Simulated learners in peers assessment for introductory programming courses

Alexandre de Andrade Barbosa[1,3] and Evandro de Barros Costa[2,3]

[1] Federal University of Alagoas - Arapiraca Campus, Arapiraca - AL, Brazil
[2] Federal University of Alagoas - Computer Science Institute, Maceio - AL, Brazil
[3] Federal University of Campina Grande, Campina Grande - PB, Brazil

{*alexandre.barbosa@arapiraca.ufal.br*, *ebc.academico@gmail.com*}

Abstract. Programming is one of the basic competences in computer science, despite its importance, it is easy to find students with difficulties to understand the concepts required to use this skill. Several researchers report that the impossibility to achieve a quick and effective feedback, is one of the motivators for the problematic scenario. The professor, even when helped by the TAs, is not able to perform the reviews quickly, for this activity requires a huge amount of time. Fast feedback is extremely important to enable the learning of any concept. Some researches suggest the use of peer assessment as a means of providing feedback. However, it is quite common that the feedback provided by peers is not adequate. In this paper, we propose the use of simulated learners in a peer assessment approach as part of the teaching and learning processes of programming. Currently a software tool is being developed to include the proposal described in this paper.

1 Introduction

Programming is one of the basic competences in computer science, it is the basis for the development of several other competences required for professionals in the area. However, despite its importance, it is easy to find students who are demotivated and with difficulties to understand the concepts required to use this skill [7]. These difficulties causes a large number of failures, dropouts or the approval of students without the required level of knowledge [14] [6] [5].

Many factors are identified in literature as causing the problematic scenario related to programming courses. Several researchers report that the impossibility to achieve a quick, effective and individualized feedback, is one of the motivators for the problematic scenario [10] [12]. An individual follow up is impossible due to many students enrolled in the courses. In addition, there is a great complexity involved in the evaluation of a program, for it is necessary to understand how the programmer has developed the algorithm, so the professor needs to comprehend the line of reasoning adopted by the student. In this way, the professor, even when helped by the TAs, cannot provide an adequate and fast feedback about the solutions created by the students. This activity will require a huge amount

of time to manually open the code, compile, run and verify the output of every students solution for programming assignment. If the grading depends on the structure and the quality of code, in addition to program output correctness, the situation is a lot worse. Traditionally the real comprehension state of the contents of a programming course is known only months after the beginning of the course, when an evaluation activity is performed. After an evaluation it may be too late to make any intervention.

Fast feedback is of extreme importance to enable the learning of any concept [12]. Thus, some researches have been developed with the aim to propose methods and tools to facilitate the monitoring of the activities of students in programming courses. Some of these researches, such as [9][11][13], suggests the use of peer assessment as a means of providing fast and effective feedback. This solution is broadly used in Massive Open Online Courses (MOOCs), as described in [3][8], where the courses are applied to hundreds or thousands of people enrolled in them, and just as occurs in the context of programming, it is impossible for the professor to evaluate each solution. However, the peer assessment approach as a means of providing feedback has some problems. Many times the feedback provided by peers is not adequate, because the results are often not similar to the analysis of an expert [8]. It is quite common to find comments that are summarized to a phrase of congratulation or critique.

The reasons related to lack of effectiveness of feedback provided are quite distinct, these may occur due to poor understanding of the content of the activity, because of the student's low motivation, or due to the short time that one has available for the activities.

In [2] paper, it was observed the impact of learning was observed when a student is influenced by the performance of their peers, the authors describe that some students are encouraged to perform better, but others experiencing the same situations end up discouraged to perform better.

In this paper is proposed the use of simulated learners in a peer assessment approach used as part of the teaching and learning processes of programming. Two concerns are explored in this proposal: the first is related to the search of methods that enable a positive influence between students; the second concern is related to an approach that allows a less costly way of testing any proposal of applicability of peer assessment approach.

This paper is divided in five sections. In Section 2 the concept of peer assessment is presented. Observations on the implementation of peer assessment in a programming course context are shown in Section 3. The proposal of using simulated learners in the context of peer assessment for introductory programming is presented in Section 4. Finally the conclusions and future work are shown in the last section.

2 Peer Assessment

Peer assessment, or peer review, is an evaluation method where students have responsibilities that traditionally belong to professors only. Among these respon-

sibilities there are the review and the critique of the solutions proposed by their peers. This way, they can experience the discipline as students and also from the perspective of a TA. Usually in a peer assessment environment, students also conduct self assessment. This way, they can reflect on their solution when compared to other solutions, develop their critical thinking skills and improve understanding of the concepts covered in the course.

In traditional approach, the professor, even when helped by TAs, can not provide fast and adequate feedback for each solution proposed by the students. The comments provided by the professor are generic observations based on observation of all students solutions.

In accordance with [9], peer review is a powerful pedagogical method, because once students need to evaluate the work of their peers, they begin to teach and learn from each other. Thus, the learning process becomes much more active, making the learning qualitatively better than the traditional approach. Sstudents can spend more time on analysis and construction of their comments, creating more particular descriptions on a given solution and enriching discussion about the topic studied.

Thus, the use of peer review can reduce the workload on the professor, permitting the professor to focus on other pedagogical activities [9][3]. This evaluation approach can also enable the evaluation of large-scale complex exercises, which can not be evaluated in a automatically or semi-automatic fashion [3][8].

The success of peer assessment approach is strongly influenced by the quality of feedback provided. However, this feedback if often not adequate, the results are often not similar to the analysis of an expert [8]. In [8] is described that in many cases the evaluations of the students are similar to the TAs evaluation, however there are situations where the evaluations are graded 10% higher than the TAs evaluation, in extreme cases the grades could be 70% higher than the TAs evaluation. In [3] is mentioned that in general, there is a high correlation between the grades provided by students and TAs, but often in the evaluations from students the grades are 7% higher than the grades given by TAs.

Thus, we can conclude that peer assessment approach is a promising evaluation method, however there are improvements and adjusts to be applied to obtain richer discussions and more accurate assessments.

3 Peer Assessment in introductory programing courses

Human interaction is described as an essential feature for learning in many domains, including the introductory programming learning [13]. In classroom programming courses the contact between students occurs on a daily basis, allowing, for example, the discussion of the problems presented in the exercise lists, the developed solutions and the formation of groups for the projects of the course. This contact is many times inexistent in online programming courses, interactions in this environment are the human-machine type. Thus, using the peer assessment approach may enable human interaction on online courses, or enhance the interaction between humans in presencial classroom courses.

To encourage the assimilation of the topics, the use of practical exercises is quite common in programming courses, the practice of programming skills is crucial for learning. Many researchers also argue that the programming learning involves the reading and understanding of third-party code. Through peer assessment approach both characteristics can be obtained. The professor can develop new exercises, or choose problems proposed by others, while students will have to observe, understand and evaluate the codes of their peers, as well to compare these codes with their solution.

In [11] the use of a peer assessment approach to the context of programming courses is described, this approach is supported by a web application. The results described on the paper have a high correlation between the evaluations of the TAs and students, the correlation is lowest when the complexity of the exercise is higher.

An approach of peer assessment evaluation for the context of programming learning, also supported by a web application is presented in [9]. Five activities where graded using peer assessment, the occurrence of conflicts ranged from 61 % to activity with a lower incidence of conflict, up to 80 % for the activity with the highest occurrence of conflicts. The system considers that a conflict occurs when the student does not agree with the assessment provided.

In [9] the authors describes that if the peer reviews are conducted in an inadequate way, the failure rates can increase. For the teaching approach used at the programming course described in [13] there are two types of activities that require assessment, quizzes and mini projects. Among these activities only the mini projects are evaluated through peer assessment. Thus, students are not overloaded and the approach can be used appropriately.

Another problem that can emerge with the use of peer review in a programming context, is the increase of plagiarism. Once the assessment activity will be distributed among the students, the similarities of self-identification of codes can become more complicated. However, solutions are widely used to carry out the detection automatically similarities, such as MOSS [1] e GPLAG [4].

4 Simulated learners as peers in a peer assessment environment for introductory programing courses

In previous sections the advantages and disadvantages associated with the use of peer assessment in a general context, and when applied to the context of programming courses have been described. In both cases, the success of the approach is strongly influenced by the quality of the feedback given. Therefore, it is necessary to identify situations where there is inadequate feedback as well as conflict situations. Situations where inadequate feedback occurs are when, for any reason, the feedback does not help in the learning process. Conflict situations occur when the student does not agree with the assessment provided, or when there are huge variations on the evaluations provided. To perform a validation of this proposal or of any proposal involving peer assessment, it is necessary to

allocate the resources of time, physical space and adequate human resources. Thus, it can be said that the test of this approach is a costly activity.

Two concerns are explored in this proposal: how we can achieve methods that enable a positive influence between students in peer assessment environments, in other words, how a student can give a high quality feedback to their peers; and how a peer assessment approach can be tested with a lower cost, since any validation of these assessment approaches requires a huge amount of resources.

4.1 A scenario of use of peer assessment with simulated learners

Traditionally in a peer assessment environment, the professor must create the assignment and a set of assessment criteria. Then students develop their solutions observing the assessment criteria and submitting the solution to be evaluated by their peers. Each student's evaluation must meet the assessment criteria. The students should provide comments to peers explaining the reasons associated to the outcome and a grade or an evaluation concept (eg. A-, B+, C). Each student will have their code evaluated by their peers, and should assess the codes of other students. In Figure 1, it is illustrated the scenario previously described. There are variations in ways peer assessment approach is used, the scenario just mentioned has many characteristics which are similar to all the variations.

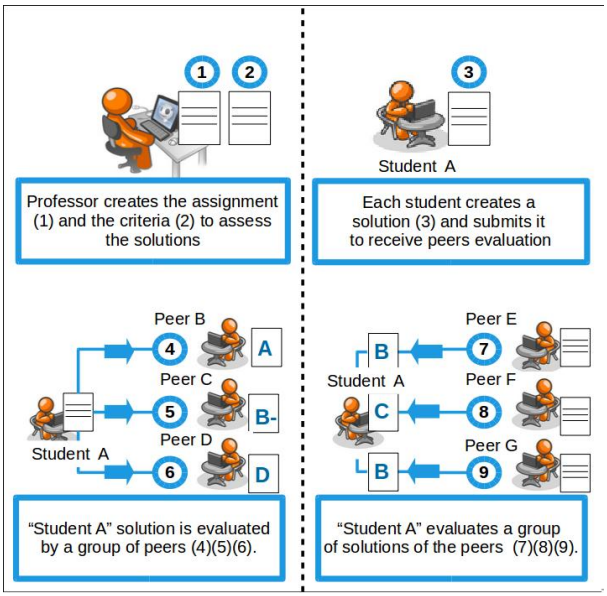

Fig. 1. A traditional peer assessment environment

In any peer assessment approach, it is possible to adopt pairing algorithms. Thereby, it is assured that evaluations are conducted by students with different

levels of knowledge. A student with low understanding of the subject will not be allocated to evaluate the work of another student in the same situation. Students with difficulties can clarify their doubts, while students with good understanding of the content should provide a good argumentation about their knowledge. However, it is not possible to ensure that a student evaluates the code that is the ideal for his/her learning and level of knowledge. As an example, in Figure 1, it is not possible to know if student "A" code is the best for peers "B", "C" and "D".

When a student does not agree with the evaluation provided by their peers, he/she will be able to request the intervention of the professor. This conflict situations are identified in [9]. However, in traditional peer assessment approach is not possible to identify incorrect evaluations provided by a student, or students that create biased evaluations only to help their fellows. As an example, in Figure 1, it is possible to see that different grades were given, but it is not possible to determine if the correct evaluations were given by peer "B", "C" or "D".

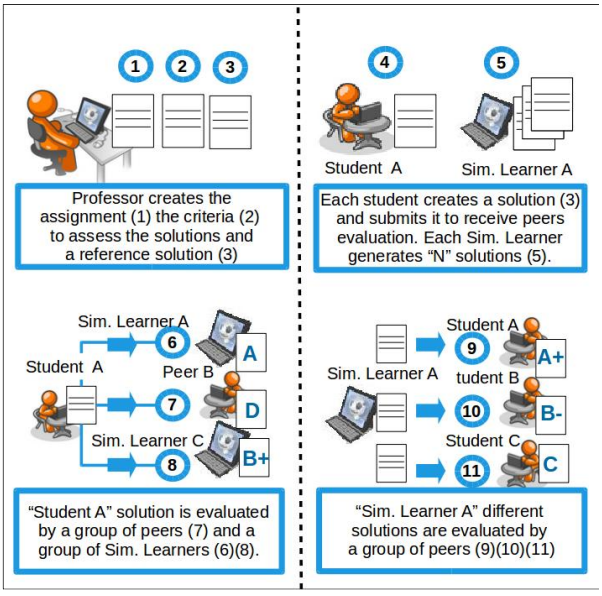

Fig. 2. A peer assessment environment using simulated learners

In a peer assessment environment that uses simulated learners, it is possible to solve the previous problems. As in traditional approach, the professor must create the assignment and a set of assessment criteria; in addition to that, he/she should provide a reference solution. Then, students develop their solutions, observing the assessment criteria and submitting the solution to be evaluated by their peers. At the same time, once a pairing algorithm can perform pairing of

the evaluators, each simulated learner must generate a code that is ideal for the learning and appropriate to the level of knowledge of each one of their peers, in this case, real students. Each student will have their code evaluated by their peers and by simulated students, and they should assess codes of other students, and codes of simulated students. In Figure 2 it is illustrated peer assessment environment with simulated learners. As an example, in Figure 2, it is possible to see that the simulated learner "A" generates a set of codes that are ideal for each student: "A", "B" and "C".

The identification of incorrect evaluations provided by a student, as well as students, who perform biased evaluations, could be carried out through the comparison of student's evaluations and the simulated student's evaluations. As an example, in Figure 2, it is possible to see that student "B", made an evaluation that is very different from the simulated learner's evaluations. In this way, it is possible to verify if the student did not understand the solution, or if the evaluation was created to help their fellows only.

Providing useful solutions to student learning A useful solution for the student learning does not always match the presentation of a correct and efficient code. Within the context of peer review may be more useful to display an incorrect code, as a way to make students to provide a set of review observations. To identify which type of code is best for a student; simulated learners can consult the representation of their cognitive status. In that way, it will be possible to the simulated learner identify the student misconceptions and errors in previous assignments, and generate variations of the reference solution that suits best for the student. Since multiple simulated students will be used, the codes that will be shown to students can range from efficient, correct and complete solutions to incorrect and/or incomplete solutions. Like that, it will be possible to check if students have different skills related to the content. To generate the variations from the reference solution, it is possible to combine testing techniques, such as mutant generation. Each code can be generated through the use of data related to the most common student's mistakes, emulating these behaviors and creating codes that are useful to learning. Once the research is in a preliminary stage, it is still not clear which artificial intelligence approaches should be used on the implementation of simulated students behaviors.

Assessment of students solutions Unlike what occurs in other contexts, for programming the evaluation of a solution can be automated or semi-automated. Typically a set of unit tests is applied to the code proposed by a student, who receives an indication that his/her code may be correct or incorrect, but no hint or comment is provided. Some researchers have investigated the use of different techniques to help assessment of codes and provide some guidance; these techniques usually employ software engineering metrics. Thus, simulated learners must be able to identify which subset of metrics can be used to perform the evaluation of the proposed solution for a student. The simulated learner should select the set of metrics that fits best to the objectives of the assignment and

the level of understanding that the student has at that moment. For each level of learning the same student can learn better if the set of metrics is properly selected. Each simulated student will use different strategies to evaluate the solutions provided by real students. Therefore, a variation between evaluations of simulated students is expected to occur. If an evaluation provided by a student has a very large variation in relation to the set of evaluations of simulated students, it will be necessary to investigate the motivation of this disparity. An acceptable variation threshold can be used to identify incorrect evaluations provided by students.

Discussing assessment criterias Once software engineering metrics were used in the evaluation, the explanation given by the simulated learner throughout the presentation of a set of metrics, is associated to the explanation of the metric choice and, possibly, of the snippet of the code where the observation is pertinent. Thereby, the simulated learner can help the professor to identify inadequate feedback, whenever an evaluation of a student is very different from the evaluation of a simulated learner, the professor and his tutors can then intervene.

4.2 Validation of peer assessent using simulated learners

Any validation of peer assessment approaches requires lots of physical space and a huge amount of human resources. As an example, if a validation of a pairing algorithm has to be done, it will be necessary to use a set of N students; this set must allow the creation of different profiles for evaluation of the pairing alternatives. The greater the possibilities of matching, the greater the amount of students required. Through the use of simulated learners any operational proposal of peer assessment can be tested at a much lower cost, since the physical space and human resources are drastically reduced. The researcher can determine how much of human resource will be available, replacing the students with simulated students. The researcher can also specify the desired behavior of students; the simulated students should emulate students with a high degree of understanding of the contents or with low understanding. After obtaining initial results with the use of simulated learners, the number of human individuals participating in an experiment can be increased, since it may be interesting to obtain a greater statistical power associated with the conclusions.

5 Conclusions and further work

In this paper, we have proposed the use of simulated learners in a peer assessment approach adopted as a support part of a programming course. The use of simulated learners as presented in this proposal aims to two goals: influence the students to provide better quality feedback; and allow for a less costly validation for peer assessment applied to programming contexts.

The research associated with the proposal presented in this paper is in a preliminary stage. Thus, the effectiveness of this proposal will be further evaluated in controlled experiments executed in the future. An open source software tool is being developed to include all aspects described throughout this proposal.

References

1. Bowyer, K., Hall, L.: Experience using "moss" to detect cheating on programming assignments. In: Frontiers in Education Conference, 1999. FIE '99. 29th Annual. vol. 3, pp. 13B3/18–13B3/22 (Nov 1999)
2. Frost, S., McCalla, G.I.: Exploring through simulation the effects of peer impact on learning. In: Proc. of the Workshops at the 16th International Conference on Artificial Intelligence in Education AIED 2013, Memphis, USA, July 9-13, 2013 (2013), http://ceur-ws.org/Vol-1009/0403.pdf
3. Kulkarni, C., Wei, K.P., Le, H., Chia, D., Papadopoulos, K., Cheng, J., Koller, D., Klemmer, S.R.: Peer and self assessment in massive online classes. ACM Trans. Comput. Hum. Interact. 20(6), 33:1–33:31 (Dec 2013)
4. Liu, C., Chen, C., Han, J., Yu, P.S.: Gplag: Detection of software plagiarism by program dependence graph analysis. In: Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 872–881. KDD '06, ACM, New York, USA (2006)
5. Mason, R., Cooper, G.: Introductory programming courses in australia and new zealand in 2013 - trends and reasons. In: Proc. of the Sixteenth Australasian Computing Education Conference - Volume 148. pp. 139–147. ACE, Darlinghurst, Australia (2014)
6. Mason, R., Cooper, G., de Raadt, M.: Trends in introductory programming courses in australian universities: Languages, environments and pedagogy. In: Proc. of the Fourteenth Australasian Computing Education Conference - Volume 123. pp. 33–42. ACE, Darlinghurst, Australia (2012)
7. McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y.B.D., Laxer, C., Thomas, L., Utting, I., Wilusz, T.: A multi-national, multi-institutional study of assessment of programming skills of first-year cs students. In: Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education. pp. 125–180. ITiCSE-WGR, New York, USA (2001)
8. Piech, C., Huang, J., Chen, Z., Do, C.B., Ng, A.Y., Koller, D.: Tuned models of peer assessment in moocs. CoRR abs/1307.2579 (2013)
9. de Raadt, M., Lai, D., Watson, R.: An evaluation of electronic individual peer assessment in an introductory programming course. In: Proc. of the Seventh Baltic Sea Conference on Computing Education Research - Volume 88. pp. 53–64. Koli Calling, Darlinghurst, Australia (2007)
10. Singh, R., Gulwani, S., Solar-Lezama, A.: Automated feedback generation for introductory programming assignments. In: Proc. of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation. pp. 15–26. PLDI, New York, USA (2013)
11. Sitthiworachart, J., Joy, M.: Computer support of effective peer assessment in an undergraduate programming class. Journal of Computer Assisted Learning 24(3), 217–231 (2008)
12. Stegeman, M., Barendsen, E., Smetsers, S.: Towards an empirically validated model for assessment of code quality. In: Proc. of the 14th Koli Calling Int. Conf. on Computing Education Research. pp. 99–108. Koli Calling, New York, USA (2014)

13. Warren, J., Rixner, S., Greiner, J., Wong, S.: Facilitating human interaction in an online programming course. In: Proc. of the 45th ACM Technical Symposium on Computer Science Education. pp. 665–670. SIGCSE, New York, USA (2014)
14. Yadin, A.: Reducing the dropout rate in an introductory programming course. ACM Inroads 2(4), 71–76 (2011)