A "light" application of Blended Extreme Apprenticeship in teaching Programming to Students of Mathematics

Ugo Solitro[1], Margherita Zorzi[1], Margherita Pasini[2], Margherita Brondino[2]

[1] Department of Computer Science
[2] Department of Philosophy, Education and Psychology
University of Verona (Italy)
{ugo.solitro|margherita.zorzi|margherita.pasini|margherita.brondino}@univr.it

Abstract. In this paper we analyze an application of eXtreme Apprenticeship (XA) methodology, in a blended form with a reduced set of software and human resources. The study was conducted at University of Verona, in the context of the course "Programming with Laboratory" with 170 participants enrolled at the first degree in Applied Mathematics, throughout three different academic years. We analyze the very first period of lessons, when the fundamentals of programming are introduced. During the first two years, students were trained with a traditional teaching method; the lasts group was trained using the XA teaching model. Results showed a real improvement of learning outcomes in students trained with XA compared with the traditional teaching method. Possible refinements of XA method in our case study and in other educational contexts are discussed.

1 Introduction

The ability of analyzing problems, designing a solution and finally realizing it using with the available resources (in short, programming) require a set of abilities related to the so-called computational thinking whose importance has been largely recognized [15, 8]. So teaching programming has proven to be a critical task and becomes particularly interesting and challenging when this matter is proposed to students of non vocational curricula. In Italy informatics is a marginal discipline in most (non technical) high schools. As a consequence many college students of scientific discipline (different from informatics) have little knowledge of informatics. Freshmen typically address basic programming courses as beginners, without previously acquired skills. Notwithstanding, computational thinking attitude is nowadays essential and some basic masteries in this field are unavoidable, in particular for technical and scientific subjects. In mathematics curricula the computer (more precisely the tools based on it) is a key device for the students. The general shortage of primary informatics skills (algorithmic thinking, basics coding abilities, experiences in problem-solving activities) and the heterogeneous composition and formation of the students group,

determine a number of initial "problems" such as misconception in the basics of informatics, difficulties in conceiving algorithmic solutions, designing the corresponding code, understanding mistakes, and more. Many students complete the first course in informatics with sufficient knowledge of the topics, but not fully competent in the applications. For this reason we decide to improve our teaching methodology for the students of the course Computer Programming with Laboratory, enrolled at the first degree in Applied Mathematics, by adopting some techniques inspired by the eXtreme Apprenticeship (XA) [13]. We also adapted the method to the particular context of the University of Verona making use of the available E-learning platform (elearning.univr.it) based on Moodle (moodle.org). In this paper, we analyse the ongoing experience of teaching programming to first year students of the Bachelor Degree in Applied Mathematics from the University of Verona, following the ideas of XA method. Students of mathematics are less familiar than expected in informatics. With a new teaching approach we expect a reduction of the initial difficulties in learning programming for not trained students, improving the acquired skills. We will measure the effectiveness of this teaching model when applied with a reduced set of resources.

1.1 The eXtreme Apprenticeship method

A promising perspective in promoting computational thinking is the Cognitive Apprenticeship (CA) learning model [3], a method inspired by the apprentice-expert model in which skills are learned within a community, through direct experience and practice guided by an expert of the skill. The main idea of Cognitive Apprenticeship is to focus on the teaching/learning process rather than just on the final "product". Cognitive apprenticeship is based on three separate stages: *modeling*, *scaffolding* and *fading*. In the modeling stage, the teacher gives students a conceptual model of the process. Lessons are principally based on presenting work examples in an interacting and active manner: the teacher explains the decisions made during the process step by step. During scaffolding stage students solve exercise under the guidance of an experienced instructor. Students receive hints to be able to discover the answers to their questions themselves. The fading stage of apprenticeship learning is reached when students are able to master tasks by themselves.

Cognitive Apprenticeship has had many applications in teaching programming. *eXtreme Apprenticeship* (XA) is an extension of the Cognitive Apprenticeship model, which emphasizes communication between teacher and learners during the problem-solving process. This approach promotes learners' intrinsic motivation, which is a positive antecedent of performance. This methodology has been developed and being actively practised since 2010 at the University of Helsinki (Finland) [13, 14] for teaching programming. In XA hours devoted to frontal lessons are drastically reduced, and students are encouraged to solve problems themselves in a guided manner and in a non-interfering environment. Exercises are the most important aspect of the learning process: student's apprenticeship starts immediately, gradually, and it is continuously monitored, by assigning programming exercises. The difficulty of exercises has to be slowly

incremental: as pointed out in [13], each new exercise has to master a minimum amount of new material on top of previous exercises. In this way, students acquire new skills facing with a measurable amount of work to be done. Programming exercises have also a positive impact on the motivational side: there is no need for any external motivating factors, since success in learning itself feels good. See, for example, [2, 1, 10]. With the application of XA method, the finning research team RAGE [13] has obtained excellent results in programming introductory courses. The method has also been applied in a blended way (that is with online support) in Bolzano (Italy) in teaching operating systems and informative system [7, 5, 6, 4].

XA is currently applied in two programming course in Verona, offered at the bachelor degree in Computer Science and at the bachelor degree in Applied Mathematics respectively. As previously said, the latter represents our case study.

2 Method

2.1 Participants

The study was conducted at University of Verona, with 170 participants (50% males, mean age: 19) enrolled at the first degree in Applied Mathematics; the course is Computer Programming with Laboratory, throughout three different academic years (2013/14; 2014/15; 2015/16). We analyze the very first period (8 weeks) when the fundamentals of programming are introduced.

During the first two years, students were trained with a traditional teaching method (TT); the lasts group was trained using the XA teaching model. Some students of the last group actually didn't participate in the training and delivered no one of the expected programming tasks connected with the XA experience, and for this reason were excluded from the XA group. The final sample consists of 114 students (50% males) for the TT condition and 48 students (54% males) for the XA condition.

2.2 Procedure

The first two months are spent to an introduction to algorithms and programming, partly using the programming language Python. The teaching is organized in two kinds of activities.

- the "theoretical" part: a wide introduction to programming in a lecture hall;
- the "programming" part: practical experience in the computer laboratory.

In the TT way (A.A. 2013/14 and 2014/15) the two activities are distinct and with different focus. In the lecture hall the teacher introduced the general concept and some exercises. In the laboratory a few exercises are presented with a full or partial solution; the students are encouraged to complete some exercises and, in addition, some more are proposed as homework. The solution of some

of exercises can be presented later. Teacher and assistants (graduate or Ph.D. students) give some practical support in the laboratory.

In the XA way, the teacher in the "non practical" lessons usually introduce the tools necessary to understand the practical activities and some general concepts about informatics and programming. When possible, some "unplugged" exercises are proposed. The students have to attend introductory lessons and take part to exercise sessions. During home works they can receive feedback and support in the laboratory or through the E-learning platform from the teacher or the assistants. Every two weeks students are asked to submit the result of an activity that will be evaluated.

The full application of XA method requires a lot of human resources [11] and/or a semi-automated correction tools (e.g. "Test My Code" [12]). In order to overcome some restrictions/problems we encounter in our educational context (few hours-with respect to XA standard– of practical lessons, limited support by laboratory assistants), we have adapted the teaching routine to the "Verona setting": in some sense, our XA method can be defined "light". We were not able to fully apply XA paradigm, but notwithstanding we followed its main guidelines. We claim and prove that XA also works in this particular situation, which is otherwise realistic: a "typical" educational setting has to face a limited amount of resources and a flexible adjustment of the teaching method may be mandatory.

2.3 Research design and data analysis

At the end of the first period students are encouraged to take a partial exam ("test") considered as part of the final exam. The following parameters are considered for the evaluation: correctness of the solution, logical structure and good programming practices. This test it the only learning outcome available for the three academic years in this study, and for this reason was considered to verify the efficacy of XA compared with the traditional teaching model. This test consists of two parts: a general "theoretical" section, in which the knowledge of fundamental notions (e.g. the definitions of compiler, interpreter, specification) are verified; a practical "programming" section where students must solve a few exercises of increasing difficulty about programming competences and problem solving skills.

The evaluation in this test produced a quantitative score, which was normalized in the range 0-1 to allow the comparison among the three different academic year and the comparison between the theoretical and the programming outcomes.

At the end, three different-even if related-quantitative dependent variables were considered: total score (TOT), theoretical score (TH), and programming score (PR). Two different data analyses were carried on.

1. Considering the whole sample, a quasi-experimental design was used, with the teaching method as the independent variable, with two conditions (XA vs TT), and the three learning outcomes as the dependent variables.

2. Considering the XA group as a sub-sample, a quasi-experimental design was used, with total number of delivered programming tasks as an independent 3 level variable (0/1; 2; 3), and the same learning outcomes.

In a third descriptive analysis, the quantitative score was also categorized in 6 different judgements A, B, C, D, E, F where 'A' represents the highest level and 'F' the lowest one.

3 Results

A first set of analyses considered the entire sample. Figure 1 shows the average exam score considering the three learning outcomes (TOT, TH and PR) for the two groups (XA and TT). Three separate two-way ANOVAs were run, with the teaching method as a two-level between-group factor, also considering the two-level factor "sex", in order to control for a possible sex effect. First of all, no sex effect was found, nor interaction sex by teaching method for the learning outcome TOT nor for the two separate learning outcomes TH and PR. The main effect of the teaching method was found for TOT ($F(1,158)=17,23$; $p<.001$) even if with a small effect size ($\eta^2 = .10$), and also for TH ($F(1,158)=29.55$; $p<.001$), with a medium effect size ($\eta^2 = .16$), and for PR ($F(1,158)=4,95$; $p<.05$), with a small effect size ($\eta^2 = .03$). In all cases, XA group showed the best results (dependent variable "TOT", XA: mean=.73, SD=.03; TT: mean=.59, SD=.03; Dependent variable "TH", XA: mean=.79, SD=.03; TT: mean=.62, SD=.02; Dependent variable "PR", XA: mean=.61, SD=.03; TT: mean=.52, SD=.02).

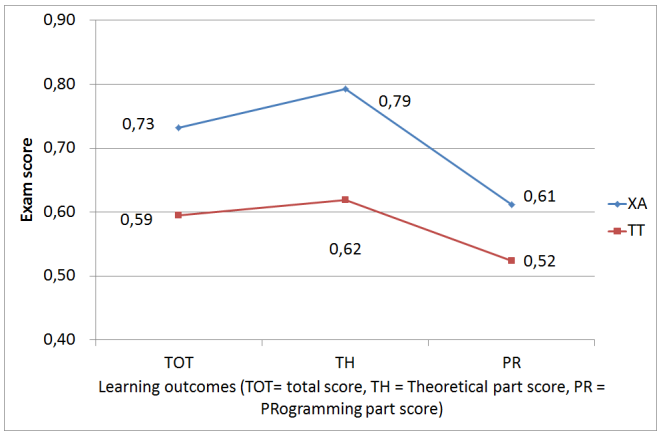

Fig. 1: Average exam scores in the two groups - Traditional Teaching method group (TT) and XA group - considering the overall score (TOT), and the score in the two parts: Theoretical (TH) and Programming (PR) part.

These results seem in line with the expected better outcomes due to the XA teaching model, even if the last result with a small effect size on programming task, needs to be more deeply explored.

In order to verify whether the performance, in the XA group, is connected with the persistence in doing the assigned tasks, another ANOVA was performed, in the sub-sample of the XA group, using the three-level variable with the total number of delivered programming tasks (0/1; 2; 3) as the between-group factor and the three exam scores as the dependent variable. As expected, the higher the number of delivered programming tasks, the better the performance (see Figure 2). This effect is significant for TOT score, due to the effect on TH and not on PR (TOT: $F(2,53)=3{,}98$, $p<.05$; TH: $F(2,53)=4{,}75$, $p<.05$).

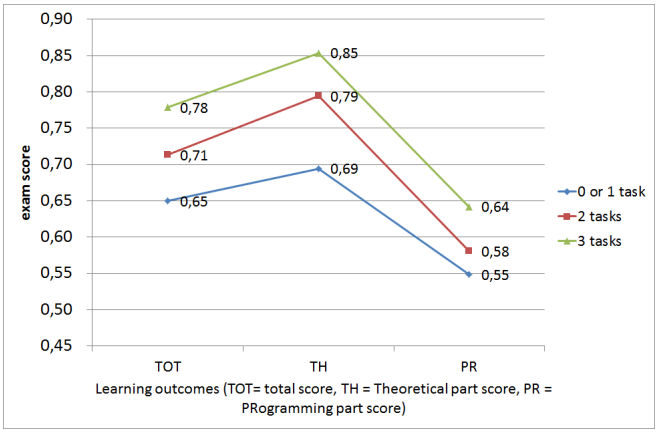

Fig. 2: Average exam scores in the three XA-level groups considering the overall score (TOT), and the score in the two parts: Theoretical (TH) and Programming part (PR).

The last analysis concerns the judgement in the exam, in 6 levels, from A to F. Figure 3 shows that judgements A and B are more frequent for XA students, whereas judgements E and F are more frequent for TT students. Percentages of central categories C and D are similar in the two groups.

4 Conclusions

The aim of the present study is to explore the potential advantages of the XA teaching model on the learning outcomes in teaching programming to university students.

Our first results are encouraging: the XA method drastically reduces the number of low results. different teaching perspective to programming seems to

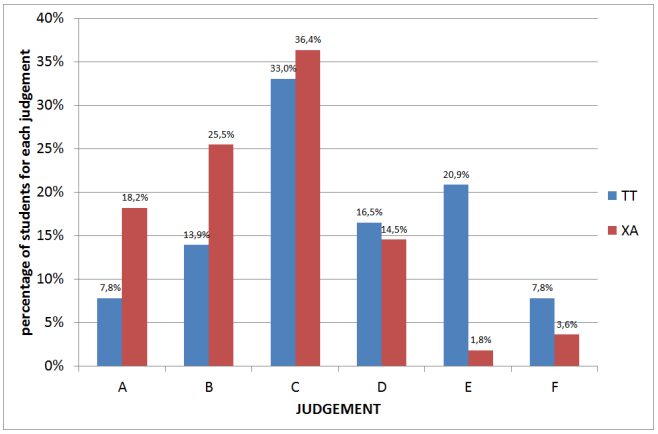

Fig. 3: Percentage of students for each judgement (from A=excellent to F=poor), separately for TT group and XA group

have a positive effect both in the "practical" part and in the "theoretical" one. The improvement of the performance is less evident in more advanced exercises that in general require more experience and, a posteriori, appear to be slightly more difficult for the current year.

The combination of a more practical approach, an active support, a constant stimulation to cope with the difficulties, the opportunity to receive feedback in the lab and through the e-learning platform have contributed to a more effective participation of the students in the activities.

We are planning to extend the XA method to the teaching of programming in different curricula and also explore new potential application. A wider application of the methodology will require the solution of some relevant problems in connection with the shortage of human resources and the necessity of a significant revision of the e-learning services.

In the next years, we hope to collect more data from different XA trained classes, and obtain fruitful hints in order to refine XA methodology in our case study and in other contexts. Database courses addressed to students of liberal arts could be, for instance, an intriguing context to investigate.

Finally, we think that eXtreme Apprenticeship offers benefits both to students and teachers. During the lessons period, an XA teacher collects a huge amount of delivered programming exercises. This offers an unprecedented overview about errors students commit in solving programming tasks. A careful analysis of students' failures will provide a precious feedback about learning cognitive processes and teaching methodology.

We think that this investigation, together with an analysis of students' motivations and emotions, will provide useful tools for overcoming the beginner diffculties in learning programming.

References

1. S. Bergin and R. Reilly. *The influence of motivation and comfort-level on learning to program*, Proceedings of the 17th Workshop on Psychology of Programming Interest Group (PPIG'05), 293–304, 2005.
2. M. Brondino, G. Dodero, R. Gennari, A. Melonio, M. Pasini, D. Raccanello, S. Torello, *Emotions and inclusion in co-design at school: Let's measure them.* Methodologies and intelligent systems for technology enhanced learning, Springer, 1–8. 2015.
3. A. Collins, J. Brown, and A. Holum, *Cognitive apprenticeship: Making thinking visible*, American Educator, 6, 38-46, 1991.
4. V. Del Fatto and G. Dodero, *Experiencing a new method in teaching Databases using Blended eXtreme Apprenticeship*, Proceedings of 21st International Conference on Distributed Multimedia Systems (DMS'2015), 2015.
5. V. Del Fatto, G. Dodero and R. Gennari: *Operating Systems with Blended Extreme Apprenticeship: What Are Students' Perceptions?*, IxD&A, 23, 24-37, 2014.
6. V. Del Fatto, G. Dodero and R. Gennari: *Assessing Student Perception of Extreme Apprenticeship for Operating Systems*, Proceedings of14th International Conference on Advanced Learning Technologies (ICALT), 459-460, 2014.
7. G. Dodero and F. Di Cerbo, *Extreme Apprenticeship Goes Blended: An Experience*, Proceedings of 12th International Conference on Advanced Learning Technologies (ICALT), 324-326, 2012.
8. W. Gander et all, *Informatics education: Europe cannot afford to miss the ACM*, available at: http://europe. acm. org/iereport/ie. html, 2013
9. T. Hautala, T. Romu, J. Rämö and T. Vikberg, *Extreme Apprenticeship Method in Teaching University-Level Mathematics*, Proceedings of 12th International Congress on Mathematical Education Program Name, 8-15 July, 2012, COEX, Seoul, Korea, 2012.
10. T. Jenkins, *The motivation of students of programming*, Proceedings of the 6th annual conference on Innovation and technology in computer science education, Canterbury (UK), 53-56, 2001.
11. J. Kurhila, and A. Vihavainen, *Management, structures and tools to scale up personal advising in large programming courses*, Proceedings of the 2011 conference on Information technology education, 3–8, ACM, 2011.
12. M. Pärtel, M. Luukkainen, A. Vihavainen and T. Vikberg, *Test My Code*, International Journal of Technology Enhanced Learning 2, 5(3-4), 271–283, 2013.
13. A. Vihavainen, M. Paksula and M. Luukkainen, *Extreme Apprenticeship Method in Teaching Programming for Beginners*, Proceedings of the 42nd ACM technical symposium on Computer science education (SIGCSE '11), 93-98, 2011.
14. A. Vihavainen and M. Luukkainen, *Results from a three-year transition to the extreme apprenticeship method*, Proceedings of IEEE 13th International Conference on Advanced Learning Technologies (ICALT), 336–340, 2013.
15. J. Wing, *Computational Thinking*, Communications of the ACM - Self managed systems 49(3). 33–35, 2006.