programming skills and to learn different programming techniques, thanks to reviewing programmes written by other students. Additionally, almost 80% of the students were happy to support the rest of the class with the review process. These are positive results in terms of influencing cooperation among students, improving their thinking skills and deepening their understanding.

Time management capabilities. The assessment mechanism works by setting three different deadlines for each weekly assignment set by the teacher: uploading the source code, submitting their reviews and uploading their revised code should all be completed by each student by the deadline. The system data showed that almost all of the students submitted their assignments and completed the review process on time. At the end of the semester, most of the students reported that they had developed solid time management skills. This result is very good, considering that learning effective time management is critical for students; the proposed, rigorously controlled process helps them to strengthen their time management capabilities and favours the incremental learning of presented concepts over the duration of the course. Notoriously, one of the biggest difficulties in most programming courses is to capture the focus and commitment of the students from the very beginning, and to constantly maintain it throughout the course's development.

Increased student engagement could be very important when it comes to then increasing the chance that students will reach the desired outcomes in his/her academic education. Chickering and Gamson (1987) proposed seven principles for good practice in undergraduate education, all of which are related to student engagement. They are: (1) student/faculty contact; (2) cooperation among students; (3) active learning; (4) prompt feedback; (5) emphasising time on task; (6) communicating high expectations; and (7) respecting diversity. Kuh (2009) reported that higher education institutions can directly influence engagement by implementing these seven principles. The system proposed in this paper implements several of these principles. In particular, it encourages cooperation among students, it gives prompt feedback, thereby responding to students' need for frequent opportunities to perform and receive suggestions for improvement, it helps to strengthen students' time management capabilities and it encourages active learning by applying what they are learning to their daily lives.

Furthermore, we offered the opportunity for mini-feedback sessions focussing on a specific concept that may be most helpful to the learner at that time, rather than flagging every potential error. This should be especially beneficial for introductory courses.

Conclusions

With the aim of improving engagement and learning outcomes in a programming course, we have developed a novel web-based peer assessment tool. It has advantages over ordinary automatic assessment as well as over our previously developed system in terms of its impact on students' programming competence, time management capabilities and student engagement.

In order to positively influence students' learning outcomes, it is necessary to take into account several approaches to enhance the traditional methodology of teaching computer programming. These approaches all involve peer evaluation mechanisms. We demonstrated that students'

programming competence significantly improved in the code review stage, as this process improves their programming skills; furthermore, reviewing programmes written by other students helps them to learn different programming techniques. Moreover, the peer assessment process encourages the development of students' deep learning skills in programming through making judgements and providing feedback on other students' work. It provides them with opportunities to compare and discuss what constitutes a good or bad piece of work, thereby helping students to improve their programming style and to think more deeply about the quality of their work.

The course website is a very useful tool for students; they get into the habit of checking in consistently, even daily, to check for instructor announcements and/or review course materials and submit their programming assignments or verify received peer reviews. They begin to feel part of a community and enhance their learning experience at the same time. This also applies to working non-attending students, which is a reality that is present on our course. Furthermore, the course website can be considered a valid tool, as it allows for the adoption of the described model by other people who wish to replicate our approach within their own class.

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

Disclosure statement

No potential conflict of interest was reported by the author.

Notes on contributor

Antonella Carbonaro is a confirmed Associate Professor of Computer Science at the Computer Science Degree Course of the University of Bologna. In 1997 she finished her Ph.D. studies at University of Ancona on Artificial Intelligent System. From 1997 to 1999 she received, beside the Computer Science degree Course of the University of Bologna, a research fellowship with theme of search "Artificial Intelligence". Since 2000 she is first assistant professor and then associate professor. Her current research interests concern semantic web technologies, data representation and personalised learning environment.

ORCID

Antonella Carbonaro http://orcid.org/0000-0002-3890-4852

References

Carbonaro, A. (2009). Collaborative and semantic information retrieval for technology-enhanced learning. *Social Information Retrieval for Technology-Enhanced Learning*, *535*, 55.

Carbonaro, A. (2010). Towards an automatic forum summarization to support tutoring. *Technology Enhanced Learning. Quality of Teaching and Educational Reform*. Springer Berlin Heidelberg, *73*, 141–147.

Carbonaro, A., & Ferrini, R. *Personalized information retrieval in a semantic-based learning environment*. Social Information Retrieval Systems: Emerging Technologies and Applications for Searching the Web Effectively (pp. 270–288).

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*, *13*(3), 121–129.

Chickering, A. W., & Gamson, Z. F. (1987). Seven principles for good practice in undergraduate education. *AAHE Bulletin*, *36* (8), 3–7.

Chinn, D. (2005). Peer assessment in the algorithms course. *ACM SIGCSE Bulletin*, *37*(3), 69–73.

Kuh, G. D. (2009). What student affairs professionals need to know about student engagement. *Journal of College Student Development*, *50*, 683–706.

Kuh, D. G., Cruce, T. M., Shoup, R., Kinzie, J., & Gonyea, R. M. (2008). Unmasking the effects of student engagement on first-year college grades and persistence. *The Journal of Higher Education*, *79*(5), 540–563.

Kulkarni, C., Wei, K. P., Le, H., Chia, D., Papadopoulos, K., Cheng, J. … …, S. (2015). Peer and self-assessment in massive online classes. *Design Thinking Research*, *20*, 131–168. Springer International Publishing.

Murakami, C., Valvona, C., & Broudy, D. (2012). Turning apathy into activeness in oral communication classes: Regular self- and peer-assessment in a TBLT programme. *System*, *40*(3), 407–420.

Pon-Barry, H., Packard, W. B., & St. John, A. (2017). Expanding capacity and promoting inclusion in introductory computer science: A focus on near-peer mentor preparation and code review. *Computer Science Education*, *27*, 54–77.

Riccucci, S., Carbonaro, A., & Casadei, G. (2005). An architecture for knowledge management in intelligent tutoring system. *CELDA*.

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.

Søndergaard, H., & Mulder, R. (2012). Collaborative learning through formative peer review: Pedagogy, programs and potential. *Computer Science Education*, *22*, 343–367.

Vickerman, P. (2009). Student perspectives on formative peer assessment: An attempt to deepen learning? *Assessment & Evaluation in Higher Education*, *34*(2), 221–230.

Wang, Y., Li, H., Feng, Y., Jiang, Y., & Liu, Y. (2012). Assessment of programming language learning based on peer code review model: Implementation and experience report. *Computers & Education*, *59*(2), 412–422.

Wang, Y., Liang, Y., Liu, L., & Liu, Y. (2015). A multi-peer assessment platform for programming language learning: Considering group non-consensus and personal radicalness. *Interactive Learning Environments*, *24*, 1–20.

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*, 366–377.