Early Training in Programming: From High School to College

Ugo Solitro[1](✉), Margherita Zorzi[1], Margherita Pasini[2], and Margherita Brondino[2]

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

Abstract. Informatics is recognized as a fundamental discipline in education at all levels. It is also an indispensable subject for scientific and technical studies. Some abilities connected to informatics learning *(computational thinking)* has being considered to provide "fundamental skills for everyone". Programming or, more generally, the ability of solving problems by algorithmic methods is one of these skills. In Italy, many scientific degree courses offer, at the first year, at least an introductory course in programming. Digital expertize and a basic attitude to computational thinking are in general expected. The present study, has been conducted at the University of Verona, in the context of the course *Programming with laboratory* of Applied Mathematics curriculum. We focus on first period of lessons, when the fundamentals of programming are introduced. Most of the students come from secondary schools, in particular *Liceo*, a secondary school with emphasis science or humanities, and where the role of informatics is in general not central. So, an academic course in programming can be a difficult task for students. In this paper, we analyze how the *"cultural"* background influences the learning of programming and the performance of students.

Keywords: Computational thinking · Programming · Coding · Extreme apprenticeship · Motivation

1 Introduction

The relevance of informatics as an autonomous discipline and the computational thinking as a general methodology has been pointed out in several papers and reports [6,9,16].

In the last twenty years, the importance of information technology in school courses has been emphasized also in the Italian educational landscape: this attention involves school courses at all levels, from primary school to University. The introduction of informatics in Italian school has a long history. In the Nineties,

O. Gaggi et al. (Eds.): GOODTECHS 2016, LNICST 195, pp. 325–332, 2017.
DOI: 10.1007/978-3-319-61949-1_34

an experimental project called **PNI** - *Piano Nazionale dell'Informatica* (in English, Computer Science National Project) has been proposed and implemented as a pedagogical avant-garde: the main objective was to align Italian curricula to European and American educational trends, in which "technological skills" have a central role.

However, the effective presence of informatics in different courses and levels is heterogeneous, with an emphasis on digital literacy and the use of digital tools. Even if the role of computer science education has been largely recognized (recent Italian ministerial recommendations are clearly ITC-oriented), the acceptance of the "digital culture" as a central subject of didactical curricula still encounters some obstacles. Many difficulties are related to the lack of resources and teachers' specific educational training. Also, computer science curricula for high school are diversified; and, with only a few exceptions, informatics is still treated as a "subordinate science". Consequently, the majority of students face academic courses without a solid preparation in principles of informatics. Regarding that issue, a crucial and interesting case study is represented by first-level courses of programming [9,11].

Programming abilities and computational thinking involve many skills, such as, for instance: reading comprehension, critical and systemic thinking, cognitive meta-components identification, planning and problem solving, creativity and intellectual curiosity, mathematical skills and conditional reasoning, procedural thinking and temporal reasoning, analytical and quantitative reasoning, as well as analogical, syllogistic and combinatorial reasoning. Teaching programming at university, particularly in introductory courses, seems to be a challenge. A promising perspective in promoting computational thinking is the *Cognitive Apprenticeship* (**CA**) learning model, 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 subject [7]. An extension of the Cognitive Apprenticeship model, *eXtreme Apprenticeship* (**XA**), was proposed in introductory programming courses by a team of University of Helsinki [14,15]; **XA** emphasizes communication between teacher and learners during the problem-solving process. This methodology has been successfully applied also in different contexts: operating systems [2,3], databases [4], mathematics [8] and, more recently, in secondary schools. A "light" version of XA technique has been introduced for students in *Applied Mathematics* and *Computer Science* at the University of Verona [13].

Students' cultural background could impact the effectiveness of teaching programming. This cultural background, mainly connected with the educational experience during the secondary school, could also define in some way the possibility to achieve good results. Some authors classify students as *science-oriented* and *humanities-oriented* according to their approach to problem solving [1]. Students can also be classified on the bases of the teaching method adopted (traditional teaching method, cognitive apprenticeship, extreme apprenticeship and so on). In this paper we analyze early performances in learning programming of students enrolled at the first year of the course in *Applied Mathematics*. The aim of this pilot study is to verify the impact of different educational experiences on students' achievement measured by a partial test covering the first part of the course Programming with Laboratory. A subset of students was trained with a traditional teaching methods, whereas a second subset was trained using **XA**.

2 Method

The sample consisted of 140 students (51.4% males, mean age 20.3) enrolled at the first year of the bachelor degree in Applied Mathematics in Verona throughout the last three academic years. The traditional teaching methodology of the past years has been updated in the current course introducing the eXtreme Apprenticeship (XA), so that some students of the current year in the sample (68,6%) was trained with a traditional teaching method and the others with XA teaching method. Concerning students' cultural background, participants came from different kind of schools. Most of them (59,3%) are from a secondary school which emphasizes mathematics and sciences. Another group of students (27,1%) come from technical and professional schools, with different curricula emphasizing technical and professional skills. The third group, the smallest one (13,6%), comes from humanistic studies (humanities, language and social sciences high schools). In order to preliminarily evaluate students' previous skills, we asked them to fill out a questionnaire to collect informations about their school experience in informatics. We can qualitatively describe the result in the following way: in most cases (about 75%) informatics was not a subject of the last year and in general was not treated autonomously; about 50% of them (think to) know the notion of algorithm and programming language, but the great majority have no relevant experience in programming in a specific language.

2.1 Research Design and Data Analysis

At the end of the initial period of the course Programming with Laboratory, students took a partial exam (in the following, *test*) considered as part of the final examination. The following parameters are considered for the evaluation: correctness of the solution, logical structure and good programming practices. The test consists of two parts: a general theoretical section (TH), in which the knowledge of fundamental notions (e.g. the definitions of compiler, interpreter, specification...) are verified; a practical, programming section (PR), where students must solve a few exercises of increasing difficulty about programming competences and problem solving skills. The evaluation of the test produced a quantitative score, which was normalized in the range 0–1 to allow the comparison among the three different groups of cultural background and the comparison between the theoretical and the programming outcomes. At the end, two different-even if related-quantitative dependent variables were considered: theoretical score (TH), and programming score (PR). A quasi-experimental design was used, with two between-subject factors: 1. educational background (with three conditions: science, humanities, and technical/professional); 2. the teaching method (with two conditions: XA and TT), and the two learning outcomes as the dependent variables (TH and PR). A mixed 3X2X2-ANOVA (Fisher's ANalysis Of VAriances, see e.g. [5]) were run, with the school group as a three-level between-subject factor, the teaching method as a two-level between-subject factor, and the two different scores as the within-subjects factor (hence the classification 3X2X2).

In the following, we will adopt the standard general form to write ANOVA results, i.e. $F(a,b) = c; p \bullet d$, where: $\bullet \in \{<, >, =\}$; the ratio c of the F-statistic depends on a, b, that represent the degrees of freedom of the between-subject variables and of the within-subject variables respectively; p is the *p-value* and d is a threshold (traditionally set to .05). When $p < d$, observed data can be considered statistically meningful. Another statistical measure we use is the *effect size*, a standard measure that can be calculated from any number of statistical outputs. Informally, effect size expresses the mean difference between two groups in standard deviation units, and will be denoted as η^2. For further details, see [5].

2.2 Results

A mixed 3X2X2-ANOVA was run to check for statistical differences among groups' means. The main effect of the secondary school was significant (F(1,134) = 3.424; $p < .05$) even if with a small effect size ($\eta^2 = .05$), with students from scientific high school performing better than technical/professional students, and than humanities students, which showed the worst overall

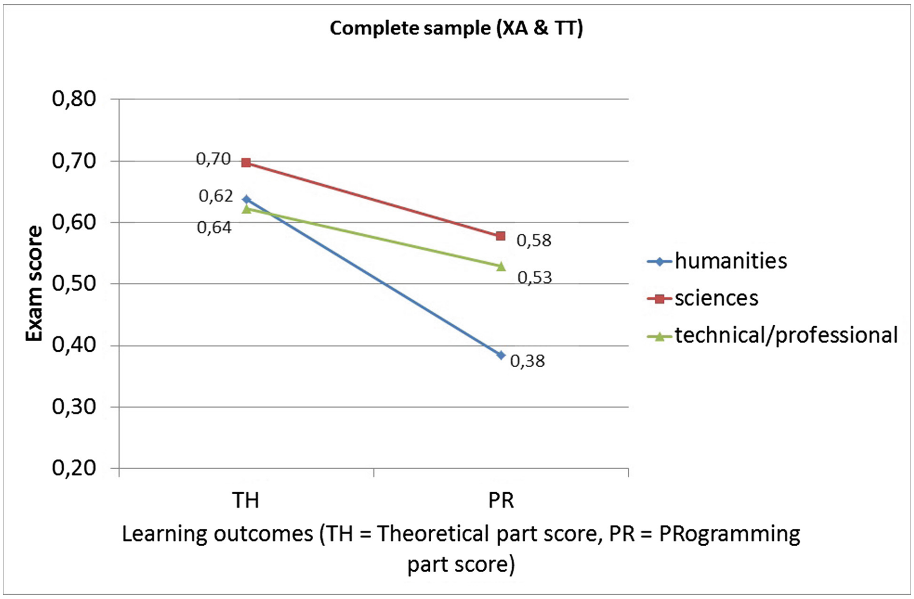

Fig. 1. Average test scores considering the two learning outcomes (TH and PR) for the three school groups (sciences, technical/professional, humanities) in the whole sample.

performance. The main effect of the teaching method was also significant (F(1,134) = 20.979; $p < .001$) with a medium effect size ($\eta^2 = .14$): students trained with XA performed better than students trained with the traditional teaching method. The main effect of test (F(1,134) = 60.003; $p < .001$) with a medium effect size ($\eta^2 = .31$) showed students performed better on the theoretical part than in the programming part. An interesting result concerns the interaction between the kind of learning outcomes and the secondary school (F(1,134) = 3.68; $p < .05$) even if with a small effect size ($\eta^2 = .05$). This effect is showed by Fig. 1, which represents the average test score considering the two learning outcomes (TH and PR) for the three groups (sciences, technical/professional, humanities). For all the three groups the programming part is more difficult than the theoretical one, but for students from humanities schools this gap was higher. No other results were statistically significant, but an interesting trend is shown by Figs. 2 and 3, showing learning outcomes in TH and PR in the three different school groups distinguished by teaching method. XA seems more effective for students coming from humanities high school, mainly for the theoretical test. Students coming from technical/professional schools showed the lowest level of benefit from the XA teaching methodology.

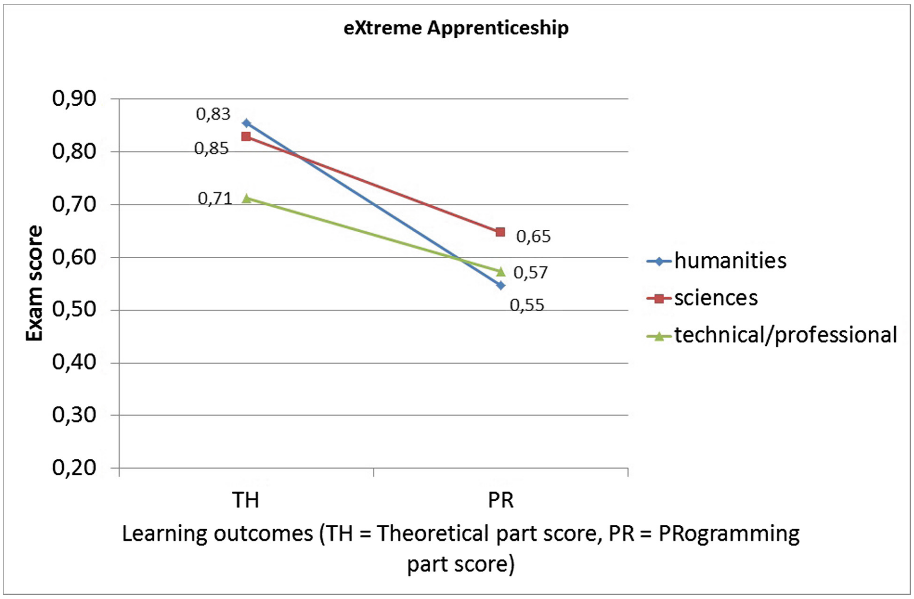

Fig. 2. Average test scores considering the two learning outcomes (TH and PR) for the three school groups (sciences, technical/professional, humanities) for students with XA.

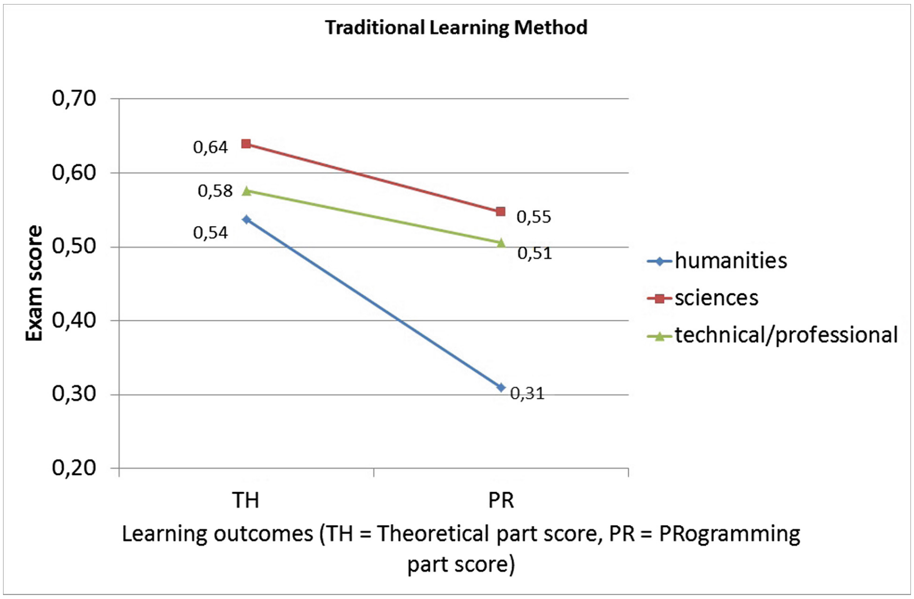

Fig. 3. Average test scores considering the two learning outcomes (TH and PR) for the three school groups (sciences, technical/professional, humanities) for students trained in the traditional learning method.

3 Conclusions and Future Work

Programming abilities require a set of knowledges and skills related to the so-called computational thinking, whose importance has been largely recognized. Teaching programming has proved to be a critical task, and becomes particularly interesting and challenging when this matter is proposed to students of non-vocational curricula. Students at the beginning of their university studies come from different educational backgrounds, and some of the skills connected with computational thinking could be differently owned by students on the basis of their secondary school experience. For instance, some of these skills could be less familiar to humanities-oriented learners than to their sciences-oriented colleagues.

The aim of this pilot study was to verify the impact of different educational experiences on students' performance in developing programming abilities. Results highlighted that human-oriented learners showed more difficulties than sciences-oriented colleagues, more in practical tasks than in theoretical ones. Is this gap insurmountable? A promising perspective to face this issue seems to be the use of appropriate teaching methodology, as for example the Cognitive Apprenticeship learning model, and particularly eXtreme Apprenticeship, a learning model which emphasizes communication between teacher and learners during the problem-solving process. In the present pilot research, the gap between humanities-oriented and sciences-oriented students has been reduced in the group who attended programming courses with XA.

An improved of teacher and peer-wise support combined with automated scaffolding [10] and an enhancement on the motivation side [12] could produce better results.

Further studies are needed. First we aim to verify whether this result in this small sample can be replicated and generalized. Second, it will be interesting to enrich the ANOVA with further subjects such as genres and high-school final evaluation. Finally, we plan to better understand the underlying mechanism which allows this improvement in computational thinking.

References

1. Billington, J., Baron-Cohen, S., Wheelwright, S.: Cognitive style predicts entry into physical sciences and humanities: questionnaire and performance tests of empathy and systemizing. Learn. Individ. Differ. **17**(3), 260–268 (2007)
2. Del Fatto, V., Dodero, G., Gennari, R.: Assessing student perception of extreme apprenticeship for operating systems. In 2014 IEEE 14th International Conference on Advanced Learning Technologies (ICALT), pp. 459–460. IEEE (2014)
3. Del Fatto, V., Dodero, G., Gennari, R.: Operating systems with blended extreme apprenticeship: what are students' perceptions? Interact. Des. Archit. J. (IxD&A), special issue (2015)
4. Del Fatto, V., Dodero, G., Lena, R.: Experiencing a new method in teaching databases using blended extreme apprenticeship. Technical report (2015)
5. Freedman, D.A.: Statistical Models. Theory and Practice, 2nd edn. Cambridge University Press, Cambridge (2009)
6. Gander, W., Petit, A., Berry, G., Demo, B., Vahrenhold, J., McGettrick, A., Boyle, R., Mendelson, A., Stephenson, C., Ghezzi, C., et al.: Informatics education: Europe cannot afford to miss the boat. ACM (2013)
7. Ghefaili, A.: Cognitive apprenticeship, technology, and the contextualization of learning environments. J. Educ. Comput. Des. Online Learn. 1–27 (2003)
8. Hautala, T., Romu, T., Rämö, J., Vikberg, T.: Extreme apprenticeship method in teaching university-level mathematics. In: Proceedings of the 12th International Congress on Mathematical Education, ICME (2012)
9. Katai, Z.: The challenge of promoting algorithmic thinking of both sciences-and humanities-oriented learners. J. Comput. Assist. Learn. **31**(4), 287–299 (2015)
10. Pärtel, M., Luukkainen, M., Vihavainen, A., Vikberg, T.: Test my code. Int. J. Technol. Enhanc. Learn. 2 **5**(3–4), 271–283 (2013)
11. Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Devlin, M., Paterson, J.: A survey of literature on the teaching of introductory programming. In: Working Group Reports on ITiCSE on Innovation and Technology in Computer Science Education, ITiCSE-WGR 2007, pp. 204–223. ACM, New York (2007)
12. Solitro, U., Pasini, M., Brondino, M., Raccanello, D.: The challenge of learning to program. In: PPIG 2016, September 2016

13. Solitro, U., Zorzi, M., Pasini, M., Brondino, M.: A “light” application of blended extreme apprenticeship in teaching programming to students of mathematics. In: Caporuscio, M., De la Prieta, F., Di Mascio, T., Gennari, R., Rodríguez, J.G., Vittorini, P. (eds.) Methodologies and Intelligent Systems for Technology Enhanced Learning. AISC, vol. 478, pp. 73–80. Springer, Cham (2016). doi:10.1007/978-3-319-40165-2_8
14. Vihavainen, A., Luukkainen, M.: Results from a three-year transition to the extreme apprenticeship method. In: 2013 IEEE 13th International Conference on Advanced Learning Technologies (ICALT), pp. 336–340. IEEE (2013)
15. Vihavainen, A., Paksula, M., Luukkainen, M.: Extreme apprenticeship method in teaching programming for beginners. In: Proceedings of the 42nd ACM Technical Symposium on Computer Science Education, pp. 93–98. ACM (2011)
16. Wing, J.M.: Computational thinking. Commun. ACM **49**(3), 33–35 (2006)