Fostering Students-Driven Learning of Computer Programming with an Ensemble of E-learning Tools

Ricardo Queirós[1] and José Paulo Leal[2],

[1] Department of Informatics, Polytechnic of Porto & CRACS & INESC-Porto LA,
[2] CRACS & INESC-Porto LA, Faculty of Sciences, University of Porto

ricardoqueiros@esmad.ipp.pt
zp@dcc.fc.up.pt

Abstract. Learning through practice is crucial to acquire a complex skill. Nevertheless, learning is only effective if students have at their disposal a wide range of exercises that cover all the course syllabus and if their solutions are promptly evaluated and given the appropriate feedback. Currently the teaching-learning process in complex domains, such as computer programming, is characterized by an extensive curricula and a high enrolment of students. This poses a great workload for faculty and teaching assistants responsible for the creation, delivering and assessment of student exercises. In order to address these issues, we created an e-learning framework - called Ensemble - as a conceptual tool to organize and facilitate technical interoperability among systems and services in domains that use complex evaluation. These domains need a diversity of tools, from the environments where exercises are solved, to automatic evaluators providing feedback on the attempts of students, not forgetting the authoring, management and sequencing of exercises. This paper presents and analyzes the use of Ensemble for managing the teaching-learning process in an introductory programming course at ESEIG - a school of the Polytechnic of Porto. An experiment was conducted to validate a set of hypotheses regarding the expected gains: increase in number of solved exercises, increase class attendance, improve final grades. They support the conclusion that the use of this e-learning framework for the practice-based learning has a positive impact on the acquisition of complex skills, such as computer programming.

Keywords: E-Learning Interoperability, Distributed and interactive learning environments, Programming and programming languages, Teaching/learning strategies.

1 Introduction

For someone to acquire, improve or even maintain a complex skill it is necessary to practice it on a regular basis [1,2]. The amount of practice required depends on the nature of the activity and on each individual. How well an individual improves with practice is directly related with her or his inherent aptitudes, previous know-how and on the feedback. If feedback is either non-existent or inappropriate, then the practice tends to be ineffective or even detrimental to learning.

There are several complex skills that require constant practice, where exercise solving is a key component such as management, health sciences, electronics. Playing business games in management courses, or simulating a human patient in life sciences courses, or simulating an electronic circuit in electronics courses are examples of learning processes that require the use of special authoring, rendering and assessment tools. These tools should be integrated in instructional environments to provide a better learning experience. However, these tools would be too specific to incorporate in an e-learning platform. Even if they could be provided as pluggable components, the burden of maintaining them would be prohibitive to institutions with few courses in those domains.

The motivation for this work comes from yet another domain with complex evaluation: computer programming. Introductory programming courses are generally regarded as difficult and often have high failure and dropout rates [3,4,5]. Researchers pointed out several causes for these rates [6]. The most consensual are:

- **Teaching methods** - lectures and programming language syntaxes [7,8];
- **Subject complexity** - learning how to program means to integrate knowledge of a wide variety of conceptual domains such as computer science and mathematics while developing expertise in problem understanding, problem-solving, unit testing and others. Additionally, students petered out when they need to understand and apply abstract programming concepts like control structures or to create algorithms that solve concrete problems [6];
- **Student motivation** - the public image of a "programmer" as a socially inadequate "nerd" [9] and the reputation of programming courses as being extremely difficult affects negatively the motivation of the students [10]

Many educators claim that "learning through practice" is by far the best way to learn computer programming and to engage novice students [1,2]. Practice in this area boils down to solving programming exercises. Nevertheless, solving exercises is only effective if students receive an assessment on their work. An exercise solved wrong will consolidate a false belief, and without feedback many students will not be able to overcome their difficulties.

2 State of the Art

Assessment plays a vital role in learning [3]. However, automatic assessment of exercises other than multiple choice can be a rather complex task. This kind of evaluation differs significantly from evaluations supported by most LMSs, encoded in the IMS Question \& Test Interoperability (IMS QTI) specification[1]. The data model of QTI was designed for questions with a set of pre-defined answers and cannot handle evaluation domains with specialized requirements such as the computer programming. For instance, the assessment of programming exercises requires tests

[1] IMS QTI Web site: http://www.imsglobal.org/question/

cases, program solutions, compilation lines and other data that cannot be encoded in QTI. Besides the lack of a formal description for programming exercises, the interaction of assessment tools with other systems is not mature enough since there are no communication specifications as stated in several surveys [11,12].

Automatic assessment in computer programming domains can be applied in two distinct learning contexts: curricular and competitive learning. Introductory programming courses are part of the curricula of many engineering and sciences programs. These courses rely on programming exercises, assignments and practical examinations to consolidate knowledge and evaluate students. The enrolment in these courses is usually very high, resulting in a great workload for the faculty and teaching assistants responsible for assessing student programs.

While the concept of "winners and losers" can hinder the motivation of students [13], competitive learning is a learning paradigm that relies on the competitiveness of students to increase their programming skills [14,15]. This is the common goal of several programming contests where students at different levels compete such as: the International Olympiad in Informatics (IOI)[2], for secondary school students; the ACM International Collegiate Programming Contest (ICPC)[3], for university students; and the IEEExtreme[4], for IEEE student members.

In this context, several tools are used to allows students to train or participate in programming contests. These tools such as Programming Contests Management Systems (PCMS) and Online Judges (OJ) rely also on the assessment of programming exercises.

In both scenarios the manual assessment of programming assignments poses significant demands on the time of teachers [16]. Apart from being time-consuming, manual assessment hinders the consistency and accuracy of assessment results as well as it allows unintended biases and a diverse standard of marking schemes [17]. This demand stimulated the development of automated learning and assessment systems in many universities [3] as a means for grading the programming exercises of students as well as giving feedback on the quality of their solutions [18,19]. This feedback support is crucial for the computer programming learning [20,21], especially for first year students that need to be adequately engaged to learn programming [9]. Furthermore, immediate feedback motivates students to continue practicing [22,23].

Beyond the automatic assessment other relevant topic in this domain is the availability of programming exercises. It is important that an e-learning system provides a collection of exercises covering a course syllabus and with different levels of difficulty. It has been shown that this can improve the performance of students and their satisfaction levels [20]. Students with lower computer skills can begin by solving easier problems to learn progressively and to stay motivated to solve the harder problems later [24]. At the same time this gives them experience that is one of the factors that has a greater influence on the student success in learning programming. In recent years, many programming exercises have been developed and published mostly for use in programming contests. These exercises are generally stored in proprietary systems (e.g. Online Judges) for their own use. Despite some efforts [25] to define a

[2] IOI Web site: http://ioinformatics.org
[3] ICPC Web site: http://icpc.baylor.edu/
[4] IEEExtreme Web site: https://tinyurl.com/ycm8pabf

common format to describe programming exercises, each of these systems has its own exercise format, making it difficult to share among instructors and students. This poses several issues on the interoperability of the assessment systems with other e-learning systems.

Many learning tools and environments have been built to assist both teachers and students in introductory programming courses.

Rongas, Kaarna, and Kalviainen [26] established a classification for these tools dividing them into four categories: 1) integrated development interfaces, 2) visualization tools, 3) virtual learning environments, and 4) systems for submitting, managing, and testing exercises. To the best of the author's knowledge, no e-learning environment described in the literature integrates all these facets [6,10,27].

Several systems [27,28,29] try to address this issue allowing the integration of automatic assessment tools with course management systems but these approaches rely on ad hoc solutions or proprietary plug-ins rather on widely accepted international specifications for content description and communication among systems.

3 The Ensemble framework

The cornerstone of this approach is an e-learning framework - called **Ensemble** - that acts as a conceptual tool in the definition and deployment of such kind of e-learning network, relying on interoperability content and communication standards and specifications. This section provides a brief introduction to the Ensemble framework. Details on this framework can be found elsewhere [30].

The Ensemble e-learning framework (EeF) is exclusively focused on the teaching-learning process. Existing frameworks cover areas that go beyond the scope of e-learning, from course to financial management. In this framework the focus is on the coordination of pedagogical services that are typical in everyday life of teachers and students at schools such as the creation, delivery, resolution and evaluation of assignments. This framework emphasizes the use of assessment services to automatically evaluate the attempts of students to solve exercises and to produce relevant feedback on their quality. The need for automatic assessment exists in different domains, for instance:

1) an electronic circuit evaluator receives a description of a circuit, injects input signals, simulates the circuit and compares output signals;
2) a diagram evaluator receives a description of a diagram (e.g. UML) - a typed graph - and tries to create a graph homomorphism with a solution;
3) a programming exercise evaluator receives an attempt of a student to solve an exercise and the program is executed against test data, and then the output of the program (or return value) is compared with a solution model.

Another distinctive feature of EeF is its architectural model. The EeF uses a model that can be described as a "decentralized orchestration". An implementation of the EeF uses a pivot component that orchestrates the communication with other services but is replicated and deployed for each end user. This novel approach avoids any single-point-of-failure issues that occur in central orchestrations.

Based on this framework a network of systems and services was created and deployed for a specific domain - the computer programming domain. This framework instance comprises several systems and services and their integration poses interoperability issues on two levels: content and communication. Content issues are tacked with a standard format to describe programming exercises as learning objects. Communication is achieved with the extension of existing specifications for the interoperation with several systems typically found in an e-learning environment such as learning management systems, learning objects repositories and assessment systems. Some of these systems were created from scratch and others were adapted to support the new content and communication specifications.
The overall architecture of a network of e-learning systems and services participating in the automatic evaluation of programming exercises is depicted in Figure 1.

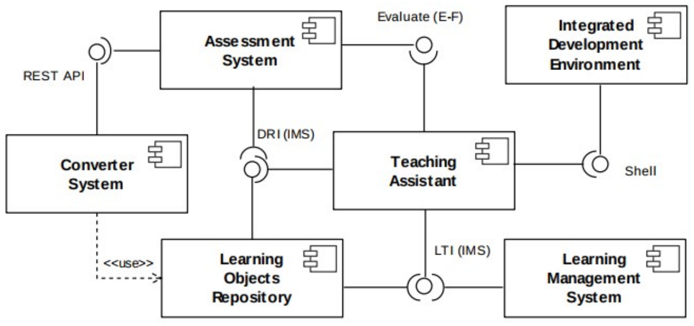

Fig. 1. Network component diagram.

The network is composed by the following components: 1) Teaching Assistant (TA) to interface the users with the network and to mediate the communication among all components; 2) Integrated Development Environment (IDE) to code the exercises; 3) Assessment System (AS) to evaluate exercises of students; 4) Learning Objects Repository (LOR) to store and retrieve exercises; 5) Learning Management System (LMS) to present the exercises to students; 6) Conversion System (CS) to convert exercises formats.

4 Experiment

The experiment took place at the Escola Superior de Estudos Industriais e de Gestão (ESEIG) - a school of the Polytechnic Institute of Porto - during the months of October and November of 2013. The participants were students from the first-year of the course Algorithms and Programming and their teachers. This course is offered to the degree in Mechanical Engineering and aims to introduce students to programming concepts.
The experiment methodology followed the experimental research method. For this purpose, experimental and control groups (two classes from the same course) were

settled. The first class (the experimental group) had 21 students and the second class (the control group) had 19 students.
Students of both groups have similar characteristics such as the gender and previous background. For instance, the gender in both groups were well distributed, respectively 62% and 57% of females in both groups.
The conditions of the experiment were also equal for both classes (e.g. syllabus, teaching times, teacher, labs, technical means).
Although it could be assumed that the population of the classes were almost randomly formed, strictly the experiment should be called a *quasi-experiment*. A completely randomized design was not possible due to operational reasons. Based on this type of design, a **static group comparison design** was followed where students of class A used Ensemble (the experimental group) and students of class B did not use it (the control group).
The course has an average enrolment of 40 students per year divided in two classes. The course is organized in two lectures of one hour each and one lab session of 4 hours per week. The experiment occurred in 6 lab sessions. In each lab session both groups (a total of 40 students) had 3 exercises to solve. In the experimental group the teacher only intervenes to solve operational issues related to the use of Ensemble and does not give any feedback to students regarding the exercises. Prior to the experiment, teacher and students were prepared for the experiment.

The instruments used for collecting data on the experiment were the following: surveys (session l& final survey), service logs, students' attendance logs and grades.

The **surveys** were fulfilled and collected on-line using Google Forms[5]. Two types of surveys were presented to students: session and final survey. The former was filled in by both groups of students after each lab session. The questionnaire[6] includes questions on the number of solved exercises and the feedback impact. It had an average of 38 responses per session (the equivalent to 95\% of the total of students). The latter was presented to the experimental group at the end of the experiment. The questionnaire includes questions on the Petcha (the central component of the system acting as a Teaching Assistant tool) usefulness and reliability. The final survey was completed by all the students from the experimental group.

After each lab session both classes were surveyed on the number of solved exercises and the feedback impact. Table 1 aggregates the answers given by students.

Table 1. Survey results averages.

Assertions	Experimental Group	Control Group
Exercises started	89%	81%
Exercises complete	83%	74%
Exercises effectively solved	82%	66%
Exercises with feedback	59%	62%
Feedback helpfulness	55%	62%

[5] http://www.google.com/google-d-s/forms/
[6] http://goo.gl/AlhsL

Data collected in the experimental class surveys was checked against the logs of Petcha and other systems in the network to attest the truthfulness of the survey responses. An average discrepancy of 4.6% between these two sets of values was found. This value suggests that students filled the survey carefully.

Firstly, statistical tests called Shapiro–Wilk were made to test if the samples came from a normally distributed population. After this validation, statistical hypothesis tests were used to assess whether the means of these two groups are statistically different from each other. The *t-test* is commonly used to verify if differences in observations can be explained by chance with a certain significance level p, typically 0.05 or 0.01.

For instance, using the .05 significance level means that one will accept as a real difference only those that occurred by chance only 5 times in 100 (i.e. in 95% of cases not due to chance). The statistical tests on the two sets of data (survey data and logs data) were satisfactory since the probability of the differences being the result of chance is higher than 5% (p=0.312 → p > 0.05$). Hence, we cannot distinguish between the two averages since their difference most likely occurred by chance.

The **service logs** were used to attest the accuracy of the experimental group questionnaires responses. The data collected in the surveys of the experimental class was checked against the logs of Petcha and other systems in the network. An average discrepancy of 4.6\% between these two sets of values was found.

The **student attendance** and **student outcomes** (programming module and semester grades) were collected through the Academic Management System used at ESEIG. The data was exported to a spreadsheet to simplify the data processing.

Due to page limit constraints, others statistic were omitted, such as: exercises solving, feedback, attendance and grades.

5 Conclusions and Future Work

This paper presents and analyzes the results of an experiment on the use of an e-learning framework called Ensemble that acts as a conceptual tool in the definition and deployment of e-leaning networks using complex evaluation. The experiment occurred in ESEIG - a school of the Polytechnic of Porto. The framework instance was deployed for use in practical classes of undergraduate programming courses.

The participants were students from the first-year of the course Algorithms and Programming and their teachers. This course is offered to the degree in Mechanical Engineering and aims to introduce students to programming concepts.

The results of the experiment showed an increase on exercises solving, attendance and grades when Petcha (we only show statistics on the first due to page limit restrictions) replaced a human Teaching Assistant (TA). However, these results show also that the automatic feedback provided by Petcha is less effective than that of a human TA. There is clearly room for improving automatic feedback in Petcha, although it can be argued that automated feedback is still a remedy for situations where a human TA is not available.

We can conclude that this Ensemble instance is a stable and reliable environment. Although there are still several aspects to improve the process of teaching and learning in domains with complex evaluation, such as computer programming.

Thus, the main conclusion is that the Ensemble instance improves student outcomes and this effect persists in the subsequent years after the introduction of the evaluation method.

The computer programming instance of Ensemble is currently being used in the practical classes of undergraduate programming courses at ESEIG and will continue to be used in the next academic year. Nevertheless, the evaluation of Ensemble and the validation of this work highlighted a number of issues that must be resolved. Several improvements are planned for immediate implementation based on the experiment results and based on the suggestions of teachers and students after the experiment. They are the improvement of the feedback, the extension of Ensemble for new domains and the upgrade of the pivot component.

The quality of feedback is one of the major issues detected in the evaluation phase of Ensemble. One solution is to improve the feedback mechanism based on, for instance, the use of static analysis over the students' code. Existing work in this area [31] can be used to improve the feedback given to students after submission.

Other facet to be improved is related with the Ensemble support for multi-domains. Ensemble as an e-learning framework can be specialized to other domains. Still, we only instantiate for a single domain - the computer programming domain. However, it was always kept in mind that the proposed concepts and tools could be used in other domains. The main opportunity for future research comes from extending this framework to other domains and requirements. One interesting domain is serious games applied to management courses where students develop their skills using simulation. Business simulation games improve the strategic thinking and decision-making skills of students in several areas (e.g. finances, logistics, and production). Through these simulations students compete among them as they would in a real-world company. A business simulation service fulfils a role like that of the assessment systems in programming exercises and it also requires a repository containing specialized LO describing simulations. Thus, this specific domain poses challenges not only in the development of the network TA, but also in the refinement of the framework specifications and services (e.g. repository, assessment system) to meet the new evaluation domains requirements.

The central piece of Ensemble instance network and the visible system of the network is a pivot component called Petcha. Based on the suggestions of teachers and students after the experiment, we plan to make some improvements. First, we will make the user interface more intuitive and flexible. We will also improve the visualization of the evaluation reports using new formats (e.g. PDF) and improve statistical data on student activity (e.g. time to solve, rankings). Finally, we will extend the documentation to guide users.

Acknowledgments

This work is partially funded by the ERDF through the COMPETE 2020 Programme within project POCI-01-0145-FEDER-006961, and by National Funds through the FCT as part of project UID/EEA/50014/2013.

References

1. Gross, P. and Powers, K. (2005). Evaluating assessments of novice programming environments. In Proceedings of the first international workshop on Computing education research, ICER '05, pages 99-110, New York, NY, USA. ACM.
2. Eckerdal, A. (2009). Novice Programming Students' Learning of Concepts and Practise. PhD thesis, Uppsala, Uppsala University, Division of Scientific Computing, Numerical Analysis
3. Ala-Mutka, K. (2005). A survey of automated assessment approaches for programming assignments. Journal of Computer Science Education, 15(2):83-102. http://www.tandfonline.com/doi/pdf/10.1080/08993400500150747.
4. O'Kelly, J. and Gibson, J. P. (2006). Robocode & problem-based learning: a non-prescriptive approach to teaching programming. SIGCSE Bull., 38(3):217-221.
5. Robins, A., Rountree, J., and Rountree, N. (2003). Learning and teaching programming: A review and discussion. Computer Science Education, 13:137-172.
6. Esteves, M., Fonseca, B., Morgado, L., and Martins, P. (2010). Improving teaching and learning of computer programming through the use of the Second Life virtual world. British Journal of Educational Technology.
7. Lahtinen, E., Ala-Mutka, K., and J arvinen, H.-M. (2005). A study of the difficulties of novice programmers. SIGCSE Bull., 37(3):14-18.
8. Schulte, C. and Bennedsen, J. (2006). What do teachers teach in introductory programming? In Proceedings of the second international workshop on Computing education research, ICER '06, pages 17-28, New York, NY, USA. ACM.
9. Jenkins, T. (2002). On the Difficulty of Learning to Program. In 3rd annual Conference of LTSN-ICS, Loughbourgh.
10. Gomes, A. and Mendes, A. J. (2007). Learning to program - difficulties and solutions. Proceedings of the International Conference on Engineering Education.
11. Leal, J. P. and Queirós, R. (2010). elearning frameworks: a survey. In International Technology, Education and Development Conference, Valencia, Spain.
12. Queirós, R. and Leal, J. P. (2011). A survey on elearning content standardization. In "World Summit on the Knowledge Society", WSKS'11. Springer Verlag.
13. Vansteenkiste, M. and Deci, E. L. (2003). Competitively contingent rewards and intrinsic motivation: Can losers remain motivated? Motivation and Emotion, 27:273-299. 10.1023/A:1026259005264
14. Burguillo, J. C. (2010). Using game theory and competition-based learning to stimulate student motivation and performance. Comput. Educ., 55(2):566-575.
15. Siddiqui, A., Khan, M., and Akhtar, S. (2008). Supply chain simulator: A scenario-based educational tool to enhance student learning. Comput. Educ., 51(1):252-261
16. Douce, C., Livingstone, D., and Orwell, J. (2005). Automatic test-based assessment of programming: A review. J. Educ. Resour. Comput., 5.

17. Romli, R., Sulaiman, S., and Zamli, K. (2010). Automatic programming assessment and test data generation a review on its approaches. In Information Technology (ITSim), 2010 International Symposium in, volume 3, pages 1186-1192.
18. Tremblay, G., Guérin, F., Pons, A., and Salah, A. (2008). Oto, a generic and extensible tool for marking programming assignments. Softw. Pract. Exper., 38(3):307-333.
19. Spacco, J., Hovemeyer, D., Pugh, W., Emad, F., Hollingsworth, J. K., and Padua-Perez, N. (2006). Experiences with marmoset: designing and using an advanced submission and testing system for programming courses. SIGCSE Bull., 38(3):13-17.
20. Wang, F. L. and Wong, T.-L. (2008). Designing programming exercises with computer assisted instruction. In Proceedings of the 1st international conference on Hybrid Learning and Education, ICHL '08, pages 283-293, Berlin, Heidelberg. Springer-Verlag.
21. Mory, E. H. (2007). Feedback research revisited. Handbook of Research for Educational Communications and Technology.
22. Daly, C. (1999). Roboprof and an introductory computer programming course. SIGCSE Bull., 31(3):155-158.
23. Truong, N. K. D. (2007). A web-based programming environment for novice programmers. PhD thesis, Queensland University of Technology
24. Lee, F. and Heyworth, R. (2000). Problem complexity: A measure of problem difficulty in algebra by using computer.
25. Queirós, R. and Leal, J. P. (2012b). Petcha - a programming exercises teaching assistant. In ACM SIGCSE 17th Anual Conference on Innovation and Technology in Computer Science Education, Haifa, Israel. ACM.
26. Rongas, T., Kaarna, A., and Kalviainen, H. (2004). Classification of computerized learning tools for introductory programming courses: Learning approach. In Kinshuk, Looi, C.-K., Sutinen, E., Sampson, D. G., Aedo, I., Uden, L., and Kaahkaonen, E., editors, ICALT. IEEE Computer Society.
27. Verdú, E., Regueras, L. M., Verdú, M. J., Leal, J. P., de Castro, J. P., and Queirós, R. (2011). A distributed system for learning programming on-line. Computers & Education.
28. Xavier, J. and Coelho, A. (2011). Computer-based assessment system for e-learning applied to programming education. In ICERI2011 Proceedings, 4th International Conference of Education, Research and Innovations, pages 3738-3747. IATED.
29. Guerreiro, P. and Georgouli, K. (2008). Enhancing elementary programming courses using e-learning with a competitive attitude. International Journal of Internet Education, 10.
30. Queirós, R. and Leal, J. P. (2012). Orchestration of e-learning services for automatic evaluation of programming exercises. 18(11):1454-1482. http://www.jucs.org/jucs1811=orchestrationofelearningservices:
31. Nielson, F., Nielson, H. R., and Hankin, C. (1999). Principles of Program Analysis. Springer-Verlag New York, Inc., Secaucus, NJ, USA