Computational Thinking in Primary Schools: Theory and Causal Models

Christine Bescherer, Andreas Fest

Computational Thinking in Primary Schools: Theory and Causal Models

Christine Bescherer and Andreas Fest

Ludwigsburg University of Education, Germany,
bescherer@ph-ludwigsburg.de, fest@ph-ludwigsburg.de

Abstract. During a one-year-subproject, student teachers develop and pilot learning scenarios and materials in mathematics classrooms for third and fourth graders fostering 'Computational Thinking'. To evaluate these interventions regarding the impact on the student teachers as well as the schoolchildren 'impact models' are used. These models based on program impact theory will be continuously refined to converge to a 'proof of the success' of the project.

Keywords. Computational Thinking in Primary Schools, Half-baked Microworlds, Evaluation Theory, Impact Models

1. Introduction

The project 'Digital Learning in Primary Schools Stuttgart / Ludwigsburg' ('Digitales Lernen in der Grundschule Stuttgart / Ludwigsburg' dileg-SL) at the Ludwigsburg University of Education is funded by Deutsche Telekom Stiftung. It aims at the development of digital learning scenarios at primary schools. Beneath the productive and critical exposure to digital media in different contexts and subjects like German and English language, biology, music, physical education, another important objective is the development of primary schoolchildren's basic competencies in computer science and algorithmic thinking.

Student teachers learn in a special seminar about 'computational thinking' themselves and then develop learning scenarios to foster algorithmic and computer science related competencies in mathematics classrooms for third or fourth grade. These student teachers will test their scenarios in 3rd and 4th grade classrooms and reflect on their experience.

Projects like this where classroom related, rather complex learning scenarios are developed and implemented always pose the problem how to show they are successful. The number of student teachers and primary schoolchildren are here too small to allow sophisticated quantitative research statistics. In addition, in this case because of the structure of the project an experimental design is not possible either. So how to evaluate the learning scenarios or to describe and even measure the impact? Further, questions like whether there are different impacts for different groups (i.e. girls/boys or low / high computer use) also need to be answered.

The whole project dileg-SL started in 2016 and the special seminar in math teacher education referred to will take place from October 2017 until July 2018 (two

semesters). The schoolchildren will have use of iPads and/or Convertibles supplied by the university.

2. Theoretical Background

The theoretical background covers the teaching and learning of computer science and algorithmic thinking in (primary) schools as well as program impact theory. In this paper the aspects of media pedagogy are not discussed even though they play a major role in the umbrella project.

2.1 Computational Thinking/FITness

'Computational thinking' is a rather inflationary used concept, which is still not well defined. Wing [1] describes 'conceptualization' and 'thinking on multiple levels of abstraction' as important aspects of computational thinking. Grover and Pea [2] give in their review of the state of the field on computational thinking in K-12 the following list as widely accepted elements of computational thinking

- 'Abstractions and pattern generalizations (including models and simulations)
- Systematic processing of information
- Symbol systems and representations
- Algorithmic notions of flow of control
- Structured problem decomposition (modularizing)
- Iterative, recursive, and parallel thinking
- Conditional logic
- Efficiency and performance constraints
- Debugging and systematic error detection' [2, p. 39/40]

Obviously, these elements cover very different levels of knowledge: there are some factual knowledge (i.e. conditional logic, efficiency and performance constraints), some are basic concepts of computer science thinking (i.e. algorithmic notions of flow of control or iterative, recursive, and parallel thinking) as well as typical procedures computer scientists use (like modelling and simulation, modularization, debugging and systematic error detection). Therefore, these elements are not a list to go through step by step to learn computational thinking, rather the factual knowledge and basic concepts should be understood and deepened by working like a typical computer scientist. Figure 1 gives an example for the combination of the basic concept 'recursive thinking' and 'debugging an systematic error detection'. It shows a series of screenshots of a 4th grader working on programming a recursive binary tree using Scratch.

An older definition of what is necessary to be 'FIT' in using IT is the 'Fluency with Information Technology' ('FITness') concept [3] from 1999. There the authors define three kinds of knowledge: contemporary skills, foundational concepts, and intellectual capabilities. Further, they stated the role of programming in becoming 'fit' with information technology. Programming and mathematics learning in primary school have a long tradition [4]. Gadanidis, Hughes, Minniti, & White [5] describe a concept for introducing computational thinking in primary mathematics classrooms where grade 1 pupils experience several aspects of computational thinking.

During the ScratchMath project a ‘framework for action’ to bring together primary programming and mathematics was developed by Benton, Hoyles, Kalas & Noss [6]. It consists of the ‘5Es’ – ‘explore’, ‘explain’. ‘envisage’, ‘exchange’ and ‘bridgE’ which scaffold the design of learning scenarios combining programming and math learning.

2.2 Half-baked microworlds

Writing program code on the empty screen is demotivating or not following learning theories like constructivism. Primary schoolchildren are not supposed to become experts in programming ‘full grown’ computer languages but they should experience how computers work and develop basic concepts. Later in secondary schools, they can build their computer science skills and competencies based on these fundamental exposures.

Therefore, learning scenarios based on educational theories like Seymour Papert´s microworlds [4] qualify for the project. Another suitable concept are different levels of worked examples with either intentional mistakes or cloze which learners have to identify and fill out [5]. Learning with worked examples has been successful for learning mathematical concepts [6] and could similarly work while learning programming. A combination of these learning scenarios are the half-baked microworlds proposed by Chronis Kynigos [7].

Kynigos understands half-baked microworlds ‘designed to facilitate communication between researchers, technicians, teachers and students as they become engaged in changing them’ [7, p. 335].

2.3 Program Impact Theory

‘Program Theory-Driven Evaluation Science is the systematic use of substantive knowledge about the phenomena under investigation and scientific methods to improve, to produce knowledge and feedback about, and to determine the merit, worth, and significance of evaluands such as social, educational, health, community, and organizational programs.’ [11, p. 7] One way of measuring the success of a pedagogical intervention is to set up a ‘casual model’, which gives the statistical correlations between different input- and output-factors. This would implicate that there is a measurable cause with correlation and high effect size, which in small-scale real classroom settings is often just not possible.

To develop, to describe and to communicate the impact(s) of a program or a complex learning scenario impact models [12] are used. Impact models do not describe cause and effect, but they allow closing in on an empirical and well-founded causal relationship.

An impact model fitting to classroom interventions usually consist of theoretical input – activities/learning scenarios – expected or measured outcome(s) (Fig. 1). As the project develops further this initial model will be iteratively refined with every new finding. The series of improved impact models show the impact of the project work thus converging to a conclusive argument. This is not an analytical or empirical proof but it comes close to one.

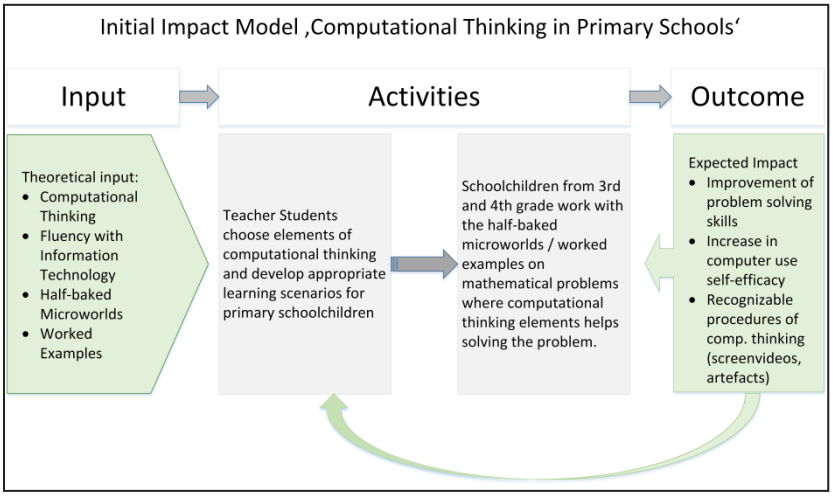

Fig. 1. Initial Impact Model

3. Methods and Implementation

In the project, the student teachers will work in the seminar with half-baked microworlds as well as worked examples to learn about the elements of computational thinking. In the second half of the semester, they will develop analogues scenarios for the schoolchildren in 3rd and 4th grade. These scenarios will be used in real classrooms in a primary school with a special focus on media education close to the university. Afterwards the student teachers will reflect on the teaching experience according to previously discussed hypotheses.

To see whether a rather complex project involving different target groups like student teachers and schoolchildren works at all or to determine for which subgroups it fits better different ways of measuring or research approaches can be used. Since only a small numbers of participants will take part in the project and an experimental design is not possible we decided to work with impact models of the program theory evaluation approach.

One of the measures beside the artefacts the student teachers and the pupils will produce (i.e. Logo or Scratch programs, descriptions of the learning scenarios, worksheets,…) computer use self-efficacy questionnaires will be used to determine whether there are pre-post-differences. Figure N° 1 shows the initial causal model for the subproject described in this paper, which will be refined further during the project.

4. Outlook and Summary

The seminar and the piloting in the primary school will start in October 2017. Until then the half-baked microworlds, the worked examples for the student teachers, the questionnaires and preparations for the analysis of the artefacts and the screenvideos

will be developed. This paper gives an idea about the theoretical input and the planed methods including an initial impact model according to program theory evaluation science.

Acknowledgments. The authors would like to thank the Deutsche Telekom Stiftung (https://www.telekom-stiftung.de/en) for financing the project.

References

1. Wing, J. M.: Computational thinking. Communications of the ACM 49(3), 33-35 (2006)
2. Grover, S., Pea, R.: Computational Thinking in K–12 A Review of the State of the Field. Educational Researcher, 42 (1), 38-43 (2013)
3. National Research Council: Being fluent with information technology. National Academies Press (1999)
4. Benton, L., Hoyles, C., Kalas, I., Noss, R.: Bridging Primary Programming and Mathematics: some findings of design research in England. Digital Experiences in Mathematics Education. 3, 115–138 (2017)
5. Gadanidis, G., Hughes, J. M., Minniti, L., White, B. J.: Computational Thinking, Grade 1 Students and the Binomial Theorem. Digital Experiences in Mathematics Education. 3, 77–96 (2017)
6. Benton, L, Hoyles, C, Kalas, I, Noss, R.: Building mathematical knowledge with programming: insights from the ScratchMaths project. Constructionism in Action 2016: Conference Proceedings. 25-32 (2016)
7. Papert, S.: Teaching Children Thinking. Journal of Structural Language 4, 219-229 (1975)
8. Renkl, A.: Worked-out examples: Instructional explanations support learning by self-explanations. Learning and instruction 12(5), 529-556 (2002)
9. Scherrmann, A.: Learning with worked examples – how does it work in a real classroom setting? Proceedings 10th Congress of European Research in Mathematics Education. (2017) https://keynote.conference-services.net/resources/444/5118/pdf/CERME10_0322.pdf, last retrieved 2017/08/30
10. Kynigos, C.: Half-baked logo microworlds as boundary objects in integrated design. Informatics in Education - An International Journal. 6(2), 335-359 (2007)
11. Donaldson, S. I.: Program theory-driven evaluation science: Strategies and applications. Routledge (2012)
12. Rogers, P. J.: Causal models in program theory evaluation. New directions for evaluation. 87, 47-55 (2000)