Nr 14: Nortvedt, G. A., Buchholtz, N. F., Fauskanger, J., Hreinsdóttir, F., Hähkiöniemi, M., Jessen, B. E., Kurvits, J., Liljekvist, Y., Misfeldt, M., Naalsund, M., Nilsen, H. K., Pálsdóttir, G., Portaankorva-Koivisto, P., Radišić, J. & Wernberg, A. (Eds.) (2021). Bringing Nordic mathematics education into the future. Preceedings of Norma 20. The ninth Nordic Conference on Mathematics Education, Oslo, 2021. NCM & SMDF.

Teachers' arguments for including programming in mathematics education

Cecilia Kilhamn[1], Kajsa Bråting[2] and Lennart Rolandsson[2]

[1]University of Gothenburg, Sweden. cecilia.kilhamn@gu.se

[2]Uppsala University, Sweden; kajsa.brating@edu.uu.se; lennart.rolandsson@edu.uu.se

In recent years, programming has been inserted into mathematics curricula in many countries. This paper reports on interviews with 20 Swedish mathematics teachers who, as early adopters, teach programming within the frames of their ordinary mathematics lessons. Qualitative analyses of data identified four types of arguments for teaching programming in mathematics: to develop computational thinking; to increase engagement; to learn mathematics; or simply because it is a powerful tool. We conclude with some implications of these different arguments.

Keywords: Mathematics, computational thinking, programming.

During the last decade, programming has been given a place in school curricula in many countries. Some countries allocate time for a specific subject (e.g. England with Computing) while others incorporate programming and computational thinking into existing subjects, primarily into mathematics (Mannila et al., 2014). In Sweden, *programming* is to be taught in mathematics and applied in technology, while a more general term in the curriculum is *digital competence* (Swedish National Agency of Education, 2018). The present study is part of a recently started research project regarding the ongoing integration of programming in school mathematics (Bråting et al. 2020). The project as a whole is theoretically embedded in Chevallard's (2006) framework about transposition of knowledge, which describes a praxeology of *what is taught* and *why it is taught* and how this changes as knowledge is transposed to different levels of the educational system. In 2018, when programming was added to the Swedish national mathematics syllabus, all mathematics teachers were obliged to teach programming. Official arguments as to why are scarce, except vaguely that it is to help increase students' digital competence. By exploring how teachers talk about introducing programming, our aim in this paper is to better understand the know-why on the teacher level, and to identify challenges in relation to the integration of programming in school mathematics. We ask the question: What arguments do teachers, who are early adopters of programming in primary and secondary school, give for including programming in mathematics lessons?

Computational thinking

The term *computational thinking* (CT) was first introduced by Papert (1980) when he developed Logo programming. Although the terms CT and *programming* are used differently in research literature, it is clear that they are closely connected. Moreno-Léon et al. (2019) make a distinction between CT as a cognitive ability and programming as just one way of developing that ability. The Swedish national curriculum states that "pupils should be given opportunities to develop knowledge in using digital tools and programming to explore problems and mathematical concepts, make calculations and to present and interpret data" (Swedish National Agency of Education, 2018, p. 55). The curriculum prescribes the use of programming using non-digital activities and visual as well as text-based environments, but does not mention thinking skills or cognitive abilities. Hence, how teachers

interpret what programming is, and why they should teach it is embedded in the question of transposition of knowledge.

There are many attempts world-vide to conceptualize computational thinking skills and show how digital technologies could enhance cognition. One such framework by Brennan and Resnick (2012) proposes three dimensions of computational thinking*: computational concepts* (sequences, loops, parallelism, events, conditionals, operators, and data), *computational practices* (being incremental and iterative, testing and debugging, reusing and remixing, abstracting and modularizing), and *computational perspectives* (i.e., expressing, connecting, and questioning). We consider this framework useful, as it broadens the perspective on computational thinking in the era of the 21st century. Brennan and Resnick's framework emerged from their studies of young interactive media designers using Scratch, a common environment in educational settings. We will use the term computational thinking (CT) in line with Brennan and Resnick (2012) throughout the paper.

Research on teacher's views on programming in mathematics

The incorporation of programming in school is a new phenomenon, with roots in early initiatives in the 1980's when Papert (1980) introduced Logo programming as a way to develop mathematical understanding. Little is yet known about what is taught and why, in the revival of programming in schools taking place in the last decade. There are a few studies on teacher's views, mostly based on written surveys, for example by Mannila et al. (2014), who gathered data from 961 teachers in five European countries. The researchers asked explicitly about CT skills and concepts, and concluded that teachers already were involved in activities with potential for introducing aspects of CT. Misfeldt et al. (2019) collected data from 133 Swedish teachers, showing that, although teachers were positive towards working with programming in mathematics, not all could see the relationship between the two, nor the relevance for doing it. From an on-line questionnaire given to Finnish primary school teachers, Pörn et al. (2020) analysed 91 written answers to the questions "What is programming". They found that the teachers primarily emphasized "writing, giving and following instructions", but also mentioned mathematical skills such as logical thinking, problem solving and identifying patterns, as well as using modern technology and preparing their students for future work and studies.

In England, the introduction of computing as a new compulsory school subject in 2014, triggered a large research project about using Scratch in mathematics (Benton et al. 2017). While the English curriculum emphasizes CT skills, the study showed that it is also possible to teach mathematical ideas while teaching CT, highlighting the importance of teachers making explicit links between programming and mathematics.

In a related study to the one presented here, Nouri et al. (2019) conducted interviews with 19 teachers with approximately two years' experience teaching programming in school, although not only in mathematics. They found that the teachers talked about developing skills that corresponded well with Brennan and Resnick's (2012) three dimensions of CT, but identified also four other types of skills that teachers wanted their students to develop; cognitive skills, language skills, creative problem solving skills and collaborative skills. No clear mention is made in the article of developing mathematical understanding through programming.

Method

Data for this study was collected through semi-structured interviews with 20 teachers, classified as "early adopters" (EA) because they were enthusiastic about the implementation of programming in school and already had experience in teaching it. They were recruited through different teacher networks and identified themselves as early adopters. All the EA's teach mathematics, 15% in grades F-3, 30% in grades 4-6 and 55% in grades 6-9. They have 6–35 years of teaching experience, 75% of them also teach technology and 55% have an extended responsibility to implement digital tools and programming in their schools. Their programming competence is diverse; three have an engineer exam, some have participated in one or two programming courses for teachers, but 40% are self-taught with no programming credentials at all. They work in 14 different municipalities well spread around the country and in schools of various sizes.

The interviews were audio recorded and transcribed verbatim. Two of the authors conducted the interviews, initially doing three together and the rest separately. The interviews took approximately 30 minutes and were structured around eight questions that had been supplied in advance with the intention of capturing different aspects of teachers' talk about programming in mathematics. Following four background questions the interview guide included the following topics: 5) What is the role of programming in mathematics? 6) Where do you find inspiration and ideas? 7) Can you give an example of a good programming activity that you have tried? 8) What programming concepts are important to bring up in mathematics? As data is rich, we will focus in this paper on the explicit connections teachers made between programming and mathematics throughout the interview.

A data-driven analysis of the transcripts was made using NVivo software, identifying and sorting quotes into categories illustrating different types of arguments. Categories were identified through a thematic analysis (Braun & Clarke, 2006) in an inductive bottom-up process. In an iterative process the categories were identified by the first author, then revised several times when discussed, compared and validated within the research team. Once identified, the categories were related to the CT framework of Brennan and Resnick (2012). In each category we tried to capture a unique aspect of what the teachers brought up as reasons to work with, or benefits of, programming. Although distinctly different, the categories are sometimes intertwined. While focus is on the phenomenon, not the individual teacher, each interview could generate several arguments.

Results

We found four main categories of arguments, with some sub-categories. Each category is described in short and instantiated below in a few typical quotes for each category.

Category 1. Programming is a potentially powerful tool

Teachers see programming as a pedagogical tool in addition to other tools that are potentially useful in mathematics once you master them sufficiently. They think that teaching programming will change with time, since they and their students initially need to spend time learning how to use the tool.

> You need to think of it as a pedagogical tool, just like some years ago when we were making films. See it as a pedagogical tool, that I could use even if I teach Geography, or History. […] I can use it as a working tool for the children, so that somewhere down the line they need to learn how it works. (EA12)

> To use programming as a tool for doing mathematics, like GeoGebra, I think that is great. […] All tools that support learning are good, and I think programming is one of them. (EA18)

> Well, on this level you don't really need programming to solve the problems. I think it is more that you see that it is possible to use programming to solve problems […] for the students, maybe paper and pencil is faster than using a computer […] when you get to more complex problems programming could be useful, so then you need to learn it on these easier problems. (EA20)

Category 2. Programming increases engagement

Teachers say that students find programming interesting, that they engage better with programming tasks than with traditional mathematics tasks. Students become more engaged because it is interesting and fun (2.1) or connects to reality (2.2). They may not learn mathematics through programming, but implicitly the teachers believe increased engagement will enhance mathematical learning.

Category 2.1 Programming is interesting and fun

> Some students who become interested in programming, they learn a lot of other things too. […] it definitely increases motivation. […] And to develop that interest they need to learn the mathematics, to be able to move on. (EA01)

> Well, content wise, at this level I don't think it contributes much but it does add a new dimension, the students get more inspired, more motivated to do the work. (EA17)

> Fun, demanding, challenging and inspiring. (EA19)

Category 2.2 Programming connects to reality

> That's what I find so fascinating, that it is connected to something they have done before, or seen as useful. Well, maybe not always useful, but they have seen it in reality in some way. (EA13)

> The students get inspiration, get really interesting discussion going, for example during my maths lesson, about future occupations. I haven't very often talked about that before […] now they have a feeling that: this is programming, and that's a job I could have. (EA17)

Category 3. Programming develops computational thinking

Teachers describe programming as a new way of thinking and working, which is different from traditional mathematics but relevant for mathematics education. More or less explicitly, they refer to the development of computational thinking practices as described by Brennan and Resnick (2012), where being incremental (3.1) is pointed out as a generic practice, not restricted to mathematics.

Category 3.1 Programming teaches students to break down instructions and problems into small sequential steps. This category connects to the practice of being incremental.

> It is a new way of thinking for the children. They have to get into thinking in small steps and giving instructions. Even if we work a lot with instructions in Swedish language class, writing and giving instructions, here is another connection because they need to be so precise. (EA12)

> Well it is a logical thinking process, I think. We work a lot with all these concepts, like loop, algorithms. How to create algorithms, that all goes with programming, but also with Swedish language class, like cutting a story into small parts and then putting them all in the right order.

That's how you get computational thinking. That's how I work with the six steps of computational thinking. (EA15)

To me, an algorithm is like a cooking recipe, it is a step-by-step instruction. That is the thinking behind algorithms, I think, and that is a very mathematical way of thinking. (EA16)

Category 3.2 Programming encourages testing, debugging and modifying. Teachers highlight that students learn new practices from programming that are useful when doing mathematics, such as testing, failing, and working iteratively, as well as the importance of being persistent and meticulous.

They think it is easy in programming, looking for mistakes and trying again and again and again, many, many times. I would like them to take that with them when they work with other types of mathematical problems. We talk a lot about that. Because sometimes they have the patience, sometimes they don't. […] It is absolutely a feature of computational thinking, to try and fail, and then try again, make small changes … (EA17)

For once they are in a position where it is acceptable to fail, you need to try, […] In programming some things always go wrong. It is a natural part of the process to "hit the wall", to search for mistakes and that sort of thing. [EA18]

I want my students to understand code, to be able to read code, to modify code. It is the same with GeoGebra, it is all about modifying something. You don't really need to know what happens in the background, but you need to be able to modify it to fit the problem you have right now. That's what you need to know. In Excel too. Here is a context where students really find themselves in a position where it is extremely important to be meticulous, where the tiniest mistake can make it all go wrong. (EA18)

Category 4. Programming is a way of learning mathematics

Teachers want students to learn mathematics through programming. It can be a tool for learning some specific mathematical content (4.1) or it can be considered as mathematics in itself (4.2).

Category 4.1 Programming is a tool to learn mathematics. In this category programming is described as a tool for learning some other mathematical content rather than being fundamentally mathematical in itself. However, the teachers often describe it as an ideal, a difficult goal and not yet reached.

The programming helped [students understand] the concept of variable. (EA11)

In mathematics it is more a tool to reach mathematical understanding. Which makes it more difficult to choose good examples in mathematics. (EA01)

Well, mathematics and programming don't really go together. You can be a good programmer and lousy at maths, or the opposite. But you can use programming as a tool to enforce mathematical concepts, of that I am certain. (EA18)

It [Desmos] is one of the best I think we can find right now for mathematics and programming. Because it focusses on programming to make the mathematics visible. (EA05)

By using Excel, Ipads and GeoGebra, the students could investigate relationships between area and perimeter. (EA03)

Category 4.2 Programming is a genuine mathematical activity. One of the teachers talk about programming as a mathematical activity in itself.

> I think teaching has been too instrumental, where you never get to the heart of mathematics, like what is the beauty of mathematics, what it is that is interesting in mathematics. I think it is puzzles and patterns and mysteries. And like how we can figure this out and how this relates to that. And I think programming makes it easier to get to that heart. That's how I think about it. (EA09)

Discussion

Although we did not explicitly ask the teachers to define programming, a lot of their arguments implied a view quite similar to what was found among Finnish teachers (Pörn et al., 2020), as primarily to learn about *giving and taking instructions expressed in small sequential steps* (see Category 3.1). This is a generic skill, not solely mathematical, although fundamental in mathematics. But should not programming in mathematics teach mathematics? The four types of arguments that emerged from our data highlight an important dichotomy in relation to mathematics.

On the one hand Categories 1 and 2 describe programming as *useful and engaging on a general level*, not necessarily connected to mathematics, in line with the teachers in Nouri et al.'s (2019) study who pointed mainly to student's development of general skills, not to programming as a mathematical activity. Being early adopters, the teachers in this study are positive to programming activities, but like teachers in the study reported by Misfeldt et al. (2019), they do not always see the clear connection to mathematics. In fact, some actually think it is easier to include programming in other subjects, which we see also in category 3.1. This could be a result of the fact that visual environments, such as Scratch, are more closely adapted to storytelling activities and animation than to mathematical activities (see also Bråting et al., 2020). Consequently, promoting visual environments while at the same time placing programming in the subject of mathematics is a challenge. This was also shown by Benton et al. (2017) who emphasized the importance of designing Scratch activities with specific mathematical goals, and the teacher's role in making mathematical connections explicit. Using Scratch is in itself not necessarily a mathematical activity.

On the other hand, arguments in Categories 3 and 4 proclaim that programming could help students learn about mathematical ideas and develop practices that are fruitful for mathematical work. We see some teachers emphasizing the development of computational thinking, actually highlighting that CT skills are valuable for mathematics and may be instrumental in changing students' way of working and thinking in mathematics in a good way. By introducing practices like debugging and modifying, suggested in the Brennan and Resnick (2012) framework of CT, and by encouraging students to see failure as a natural part of a problem solving process, the teachers believe that mathematics learning will benefit from programming activities. It is interesting that only one quote from all the 20 interviews describe programming as a mathematical activity in itself (Category 4.2). We see here a challenge for mathematics education researchers and developers: could we come up with programming activities that are more mathematical in nature?

Considering the arguments in Category 1, we conclude that some teachers trust new technology, embracing arguments about the usefulness of programming from other levels of the educational system, but without actually experiencing its usefulness themselves. They say that at the moment we need to spend time learning the tools, referring to themselves as well as their students. Presently the

whole school system is faced with a big challenge when teachers are asked to teach something they do not yet master. Perhaps really seeing how programming can promote mathematics will only be possible when programming had become a tangible feature of everyday life. Some teachers feel that they need to get to that stage first. However, if the main purpose is to learn to use a new tool, it is questionable if the time spent on programming is wasted from a mathematics education point of view. When computers were introduced as new tools in school, learning to use keyboards, search engines and word processors was not included in the mathematics syllabus.

In accordance with the theory of transposition of knowledge described by Chevallard (2006), we can see that the know-why of programming in mathematics has changes as it moved from the curriculum level to the teacher level. While the curriculum mentions programming to explore problems and mathematical concepts, the early adopters tend to focus more on the tool itself and talk about the benefits of programming in general terms. In schools there are many teachers who know much less about programming than these early adopters and who may well have even more general arguments for, or even against, introducing programming in mathematics. As another part of this research project we are currently collecting data from such teachers as well. Although our early adopters express ideas about programming as a powerful tool for engagement and development of thinking skills, it does not necessarily imply learning in mathematics. To benefit from greater student engagement, teachers need to be able to discern powerful mathematical ideas in programming, as well as computational aspects of mathematics, which will require good skills in both programming and mathematics.

In our analysis we easily found the first two dimensions described in the framework by Brennan and Resnick's (2012); computational concepts and computational practices. Working within the digital era with the 21st century skills (including programming) could initiate a transformation of mathematics education to embrace errors as resources for scrutiny of taken for granted mathematical concepts. Such a transformation would most likely change teachers' epistemological beliefs about mathematics into an experimental and practical curriculum, necessary if computational practices and perspectives are to flourish in school mathematics. But this is demanding, and meanwhile, programming takes time from more traditional mathematics learning. Drawing on the early adopters' arguments, we conclude that programming could, but may not, enhance mathematics, depending on whether or not the teachers look for opportunities where it could, and are open to changes in mathematical thinking practices. This is a great challenge for teachers who are not early adopters.

Acknowledgment

This work was supported by the Swedish Research Council [Grant no. 2018-03865].

References

Benton, L., Hoyles, C., Kalas, I., & Noss, R. (2017). Bridging primary programming and mathematics: Some findings of design research in England. *Digital Experiences in Mathematics Education, 3*(2), 115–138.

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative research in psychology*, *3*(2), 77–101.

Brennan, K., & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking*. Paper presented at the 2012 annual meeting of the American Educational Research Association, Canada.

Bråting, K., Kilhamn, C., & Rolandsson, L. (2020). *Integrating programming in Swedish school mathematics: description of a research project*. Paper presented at the twelfth research seminar of the Swedish Society for Research in Mathematics Education, MADIF12, Linnaeus University, Växjö, January 14–15, 2020.

Chevallard, Y. (2006). Steps towards a new epistemology in mathematics education. In M. Bosch (Ed.), *Proceedings of the Fourth Congress of the European Society for Research in Mathematics Education, CERME 4* (pp. 21–30). Barcelona: FUNDEMI IQS-Universitat Ramon Llull.

Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., & Settle, A. (2014). Computational thinking in K-9 education. In A. Clear & R. Lister (Eds.), *Proceedings of Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference* (pp. 1–29). ACM.

Misfeldt, M., Szabo, A., & Helenius, O. (2019). Surveying teachers' conception of programming as a mathematical topic following the implementation of a new mathematics curriculum. In U. Jankvist, M. Van den Heuvel-Panhuizen, & M. Veldhuis, M. (Eds.). *Proceedings of CERME11* (pp. 2713–2720). Utrecht: Freudenthal Institute, Utrecht University and ERME.

Moreno-León, J., Robles, G., Roman-González, M. & Rodrígues, J.D. (2019). Not the same: a text network analysis on computational thinking definitions to study its relationship with computer programming. *Revista Interuniversitaria de Investigación en Tecnología Educativa, 7,* 26–35.

Nouri, J., Zhang, L., Mannila, L., & Norén, E. (2019). Development of computational thinking, digital competence and 21st century skills when learning programming in K-9, *Education Inquiry*, *11*(1), 1–17.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas.* Basic Books.

Pörn, R., Hemmi, K., & Kallio-Kujala, P. (2020). *"Programming is a new way of thinking" – teacher views on programming as a part of the new mathematics curriculum in Finland*. Paper presented at the twelfth research seminar of the Swedish Society for Research in Mathematics Education, MADIF12, Linnaeus University, Växjö, January 14–15, 2020.

Swedish National Agency of Education (2018). *Curriculum for the compulsory school, preschool class and school-age educare 2011*. Elanders Sverige AB.