Informatics Concepts For Primary Education: Preparing Children For Computational Thinking

Barbara Sabitzer
Alpen-Adria-Universität Klagenfurt
Universitätsstraße 65-67
9020 Klagenfurt, Austria
+43(463)27003517
barbara.sabitzer@aau.at

Peter K. Antonitsch
Alpen-Adria-Universität Klagenfurt
Universitätsstraße 65-67
9020 Klagenfurt, Austria
+43(463)27003519
peter.antonitsch@aau.at

Stefan Pasterk
Alpen-Adria-Universität Klagenfurt
Universitätsstraße 65-67
9020 Klagenfurt, Austria
+43(463)27003517
stefan.pasterk@aau.at

ABSTRACT

In Austria Informatics is not taught in primary schools, but the curriculum includes some issues in several subjects that are related to computational thinking. Teachers are not aware that they already teach and use informatics concepts in their daily lessons. Informatics didactics experts and teacher trainers have to inform them and reveal connections between their primary school contents and different informatics concepts. Furthermore, one general educational aim of the curriculum is the acquisition of elementary cultural techniques including a child-friendly approach to modern information and communication technologies (ICT). The aim of this paper is to show how this is possible in practice and lists informatics concepts already "hidden" in the primary school curriculum. It reports on different initiatives that aim at introducing informatics in primary schools as well as a sample project on computational thinking funded by the regional teacher support program *Teaching Informatics creatively*. The qualitative results of this and other primary school projects show that it is possible and worth integrating informatics already at this early stage.

Categories and Subject Descriptors

K.3.2 **[Computers and Education]:** Computer and Information – Science Education – *computer science education*

General Terms

Algorithms, Human Factors, Standardization

Keywords

Computational thinking, informatics, primary education

1. INTRODUCTION

Over several years a discussion arose if informatics should be introduced in primary schools or not. With the publication of the Report of the joint Informatics Europe & ACM Europe Working Group on informatics education [1] in the year 2013 strong arguments for teaching informatics in primary schools were presented.

WiPSCE'14, November 05 - 07 2014, Berlin, Germany
Copyright 2014 ACM 978-1-4503-3250-7/14/11…$15.00
http://dx.doi.org/10.1145/2670757.2670778

In the conclusion of this report it is pointed out that "[…] European nations are harming their primary and secondary school students, both educationally and economically, by failing to offer them an education in the fundamentals of informatics." [1, p 17]. The report further clearly distinguishes between digital literacy and informatics, the former covering fluency with computer tools and the Internet, while informatics covers the science behind information technology [1, p 3]. Both of these should be considered in primary and secondary education. Above all, computational thinking can be considered a promising concept to introduce informatics at that age. In short, computational thinking represents a set of problem solving techniques and intellectual practices that aim at including digital devices so support the process of problem solving [1, 2]. Wing further postulates, that "[…] computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability" [2, p 1]. A positive effect of an early access to the content of informatics could lead to the achievement of important aims like increasing the interest for technology, preparing children for computational thinking and fostering problem-solving strategies in general. Interest for technics and new developments definitely exists in primary schools and should be promoted as well. Especially the girls' interest for technical subjects like Informatics should be stimulated at that age, as their interest diminishes during puberty and - maybe therefore - they were often neglected in respect of technics so far. Various initiatives like *CS Unplugged* [3] and *Experiencing Informatics* (Informatik erLeben) [4] show that introducing informatics concepts in primary schools is possible and successful, when teaching and learning can be done in an age-appropriate way. Studies on the feasibility of several informatics concepts related to the cognitive development of children are summarized in [5]. They confirm, that many fundamental ideas of informatics e.g. divide-and-conquer, the greedy-approach or even recursion can be taught and understood in primary school.

2. INFORMATICS IN AUSTRIAN PRIMARY SCHOOLS

2.1 The Curriculum

The Austrian curriculum for primary education [6] is divided into three parts: the pre-school (age 5-6 years), the primary school stage I (grades 1 and 2, age 6-8 years) and stage II (grades 3 and 4, age 8-10 years). It does not contain Informatics as a subject, but the learning fields technology and technical work, where children should acquire digital literacy and learn the reasonable handling of new technologies. Informatics concepts are not explicitly indicated, but embedded respectively “hidden” in various subjects, as an analysis of the curriculum revealed. Primary school teachers practice informatics concepts regularly without knowing it, e.g. *algorithmization* (picture stories, step-by-step-instructions, orientation games, describing way to school), *logical operations* (true/false tasks), *data representation* (presenting information in text, pictures, tables etc.), *modeling* (working with models, constructing models), *encryption* (secret language), *coding* (traffic lights and symbols, finding symbols), *hierarchization/categorization* (finding generic terms, recognizing and describing relations), *divide and conquer* (sorting different kinds of objects). A more detailed list and the corresponding tasks are being elaborated in the project *Informatics – A Child’s Play*, containing initiatives for children and teachers described in the next section.

2.2 The Project Informatics – A Child’s Play

Partly based on “CS Unplugged” [3] and “Informatik erLeben” [4] a new project was started at the Department of Informatics Didactics of our university: Informatics – A Child’s Play. It includes different initiatives for children and teachers aiming at increasing interest for technology, showing core concepts of informatics in a playful way and introducing it in primary schools for a long-term. The project started in 2013 and is divided in three phases:

1. development of teaching units and materials (2013-14)
2. implementation in different partner schools (2014-16)
3. completion phase with a final evaluation and the publication of the developed materials (2016) [7].

The main aims of the project are raising and fostering interest for technology and computer science, particularly in girls; Integrating core concepts of computer science in the long-term in primary education; Laying the basis for problem solving strategies, in particular for computational thinking; Fostering activity and creativity of the children; and Improving understanding of complex contents by considering neurodidactical principles [7].

2.2.1 Initiatives For Children

As a part of the pilot phase we invited children, adolescents and adults to join our Informatics lab at the Department of Informatics Didactics (four weeks in July 2014). The visitors get to know the core concepts of informatics in several stations or in different workshops. Furthermore, they are actively involved (if they want) and create individual materials, e.g. websites, games, quizzes etc., or contribute to the adaptation of existing and development of new teaching units and materials. Since 2013 some of the teaching units are already offered in a series of workshops, in part independent from the research project. The units are developed and carried out of the Regional Informatics Didactics Centre of Carinthia, a Regional Educational Competence Center, in cooperation with the hands-on museum “wissens.wert.welt – blue cube & kidsmobil”. These mobile workshops for primary schools can be booked by the teachers and are held in their familiar classroom environment.

2.2.2 Initiatives For Teachers: Training & Projects

Apart from teaching informatics directly to the children, it is necessary to bring it to the teachers, too. Some basics should be part of teacher training; therefore we (the Regional Informatics Didactics Centre) developed a one-year course as additional training for primary school teacher education and in-service training, carried out at the University College of Teacher Training Carinthia. In the first year (2012/13) twelve students attended the course, but only six finished it with success. The high drop-out rate was mainly due to organizational problems and the high workload for teachers besides their lessons at school. The evaluation results taken from open interviews and the official course feedback revealed that they were mainly satisfied with the contents and consider them reasonable and practicable. They had no problems in understanding the presented informatics concepts, but all teachers, who followed the course besides their full teaching assignment at school, complained the high presence phases (six weekends).

Another initiative for promoting informatics in primary schools is the teacher support program of *Informatik kreativ unterrichten* (Teaching Informatics creatively), a regional part of the Austrian support system IMST - Innovations Make Schools Top. In the last years – from 2010 until 2014 totally 52 projects were funded and supervised. Eleven of these projects were conducted in primary schools and further three were carried out as cooperation between each a primary and a secondary school. These are 27% of all projects, which is a good rate for primary schools, where Informatics doesn’t exist as a subject. One sample project on computational thinking as a cooperation between a primary school and a vocational high school is described in the following section.

3. Sample Project Computational Thinking

3.1 Project Description

The IMST project *Aspects of Computational Thinking in Primary Education* [8] has been chosen as sample project because it considered important aspects of how to introduce basic Informatics into primary education. Designed as a pilot study to find out about peculiarities of primary education, the project focused on algorithmization as one major aspect of solving problems by means of computing devices. Furthermore, this project considered prerequisites of teachers, the selection of learning topics and the tradition of classroom organization.. These issues have to be considered, especially when introduction of Informatics content into primary education (still) relies on individual initiatives.

3.1.1 Preliminaries

Being inspired by suggestions within the CSTA K-12 standards paper [8], and by Scratch-based case studies published at [9] the project aimed at introducing 3rd grade primary school children to translate a given problem into a semi-formalized representation that can be animated by means of BYOB/Snap!. This Scratch-based software was preferred to Scratch as it offers the additional functionality to create subprograms. In advance of the project, fourteen teachers of the selected primary project-school attended a half-day teacher training as an introduction into the software to be used and some didactic concepts for integrating it into everyday primary school learning situations. Although this has to be considered an extra-lightweight version of a one year teacher training course like the one mentioned above, it also highlights the limits of such an initiative: Only one of the teachers volunteered for the following year’s project.

3.1.2 Learning Topics

Regarding the current Austrian curriculum, primary education still has to focus mainly on the competences of writing, doing basic calculations and reading [6]. The latter is of particular importance because many problems at higher school level originate in insufficient text comprehension. Various resources like the CSTA Computational Thinking Teacher Resources [10] suggest that algorithmization can be integrated into these traditional topics smoothly, e.g., by animating stories that have been planned with hand-drawn storyboards augmented by a textual description of the plot. To do so, appropriate topics were identified (see 2.1 The Curriculum).

3.1.3 Didactic Framework

The pilot study partly followed the didactic principle of ongoing schematization known from mathematics didactics for primary schools [11]. This principle states that basics of mathematical operations should be learned within environments of sufficient complexity, providing demanding situations that motivate the learners to solve problems, for instance, by making repeated use of simple known operations like iterative addition. During the learning process, the sequence of simple operations has to be substituted by the corresponding higher operation, say, multiplication. Transferring this principle to the field of algorithmization we introduce the principle of ongoing formalization. There, the learning process to formulate algorithms has to be understood as a sequence of the following (overlapping) learning steps:

1. Understanding and creating ordered sequences (of instructions), given in everyday language.

2. Understanding an creating sequences of instructions making use of a (semi-) formal language (like graphic instruction blocks in programming environments designed for children).

3. Automating sequences of formalized instructions.

3.1.4 Course Of Action

The project was carried out with a single class of 8 boys and 14 girls, spanned the period from September 2013 until May 2014,and was divided into four phases of ongoing formalization:

The first phase (September until mid of November 2013) was dedicated to understanding, following and creating (rather) short sequences of instructions using everyday language. All learning tasks of this phase were taken from the pool of well-tried primary education learning material, hence connecting to traditional learning situations in primary education. The tasks included instructions to conduct simple science experiments [8], written and/or drawn handicraft instructions, or instructions how to build a flipbook. Consequently, one flipbook (which can be seen as a sequence of ordered drawings telling a story or describing some sort of movement) had to be manufactured by each of the learners. Further tasks to create ordered sequences included inventing a short story by one's own and telling it by means of a storyboard, i.e. a series of pictures combined with text. Flipbooks and storyboards linked the first phase to the second, where the presentation of ordered sequences (of instructions) became more and more formalized, regarding two areas:

Firstly, it was decided to use the software Blinkenpaint to formalize and animate (virtual) flipbooks. This software displays a house with 8x18-windows within a rectangular area. By switching a window color from dark (blue) to light (yellow), the programmer can create patterns and the impression of changing patterns by combining them movie-like within a sequential order. Simply saying, this software reduces the drawing of a flipchart to coloring the fields of an 8x18-matrix or table. Therefore, to prepare the process of planning before implementation, the learners where handed sheets with 8x18 tables to draw Blinkenpaint-like flipbooks, which, later on, were to be animated by use of the software. This part of the second phase also included exercises about navigating the directory tree when opening or saving the programming projects and lasted from the end of October until the end of November.

Secondly, and in parallel, the learners were trained to understand and create sequences of instructions using BYOB/Snap! command blocks. To connect with common primary school tasks, these exercises were preceded by exercises on finding and telling the way in an imaginary city (see [12] for corresponding worksheets; similar tasks can be found in [3], [4]). Moreover, the BYOB/ Snap! exercises were done without using the software itself. On one hand this was achieved by using task sheets displaying screenshots of BYOB/Snap! programs and corresponding tasks that guided the learners to read these programs. On the other hand there were activities to assemble a sequence of instructions to solve a given "problem" by means of printed, laminated and cut BYOB/Snap! command blocks. Thus, this second part of the second phase did not make use of the computer and lasted from mid November until the beginning of January.

During the third phase (January, February) the learners were introduced to the software BYOB/Snap! Due to a shortage of computers at primary school the children visited a secondary higher vocational school twice so that all of them were able to learn in parallel how to operate the software. In between these learning phases the primary school children did occasional exercises with Scratch Activity cards and practiced to create BYOB/Snap! programs as part of free-choice learning phases. These exercises were done in primary school, where only four laptops were available to the learners.

Having learned the principles of file organization and how to use the BYOB/Snap! programming-environment (which, besides writing simple programs, includes the choice and/or creation of appropriate sceneries and characters as well) the learners were expected to animate their own stories throughout the last project phase. Again, this was one possible activity within their weekly free choice learning phases. Hence, informatics content once again could be integrated seamlessly into the stream of common primary education learning tasks, making the computer just another tool for working in a creative way. Moreover, with only four computers available, the learners had to plan their computer activities first instead of relying on the strategy of trial and error alone. Nevertheless, review of intermediate results showed that the learners' programming skills where less developed than expected. Therefore, this last phase included two instruction units (early April, mid May) where each of the learners was given about one hour time to review and/or to deepen programming basics by altering and/or extending elaborated BYOB/Snap! projects.

3.2 First Results

The pilot study aimed at identifying existing primary school learning tasks computational thinking could connect to, and at adding new tasks and activities to introduce algorithmization into traditional primary-school learning environments. As the first project goal has been documented in sections 2.1 and 3.1, we now concentrate on the learners' progress concerning algorithmization.

3.2.1 Algorithmization At Primary Level

Evaluation of the learners' progress was based on teacher's observations, the learners' response to short test tasks and one start-up questionnaire to find out about the learners' general prior knowledge concerning computers and informatics. In summary, the children had some basic knowledge about how to operate a computer and how to use some standard software, mainly web browsers to connect to friends or to play some online games. All of them had no clear idea what the term "informatics" might stand for.

At the end of the first project phase the primary school children had to work on a single and simple test task to check their ability to find and describe the correct order of action: They were handed a series of five out-of-order-pictures showing the development from egg to chick and given the task whether to cut the pictures and rearrange them in correct order (sequencing cards task), to write the correct order into a prepared table, or to write a whole story about a chicks hatching. All of the children completed the task in one way or the other, but none of them chose the option to fill the table. The hypothesis is, that children of that educational age are not yet used to formalized forms of data representation. Therefore additional exercises concerning tables or tree structures were added to subsequent task sheets (see section 3.1).

The BYOB/Snap! based learning units were framed by a pre- and a posttest on formal representation of step-by-step instructions. The pretest had some exercises utilizing everyday language, some exercises with a predefined set of BYOB/Snap!-style commands in everyday language (like: "make 5 steps" or "turn right") and some exercises based on BYOB/Snap! command blocks. While most of the learners achieved good results with everyday-language-tasks, none of them could make sense of the small programs using BYOB/Snap! command blocks. As all the tasks were of similar complexity, we concluded that the acquaintance with a (formal) representation system is more important than we had assumed. This was proven right by the posttest, one part of which consisted of short BYOB/Snap!-programs only, none of the learners had problems to understand.

Furthermore, the posttest included exercises to store a file at or retrieve a file from a specified subdirectory. Only three of the primary school children were able to solve at least one of these exercises correctly. The hypothesis here is: In spite of several corresponding learning tasks concerning directory structure, most of the learners were not able to create a mental representation of the formal structure "file tree". This fits to observations during the phases when the learners were working with the computers: Most of the problems occurred when storing or retrieving files.

3.2.2 Didactic Framework Revisited

The outcome of pre- and posttest mentioned above indicate that the suggested principle of ongoing formalization works well with algorithmization at basic level. But, as reported in section 3.1.4, most of the learners tend to remain at that basic level when learning the process of algorithmization that way. We suppose, that mastering the various levels of complexity inside any formal (programming) language cannot be achieved by the principle of ongoing formalization alone but, needs the principle of ongoing complexity, too [11]: Simple tasks concerning a certain aspect of coding have to be succeeded by more complex tasks concerning the same aspect, just like in seemingly old-fashioned programming courses. Consequently, a didactic framework for introducing Computational Thinking into primary education must extend the principle of ongoing formalization, even more as it has to consider order structures like directory trees (see. 3.2.1) as well.

Dealing with (hierarchical) order structures might be integrated into primary education learning settings quite seamlessly easily, just thinking about organizing the learners' works in a portfolio-like style with directories for each subject matter und subdirectories for certain topics. Differently, the principle of ongoing complexity might need some further adaptions of the traditional learning setting as it demands for a series of subsequent coding units that might interrupt primary education learning business as usual.

4. CONCLUSION AND OUTLOOK

Introducing informatics in primary schools is possible and reasonable when it is taught in a child-appropriate way. Although not anchored in the Austrian curriculum, many activities related to different informatics concepts are performed in the daily school life. Fostering these activities and relating them to informatics may be a way to prepare the children for computational thinking in its broadest sense. From the viewpoint of informatics didactics this calls for the development of a corresponding didactic concept to extend traditional primary education so as to include these new aspects but preserves well tried teaching and learning practice concerning competences in basic literacy. The pilot project we reported about in detail will be prolonged for one further year and will continue working in this direction.

5. REFERENCES

[1] Joint Informatics Europe & ACM Europe Working Group on Informatics Education. *Informatics education: Europe cannot afford to miss the boat.* 2013. http://europe.acm.org/iereport/ACMandIEreport.pdf

[2] J.M. Wing. Computational Thinking. In *Communications of the ACM*, March 2006/Vol 49 No. 3. 2006. http://www.cs.cmu.edu/~15110-s13/Wing06-ct.pdf

[3] T. Bell, I.H. Witten and M. Fellows. *Computer Science Unplugged. An enrichment and extension programme for primary-aged children.* 2010. http://csunplugged.org/ (2. 7. 2014)

[4] R.T. Mittermeir, E. Bischof and K. Hodnigg. Showing Core-Concepts of Informatics to Kids and Their Teachers. In Hromkovič, J., Královič, R., Vahrenhold, J. (eds.) *Teaching Fundamental Concepts of Informatics ISSEP 2010*, LNCS 5941, pp.143-154. Springer, Heidelberg, 2010.

[5] A. Schwill. Ab wann kann man mit Kindern Informatik machen? *INFOS2001-9. GI-Fachtagung Informatik Und Schule GI-Edition*, 13–30, 2001.

[6] BMBF. LEHRPLAN DER VOLKSSCHULE. BGBl. Nr. 134/1963 in der Fassung BGBl. II Nr. 303/2012; 13. September 2012. In https://www.bmbf.gv.at/.

[7] B. Sabitzer and S. Pasterk. Informatics – A Child's Play. *Proceedings of the 6th International Conference on Education and New Learning Technologies (EDULEARN*), Barcelona, Spain, July 2014.

[8] P. Antonitsch. A Media-Reduced Approach towards Informatics at Primary Level. *Proceedings of the 8th Workshop in Primary and Secondary Computing Education*, WIPSCE, Aarhus, 2013.

[9] Case Studies according to the Northern Ireland Curriculum, Using ICT – Scratch projects, available at: http://www.nicurriculum.org.uk/

[10] Computer Science Teachers Association. Computational Thinking teacher resources, second edition. 2011. Available at: http://csta.acm.org/

[11] M. Glade. Vom Zeichnen zur Rechenregel – Individuelle Prozesse der fortschreitenden Schematisierung zum Anteil vom Anteil. Presentation at the 45th Conference on Mathematics Didactics, Freiburg, 2011. Available at: http://www.mathematik.tu-dortmund.de/ieem/bzmu2011/_BzMU11_2_Einzelbeitraege/BzMU11_GLADE_M_Schematisierung.pdf

[12] E. Hengarter, U. Hirt and B. Wälti. Lernumgebungen für Rechenschwache bis Hochbegabte. Natürliche Differenzierung im Mathematikunterricht, 2. Auflage, Klett und Balmer Verlag AG, Zug. Referenced worksheet available at: http://www.mathe-projekt.ch/index2.htm (in German), 2010.