TEACHING PROGRAMMING

Aristides Dasso[‡], Ana Funes[‡], Daniel Riesco[‡], Germán Montejano[‡], Mario Peralta[‡], Carlos Salgado[‡]

[‡]Universidad Nacional de San Luis,
Argentina
Tel.: +54 (0) 2652 42 4027 Fax: +54 (0) 2652 43 0224
{arisdas, afunes, driesco, gmonte, mperalta, csalgado}@unsl.edu.ar

Abstract. Programming –as Programming Fundamentals– has been and still is a basic part of most Computing Curricula including all of the Computing Disciplines – we use here the terminology introduced in [1]. It is also considered by many a problem solving teaching methodology. There are several facets or issues of Teaching Programming that we think are very interesting and must be taken into account and that are answers to the questions of *why*, *what*, *how* and even *when* to teach programming. But even those questions receive different answers. We consider some in this paper.

Keywords: Teaching Programming. Programming Languages. Programming Paradigms.

1 Introduction

There was and still is a raging dispute in Teaching Programming about several issues that have to do with the method, programming language to use, and these have as well considerations such as: Do we use a widely employed language –at the time the teaching is taking place, of course–or an ad hoc teaching programming language which is supposedly academically more suitable.

We treat some of these issues in the next section dividing them into questions namely "why teach Programming?" in section 2.1, "what to teach in Teaching Programming?" in section 2.2, "how to teach Programming?" in section 2.3, and finally "when to teach Programming?" in section 2.4. In section 3 we give some conclusions.

2 Some Issues

We explore here some of the issues mentioned above. The order is completely meaningless and immaterial. We do not pretend to be thorough or consider that the subject is closed. On the contrary there is room –a lot of it! – for further discussion on these topics and others.

2.1 Why Teach Programming

There are generally two answers to the question: (a) because is needed in some professional context; (b) because is considered a good problem solving methodology.

Why to teach Programming takes us to explore whether Programming is going to be taught because is considered part of the professional curricula or just as a sort of 'training' in problem solving.

Both issues have their pros and cons.

It is not always clear whether Programming is really necessary to the professional curricula considered. Some of the Computing Disciplines present no doubts about it, for others are harder to decide. Take Information Systems for example. Do students of this discipline need to know how to program? Are they ever going to make a program? However Programming Fundamentals appears in all five Computing Disciplines in the "Computing Curricula 2004", see [1, table 3.1, p. 28]

In any case the question here is how much of the curricula time is taking up in learning Programming.

On the other hand it is believed by many that Programming is a good problem solving methodology and so it can be used as a way of training –especially young minds [6].

But if programming is 'like' problem solving, is there a way to teach it? Can there really be a general problem solving methodology? Can we go further than sharpening the natural ability of somebody to solve problems and teach her/him how to solve it? [7]

2.2 What to Teach

This question encompasses several issues that have to do with (a) the paradigm in which to teach; (b) consequently the language used; (c) problems to use as examples; (d) programming methodology; (e) and now with the advent of the Internet appears Web Programming –is it all that different from the "normal" programming ?

Programming Paradigms and Languages is one of the most disputed issues –if not the most disputed issue– in this area. See [9, 10]

We can recognize four Paradigms: Imperative, Object Oriented, Functional and Logical.

Each have its own set of languages and if to choose a paradigm to teach is a hot subject there is also a big dispute on which language is the most suited to the paradigm. So for instance if one would choose what is probably the most common paradigm –the imperative– then the dispute over which language to choose goes from those currently employed by the industry to those that seems more appropriate academically speaking. See for instance [11].

There are a number of publications criticizing one or another language either as unsuitable for teaching programming or as plain unsuitable for programming. For example see [12, 13, 14]

In "Computing Curricula 2001, Computer Science" [8] three approaches are considered in introducing Programming Fundamentals each relating to one of the paradigms mentioned above –the logical paradigm is sometime included in the functional paradigm– namely 'Imperative-first', 'Objects-first' and 'Functional-first'. The 'Breadth-first' approach –that calls for a more holistic point of view– although it has been strongly recommended has been difficult to implement in practice.

There is another approach –Algorithms-first– that recommends using pseudo-code instead of a particular language but it is as opposed to a Hardware-first approach.

For a more detailed discussion on these approaches see [8, pgs 28 and following]

To add to this lately web programming has appeared for some as a separate area in programming. However this seem to recognized the need for Programming Fundamentals as a base to build the specific knowledge to be able to program web applications using some of the languages that have grown out of the need for programming web applications

2.3 How to Teach Programming

From textbooks –of which there is a plethora, to creating special learning environments [2, 3, 4], to just plain amusement –some may even considered 'serious' amusement [5], there is a lot to choose from when deciding how to teach programming.

Here we are confronted by those that insist that programming must be taught using examples and particularly using interactive media on a computer on a hands-on approach and those that insist on teaching the theoretical fundamentals of programming leaving the practice using a particular language for when the students have mastered the theoretical fundamentals of programming.

2.4 When to Teach Programming

Meaning at which level. Do we teach programming: (a) in secondary school, (b) at university level, (c) for students doing something else that have nothing to do with Computer Science?

Some authors believe that Programming can be a good methodology to teach problem solving (see section 2.1, above) and so they include programming in the secondary school curricula. It is not clear whether teaching programming develops problem-solving skills or whether those skills are necessary to learn programming.

On the university level, does Programming have to be taught early on in Computing Curricula or the introductory curriculum has to have a wider base and programming must be taught as a more advanced course?

The programming-first approach that was –and still mainly is– the traditional historical approach to the introductory curriculum in Computing has had many objections (see [8, p.22, and following] for more on this), however it is recognized that "the programming -first model is likely to remain dominant for the foreseeable future." [8, p.24]

Is it convenient for students of other disciplines that have nothing to do with Computing Science to learn programming? Are they ever going to program anything or just used packaged software? Lately these questions seem to have been answered in the negative. Students of other disciplines do not need to learn programming since it is very unlikely that they will ever have a need for it.

3 Conclusion

We have presented here a few of the issues in Teaching Programming, mostly in the form of questions with in some cases some suggested answers. We do not think that this covers the whole field nor that ii provides a closure to the subject.

On the contrary we truly expect to continue the study and discussion on the field and hope that the present paper can serve that purpose.

References

1. Joint Task Force for Computing Curricula 2004, "Computing Curricula 2004, Overview Report including A Guide to Undergraduate Degree Programs in Computing". The Association for Computing (ACM), The Association for Information Systems (AIS), The Computer Society (IEEE-CS), November 22, 2004.
2. Victor Adamchik, Ananda Gunawardena, "A Learning Objects Approach to Teaching Programming". Carnegie Mellon University.
3. Peter Van Roy, "A concepts -based approach for teaching programming". SIGCSE 2004.
4. Ray Kemp and Ben du Boulay (Chairs), "Innovations in Teaching Programming". VOLUME VII OF AIED2003 SUPPLEMENTARY PROCEEDINGS. 11th International Conference on

Artificial Intelligence in Education.

5. The following URL has some links to the different Esoteric Programming Languages: http://en.wikipedia.org/wiki/Esoteric_programming_language
6. S. Pappert, "Mindstorms".
7. G. Polya, "How to Solve it". Second Edition. Doubleday Anchor Books. Doubleday & Co. Inc. New York, 1957.
8. The Joint Task Force on Computing Curricula, IEEE Computer Society, Association for Computing Machinery, "Computing Curricula 2001, Computer Science". Final Report, (December 15, 2001)
9. Peter Van Roy (Moder.), Joe Armstrong, Matthew Flatt, Boris Magnusson, "The Role of Language Paradigms in Teaching Programming". Panel in SIGCSE 2003, February 19-23, 2003, Reno, Nevada, USA.
10. R. Jernigan. B. W. Hamill. D. M. Weintraub, editors, "The Role of Language in Problem Solving". North Holland, Amsterdam, New York, Oxford, 1985.
11. J. N. P. Hume, "A Guide to the PC-Turing Interpreter" along with R. C. Holt and J. R. Cordy "The Turing Language Report" and J. R. Cordy and T. C. N. Graham "Commands for Turing Programming Environment". Computer Systems Research Institute, University of Toronto, Toronto, Canada, 1986.
12. Jacqueline L. Martin, "Is Turing a better language for teaching programming than Pascal?", Honours Dissertation, January 1996, University of Stirling, Department of Computing Science. http://www.holtsoft.com/turing/essay.html
13. Ian Joyner, "A Critique of C++", Unisys ACUS, 115 Wicks Road, North Ryde, Australia, 1992.
14. Brian W. Kernighan, "Why Pascal is Not My Favorite Programming Language". April 2, 1981, http://www.lysator.liu.se/c/bwk-on-pascal.html#bwk