<http://www.diva-portal.org>

Postprint

This is the accepted version of a paper presented at *11th annual International Conference of Education, Research and Innovation*.

Citation for the original published paper:

Mozelius, P. (2018)

Teaching The Teachers To Teach Programming: On Course Design and Didactic Concepts

In: *ICERI2018 Proceedings* (pp. 8031-8037). Sevilla: The International Academy of Technology, Education and Development

Proceedings of International Conference of Education, Research and Innovation

<https://doi.org/10.21125/iceri.2018.0445>

N.B. When citing this work, cite the original published paper.

Permanent link to this version:

<http://urn.kb.se/resolve?urn=urn:nbn:se:miun:diva-34884>

TEACHING THE TEACHERS TO TEACH PROGRAMMING – ON COURSE DESIGN AND DIDACTIC CONCEPTS

P. Mozelius

Department of computer and system science, Mid Sweden University (SWEDEN)

Abstract

Involving computer programming in primary and secondary school is an urgent issue in many countries, and in Sweden this should be rapidly implemented during 2018 and 2019. First subjects that should implement programming in the teaching and learning activities are mathematics and technology. Some teachers have earlier experiences of programming, but for many primary and secondary school teachers programming is a new discipline. The aim of this paper is to describe and discuss how requirements have been gathered and implemented in a course for teacher training on programming for mathematics and technology in K-12 education.

The study was carried out as three phases of a development process inspired by design science. In the first phase the main problem was formulated, and in the second phase requirements were gathered and discussed in a brainstorming workshop where the course developers tried to align learning outcomes with suitable didactic ideas. In the third phase the requirements from the first phase were implemented in the actual course development.

Conclusions are that the idea of teaching the teachers needs a different course outline than how traditional design of programming courses. Higher education has a tradition of training students for system development in the industry. Teacher training should have a focus on didactic concepts that later could be reused in courses on mathematics and technology in K-12 settings. Examples of such didactic concepts are, computational thinking, pair programming, visualisation and game-based learning. Important also that teachers taking the course should produce material that can be useful in their future daily teaching.

Keywords: Programming education, Teacher training, K-12, Course development, Digitalisation.

1 INTRODUCTION

Involving programming and computational thinking in primary and secondary school is an urgent issue in many countries [1, 2], and in Sweden, this should be implemented during 2018 and 2019 [3]. In the subjects of Mathematics and Technology, this will be a rapid and mandatory process for teachers that often lack earlier programming experience. All teachers in Mathematics and Technology should take a 7.5 ECTS course in fundamental programming at their local university. Subject matter experts at various Swedish universities have constructed a general course framework, but there are no specific didactic guidelines. Each university is free to choose programming languages, didactics and course design.

Programming courses in higher education has, for natural reasons earlier had a strong focus on skills and knowledge that are useful for a professional programming career. For a younger primary and secondary audience research studies have shown that the focus should rather be on computational thinking (CT), than code construction [1, 4]. CT is a concept that involves problem solving with the use of generic computer science techniques, and something that a teacher can organise with activities that does not necessarily need to involve any computers [4, 5]. The definition of CT was refined in 2014 by Selby and Woollard [6] to consist of the seven aspects of 1) A thought process, 2) Abstraction, 3) Decomposition, 4) Algorithmic design, 5) Evaluation, 6) Generalisation and 7) Automation.

This study emphasised on the aspects of Abstraction, Decomposition, Algorithmic design and Generalisation. Abstraction that according to Wing [7] is the most distinctive aspect of CT that differs from ordinary thinking. Abstraction, here defined as it was by Faber [8], something *that “lets an individual look at certain aspects of a problem or situation by hiding complex aspects of said problem or situation. By removing the aspects of a problem that are not relevant, the individual is not distracted and can direct all attention to the important aspects of the problem”*. Decomposition, which is related to abstraction [8], is the idea of dividing a complex problem into smaller problems much like the ‘divide and conquer’ technique that is frequently used in computer science [9]. Furthermore, algorithmic design, another

concept with its origin in computer science is of special importance in a course where all participants in some way use algorithms in their daily teaching. Finally, the concept of generalisation has also, from the very beginning been a fundamental part of computer science. Generalisation can, in a broader definition be seen as the concept of reusing found solutions in other contexts.

1.1 Problem and aim

The aforementioned concepts of abstraction, decomposition, algorithmic design and generalisation are all core concepts in CT as well as in computer science, and all of fundamental importance. In a new course context where teachers should learn to teach programming for a new younger K-12 audience, the traditional presentation and implementation of these concepts must get a new design. The aim of this study is to describe and discuss the definition and implementation of requirements in a course on teacher training on programming for mathematics and technology in K-12 education.

2 METHODOLOGY

The overall research strategy has been Design science inspired by the fivefold process that has been outlined by Johannesson and Perjons 14. This study was carried out following the first three phases of the process that is depicted below in Figure 1. Firstly, a problem was identified and formulated, secondly an artefact was designed to address the problem, and finally the design requirements were implemented as a course instance in the Moodle virtual learning environment.

Figure 1: Design science process phases [10]

Discussions at a one-day brainstorm workshop created the bedrock for defining requirements to outline the artefact. Main activities in the workshop are described in the next section.

3 THE WORKSHOP

General constrictions for the workshop was the syllabus framework from the Swedish National Agency for Education. A syllabus framework that was developed in a distance collaboration between programming teachers and pedagogues from various Swedish universities. The main objectives in the syllabus framework are, in a stepwise order to sort out 1) What is programming? 2) How to program? and finally 3) How to apply programming in teaching and learning sessions? Furthermore, the syllabus

framework contains requirements for involving subject related didactics Compared to traditional programming courses in higher education this requires a new and different course structure.

Common learning objectives in the syllabus framework are that course participants after a completed course should be able to:

- Apply basic programming techniques to create programs and to reflect over programming approaches
- Solve given problems by computational thinking and construction of executable programs
- Read, analyse, test, debug and improve programs and programming code
- Use programming techniques in teaching and learning activities to curricula for technology and mathematics

Participants in the workshop were three subject matter experts and programming teachers, the project leader and a primary school student. Part 1 of the workshop had a brainstorming character where a rich assortment of programming techniques and didactic was brought, discussed and written down on post-it notes. Later in Part 2, ideas were analysed more critically and aligned to the paper matrix that is depicted below in in Figure 2. The five heading are for the five fundamental course segments that are described and discussed one by one in the next session.

Figure 2: Workshop delivery with design concepts in a requirement draft matrix

4 RESULTS AND DISCUSSIONS

The general outline of the course is a sketch for a five-week (7.5 ECTS) course that should be given on 25% pace in twenty weeks. A blended learning design should involve 2x4 face-to-face meetings combined with the online learning that is organised in the Moodle virtual learning environment. Since the participating teachers all are working full-time a 25% study pace seems realistic and a blended mode with face-to-face meetings will hopefully stimulate collaboration and the formation of study groups. To met the main objectives in the syllabus framework the course was divided into five courses sections where the first four will be aligned to a dedicated one-day face-to-face session.

4.1 Section 1: Programming in school, why, what and how?

Unlike traditional courses where the first course section should bring up and discuss the why and what of programming before introducing how to construct programs. Experiences from early pilot courses indicates heterogeneous groups where only a minority has earlier programming experiences. To fine-tune the first section a digital questionnaire has been constructed to measure course participants pre-knowledge and programming experiences. The face-to-face meeting will also discuss the more general digitalisation of primary and secondary school. Finally, the how-part in the face-to-face session should also include installation of the programming environments for Python and Scratch.

4.2 Section 2: The fundamental building blocks of programming

Before introducing any didactic concepts there is a basic need of learning the fundamental programming techniques. As in all other introductory courses building blocks like variables, constants, data structures selection, iteration must be introduced before other more abstract concepts. This will be carried out with lectures and workshops that have a rich assortment of illustrative examples. In this section there will probably exist a huge need for facilitation as well.

4.3 Section 3: Didactics for Technology and Mathematics

When the fundamental building blocks are learnt it is time to explore more subject matter related ideas. Both for mathematics and technology it is relevant to introduce and discuss algorithms. This will be exemplified with programs generating Fibonacci numbers and programs searching for Prime numbers. More specialised for teachers in technology there will be a session on how to connect to hardware and how to control hardware from software constructions. Furthermore, there ought to be code generated visualisations of mathematical and technical concepts with the use of graphical add-on libraries.

4.4 Section 4: Didactics for programming education

Visualisation will be a part of the fourth section as well where various geometrical concepts are depicted with the use of 'Turtle graphics', a programming concept developed by Seymour Papert [11]. He was also the person who developed the idea of constructionism, a specialisation of Jean Piagets' branch of constructivism suitable for computer science and programming education [12]. With a relation to constructionism are the two didactic concepts of problem-based [13] and game-based learning [14]. They will both be exemplified and suggested as ingredients in the participants' final project work. Two programming concepts that should be presented are iterative and incremental development [15], and the widespread idea of pair-programming [16].

4.5 Section 5: Project work

Programming is a discipline where both theoretical knowledge and practical skills are of importance, and the two should be combined here in the final project work. The project work is also a way to meet the Swedish National Agency for Education objective that teachers should be able to apply programming in teaching and learning sessions. The delivery of the project should be an interactive digital artefact that later could be reused in course participants daily work. It is also a condition that the artefact should be constructed by programming and that there should be a multimodal interaction where users can learn something specific in the areas of mathematics or technology.

4.6 Computational thinking

Workshop participants reached a consensus on the importance of a focus on computational thinking and the four earlier described components of abstraction, decomposition, algorithmic design and generalisation.

4.6.1 Abstraction

One of the didactic ideas in the course design is problem-based learning, and more than in traditional courses, to also have an emphasis on general problem solving. This could also be combined with building flow-charts and writing pseudo code before the actual code implementation. For the project part the work-flow should be documented with text, images or as a video.

4.6.2 Decomposition

From the initially small and straightforward problem solving, assignments and exercises grow more complex. An established finding is that *“Humans solve problems that they have never encountered. Provided that the problem is not excessively complex”* [17]. The frequently used principle of reducing the search set will also be illustrated in an assignment where course participants should build the well-known game ‘Guess my number’. To understand the idea of decomposition would also be a useful help in the more complex project assignment.

4.6.3 Algorithmic design

Important concept since the target audience consist of math and tech teachers that often use algorithms in their daily teaching. The concept will be presented in three steps: 1) A general introduction to the use of algorithms in computer science, 2) An analysis of an algorithm that generates Fibonacci numbers and finally 3) An assignment where learners should build a Prime number sieve. The construction of two less complex algorithms is also the core problem in the very first assignment.

4.6.4 Generalisation

To be able to reusing found solutions in new problem solving is a fundamental principle in programming. Writing, testing and debugging code is often time consuming and reusability is a core part of computer science. Practically all content in the first four course sections can be reused and a part of the more complex project assignment. A rich variety of code examples will be presented and analysed during lectures and workshops. Learning by examples has been successful both for humans and in artificial intelligence. In the learning of an artificial programming language the building blocks

4.7 Motivational design

Reports from earlier pilot courses indicates that not all participants have a high motivation and also that teachers have questioned the relevance of course content. The choice of motivation model is ARCS, and to focus on Attention, Relevance, Confidence and Satisfaction as outlined by John Keller [18; 19]. Attention should be gained by active participation and with specific examples of programming artefacts such as digital games. Relevance might be met by the ARCS concept of future usefulness, where solutions to course assignments should have a potential for reuse in teachers’ future daily work. Confidence could be reached by iterative feedback and feed forward in a constant dialogue between learners and instructors. Finally, satisfaction could in the same way as relevance be established by the ARCS concept of future usefulness. Assignments in general and the project work in general should result in artefacts that could be teaching and learning material in teachers’ future daily work.

5 CONCLUSION

Teaching teachers to teach programming in primary and secondary school needs a different design than traditional introductory programming courses. The presented and discussed set of requirements looks promising, but what counts at the end of the road is the learning outcomes from the implemented course. To get feedback on the course design this paper will be part of the course content and further discussed with the teachers that will take course. In a longer perspective the new early introduction of programming and computational thinking in primary school will also create a need of redesign of introductory programming courses at university level.

6 FUTURE WORK

The implementation of the defined requirements looks promising, but the important next step is to carry out the two remaining phases in Figure 1. Course requirements were implemented with good intentions, but what finally counts are the extensions. The course will be given twice during the coming year, and both the instances will be thoroughly evaluated. Another interesting idea would be to compare this local implementation with other implementations of the common syllabus framework.

REFERENCES

- [1] L. Mannila et al., "Computational thinking in K-9 education." Proceedings of the working group reports of the 2014 on innovation & technology in Computer science education conference. ACM, 2014.
- [2] M. Saelli et al., "Teaching programming in Secondary school: A pedagogical content knowledge perspective." *Informatics in Education* 10.1: pp 73-88, 2011
- [3] Regeringskansliet (Government Offices of Sweden)., "Stärkt digital kompetens i läroplaner och kursplaner" / "Strengthened digital competence in curricula and syllabi", 2018 (Retrieved 24/09/2018) from <http://www.regeringen.se/pressmeddelanden/2017/03/starkt-digital-kompetens-i-laroplaner-och-kursplaner/>
- [4] K. Lawanto et al., "Exploring Strengths and Weaknesses in Middle School Students' Computational Thinking in Scratch." *Emerging Research, Practice, and Policy on Computational Thinking*. Springer, Cham, pp 307-326. 2017.
- [5] T. Bell et al., "Computer science unplugged: School students doing real computing without computers." *The New Zealand Journal of Applied Computing and Information Technology* 13.1 pp 20-29, 2009.
- [6] C. Selby, and J. Woollard., "Refining an understanding of computational thinking." pp 1-23. 2014
- [7] J. M. Wing., "Computational thinking and thinking about computing." *Philosophical transactions of the royal society of London A: mathematical, physical and engineering sciences* 366.1881, pp 3717-3725, 2008.
- [8] H. H. Faber et al., "Teaching computational thinking to primary school students via unplugged programming lessons." *Journal of the European Teacher Education Network* 12. pp 13-24, 2017.
- [9] J. M. Park, C. G. Looney and H. C. Chen., "Fast connected component labeling algorithm using a divide and conquer technique." *Computers and Their Applications* 4, pp 4-7. 2000.
- [10] P. Johannesson and E. Perjons., *An introduction to design science*. Springer.2014
- [11] C. J. Solomon and S. Papert. "A case study of a young child doing Turtle Graphics in LOGO." *Proceedings of the June 7-10, 1976, national computer conference and exposition*. ACM, 1976.
- [12] S. Papert and I. Harel., "Situating constructionism." *Constructionism* 36.2, pp 1-11,1991.
- [13] W. Hung, D. H. Jonassen, and R. Liu. "Problem-based learning." *Handbook of research on educational communications and technology* 3, pp 485-506, 2008.
- [14] S. Tobias, J. D. Fletcher, and A. P. Wind. *Game-based learning*. Handbook of research on educational communications and technology. Springer, New York, NY, 2014.
- [15] C. Larman and V. R. Basili. "Iterative and incremental developments. a brief history." *Computer* 36.6, pp 47-56, 2003.
- [16] M. Villamor and M. M. Rodrigo. "Predicting Successful Collaboration in a Pair Programming Eye Tracking Experiment." Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization. ACM, 2018.
- [17] C. Pearce, Y. Bai, and C. Worsfold. "Problem Solving through Successive Decomposition."
- [18] J.M. Keller., "Development and use of the ARCS model of instructional design." *Journal of instructional development* 10.3, 1987.
- [19] J.M. Keller., *Motivational design for learning and performance: The ARCS model approach*. Springer Science & Business Media, 2009.