Remaking and reinforcing mathematics and technology with programming – teacher perceptions of challenges, opportunities and tools in K-12 settings

Received 28 February 2020
Revised 26 May 2020
17 July 2020
Accepted 24 July 2020

Niklas Humble and Peter Mozelius
Department of Computer and System Science, Mid Sweden University, Östersund, Sweden, and
Lisa Sällvin
Department of Information Systems and Technology, Mid Sweden University, Sundsvall, Sweden

Abstract

Purpose – The purpose of this study is to analyse and discuss K-12 mathematics and technology teachers' perceptions on integrating programming in their teaching and learning activities, and perceptions on different programming tools.

Design/methodology/approach – The approach of a case study was used, with data collected from three instances of a professional development programming course for K-12 teachers in mathematics and technology.

Findings – The findings show that there are perceived challenges and opportunities with learning and integrating programming, and with different programming tools. Many teachers perceive programming as fun, but lack the time to learn and implement it, and view different programming tools as both complementary to each other and with individual opportunities and challenges.

Practical implications – The practical implication of the research is that it can provide guidance for teachers and other stakeholders that are in the process of integrating programming in K-12 education. Further, the research provides useful information on teachers' experiences on working with different programming tools.

Social implications – The social implication of the research is that the overall aim of the nation-wide integration process might not succeed if the challenges identified in this study are not addressed, which could have negative effects on the development of students' digital competence.

Originality/value – The value of the research is that it identifies important challenges and opportunities for the integration of programming. That is, that many teachers perceive the different programming tools

Parts of this research have been presented in early versions at conferences and in a book chapter. Comments and discussion in these forums have helped to improve the research. The authors would like to thank the conference attendees, reviewers, editors and organisers of the ECEL2019 conference, the EDULEARN2020 conference, the END2019 conference and the book Education Applications & Developments V. Lastly, the authors would like to thank the teachers that participated in the course instances. Without your participation, there would not have been a study.

available as complimentary to each other, but are hesitating about what is expected of the integration. Findings could also be valuable for future course design of the teacher professional development.

Keywords Teacher professional development, Programming, Textual programming, Block programming, Unplugged programming, K-12

Paper type Research paper

1. Introduction

The integration of programming and computer science in K-12 education has been an ongoing process for decades (Rubio *et al.*, 2015; Balanskat and Engelhardt, 2015; Tsukamoto *et al.*, 2016; Floyd, 2019; Szabo *et al.*, 2019; Nouri *et al.*, 2020). Some reasons for the growing interest in computer programming are: an increasingly need for professionals with coding skills in future society and a need for citizens with basic understanding of computer science in the digitalised society (Balanskat and Engelhardt, 2015; Enders *et al.*, 2019; Lindberg *et al.*, 2019; Admiraal *et al.*, 2019). In many countries, this change has been driven by updates to national curricula to mandate the importance of teaching and learning basic programming (Balanskat and Engelhardt, 2015; Szabo *et al.*, 2019).

In Sweden the time frame was narrowly defined when the Swedish government approved a new curriculum in March 2017, with the plan to implement programming in K-12 settings during 2018. In the Swedish context, programming should be involved as a teaching and learning tool in technology and mathematics. (Heintz *et al.*, 2017) Originally, the fundamental idea was to use programming as a problem-solving tool in these subjects, but later this has been extended with the alternatives of using programming for data processing and visualisation. Since most technology and mathematic teachers lack basic programming skills, a nation-wide teacher professional development initiative is crucial (Mozelius *et al.*, 2019). However, all schools cannot send all their teachers to the courses for professional development simultaneously. In the current plan from the Swedish National Agency for Education, the nation-wide teacher training will continue in 2020 and 2021 (Swedish National Agency for Education, 2020).

Despite the structural similarities between the systematic workflow in programming and mathematics and technology, the learning of programming is a time-consuming process. To make a large-scale change in education takes time (Heintz *et al.*, 2017), and many K-12 teachers lack earlier experiences of programming (Mozelius *et al.*, 2019). The pedagogical challenge that remains is the overhead of learning to programme for teachers with little or no pre-knowledge, involving how programming should be “mathematized” or adapted as a tool in science and technology (Sengupta *et al.*, 2018). To improve the current process, it is important to further explore the teachers’ perceptions on opportunities and challenges in learning and integrating programming; and their perceptions of different programming tools.

This study was conducted with the aim to analyse and discuss K-12 mathematics and technology teachers’ perceptions on integrating programming in their teaching and learning activities, and perceptions on different programming tools. The work was guided by the following two research questions:

RQ1. What are K-12 mathematics and technology teachers’ perceptions of challenges and opportunities in learning and integrating programming in their teaching and learning activities?

RQ2. What are K-12 mathematics and technology teachers’ perceptions of challenges and opportunities with different programming tools for their teaching and learning activities?

2. Background

Previous research has pointed out both opportunities and obstacles with learning and integrating programming in K-12 education. An opportunity is that learning programming is expected to help students acquire and develop skills that are useful, such as problem-solving, collaboration, creativity, logical and computational thinking (Balanskat and Engelhardt, 2015; Duncan and Bell, 2015; Zhang and Nouri, 2019). An obstacle is that learning and teaching programming is a resource-demanding activity, where expected skills does not develop in itself simply by learning programming (Psycharis and Kallia, 2017). Schools need to have access to teacher professional development courses and resources for the teachers to learn and integrate programming (Mannila *et al.*, 2014; Tundjungsari, 2016).

Programming can be defined as in the Cambridge Dictionary (2020): "the activity or job of writing computer programs". An activity that can be carried out in a wide variety of programming languages and programming tools. Programming also involves computational thinking, which has several definitions, one is the older broader definition by Wing (2006) that computational thinking is about solving problems and designing computer programmes. This study uses the term as it later has been elaborated and redefined by Shute *et al.* (2017), with the six main components of abstraction, algorithms, debugging, decomposition, iteration and generalisation.

This study has a focus on teaching and learning programming in the formal context of K-12 mathematics and technology, because of the direction set by the Swedish National Agency for Education (Swedish National Agency for Education, 2020). However, it is possible to teach and learn programming through other strategies and contexts. From a Swedish perspective there are earlier research studies on how to design programming activities in makerspaces in both K-12 technology education (Eriksson *et al.*, 2018) and teacher education (Kjällander *et al.*, 2018). The more global roots to the maker movement can be traced back to the MIT and Seymour Papert (Resnick and Robinson, 2017, p. 36).

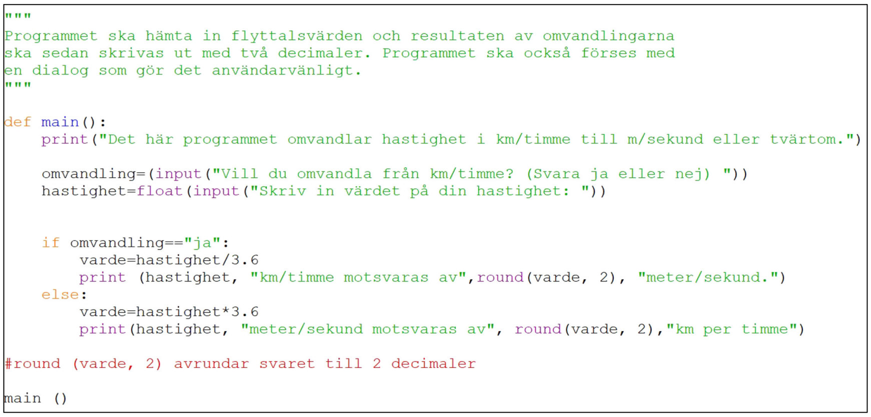
Two common approaches to teach programming are through textual programming tools and block programming tools, both requires access to a computer (Garneli *et al.*, 2015; Tsukamoto *et al.*, 2016; Gomez *et al.*, 2019). A third approach, called unplugged programming, that uses no computer is also described in previous research (Bell *et al.*, 2009; Aranda and Ferguson, 2018; Caeli and Yadav, 2020). These three types of programming tools are described more thoroughly in the sub-sections below.

2.1 Textual programming

A definition of textual programming is that it is written and modified with the help of a text editor (Chen *et al.*, 2019; Chiu, 2020). Many see this as the authentic way of programming and the approach has a more established connection to professional development than block programming tools (Garneli *et al.*, 2015; Tsukamoto *et al.*, 2016; Mladenović *et al.*, 2020). In a literature review on introductory programming in education, Szabo *et al.* (2019) conclude that the three main textual programming tools used were Python, Pascal and Java.

A common challenge in previous research is that textual programming is more complex and difficult to learn for novice programmers than other types of programming, such as block programming (Garneli *et al.*, 2015; Tsukamoto *et al.*, 2016; Lindberg *et al.*, 2019; Chiu, 2020). In a study on introducing programming to primary school students, textual programming tool did not seem to score as high as the block programming tool in regards to student motivation (Tsukamoto *et al.*, 2016). This could indicate that it is less suitable for teaching younger students programming. Figure 1 below shows a teacher's solution of a programming assignment with the use of textual programming in a course on fundamental programming for technology and mathematic teachers.

An opportunity with using textual programming tools are their connection to professional development and perception as "authentic" programming, which could be motivating and inspiring for the students (Garneli *et al.*, 2015; Tsukamoto *et al.*, 2016; Mladenović *et al.*, 2020). In a study by Tsukamoto *et al.* (2015) it is concluded that it is possible for students from grade 4 and above to use and enjoy textual programming tools. It is also suggested that the use of textual programming tools in education could attract students with higher programming ability, which could make it a suitable for after school activities (Mladenović *et al.*, 2018).



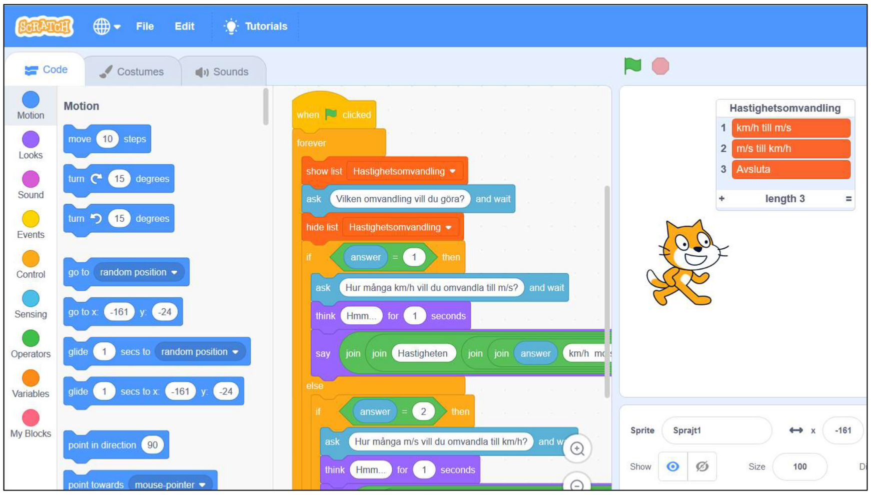
```
"""
Programmet ska hämta in flyttalsvärden och resultaten av omvandlingarna
ska sedan skrivas ut med två decimaler. Programmet ska också förses med
en dialog som gör det användarvänligt.
"""

def main():
    print("Det här programmet omvandlar hastighet i km/timme till m/sekund eller tvärtom.")

    omvandling=(input("Vill du omvandla från km/timme? (Svara ja eller nej) "))
    hastighet=float(input("Skriv in värdet på din hastighet: "))


    if omvandling=="ja":
        varde=hastighet/3.6
        print (hastighet, "km/timme motsvaras av",round(varde, 2), "meter/sekund.")
    else:
        varde=hastighet*3.6
        print(hastighet, "meter/sekund motsvaras av", round(varde, 2),"km per timme")

#round (varde, 2) avrundar svaret till 2 decimaler

main ()
```

Figure 1.
Velocity conversion assignment solved in Python (Graphics by Peter Mozelius)

2.2 Block programming

Block programming tools can be compared to building with LEGO, programmes are created by manipulating graphical elements in the environments, usually by a drag-and-drop approach (Maloney *et al.*, 2010; Lawanto *et al.*, 2017; Rodríguez-Martínez *et al.*, 2020; Chiu, 2020). According to a study, block programming tools are amongst the most commonly used in introductory programming in K-12 education (Garneli *et al.*, 2015). Figure 2 below shows a teacher's solution of a programming assignment with the use of block programming in a course on fundamental programming for technology and mathematic teachers.

One challenge mentioned by Tsukamoto *et al.* (2015) is that block programming tools might limit advanced students in developing their programming skills. In a study by Weintrop and Wilensky (2019) it was concluded that skills acquired with block programming tools did not automatically transfer to textual programming tools. This could potentially pose a challenge, since block programming tools are not commonly used in a professional context, but textual programming tools are (Tsukamoto *et al.*, 2015). Sooner or later students will have



```
when clicked
forever
    show list Hastighetsomvandling
    ask Vilken omvandling vill du göra? and wait
    hide list Hastighetsomvandling
    if answer = 1 then
        ask Hur många km/h vill du omvandla till m/s? and wait
        think Hmm... for 1 seconds
        say join join Hastigheten join join answer km/h m
    else
        if answer = 2 then
            ask Hur många m/s vill du omvandla till km/h? and w
            think Hmm... for 1 seconds
```

Figure 2.
Velocity conversion assignment solved in Scratch (Graphics by Peter Mozelius)

to make the transition to textual programming tools (Tsukamoto *et al.*, 2015), but will they be able to draw on previous knowledge in block programming tools?

An opportunity with block programming tools is their way of avoiding syntactical errors, since the environments are constructed so that blocks will only connect to each other when everything is correct (Sáez-López *et al.*, 2016). In a study on introducing programming to students in primary school, it is suggested that block programming tools might be more suitable for younger students, since they seemed to be perceived as more motivating (Tsukamoto *et al.*, 2016). Another opportunity is that block programming tools have a history of incorporating other activities to programming, such as storytelling, games, music and art (Maloney *et al.*, 2010; Cooper and Cunningham, 2010; Topalli and Cagiltay, 2018).

2.3 Unplugged programming

The act of programming without a computer is called unplugged programming (Bell *et al.*, 2009; Faber *et al.*, 2017; Heintz *et al.*, 2017; Aranda and Ferguson, 2018). The idea is that it is possible to learn basic programming through activities that do not necessarily require a computer, such as giving oral or written instructions on how to perform a task in a number of steps (Wyffels *et al.*, 2014; Faber *et al.*, 2017; Aranda and Ferguson, 2018; Miller *et al.*, 2018). Another way to approach unplugged programming is through board games that aims to teach basic programming (Tsarava *et al.*, 2018; Jagušt *et al.*, 2018). Figure 3 below shows an example of an unplugged programming exercise used in a course on fundamental programming for technology and mathematic teachers.

Bell and Vahrenhold (2018) comes to the conclusion, after reviewing previous research on unplugged programming, that it should be used in relation to other technology and not as an isolated occurrence. In a study on high school students' perceived understanding of computer science concepts and their attitudes towards the subject, the unplugged approach did not seem to have any impact (Feaster *et al.*, 2011). Previous research points out that there is a need for further research on unplugged programming to determine the effect on learning programming and relating concepts (AlAmer *et al.*, 2015; Brackmann *et al.*, 2017).

An opportunity with unplugged programming is the easy access, since a computer is not required. Brackmann *et al.* (2017) points out that this could be valuable for schools in regions that are low on technical resources. Studies suggests that unplugged programming, if used appropriately, could introduce programming and computer science to, especially younger, students in a meaningful way (Brackmann *et al.*, 2017; Bell and Vahrenhold, 2018). In a study by Wohl *et al.* (2015), it is concluded that unplugged programming can generate high level of understanding for algorithms, debugging and logical predictions amongst younger students.

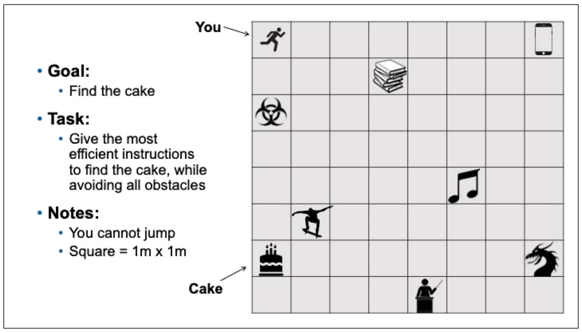

You
Goal:
Find the cake
Task:
Give the most efficient instructions to find the cake, while avoiding all obstacles
Notes:
You cannot jump
Square = 1m x 1m
Cake

Figure 3. Example of an unplugged programming exercise (Graphics by Niklas Humble)

3. Methodology

The study was conducted with a case study approach, with data gathered from three instances of a professional development programming course for K-12 teachers in mathematics and technology. The course instances were given during the Autumn Semester of 2018, Spring Semester of 2019 and Autumn Semester of 2019. The course was of 7.5 ECTS, a study pace of 25%, and included both online learning in a virtual learning environment, and face-to-face meetings with lectures and workshops. A common recommendation for the case study approach is to use multiple sources for data collection to reach deeper understanding of the studied phenomenon (Remenyi *et al.*, 2002; Denscombe, 2014, pp. 54–56; Bryman, 2016, pp. 60–61).

The data in this study were collected through three sources, with the idea of triangulation (Patton, 2007; Fusch *et al.*, 2018), to enhance the analysis of the case, these were: workshop observations, essays-assignments and online postings in the courses' discussion forums (Table 1). Content analysis, and deductive-inductive coding, was used to analyse the collected data and identify topics of interests (Drisko and Maschi, 2015, pp. 5–26; Bryman, 2016, p. 283). A content analysis is carried out by creating codes that can be applied to data for the development of meaningful categories as described by Blair (2015).

As pointed out by Stemler (2001) the coding and categorisation can be conducted inductively with codes that emerges from data, or with a deductive *a priori* approach where codes and categories have been created beforehand. In this study, the collected data were first structured with deductive coding according to the two research questions. Inductive coding was used to analyse and group emerging themes of challenges and opportunities in the collected essays and forum postings in spreadsheets, these were later compared to the analysis of the workshop observations for additional themes of interests.

The workshop observations were conducted during face-to-face meetings in all course instances (Table 1). The workshops focussed on programming assignments, discussions, and exchanging knowledge and experience. All workshops touched upon textual programming tools and block programming tools, as well as programming in general. Three of the workshops (two in spring 2019 and one in autumn 2019) also included exercises and discussions on unplugged programming.

The essay assignments were collected from the online virtual learning environment Moodle, that was used in all course instances (Table 1). The assignment was given a couple of weeks into the courses with instructions to write an essay, reflecting on opportunities and challenges in learning programming and how participants perceive the textual programming tool Python and the block programming tool Scratch.

The forum postings were collected from the online virtual learning environment in two of the course instances (Table 1). This was done midway through the courses, in relation to the workshops that touched upon unplugged programming. Besides unplugged programming exercises and discussions, the participants were encouraged to share their thoughts on the subject in the online course forum of Moodle. The instruction for the online postings was to reflect on how they perceived unplugged programming and what challenges and opportunities they could see for its application in the classroom.

Table 1.
Data collection

	Autumn 2018	Spring 2019	Autumn 2019
Number of participants	60	32	15
Number of workshop observations	8	8	4
Number of essay-assignments	31	18	6
Number of forum postings	0	28	5

All teachers participating in the course instances have been informed about the intention to use the collected material in research. They have been informed about the intention with the research and their right to withdraw contribution. To protect participants' anonymity, all extracts from forum postings and essays have been translated and rewritten, without changing the essential meanings.

4. Results and analysis

This section presents the results according to the research question that they relate to. Results that relate to the first research question are presented in the first category, "Learning and integrating programming". Results that relate to the second research question are presented in the second category, "Perceptions of programming tools".

4.1 Learning and integrating programming

Three themes of opportunities emerged in the collected essay-assignments:

(1) It is easy to connect programming to other teaching and learning activities in school, such as logic and problem-solving.

(2) There are a lot of easily accessible teaching and learning material on the Internet and in books.

(3) Programming is fun, which was the most frequent theme. These themes were amplified by the workshop observations. One teacher writes about the experience of implementing programming in grade 7 and 9 in an essay:

> My plans for the lesson cracked after 20 minutes, at the same time creativity sparked throughout the classroom. I must admit that it was uncomfortable to lose control, but at the same time it was fun to see the enthusiasm in the classroom that maintained throughout the whole lesson.

Three themes about challenges emerged in the collected essay-assignments:

(1) Lack of directives for the integration, for example, what type of programming is to be implemented, where should it be implemented and how much?

(2) Programming is difficult, for example, learning new structures and logic and much of the available material is in English.

(3) Lack of time for learning and integrating programming, which was the most frequent theme. These themes were amplified by the workshop observations, where teachers discussed and shared experiences about implementing programming and trying to get directives from schoolboards and the Swedish National Agency for Education. One teacher expresses the frustration about this in an essay:

> It is simply not possible for me to put the amount of time on this that I think it requires. From where am I supposed to take time in an already full schedule? [. . .] As I have mentioned earlier, I get no time from my employer to work with the integration of programming in my subjects and because of this the time must be taken from something else. I find it extremely difficult to have to prioritise away something that my student might need and, in that way, potentially affect their learning negatively.

4.2 Perceptions of programming tools

The main opportunity with the textual programming tool Python is its perceived freedom and potential compared to the block programming tool Scratch, for example in relation to other subjects, future education and labour market. The main challenge is that Python is perceived as difficult compared to Scratch, which might make the process of integration more difficult. The opportunity and challenge are mentioned in both essays and workshop observations. In an essay, one teacher writes:

> My plan is to solve all assignments in this course with Python, because it seems to be more flexible and appropriate for advanced projects. At the same time, I am hesitant to use it in my own teaching. Although it is pretty forgiving compared to other textual programming tools, for example in regards to the use of data types.

The main opportunity with the block programming tool Scratch is its perceived lower threshold and being easier than Python, especially for novice programmers. The main challenge is that Scratch is perceived as limited and not appropriate for more complex tasks compared to Python. The opportunity and the challenge are mentioned in both essays and workshop observations. In an essay, one teacher writes:

> If I write a program in Scratch the structure will be much better with the use of blocks. However, it is also harder for me to use when the programming is about mathematical calculations.

The main opportunity with unplugged programming is that it is perceived as an easy and fun way to introduce or support programming and other activities, such as mathematics, computational thinking and logical thinking. The main challenge is that it is perceived as mainly directed towards younger students, and that using it with older students might not resonate with their expectations about programming. The opportunity and the challenge are mentioned in both essays and workshop observations. In a forum posting, one teacher writes:

> Unplugged programming is a good way to show how complicated and yet easy it is to build a program. [. . .] The students need to understand that when you write a program, what you are doing is giving instructions to someone/something that have no knowledge of what your intensions are. Everything needs to be described in detail. It would be desirable if students had good experience with unplugged programming before grade 1–6. After that they could start with block programming and draw on their previous experiences when the programs start to become more complicated.

It was common in both essays, forum postings and workshop observations that teachers expressed a perceived relationship between the programming tools. In the workshops, teachers elaborated on this notion and expressed a perception of the programming tools to range from a general approach to a more specialised approach on programming. Unplugged programming was described as the most general and appropriate for the lower grades and a wider range of subjects. Python was described as the most specialised approach and more appropriate for the upper grades and the subject of mathematics. While Scratch was described in the middle, concerning subjects and grades. In one essay, a teacher writes:

> I believe that knowledge in one programming language leads to similar insights in the other language, although the problem at hand might be solved a little differently depending on what programming tool is at your disposal. Programming is a way of thinking in its own, independent of what language, it is about algorithms and the implementation of these.

5. Findings and discussion

Concerning the first research question, there are both expected and unexpected findings. The expected findings relate to findings in previous research. The first being that teachers found it easy to connect programming to other activities, which can be related to the expectancy of students getting support to develop other skills through learning programming, such as: problem-solving, collaboration, creativity, logical and computational thinking (Balanskat and Engelhardt, 2015; Duncan and Bell, 2015; Zhang and Nouri, 2019). The second being that teachers expressed concerns about programming being too difficult, which can be related to that learning and teaching programming is a difficult task; and that knowledge, resources and support is needed (Mannila *et al.*, 2014; Tundjungsari, 2016; Psycharis and Kallia, 2017).

The unexpected findings have some connections to previous research. For example, previous research expresses that programming can be fun and engaging (Tsukamoto *et al.*, 2015; Tsukamoto *et al.*, 2016; Brackmann *et al.*, 2017; Bell and Vahrenhold, 2018). But it was unexpected that it was one of the most recurring themes in the analysed material. Previous research has pointed out the need for resources, knowledge and support in learning and integrating programming (Mannila *et al.*, 2014; Tundjungsari, 2016; Psycharis and Kallia, 2017). It was unexpected that so many teachers expressed a lack of time and guidance in the collected material.

Concerning the second research question, there are both expected and unexpected findings. The expected findings relate to the findings in previous research. The first being that teachers tend to compare textual programming tools to block programming tools in regards to opportunities and challenges. Textual programming tools are often perceived as difficult compared to block programming tools, but also to have greater freedom and potential (Garneli *et al.*, 2015; Tsukamoto *et al.*, 2015; Tsukamoto *et al.*, 2016; Sáez-López *et al.*, 2016; Lindberg *et al.*, 2019; Weintrop and Wilensky, 2019; Mladenović *et al.*, 2020; Chiu, 2020). The second being that unplugged programming is perceived as fun and easy, but might be limited to younger students (Feaster *et al.*, 2011; Wohl *et al.*, 2015; Brackmann *et al.*, 2017; Bell and Vahrenhold, 2018).

The unexpected findings have some connections to previous research. Independent of which tools or approaches that are used to learn and integrate programming, the goal is often similar, to develop computational thinking and foster creativity, collaboration and problem-solving (Balanskat and Engelhardt, 2015; Duncan and Bell, 2015; Zhang and Nouri, 2019). It was unexpected that teachers, relatively early in the courses, expressed perceptions of similarity between the programming tools, and that the tools could be used to enrich and build upon each other.

6. Conclusion

The study was conducted with the aim to analyse and discuss K-12 mathematics and technology teachers' perceptions on integrating programming in their teaching and learning activities, and their perceptions on different programming tools. Both challenges and opportunities have been found regarding the learning and integration of programming in K-12 mathematics and technology, and with different programming tools. Recurring in the collected material were teachers' perceptions of programming as fun, but difficult to learn and integrate due to a lack of time and directives for the integration and their own professional development.

An interesting finding was also the perceived relationship between textual programming, block programming and unplugged programming; and that it was described in hierarchical terms. This could potentially be drawn upon in a large-scale integration of programming in K-12 education. The finding suggest that unplugged programming has a more general approach to programming, which make it appropriate for the lower grades and a wide range of school subjects. Textual programming has a more specialised approach, which make it appropriate for the upper grades and the subject of mathematics. While block programming has an approach somewhere in the middle of general and specialised, making it appropriate to integrate as a stepping stone in the grades between unplugged and textual programming.

7. Limitations and future research

This study was conducted in three instances of a programming course for K-12 teachers in mathematics and technology where the authors were both course teachers and researchers. With multiple roles for the authors, challenges and limitations follows that need to be addressed. The collection of data was conducted in a context where teachers have freely chosen to learn programming, it is likely that their attitudes towards programming do not represent that of the average teacher. The authors where both course teachers and researchers, it is possible that the authors attitudes towards programming has affected the teachers' attitudes during the courses.

An interesting next step of research would be to investigate the role of different programming tools in makerspace and after school activities. In this study, the use and perception of programming tools are influenced by the subjects in which they are to be integrated. But how are the use and perception of programming tools affected, if they are removed from the context of school subjects? Do they remain the same or do new opportunities and obstacles emerge?

References

Admiraal, W., Post, L., Guo, P., Saab, N., Makinen, S., Rainio, O., Vuori, J., Bourgeois, J., Kortuem, G. and Danford, G. (2019), "Students as future workers: cross-border multidisciplinary learning labs in higher education", *International Journal of Technology in Education and Science*, Vol. 3 No. 2, pp. 85-94.

AlAmer, R.A., Al-Doweesh, W.A., Al-Khalifa, H.S. and Al-Razgan, M.S. (2015), "Programming unplugged: bridging CS unplugged activities gap for learning key programming concepts", *2015 Fifth International Conference on E-Learning (Econf)*, IEEE, pp. 97-103.

Aranda, G. and Ferguson, J.P. (2018), "Unplugged Programming: the future of teaching computational thinking", *Pedagogika*, Vol. 68 No. 3, pp. 279-292.

Balanskat, A. and Engelhardt, K. (2015), *Computing Our Future: Computer Programming and coding*, Priorities, School Curricula and Initiatives across Europe: European Schoolnet (EUN Partnership AIBSL), Brussels.

Bell, T. and Vahrenhold, J. (2018), "CS unplugged—how is it used, and does it work?", *Adventures between Lower Bounds and Higher Altitudes*, Springer, Cham, pp. 497-521.

Bell, T., Alexander, J., Freeman, I. and Grimley, M. (2009), "Computer science unplugged: school students doing real computing without computers", *The New Zealand Journal of Applied Computing and Information Technology*, Vol. 13 No. 1, pp. 20-29.

Blair, E. (2015), "A reflexive exploration of two qualitative data coding techniques", *Journal of Methods and Measurement in the Social Sciences*, Vol. 6 No. 1, pp. 14-29.

Brackmann, C.P., Román-González, M., Robles, G., Moreno-León, J., Casali, A. and Barone, D. (2017), "Development of computational thinking skills through unplugged activities in primary school", *Proceedings of the 12th Workshop on Primary and Secondary Computing Education*, pp. 65-72.

Bryman, A. (2016), *Social Research Methods*, Oxford University Press, Oxford.

Caeli, E.N. and Yadav, A. (2020), "Unplugged approaches to computational thinking: a historical perspective", *TechTrends*, Vol. 64 No. 1, pp. 29-36.

Chen, C., Haduong, P., Brennan, K., Sonnert, G. and Sadler, P. (2019), "The effects of first programming language on college students' computing attitude and achievement: a comparison of graphical and textual languages", *Computer Science Education*, Vol. 29 No. 1, pp. 23-48.

Chiu, C.F. (2020), "Facilitating K-12 teachers in creating apps by visual programming and project-based learning", *International Journal of Emerging Technologies in Learning (iJET)*, Vol. 15 No. 1, pp. 103-118.

Cooper, S. and Cunningham, S. (2010), "Teaching computer science in context", *Acm Inroads*, Vol. 1 No. 1, pp. 5-8.

Denscombe, M. (2014), *The Good Research Guide: For Small-Scale Social Research Projects*, McGraw-Hill Education, Maidenhead.

Cambridge Dictionary (2020), *Cambridge English Online Dictionary*, available at: https://dictionary.cambridge.org/dictionary/english/programming (accessed 29 04 2020).

Drisko, J.W. and Maschi, T. (2015), *Content analysis*, Pocket Guides to Social Work R, Oxford.

Duncan, C. and Bell, T. (2015), "A pilot computer science and programming course for primary school students", *Proceedings of the Workshop in Primary and Secondary Computing Education*, pp. 39-48.

Enders, T., Hediger, V., Hieronimus, S., Kirchherr, J.W., Klier, J., Schubert, J. and Winde, M. (2019), "Future skills: six approaches to close the skill gap", *The 7th World Government Summit*, pp. 5-15.

Eriksson, E., Heath, C., Ljungstrand, P. and Parnes, P. (2018), "Makerspace in school—considerations from a large-scale national testbed", *International journal of child-computer interaction*, Vol. 16, pp. 9-15.

Faber, H.H., Wierdsma, M.D., Doornbos, R.P., van der Ven, J.S. and de Vette, K. (2017), "Teaching computational thinking to primary school students via unplugged programming lessons", *Journal of the European Teacher Education Network*, Vol. 12, pp. 13-24.

Feaster, Y., Segars, L., Wahba, S.K. and Hallstrom, J.O. (2011), "Teaching CS unplugged in the high school (with limited success)", *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*, pp. 248-252.

Floyd, S.P. (2019), "Historical high school computer science curriculum and current K-12 initiatives", *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, p. 1287.

Fusch, P., Fusch, G.E. and Ness, L.R. (2018), "Denzin's paradigm shift: revisiting triangulation in qualitative research", *Journal of Social Change*, Vol. 10 No. 1, p. 2.

Garneli, V., Giannakos, M.N. and Chorianopoulos, K. (2015), "Computing education in K-12 schools: a review of the literature", *2015 IEEE Global Engineering Education Conference (EDUCON)*, IEEE, pp. 543-551.

Gomez, M.J., Moresi, M. and Benotti, L. (2019), "Text-based programming in elementary school: a comparative study of programming abilities in children with and without block-based experience", *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, pp. 402-408.

Heintz, F., Mannila, L., Nordén, L.Å., Parnes, P. and Regnell, B. (2017), "Introducing programming and digital competence in Swedish K-9 education", *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*, Springer, Cham, pp. 117-128.

Jagušt, T., Krzic, A.S., Gledec, G., Grgić, M. and Bojic, I. (2018), "Exploring different unplugged game-like activities for teaching computational thinking", *2018 IEEE Frontiers in Education Conference (FIE)*, IEEE, pp. 1-5.

Kjällander, S., Åkerfeldt, A., Mannila, L. and Parnes, P. (2018), "Makerspaces across settings: didactic design for programming in formal and informal teacher education in the nordic countries", *Journal of Digital Learning in Teacher Education: Special Issue: Maker Spaces in Teacher Education*, Vol. 34 No. 1, pp. 18-30.

Lawanto, K., Close, K., Ames, C. and Brasiel, S. (2017), "Exploring strengths and weaknesses in middle school students' computational thinking in scratch", *Emerging Research, Practice, and Policy on Computational Thinking*, Springer, Cham, pp. 307-326.

Lindberg, R.S., Laine, T.H. and Haaranen, L. (2019), "Gamifying programming education in K-12: a review of programming curricula in seven countries and programming games", *British Journal of Educational Technology*, Vol. 50 No. 4, pp. 1979-1995.

Maloney, J., Resnick, M., Rusk, N., Silverman, B. and Eastmond, E. (2010), "The scratch programming language and environment", *ACM Transactions on Computing Education (TOCE)*, Vol. 10 No. 4, pp. 1-15.

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

Miller, B., Kirn, A., Anderson, M., Major, J.C., Feil-Seifer, D. and Jurkiewicz, M. (2018), "Unplugged robotics to increase K-12 students' engineering interest and attitudes", *2018 IEEE Frontiers in Education Conference (FIE)*, IEEE, pp. 1-5.

Mladenović, M., Boljat, I. and Žanko, Ž. (2018), "Comparing loops misconceptions in block-based and text-based programming languages at the K-12 level", *Education and Information Technologies*, Vol. 23 No. 4, pp. 1483-1500.

Mladenović, M., Mladenović, S. and Žanko, Ž. (2020), "Impact of used programming language for K-12 students' understanding of the loop concept", *International Journal of Technology Enhanced Learning*, Vol. 12 No. 1, pp. 79-98.

Mozelius, P., Ulfenborg, M. and Persson, N. (2019), "Teacher attitudes towards the integration of programming in middle school mathematics", *INTED 2019*, IATED, Valencia.

Nouri, J., Zhang, L., Mannila, L. and Norén, E. (2020), "Development of computational thinking, digital competence and 21st century skills when learning programming in K-9", *Education Inquiry*, Vol. 11 No. 1, pp. 1-17.

Patton, M.Q. (2001), *Qualitative Research & Evaluation Methods: Integrating Theory and Practice*, Sage publications,, pp. 247-248.

Psycharis, S. and Kallia, M. (2017), "The effects of computer programming on high school students' reasoning skills and mathematical self-efficacy and problem solving", *Instructional Science*, Vol. 45 No. 5, pp. 583-602.

Remenyi, D., Money, A., Price, D. and Bannister, F. (2002), "The creation of knowledge through case study research", *Irish Journal of Management*, Vol. 23 No. 2, p. 1.

Resnick, M. and Robinson, K. (2017), *Lifelong Kindergarten: Cultivating Creativity through Projects, Passion, Peers, and Play*, MIT Press, Cambridge.

Rodríguez-Martínez, J.A., González-Calero, J.A. and Sáez-López, J.M. (2020), "Computational thinking and mathematics using Scratch: an experiment with sixth-grade students", *Interactive Learning Environments*, Vol. 28, pp. 1-12.

Rubio, M.A., Romero-Zaliz, R., Mañoso, C. and Angel, P. (2015), "Closing the gender gap in an introductory programming course", *Computers & Education*, Vol. 82, pp. 409-420.

Sáez-López, J.M., Román-González, M. and Vázquez-Cano, E. (2016), "Visual programming languages integrated across the curriculum in elementary school: a two year case study using 'Scratch' in five schools", *Computers & Education*, Vol. 97, pp. 129-141.

Sengupta, P., Brown, B., Rushton, K. and Shanahan, M.-C. (2018), "Reframing coding as "mathematization" in the K–12 classroom: views from teacher professional learning", *Alberta Science Education Journal*, Vol. 45 No. 2, pp. 28-36.

Shute, V.J., Sun, C. and Asbell-Clarke, J. (2017), "Demystifying computational thinking", *Educational Research Review*, Vol. 22, pp. 142-158.

Stemler, S. (2001), "An overview of content analysis", *Practical Assessment, Research & Evaluation*, Vol. 7, pp. 137-146.

Swedish National Agency for Education / Skolverket (2020), "Fundamentals of programming aligned to the subject didactics / Grundläggande Programmering Med Ämnesdidaktisk Inriktning", available at: https://www.skolverket.se/skolutveckling/kurser-och-utbildningar/grundlaggande-programmering-med-amnesdidaktisk-inriktning (accessed 27 February 2020).

Szabo, C., Sheard, J., Luxton-Reilly, A., Becker, B.A. and Ott, L. (2019), "Fifteen years of introductory programming in schools: a global overview of K-12 initiatives", *Proceedings of the 19th Koli Calling International Conference on Computing Education Research*, pp. 1-9.

Topalli, D. and Cagiltay, N.E. (2018), "Improving programming skills in engineering education through problem-based game projects with scratch", *Computers & Education*, Vol. 120, pp. 64-74.

Tsarava, K., Moeller, K. and Ninaus, M. (2018), "Training computational thinking through board games: the case of Crabs & Turtles", *International Journal of Serious Games*, Vol. 5 No. 2, pp. 25-44.

Tsukamoto, H., Takemura, Y., Nagumo, H., Ikeda, I., Monden, A. and Matsumoto, K.I. (2015), "Programming education for primary school children using a textual programming language", *2015 IEEE Frontiers in Education Conference (FIE)*, IEEE, pp. 1-7.

Tsukamoto, H., Takemura, Y., Oomori, Y., Ikeda, I., Nagumo, H., Monden, A. and Matsumoto, K.I. (2016), "Textual vs. visual programming languages in programming education for primary school children", *2016 IEEE Frontiers in Education Conference (FIE)*, IEEE, pp. 1-7.

Tundjungsari, V. (2016), "E-learning model for teaching programming language for secondary school students in Indonesia", *2016 13th International Conference on Remote Engineering and Virtual Instrumentation (REV)*, IEEE, pp. 262-266.

Weintrop, D. and Wilensky, U. (2019), "Transitioning from introductory block-based and text-based environments to professional programming languages in high school computer science classrooms", *Computers & Education*, Vol. 142, 103646.

Wing, J.M. (2006), "Computational thinking", *Communications of the ACM*, Vol. 49 No. 3, pp. 33-35.

Wohl, B., Porter, B. and Clinch, S. (2015), "Teaching computer science to 5-7 year-olds: an initial study with scratch, cubelets and unplugged computing", *Proceedings of the Workshop in Primary and Secondary Computing Education*, pp. 55-60.

Wyffels, F., Martens, B. and Lemmens, S. (2014), "Starting from scratch: experimenting with computer science in flemish secondary education", *Proceedings of the 9th Workshop in Primary and Secondary Computing Education*, pp. 12-15.

Zhang, L. and Nouri, J. (2019), "A systematic review of learning computational thinking through Scratch in K-9", *Computers & Education*, Vol. 141, 103607.

Corresponding author
Niklas Humble can be contacted at: niklas.humble@miun.se