Teaching how to teach how to teach programming

Vasilios Dagdidelis *dagdil@uom.gr*
University of Macedonia, Greece

Abstract

Over these last 12 years in Greece, a very large program has gradually been implemented whose aim is to integrate ICT into education generally, and into teaching practice specifically. Recently, Informatics and Computer Science teachers were also included. Training of teachers is an important part of this project – especially Informatics teachers. The preparation of trainers of these teachers "includes" necessary three distinct levels of interaction (preparation of trainers, training teachers, teaching of pupils) and thus it requires a negotiation on three distinct levels. Several important aspects of this preparation of trainers are examined and some findings, concerning the majors problems of this preparation, are examined. Our most important finding shows some aspects of the complexity of such a preparation program.

Keywords

Teaching Programming, Training Teachers, Transposition Didactique.

Introduction

Programming as a specific field of knowledge, balancing between art (Knuth, 1974) and science (Dijkstra, 1971; Dijkstra, 1976), between the demands of a rigorous branch of applied mathematics (Hoare, 1996) and the requirements of a profession, is a highly complex mental activity. Learning to program, even at an introductory level, seems to be difficult for young students: they usually do not understand the concepts of algorithms, data structures and programs and do not appear to be able to solve problems go beyond the ordinary or obvious.

As in other scientific fields, such as Mathematics or Physics, effective teaching requires specific course organization. The teacher should have specific knowledge and skills in order to not only recognize the difficulties encountered by students, but also to use possible methods for overcoming these difficulties, as well as to transform student misconceptions into other, more functional concepts. The teaching of programming, therefore, is a very demanding process. The teacher has to teach concepts whose "economy", i.e. function, cannot be understood in class. Students can, in the context of a programming course, for example, understand how an assignment or a selection structure in a given programming language works. However, it is much more difficult for them to understand why it is necessary to program in the context of specific *paradigms*, such as structured programming or object-oriented programming. The reasons for such a choice are of a scientific or economic nature, and these conditions cannot be reproduced in the classroom.

The basic elements that the preparation of teachers of programming should include are students' spontaneous perceptions and their standard errors, which will give teachers the knowledge needed to be able to deal with any difficulties the students encounter in the classroom; also course organization in order to maximize teaching efficiency and knowledge acquisition.

At another level, when preparing their courses, trainers instructing teachers not only have to be aware of the students' difficulties in solving programming problems (i.e. what the teachers should know), but also they need to be acquainted with any difficulties that the teachers themselves may encounter in teaching programming..

In this paper, as the title implies, we present a number of practices used in the preparation of trainers, In other words practices that are related to teaching trainers how to teach teachers how to teach students programming.

The "PAKE" schools

Over these last 12 years in Greece, a program has gradually been implemented whose aim is to integrate ICT into education generally, and into teaching practice specifically. Basically, the project involves K12 education in Greece. It is worth noting that within the context of this program seminars have been attended by about 120,000 teachers, some of whom have attended more than one (i.e., at different levels). This number represents approximately 65% of all active teachers (estimated total: 180,000) Recently, Informatics and Computer Science teachers were also included, making this a very significant program affecting the entire Greek education system.

The program has three levels of training/education: the first level corresponds to basic Computer skills, such as word processing or Internet navigation. The second level corresponds to specific training for teachers on how to use ICT in the classroom. At this second level, each scientific discipline (primary teachers, teachers of Mathematics, Physics etc) follows specialized courses in pedagogy and teaching using ICT. The third level corresponds to the training of trainers to teach at the second level. This takes place in training schools with the Greek acronym "PAKE". In Greek it stands for **PA**nepistimiaka **K**endra **E**kpedefsis, «*Πανεπιστημιακά Κέντρα Εκπαίδευσης*», meaning "University Centers for Educating (Trainers)".

These centers periodically operate as schools that prepare trainers, in other words, they are a very special kind of educational establishment where trainers of teachers are instructed by highly specialized educators. The following diagram may better explain the role of PAKE:

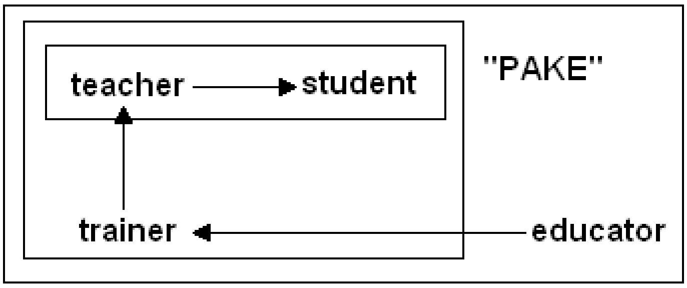

Figure 1. educating trainers in "PAKE". Each arrow means "educates"

Thus, training of trainers in PAKE requires a negotiation on three distinct levels:

- Level 1: teachers teach programming to students. Several research findings point out the difficulties faced by students when dealing with programming concepts and students' perceptions when attempting to solve problems. The concept of *teaching scenario* was created for the teaching of programming. A teaching scenario is a detailed description of a teaching module which can last more than one hour. Apart from the description of the learning objectives and the lesson itself, each teaching scenario includes information concerning classroom organization, the learning theories which the course is founded on, and when necessary, elements on epistemology. The teaching scenario attempts to provide a broad and in-depth analysis of the course that the teacher has prepared.

- Level 2: trainers instructing teachers. Besides the general principles of adult education, there are no substantial research on findings specifically on the training of Informatics and Computer Science teachers. The *training scenario* - similar to the teaching scenario- is the central idea of these courses. A training scenario describes in detail a unit (a piece of knowledge) which is taught to teachers. As is the case with the teaching scenario where a very important element is students' misconceptions of students, a significant part of the training scenarios make reference to the basic problems that may arise in the training of teachers. A characteristic problem is Greek teachers' awareness (or lack of) the necessity of such training. A typical response is: What can 144 training hours of seminars offer to teachers with 10-15 years of teaching experience? Teachers also tend to pose specific objections example, why they have to be trained in the teaching of Logo and Logo-like environments (such as Scratch and the Greek environment *Xelonokosmos*, the StarLogo and Turtle Art). The program tries_to include all the necessary elements in order to provide complete training to Informatics and Computer Science teachers.
- Level 3: training trainers (educators at PAKE instruct trainers). Few research data exist in this area – if any. In this case there is no particular scenario to follow, as is the case with levels 1 & 2 above, as the content and purpose of training trainers is too broad to fit into a single scenario model. A key element in the training of trainers was the distinction between levels 1 and 2. For instance, the trainers themselves many times questioned the need to actually create training seminars. The training of trainers does not only include preparing them to deal with new technologies that are known but have not yet been introduced in education - such as tablets and smartphones but also to prepare them for technologies that are completely unknown. How does one prepare trainers in technologies that are unknown? Among other things, certain recent changes in the Computer Science course at school, has resulted in added obstacles to training. In Greek schools, the interdisciplinary approach to the various subjects has recently been introduced along with a 'projects' approach (students work in groups on a specific project topic over the semester). Even in IT, there is an evident orientation to digital literacy or computers/information literacy. These changes are creating a new framework, a new "school ecology" in which the IT course is a part of. As is perhaps to be expected, these changes were not immediately accepted by all teachers thus compounding the task of training.

Examples, good practices, and some observations

The most important principles and applications of teaching practices that the trainers' training courses were based on are listed below.

- A comprehensive training curriculum that is versatile and long term. The duration of the curriculum of the participant trainers' course lasts over 380 hours covering many scientific fields and areas of knowledge. Thus, distance learning, synchronous and asynchronous (with environments such as Moodle platform and e-luminate) as well as issues related to the management of school laboratories and learning environments such as LAMS, are covered to a great extent. Besides technical skills, the trainers were also given an important theoretical base which included learning theories and teaching methodology, as well as issues related to adult education and modern theories about web 2.0 and Social Networking. What is termed the “General” section comprises an important component of the trainers’ training. This section of the program is shared with other disciplines, such as Math or Physics in similar PAKE seminars. It should be noted here that some trainers disputed the need for such the general section in their training. Even though as a general principle it was accepted that there is common subject matter in different academic fields, such as learning theories or certain *didactic* phenomena (for example, incorrect or incomplete perceptions of students) which can be applied to various courses, nevertheless, trainers were sceptical about being taught subjects that were not, in their view, directly related to ICT.

- The training of trainers includes a very strong component of the Didactics of Informatics Theory and particularly that of computer programming. Apart from the basic concepts of teaching, such as a the concept of *didactic contract* and *transposition didactique*, this theoretical framework also includes research data related to concepts such as variables, the structure of repetition and choice teaching recursion, empirical research on the different programming environments and programming languages, algorithms, data structures, and other concepts. This approach enabled the differences in the construction of the teaching and training scenarios to be clearly seen, since the theoretical model of reciprocal interactions and observations of a teaching system (see Figure 1) clearly defines the framework within which these teaching interactions take place. The theoretical approach also contributed to the understanding of didactic material as *phenomena*, i.e. as observable events that are not random but are produced by causes and can thus be studied. Despite the fact that the trainee-trainers seem to understand and accept the Theory of the Didactics of Informatics and in particular of programming, it is not fully certain, however, whether they found all the didactic concepts taught useful. Especially in the early years, similar reservations were expressed in other fields, such as in the Didactics of Mathematics. Today, however, very little doubt remains as the Didactics of Mathematics is an established field of research in its own right.
- Practical experience constitutes a central factor in the training of trainers. The practical aspect of the course included the trainers having to observe both teaching and training classes. In addition, they had to create both teaching and training scenarios, and then implement their teaching. Each trainer had to first teach students at a school and then teach teachers at an in-service seminar. At all times a PAKE supervisor provided assistance and guidance during these practicals. The subject matter that was taught was decided by both the supervisor and trainee-trainer. Furthermore, all the observations and instructions were carefully designed with the supervisor’s help. On completion of the practical, the trainee-trainers were required to submit a detailed report that was discussed by all the participants.
- In addition to becoming familiar with environments such as Moodle, platform e-luminate, LAMS and the like, participants are also taught to create teaching and training scenarios both for concepts related to various programming paradigms and different languages and environments. Their training includes theory and exercises related to teaching structured programming, object-oriented programming and functional programming (Logo-like style). These three options in some way correspond to the three different orientations for the teaching of programming in Greek schools: (i) programming as part of a broader informatics fluency (mainly in the first 9 years); (ii) programming as a preparatory course for high schools (structured); (iii) programming as a job qualification (object-oriented, mainly in vocational schools). The respective languages are Logo-like languages, and those for object-oriented and structured programming. In most cases, the software used has many options

which can support novices in learning programming concepts: they one to see, to some extent, the execution of the programs; they have structure editors, friendly interface and messages with precise content and clear meaning. Programming concepts are also taught indirectly in other environments, such as spreadsheets.

- The system for the evaluation of trainee-trainers includes both a formative and final evaluation Participants are required to several essays and reports throughout the program (16 in total) that vary in content and size. They are also required to participate in forae, create (and update) blogs, to upload their work in e-portfolio (Mahara), and generally to take part in a series of activities of a digital nature that are in some way related to the training program they will undertake on completion and certification. They then must sit a final test, which includes a series of closed-type multiple choice questions, and the like. However, the section of the examination that is worth the most marks is an essay of a training scenario (that includes a teaching scenario) on a particular concept or method. Throughout the PAKE course participants prepare, present, comment on dozens of both teaching and training scenarios, with the objective to become familiar with this particular way of working.
- In addition to items directly related to the teaching of Informatics generally and programming in particular, participants are also taught a number of subjects that are essential, or simply useful for their training, such as techniques in adult education.

Comments and conclusions

The PAKE training program has provided numerous qualitative and quantitative data enabling us to reach some initial conclusions.

A point worth mentioning relates to the deeper understanding of mainly theoretical concepts, that the participating trainee-trainers acquire. For example, constructivism is a widespread theory of learning that supports multiple teaching methods. However, in many cases, the constant reference to constructivism in the participants' scenarios seems to be more of a routine procedure, rather than the outcome of thorough analysis and choice. This phenomenon was also observed in other cases. For instance, the constant reference to scenarios of particular theoretical views and generally accepted assumptions integrated into the dominant paradigm, at times appears to be more the result of an "alignment" with the paradigm, rather than an informed choice. It is often the case that theoretical concepts, especially when not accompanied by a significant number of concrete examples, are totally misunderstood by participants.

The integration of programming into the school curriculum requires a specific transposition didactique: scientific knowledge and professional practice is transferred to the school and included in the educational context. Thus, the concepts are simplified to make them more easily understood; the course is organized into 45-50-minute sessions, which is the duration of a teaching period in Greek schools; the subject matter has been organized into introductory units, exercises, questions, problems. However, during this transposition of concepts into the school environment, another transformation takes place: namely, the scolarization of an entire scientific field favors certain types of problems, while marginalizing other aspects of the key concepts, such as algorithms, data structures, and programming generally. For example, data structures as a means of encoding entities of the external world, such as images, text, etc., are rarely referred to in the school environment. Trainee-trainers in many cases, did not fully acknowledge the importance of teaching or the activities on such issues, considering them of no use, since they were not directly related to current concepts of programming – at least the "school version" of programming.

Due to the fact that this program was implemented for Informatics and Computer Science for the first time in 2011, the data thus far do not allow us to evaluate further the effectiveness of the training that the participant trainee-trainers underwent. Subsequent phases of the training program, we hope will provide more in-depth information on this issue.

References

Chevallard, Yves (1994) Les processus de transposition didactique et leur théorisation, Contribution à l'ouvrage dirigé par G. Arsac, Y. Chevallard, J.-L. Martinand, Andrée Tiberghien (éds), *La transposition didactique à l'épreuve*, La Pensée sauvage, Grenoble, p. 135-180.

Dijkstra, Edsger W. (1971) EWD316: *A Short Introduction to the Art of Programming*. T. H. Eindhoven, The Netherlands, Aug. 1971.

Dijkstra, E.W. (1976), *A Discipline of Programming*, Prentice-Hall Series in Automatic Computation.

Hoare, Antony R. (1996) *Mathematical models for computing science*. NATO ASI DPD 1996: 115-164

Knuth, Donald E. (1974) Computer Programing as an Art, *ACM Turing award lectures*, ACM New York, NY, USA.