Presented at Madif–12, Växjö, January 15, 2020

Positioning of programming in mathematics classrooms – a literature review of evidence based didactical configurations

Andreas Eckert & Alexandra Hjelte
Örebro University and Gumaelius-school

This literature review looks into the roles that programming and mathematics can take in relation to each other in an educational environment. This is done by retrieving papers from Web of Science and MathEduc on mathematics and programming in education, prior to university level, and analysing their tasks through a lens of Instrumental Orchestration. The four different categories of tasks identified are manipulating a physical entity, manipulating a virtual entity, creating own interactive environment and creating, testing and refining mathematical algorithms. Depending on how mathematics and programming are positioned in the four categories, the emphasis on mathematics and programming varies, resulting in different outcomes of the lessons.

Feurzeig and Papert (1969) envisioned a future where students of all ages learn key concepts and procedures of mathematics through their newly developed programming language Logo. Now that vision seems to partly be coming true as an increasing number of countries add coding as a part of their national curriculum for elementary school to improve students' digital competence (*Key competences for lifelong learning: European reference framework*, 2007). Feurzeig and Papert (1969), amongst others, motivates this change by highlighting the skills required to master coding, and the potential of integrating learning in other subjects in the coding process. The dual purpose opens up for different ways of introducing programming in the curriculum. Sweden for example, which is the context this paper is written within, chose to include coding in existing subjects, mathematics being one of them (Skolverket, 2018).

Mathematics curricula already includes mathematical content to be learned and mathematical proficiencies to be developed, and in countries such as Sweden programming is meant to fit into this structure. Introducing programming into the teacher's practices amplifies the complexity and challenges the way he or she has taught the subject so far (Lagrange & Monaghan, 2009). Teaching mathematics becomes more complex because of the human/machine interactions as part of the learning environment, where the teacher could be seen as a conductor creating the prerequisites for an orchestra with several different instrument to harmonize (Trouche, 2004). To orchestrate teaching of mathematics, we need to look into the available components, or didactical configurations, as well as how to exploit such configurations (Drijvers, Doorman, Boon, Reed, & Gravemeijer, 2010). As programming becomes a part of the didactical configuration, we need to attend to how lessons and tasks can be designed to achieve different learning outcomes.

So, how are teachers and researcher to understand the role of coding in mathematics education? We aim to identify the shifting roles of mathematics and programming in educational settings that combines the two with this literature review. It is our hypothesis that studies have utilized different didactical configurations, and exploited those configurations in various ways, in their educational designs. Our research question is *how does different didactical configurations position programming in relation to mathematics in terms of means and goals for learning?* which we intend to answer by systematically searching and analysing recent research papers.

Theoretical background

Digital competency, often highlighted in mathematics education, are versions of Wing's (2006) computational thinking (Arnulfo, 2018). It is described as problem solving proficiency where the problem solver is aware of how for example programming can support successful problem solving across disciplines. Arnulfo (2018) identifies characteristics of computational thinking that are of value for mathematics learners. They are; confidence in dealing with complexity; persistence in working with difficult problems; tolerance for ambiguity; the ability to deal with open-ended problems; and the ability to communicate and work with others to achieve a common goal or solution. These characteristics are thought of as having a positive effect on mathematical learning, and working with mathematics through programming is thought of as having a positive effect on students' development of these characteristics. Thus, showing the bi-directional intention of including computational thinking in mathematics education. Kotsopoulos et al. (2017) argue that working in a mathematics classroom to develop these characteristics is best thought of in terms of four pedagogical experiences, unplugged, tinkering, making and remixing. Tinkering is to modify premade code, making is to write your own code and remixing is to reuse and combine existing pieces of code in new ways to create programs. These three practices could be performed both unplugged and in a computer environment. However, Kotsopoulos et al. (2017) differentiates between the unplugged experience, found in curricula for younger children, and working in a computer environment. They suggest that the four practices could be used in a sequential approach for novices to develop their computational thinking and coding skill.

It is a big step to go from the unplugged experience of computational thinking to working in a computer environment. To understand the complexity of teaching mathematics with human/machine interaction, Trouche (2004) introduced the idea of instrumental orchestration. The idea is that teachers create opportunities for students to develop their computational thinking and mathematical reasoning through organizing available artefacts, i.e. didactical configuration, and determining how to utilize these artefacts, i.e. exploitation mode. Artefacts can be physical objects in the classroom, such as a computer, but also a non-physical object, such as a programming environment which is the case in the present study. Drijvers et al. (2010) compares didactical configuration with how teachers configure their teaching settings, their intentions, and what artefacts that are available. It is the *what?* of instrumental orchestration. They further compare exploitation mode with choosing when and how each instrument (or artefact) should come into play when describing how teachers exploit the didactical configuration to work toward an intended goal. It is the *how?* of instrumental orchestration. In the context of programming, choices of how to introduce, sequence, scaffold and work through a task are all part of the exploitation mode. It is our interpretation that didactical configuration and exploitation mode are key aspects of task design. A task design needs both the prerequisites of a programming language, and the opportunities to use it to create a working classroom task.

Method

The research question of this paper is answered by the means of a systematic literature review. This means that we systematically collect relevant research papers through database (Webb of Science Core Collection and MathEduc) query and analyse the them with the purpose of answering a research question (Eriksson & Barajas, 2013).

Deciding keywords and searching for papers

To find papers that were connected to mathematics education in some way the term “math*” was combined with the following phrases that complement each other; education, learning, teaching, instruction*, activity, student*, pupil* and child*. To find papers that contained mathematics and programming, those phrases were then combined with “programming” or “programing”, together with “computational thinking” and “compute” which was found using www.thesaurus.com.

To get an overview over the resent research, the time for article publications were set to 2007-2018. The search and retrieval of papers were performed on August 20, 2018, therefor no papers published after that date was included in the analysis. The papers were also limited to papers that were published in English. The research area was set to educational research. This resulted in a hit of 318 papers from the database Web of Science. Within the database MathEduc, without the possibility of limiting the search by categories or research areas, the same search resulted in 225 papers.

Reading titles, abstracts and papers

To manually single out papers, that addressed programming and mathematics in education, relevant for this study the following inclusion criteria was used:

- The article addressed computer programming
- The article addressed mathematics education
- The article focused on programming before university level
- The article was an empirical article

The titles and parts of, or full, abstracts of the extracted papers were read by one researcher. In some cases, this was not enough to decide if the article fully satisfied the inclusion criteria. In those cases, the content of the papers was scanned, looking at research questions, method for the study and sometimes the results and conclusions to determine if the papers should be included. In a few cases there were some uncertainties if an article should be included or not. In those cases, the papers were read by both authors and discussed if it should be included based on the criteria.

Several papers addressed the combination of mathematics and programming for higher education and were excluded. Some papers only addressed programming or combined programming with other subjects but had been included in the database search due to the mention of education within STEM (science, technology, engineering and mathematics). These papers were also excluded. There were also papers that focused on mathematics education, where programming was not addressed. Often, they used the framework of computational thinking in some way, without programming, or used the phrase compute unrelated to programming. Papers that were purely theoretical or presented a task without an empirical investigation within the article were also excluded. This resulted in 15 papers from Web of Science, and 7 papers from MathEduc. However, one of the papers, written by Ke (2014), was found in both databases. This resulted in a total of 21 papers. Eight of the papers were published within journals in the area of mathematics didactics, ten papers were in journals with another didactical orientation and two papers came from journals without an explicit didactical focus.

Analysing the papers

To identify patterns, similarities and differences within the different papers the analysing tool NVivo was used to structure the data. The papers were analysed using the lens of instrumental orchestration to identify artefacts that make out the didactical configuration and their exploitation modes. The task design of the different tasks used in the studies, the learning goals of the task, together with the studies' research questions, methods and results, were also analysed. As a first step the following artefacts were identified in the papers: the subject content that the students worked with, the programming environment they used, different physical objects used and the context of the class (e.g. mathematics class or computer class). The didactical configurations and exploitation modes of those configurations where then summarized and categorized.

Three randomly selected papers were cross-analysed by both authors to compare their interpretations. The cross-analysis ensures the coherence in the interpretation and categorization of the papers (Bryman, 2011).

Results

To understand the shifting roles of mathematics and programming in previous research the tasks were categorized into four task-categories, based on comparing and grouping similar didactical configurations and exploitation modes and assigning a label. These categories were then worked through to identify the different roles of programming and mathematics. Three attributes emerged in each category, in which both mathematics and programming was positioned. The attributes were; (1) the learning goal of the task, (2) the task presented to the students and (3) the tools the students used. A summary of the results can be found in Table 1 below.

	Manipulating a physical entity	Manipulating a virtual entity	Creating an interactive environment	Creating, testing and refining algorithms
Learning goal	Mathematical topics (primarily geometry)	Mathematical topics (primarily geometry)	Mathematics and programming, with programming in the foreground	Mathematics and programming, with mathematics in the foreground
Task	"Program this physical entity"	"Program this virtual entity"	"Program a (mathematical) game"	"Create an algorithm to solve this (mathematical) problem"
Tools	Programming environment and (sometimes implicitly) mathematics	Programming environment and (sometimes implicitly) mathematics	Programming environment and sometimes mathematics	Programming environment and mathematics

Table 1: A summary of the results of the literature review showing how mathematics could be positioned within the different types of tasks.

We first expand on the types of tasks found in the literature, and then how mathematics and programming was positioned within the three attributes in each category.

Didactical configurations in the papers

The didactical configuration that were presented in the papers resulted in four categories; manipulating a physical entity; manipulating a virtual entity; creating an interactive environment and; creating, testing and refining algorithms.

Five tasks were categorized as manipulating a physical entity, contained tasks in which the students were to program a robot to do a specific thing. The artefacts used in these types of tasks where; the mathematical topic of geometry and a robot that was used to visualize the mathematical content. The programming environment differed between programming directly on the robot (e.g. Bartolini Bussi & Baccaglini-Frank, 2015) and block programming in a computer environment (e.g. Taylor, 2018). The didactical configurations consisted of using the environment to fit the task of the moving robot and formulating tasks to make the robots do different things. These configurations where then exploited in different ways to help the students develop their spatial understanding. For example, by letting the students discuss movements of the robots.

Our second category contain tasks coded as students working with manipulating a virtual entity. The virtual entity could for example be a programmable cat or a car on the screen that executed predetermined actions without further interaction of the user. 11 tasks were about manipulating a virtual entity. Within these 11 tasks there was an object, in a virtual setting, that the students worked with and manipulated in some way. The artefacts used in these types of tasks where the mathematical topic of geometry and functions and using a digital block programming environment. The didactical configurations in this category included exploiting visual outputs of programming environment in tasks that consisted of either creating geometrical objects (e.g King 2015) or to understand the changing movement, speed and acceleration of objects in some way (e.g. diSessa, 2018). It was done by letting the students work in small groups, pairs or alone. Within several of these studies, mathematics is the learning objective of the task. However, they require the students to understand the structure of programming. Using the task therefore sometimes led to a situation where the tasks' mathematical learning goal was not apparent to the students. One example of this is described by Alfieri, Higashi, Shoop, and Schunn (2015) where the students were supposed to move a robot through a virtual game and needed to calculate wheel rotations to make sure that they end up in the right place. The task is primarily a programming task (program the robot in the right way) in which mathematics was implicit. Mathematical knowledge was needed to solve the programming problem, which sometimes resulted in the development of students' mathematical understanding, or appreciation for the necessity of mathematics.

The third category contains tasks in which the students create their own interactive environment by programming a game, three tasks had this as a main objective. The games could contain one or more virtual entities; however, a user could interact with the game without doing changes to the code. In the description of the task the mathematical topic is not decided by the teacher. It is decided by the students in the development of their game. Because on the open character of the programming task, there was no mathematical area set as an artefact in the didactical configurations of these tasks. The artefact in these tasks were the block programming environment, in a computer class setting (Ke, 2014) or mathematics class setting (e.g. King, 2015). The task was to create a game for a sibling or peer. The students worked in small groups or pairs with the programming languages Scratch or TouchDevelop. Exploiting this configuration often resulted in a situation in which the students worked a lot with solving different programming problems, and developing their understanding for programming. However, Ke (2014) reported that two of the groups in their study created a game without integrating mathematics. In the cases where students had opportunities to work with, and develop an understanding for, mathematical concepts as well as programming, students often focused on the programming.

The final category focus programming and mathematical tasks where students work with the assistance of a computer to make mathematical calculations or models. In these tasks the graphics played a minor (or no) role compared to tasks in prior categories. Six papers included students working with algorithms for calculations or creating models. Students were creating, testing, refining and interpreting different algorithms, with different purposes. The artefacts in this category were the mathematical range over different areas focusing on understanding algorithms and the programming environment. The programming environment used also differed between different text-based programming languages and block programming. The context was set in a mathematics class, informatics class or technology class. The learning goals within these types of tasks focus on understanding algorithms (Grover, Pea, & Cooper, 2015), creating algorithms to solve a mathematical problem (Psycharis & Kallia, 2017) or gaining a deeper mathematical understanding of concepts by creating mathematical models with algorithms (Kahn, Sendova, Sacristán, & Noss, 2011). The students worked alone or in pairs. The tasks encouraged students to exploit programming to solve mathematical problems, and as a help to visualize (Taub, Armoni, Bagno, & Ben-Ari, 2015) or to generate calculations (Psycharis & Kallia, 2017) that the computer executes faster than the students.

Understanding the shifting roles of mathematics and programming

This literature review has presented different types of artefacts used in educational settings to work with mathematics and programming. We also described how these artefacts are put to work in respect to the learning goal, different didactical configurations, how they are exploited, and how this positions mathematics and programming within these contexts. Since the purpose of this study was to investigate the shifting roles of mathematics and programming the following three attributes where used to understand the role and positions of mathematics and programming in different tasks: (1) the learning goal of the task, what the teacher intended the students to learn when working with the task, (2) the assignment presented to the student, either a mathematical assignment or a programming assignment and (3) the tools the student use to solve the assignment.

Papers with tasks of manipulating a physical or virtual entity have one thing in common, the learning goals revolve around students developing their geometric understanding. Hence mathematics has the role of the learning goal (King, 2015). The learning goal is less specific when tasked to create an interactive environment. The students are to learn mathematics when working with programming, but programming is positioned in the foreground in the activity. The learning goals are also that the students should learn and/or develop an interest for programming (Ke, 2014). Within this category, both mathematics and programming are therefore positioned as a learning goal. The tasks in the fourth task-category, like the tasks in the first two, has more explicit mathematical goals where the students are expected to work with and understand the mathematical algorithms. However, programming is also emphasized as an expected learning outcome in the tasks of these categories.

The second attribute is the assignment of the task. Students work with tasks encouraging them to program something. It could be a robot, a figure, a game, or an algorithm. In that sense the assignments are programming assignments. However, some of the tasks in the third and fourth category explicitly said that mathematics was a part of the task; e.g. create a mathematical game (Ke, 2014) or program an algorithm to solve [this mathematical problem] (Psycharis & Kallia, 2017). That was not the case when encouraged to manipulate a physical or virtual entity

The final attribute consists of the tools the students need, and use to solve the given assignment. Within all the categories the students are using programming tools to solve their assignments. They are also using mathematics in some way, however to what extent and in what way differs both between and within the different categories. The mathematics is needed to solve the problem *how to move the robot through an environment* (Alfieri et al., 2015) or *make the cat hit the basket case* (Sung, Ahn, & Black, 2017). However, the mathematics may not be explicit to the students, and even sometimes lost. Within category three, when the students worked with games, the mathematical content was not decided by the teacher or researcher. It was set by the students, depending on how they designed their game and which mathematical problems they encountered. This sometimes meant that the students worked with mathematics and sometimes they did not, as in the case of creating an interactive environment by programming a game. Ke (2014) concludes that two of the groups created a game without having mathematics incorporated, and so the role of mathematics was lost.

Discussion and conclusions

In this paper we have identified different roles that mathematics and programming can adopt when working with the two simultaneously in an educational environment. The results indicate that the roles are not fixed but can take a variety of different positions in relation to each other. Programming can be used for working with mathematics. Mathematics can be used for working with programming. Programming and mathematics can also complement each other in solving different tasks and problems. Our results suggest that these positions are set by learning goals, formulation of the assignment, the environment in which the students can work with the problem and the tools they need to solve it. These shifting roles risk placing one in the foreground and one in the background, both in expected learning outcomes and in the actual activity that the students are working with. Without an explicit mathematical topic, aim and formulation of task, programming tends to be positioned in the foreground (Ke, 2014). But it is possible to balance both mathematics and programming so that they complement and giving each other meaning and purpose.

This paper has shown that awareness of the mathematical content that the students are expected to work with, is an important aspect to take into consideration when working with programming and mathematics. There is however a need for more research that focuses on the relationship between the two subjects. Research that can further investigate how these two subjects can be used in a way that is beneficial for both the development of mathematical understanding and knowledge of programming. The direction takes us deeper into understanding pedagogical aspects of computational thinking (Arnulfo, 2018) and how the structure of programming can be combined with mathematics. Research might bring us closer to the ideas of Feurzeig and Papert (1969), where programming can be used as a way to gain an understanding for mathematical concepts, but where mathematics also can be used to understand programming. The research project this paper is written within will use the outcomes to design research interventions on the topic, further exploring how different task designs work in a school context.

References

Alfieri, L., Higashi, R., Shoop, R., & Schunn, C. D. (2015). Case studies of a robot-based game to shape interests and hone proportional reasoning skills. *International Journal of STEM Education, 2*(1), 1–13.

Arnulfo, P. (2018). A framework for computational thinking dispositions in mathematics education. *Journal for Research in Mathematics Education, 49*(4), 424–461.

Bartolini Bussi, M. G., & Baccaglini-Frank, A. (2015). Geometry in early years: sowing seeds for a mathematical definition of squares and rectangles. *ZDM, 47*(3), 391–405.

Bryman, A. (2011). *Samhällsvetenskapliga metoder*. Malmö: Liber.

diSessa, A. A. (2018). Computational literacy and "the big picture" concerning computers in mathematics education. *Mathematical Thinking and Learning, 20*(1), 3–31.

Drijvers, P., Doorman, M., Boon, P., Reed, H., & Gravemeijer, K. (2010). The teacher and the tool: instrumental orchestrations in the technology-rich mathematics classroom. *Educational Studies in Mathematics, 75*(2), 213–234.

Eriksson Barajas, K., Forsberg, C. & Wengström, Y. (2013). *Systematiska litteraturstudier i utbildningsvetenskap: vägledning vid examensarbeten och vetenskapliga artiklar.* Stockholm: Natur & Kultur.

Feurzeig, W., & Papert, S. (1969). *Programming-languages as a conceptual framework for teaching mathematics. Final report on the first fifteen months of the Logo Project.* (Technical Report 1889). Cambridge, MA: BBN.

Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education, 25*(2), 199–237.

Kahn, K., Sendova, E., Sacristán, A. I., & Noss, R. (2011). Young students exploring cardinality by constructing infinite processes. *Technology, Knowledge and Learning, 16*(1), 3–34.

Ke, F. (2014). An implementation of design-based learning through creating educational computer games: A case study on mathematics learning during design and computing. *Computers & Education, 73*, 26–39.

Key competences for lifelong learning: European reference framework. (2007). Luxembourg: European Communities.

King, A. (2015). Reflecting on classroom practice: spatial reasoning and simple coding. *Australian Mathematics Teacher, 71*(4), 21–27.

Kotsopoulos, D., Floyd, L., Khan, S., Namukasa, I. K., Somanath, S., Weber, J., & Yiu, C. (2017). A pedagogical framework for computational thinking. *Digital Experiences in Mathematics Education, 3*(2), 154–171.

Lagrange, J., & Monaghan, J. (2009). *On the adoption of a model to interpret teachers' use of technology in mathematics lessons.* Proceedings of CERME6, Lyon, France.

Psycharis, S., & Kallia, M. (2017). The effects of computer programming on high school students' reasoning skills and mathematical self-efficacy and problem solving. *Instructional Science, 45*(5), 583–602.

Skolverket. (2018). *Läroplanen för grundskolan, förskoleklassen och fritidshemmet 2011: reviderad 2018.* (5. rev. uppl.). Stockholm: Skolverket.

Sung, W., Ahn, J., & Black, J. B. (2017). Introducing computational thinking to young learners: practicing computational perspectives through embodiment in mathematics education. *Technology, Knowledge and Learning, 22*(3), 443–463.

Taub, R., Armoni, M., Bagno, E., & Ben-Ari, M. (2015). The effect of computer science on physics learning in a computational science environment. *Computers & Education, 87*, 10–23.

Taylor, M. S. (2018). Computer programming with pre-k through first-grade students with intellectual disabilities. *The Journal of Special Education, 52*(2), 78–88.

Trouche, L. (2004). Managing the complexity of human/machine interactions in computerized learning environments. *International Journal of Computers for Mathematical Learning, 9*(3), 281–307.

Wing, J. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.