Teachers’ understanding of programming and computational thinking in primary education – A critical need for professional development

Abstract
Programming and computational thinking have (re)gained an increased focus in compulsory education worldwide, consequently demanding teachers of various subjects to engage in the teaching and learning of the two. A recent curriculum reform in Norway emphasised the development of students’ computational skills by integrating programming into four subject domains: mathematics, natural science, music, and arts and crafts. However, these requirements come without a necessary professional development programme and are based on the presumption that all concerned teachers understand the concepts of programming and computational thinking and know how to teach these skills in a sound pedagogical and didactical way. Therefore, this study investigated how teachers understand programming and computational thinking and the relationship between the two concepts. We also investigated the teachers’ approaches to teaching these concepts and their need for further professional development. To address these issues, we conducted semi-structured interviews with eight primary school teachers and thematically analysed the data. Overall, the teachers reported positive attitudes towards the new curriculum and its focus on 21st-century skills, including programming and computational thinking. However, their understanding of these concepts was narrow, focused on only one of the six pertinent subskills: algorithms. Furthermore, the teachers’ teaching approaches were limited. Finally, we observed a variety of professional development processes and practices. The teachers accentuated a critical need for professional development within these domains. Our findings showed that to fulfil the curricular expectations of developing students’ computational thinking skills, increased training of primary school teachers is needed.

Keywords: computational thinking, programming, professional development, primary school teachers, teaching approaches, thematic analysis
Læreres forståelse av programmering og algoritmisk tenkning i barneskolen – Et stort behov for kompetanseheving

Sammendrag


Nøkkelord: algoritmisk tenkning, programmering, videre-/etterutdanning, barneskolelærere, undervisningspraksis, tematisk analyse

Introduction

In the last decade, computational thinking (CT) has been identified as an important competence for students’ effective participation in future work life and society (Bocconi et al., 2016). Researchers have argued that CT has the potential to increase, for instance, students’ problem-solving skills, creativity and even learning in other subject domains (Scherer et al., 2019). Thus, CT is being integrated into the national curricula in many countries (OECD, 2016). The concepts programming and/or computer coding are most often used in the curricula to develop students’ CT skills (Sun et al., 2021). Recently, more Western countries, such as Norway and Finland, have introduced reforms that make programming and/or coding compulsory for all grades 1 to 12 students (Halinen, 2018; Official Norwegian Reports NOU, 2015). Such curricular reforms not only affect the students but also the teachers, who are now responsible for teaching programming to develop their students’ CT skills. Hence, it is critical to under-
stand the extent to which teachers understand and feel prepared to teach CT through programming and to identify their professional development (PD) needs within this domain. Hence, we address the following research questions:

1. How do primary school teachers understand programming and CT? How do they relate programming to CT?
2. What types of approaches do primary school teachers employ to develop their students’ programming competence and why?
3. To what extent do primary school teachers feel prepared to teach programming and/or CT and what are their professional development needs?

Theoretical background

**Definition of computational thinking (CT)**

Given the increased focus on CT as a vital area of competence in the last years, the number of definitions of CT and what it involves have increased. However, there is still no consensus on what the term exactly entails (Bocconi et al., 2016; Corradini et al., 2017). Wing (2011), who is considered a CT pioneer (Bocconi et al., 2016), defined CT as “the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent” (p. 1). Two aspects emerge from this definition. First, CT is considered a thought process, independent of technology (Wing, 2011). Second, as also noted by Bocconi et al. (2016), “CT is a specific type of problem solving that entails special ways of analysing problems which can be solved computationally and of developing solutions to them” (p. 10). Many definitions of CT are not as wide as that of Wing (2008). They mostly focus on the computer and the codes that the computer reads.

The Royal Society (2012) proposed a definition of CT that extended Wing’s definition to include aspects of both natural and artificial systems:

> Computational thinking is the process of recognising aspects of computation in the world that surrounds us, and applying tools and techniques from computer science to understand and reason about both natural and artificial systems and processes. (p. 29)

All these definitions, although different, agree that CT is neither only about computer science nor all about it (Wing, 2011). The different definitions emphasise particular elements of CT and skills related to it. For instance, Wing (2011) focused on abstraction, algorithm, automation, decomposition and generalisation as the core aspects of CT. Aspects such as automation, debugging, evaluation, simulation and analysis are also often mentioned as CT skills in literature (Bocconi et al., 2016).
Programming and CT
Programming is considered a part of CT; thus, CT is more comprehensive and broader than programming (Sun et al., 2021). Programming is understood as “the broad activity of analysing a problem, designing a solution and implementing it” (Bocconi et al., 2016, p. 21). Programming and coding are often interchangeably used and associated with the process of writing instructions for the computer to implement. However, programming includes more than simply writing the code, and is considered a tool for learning CT. In particular, problem analysis and problem decomposition are important elements of CT that are not included in coding (Bocconi et al., 2016, p. 21) and should therefore also be considered if the aim is to develop students’ CT skills. Wing (2006) emphasised this by stating, “Thinking as a computer scientist means more than being able to program a computer” (p. 33).

Implementation of CT and programming in curricula
Programming has been integrated into the national compulsory curricula in several countries (OECD, 2016). England integrated programming in its schools in 2014 and became a leading nation in focusing on computer science concepts, programming skills and CT (Bocconi et al., 2018). Other countries, such as Denmark, France, Finland, Ireland, Malta, Spain and Sweden, have also made CT and programming one of their key priorities for compulsory education (OECD, 2016; Vinnervik & Bungum, 2021; Yéves-Martínez & Pérez-Marín, 2019). There are mainly three approaches to implementing CT and programming in the curriculum: having a cross-curricular strategy, integrating them into existing subjects, and establishing a new subject (Bocconi et al., 2018).

In the Norwegian national curriculum, programming is being implemented as an add-on to existing subjects, namely, mathematics, natural science, arts and crafts, and music (NDET (Norwegian Directorate for Education and Training), 2020). This is being carried out through core elements (exploration and problem solving), basic skills (more specifically, digital competence) and concrete learning goals of the subjects (NDET, 2020). The implementation of the new Norwegian curriculum, Renewal of the Subjects, started in the school year 2020–2021.

Theoretical framework
As just mentioned, there are several frameworks aimed at outlining and detailing CT (Bocconi et al., 2016; Wing, 2011). In this study, we used the CT framework called the computational thinker (Figure 1; Csizmadia et al., 2015). Some of the key reasons for our selection of this framework are that it is well known internationally, aligns well with other existing frameworks (Bocconi et al., 2016; Hsu et al., 2018), is the basis of the CT model presented by NDET (2019) and has been
operationalised in the Norwegian curriculum. Thus, Norwegian teachers are expected to know this framework. Since this study analysed interviews with Norwegian teachers, it was important that the interviewer used familiar terms. Given that the Norwegian version of this model is simply a translation with no adaptations or revisions, we refer to it as shown in Figure 1. The English version of the model uses the term CT, whereas NDET uses the term algorithmic thinking (algoritmisk tenkning in Norwegian). In both the English and Norwegian contexts, algorithmic thinking has a narrower meaning than CT. Moreover, in both models, algorithmic thinking is included as a subskill of CT (Csizmadia et al., 2015; NDET, 2019). Although there is confusion in the use of the terms at several levels, in the Norwegian context, the educational authorities decided to use the concept of algorithmic thinking as synonymous with CT (Bocconi et al., 2018; NDET, 2019).

The framework computational thinker has two main areas: concepts (six sub-areas) and approaches (five sub-areas), which are considered important while working with students’ CT competence (Figure 1).

**Figure 1.** The Computational Thinker (Csizmadia et al., 2015)

**Concepts (CT subskills)**
The CT model includes six subskills: logic, algorithms, decomposition, patterns, abstraction and evaluation (Figure 1). Logic is defined as the ability to analyse and predict the result of a specific action. This allows students to use their own knowledge to draw their own conclusions. Decomposition is “a way of thinking
about artefacts in terms of their [components]” (Csizmadia et al., 2015, p. 8), in addition to understanding, solving, developing and evaluating the problems separately. Decomposition can make complex problems easier to solve mainly because it makes each problem less complex and previous knowledge easier to use. Moreover, abstraction involves removing unnecessary details from a task or a situation to focus on the most important parts of the problem and so make the problem easier to solve (Bocconi et al., 2016; Csizmadia et al., 2015; NDET, 2019). The fourth skill, pattern, involves spotting and using similarities (Bocconi et al., 2016; Csizmadia et al., 2015). Csizmadia et al. (2015) also used generalisation as a synonym for pattern and regarded it as a way of using parts of strategies used in former solutions in new and different solutions to problems. In algorithms, students make rules and step-by-step instructions (Csizmadia et al., 2015). The ability to think algorithmically involves coming up with a solution through a clear definition of the steps and through the ability to use the same step-by-step instructions every time the same problem occurs. The last skill, evaluation, refers to making judgements. “Evaluation is the process of ensuring that a solution, whether an algorithm, system or process, is a good one: that it is fit for [the] purpose” (Csizmadia et al., 2015, p. 7). This involves asking questions such as Are the answers correct? Is there any way to make the solution more effective?

Teaching approaches
Literature highlights several approaches that are advantageous for teaching CT (e.g., Scherer et al., 2020; Sun et al., 2021). The CT model denotes approaches that teachers can use to develop their students’ CT skills, such as tinkering, creating, debugging, persevering and collaborating (Csizmadia et al., 2015). When students are tinkering, they are developing CT skills by experimenting and playing. Creating refers to designing and producing an outcome, a product. Debugging refers to finding and fixing errors that occur within the activity or code. By identifying errors and correcting them through a problem-solving phase, students may well develop understanding that they did not hold before, and thus, potentially develop new knowledge. Persevering is an approach in which students “keep going”; they try one strategy for solving a problem and if it does not work, find another solution. Collaboration refers to working together as a team. The students share their thoughts and perspectives, and in so doing, can learn from each other. Collaborative programming has been identified as more efficient for developing students’ CT skills than individual programming (Scherer et al., 2020; Sun et al., 2021).

Literature review
In recent years, there has been a rise in publications focusing on CT. Many field studies have been conducted to investigate how to teach and learn CT (Hsu et al.,
Most of the studies focused on a certain way of teaching CT (e.g., an intervention) and evaluated how students responded to the approach. In a quasi-experimental study, Del Olmo-Muñoz et al. (2020) examined the impact of unplugged activities (programming without a computer) on the CT competence of 84 second-grade students. The results indicated that unplugged programming improved both students’ CT skills and their motivation to develop such skills. Similarly, Pérez-Marín et al. (2020) used pre- and post-tests to investigate the effect of (MECOPROG), a methodology that combines metaphors and Scratch programming, and found that MECOPROG is a useful approach for teaching computing concepts in primary school and for improving students’ CT skills. Özcan et al. (2021) gave yet another example of a randomised experiment on fourth-grade students who were randomly assigned to two different treatment groups to investigate the effects of learning to code on their CT, fluid intelligence and spatial orientation. Their findings showed that the students’ CT scores increased significantly only in the learn-to-code group. These relatively recent studies signified the effectiveness of different programming interventions on the development of students’ CT skills. However, the interventions (i.e., the teaching of programming) in these studies were conducted by specialists or trained teachers.

Some studies focused more specifically on teachers in primary and/or secondary schools, investigating their views on programming and CT. For instance, Hijón-Neira et al. (2017) conducted a survey among 46 teachers and found that while most of them had positive attitudes towards the integration of programming in the curriculum, 39% felt they were not capable of teaching programming. This was mainly because the teachers felt they lacked the competence needed to teach programming and they found it difficult to include programming in existing subjects. These findings were confirmed by Corradini et al. (2017) and Misfeldt et al. (2019). Furthermore, the teachers reported using Scratch or games as tools, but they emphasised that they lacked adequate methodology (Hijón-Neira et al., 2017).

Among the studies that focused on teachers, a few investigated how teachers understand CT. Umutlu (2021) conducted a qualitative study to gain an in-depth understanding of how 12 pre-service teachers’ understanding of CT changed across a seven-week course in CT. It was found that at the start of the study, the teachers did not have a deep understanding of CT. They described CT as “thinking and calculations done by computers” and argued that “computers must be used for CT” (Umutlu, 2021, p. 7). Yadav et al. (2018) investigated how nine elementary school teachers’ understanding of CT improved after a year-long development programme “from broad and generalised ideas to more elaborate versions of those ideas” (Yadav et al., 2018, p. 371). At the start of the study, the teachers often perceived CT as problem solving and/or as programming, and they also linked CT to logical thinking, algorithms and pattern recognition. After the development programme, the teachers still did not connect CT with programming,
yet their understanding of the connection between CT and problem solving expanded to include problem decomposition and algorithm efficiency (Yadav et al., 2018). Before the development programme, all the teachers described computational thinking as involving logical thinking. During the programme, their definition of logic and conditional logic (if-then-else) became more specific. Moreover, their conception of how CT involved using algorithms to solve problems also changed from generic to more specific. Before the development programme, the teachers linked efficiency to algorithms; but at the end of the course, they also linked it to abstraction and generalising. Corradini et al. (2017) also used a survey to examine similar issues in primary school teachers’ conceptions of CT. A total of 972 teachers were asked to define CT by completing the sentence “In my view, computational thinking is...” (p. 138) and by choosing their level of agreement with statements about CT. About half of the teachers included some of the fundamental elements of CT (problem solving, automation, algorithm, mental competence and informatics methods), but only 10.8% provided a complete definition, including decomposition, logical thinking, abstraction and coding (Corradini et al., 2017).

The present study
Despite the studies described, which mainly focused on teachers’ understanding of CT and attitudes towards it as part of a professional development (PD) programme, there is still limited research on how teachers understand CT and programming (Yadav et al., 2018). As most teachers in compulsory education in Norway are required to teach programming to develop their students’ CT skills, we believe that knowledge regarding their understanding of such concepts is important. This is a key to having the ability to design suitable PD programmes and to ensure that the aims of the curriculum are fulfilled. We aimed to fill this gap by conducting in-depth interviews with teachers of different subjects to investigate their understanding of programming and CT, how they teach these skills and their perceptions of PD needs in this domain.

Methods

Participants
Teachers across the country were asked through different channels (e.g., partner schools of universities, teacher education programmes at the largest universities and a web page) to volunteer for this study. The target group was grades 5–7 teachers of mathematics, science, music, and arts and crafts. The learning goals related to programming are described in these subjects and grade levels. Eight teachers joined the study, six of whom were female. Six taught mathematics and science, and two taught music and arts and crafts. Their teaching experiences varied, from being a novice to having more than 20 years of experience.
**Interviews**

In late spring of 2021, we conducted semi-structured interviews with the eight teachers using video conferencing due to the restrictions pertaining to the coronavirus disease 2019 (COVID-19) pandemic. However, only the audio of the interview was recorded with the prior approval of the interviewees. Thunberg and Arnell (2021) identified some benefits of online interviews, including reaching a larger geographical area and lower costs. Yet, they emphasised that some of the visual experiences and body language are lost to some extent with such interviews, and possible technical difficulties can pose a challenge to the interview. In our case, we did not experience technical challenges.

We developed an interview guide based on the five-step model presented by Kallio et al. (2016) that included reasoning for a semi-structured interview, applying theoretical background and knowledge, and formulating, piloting and presenting the interview guide. The pilot interview provided us with knowledge about the fit of the interview guide, apprised us of the time needed for the interview and gave us experience with the online platform. Only one researcher conducted all the interviews to minimise differences between them. However, in the first four interviews, another researcher participated as an observer to gain a common understanding and overview of the interview data. Immediately after the interviews, both researchers noted their reflections.

The interview guide included three main areas: (1) attitudes and perceptions, focusing on the new curriculum and the teachers’ attitudes towards it, particularly on the new elements related to 21st-century skills; (2) understanding of programming and CT and the relationship between the two; and (3) teaching approaches and PD. The teachers were asked open-ended questions and given time to reflect. After introducing the questions related to the teachers’ understanding of programming and CT, they were also shown the concepts in the CT model (Figure 1) and asked to further elaborate their understanding of these. This was done to reduce bias in our findings, as the teachers might simply not have recalled these concepts. Thus, they were given an opportunity to further describe their knowledge. The interviews were transcribed and analysed using NVivo software.

**Data analysis**

The thematic analysis approach (Braun & Clarke, 2006; Patton, 2015) was used to analyse the transcribed interviews to identify patterns and themes. We followed the six-step model introduced by Braun and Clarke (2006), which includes the following phases: data familiarisation, generating codes, searching for themes, reviewing themes, defining themes, and producing reports.

A coding guide was developed based on the underlying theoretical framework. However, both an inductive approach and a deductive approach were used to refine the coding guide through an iterative process. First, we identified themes based on our theoretical knowledge (e.g., from our literature review, underlying
framework and interview guide). Second, we identified themes from the data. In particular, within the category approaches and PD, we identified new themes. The data were analysed through multiple reviews of the data and coding, with the aim of moving from raw data to the development of preliminary themes and sub-themes (Braun & Clarke, 2006).

One researcher coded all the interviews so that the analysis would be homogeneous. However, the first interview was coded twice to check the validity of the analysis. It showed a high correlation between the first coding and the second coding of the interview. Furthermore, the post-interview reflection notes were used to crosscheck the preliminary results. During the several rounds of data analysis, the authors discussed and resolved inconsistencies (Corbin & Strauss, 2007).

Results

Our analysis identified 18 subthemes that we merged into four main categories: teachers’ attitudes, understanding of CT and programming, teaching approaches, and PD.

**Attitudes towards the new curriculum and programming**
The interviews were initiated by asking the respondents about their thoughts regarding the new curriculum and the integration of programming into their respective subject domains. This question was intended to open the conversation and invite the teachers to share their thoughts and attitudes towards the new curriculum, particularly programming and CT.

In general, the teachers thought positively about the new curriculum, particularly about the collaboration between teachers in implementing it and the new elements in the curriculum. Mentioned as challenges were the COVID-19 situation and the various requirements to realise its full implementation. With regard to programming, the teachers were generally positive. One teacher mentioned that introducing programming was hard because she was not so interested in it.

The teachers were also asked to rate their own competence in programming compared to that of their colleagues. All of them rated themselves as being at the same or a higher level compared to their colleagues.

**Teachers’ understanding of programming, CT and the relationship between the two**

**Teachers’ understanding of programming**
The teachers were asked about how they understood the concept of programming; see Table 1.
Table 1. Frequency and number of teachers presenting the different CT skills when asked about the concept of programming and CT

<table>
<thead>
<tr>
<th>CT skills</th>
<th>No. of teachers</th>
<th>Freq.</th>
<th>CT skills</th>
<th>No. of teachers</th>
<th>Freq.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstraction</td>
<td>CT</td>
<td>3</td>
<td>5</td>
<td>Logic</td>
<td>CT</td>
</tr>
<tr>
<td></td>
<td>Subject</td>
<td>2</td>
<td>2</td>
<td></td>
<td>CT-subj.</td>
</tr>
<tr>
<td></td>
<td>PG</td>
<td>3</td>
<td>3</td>
<td></td>
<td>CT-prog.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PG</td>
</tr>
<tr>
<td>Algorithms</td>
<td>CT</td>
<td>5</td>
<td>15</td>
<td>Pattern</td>
<td>CT</td>
</tr>
<tr>
<td></td>
<td>CT-subj.</td>
<td>3</td>
<td>12</td>
<td></td>
<td>CT-subj.</td>
</tr>
<tr>
<td></td>
<td>CT-prog.</td>
<td>4</td>
<td>5</td>
<td></td>
<td>CT-prog.</td>
</tr>
<tr>
<td></td>
<td>PG</td>
<td>7</td>
<td>37</td>
<td></td>
<td>PG-subj.</td>
</tr>
<tr>
<td>Evaluation</td>
<td>CT</td>
<td>2</td>
<td>2</td>
<td>Decomposition</td>
<td>CT</td>
</tr>
<tr>
<td></td>
<td>Curriculum</td>
<td>1</td>
<td>1</td>
<td></td>
<td>PG</td>
</tr>
<tr>
<td></td>
<td>PG-subj.</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PG</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: CT = computational thinking, PG = programming, CT-subj. = CT linked to subject, CT-prog. = CT linked to PG, PG-subj. = PG linked to subject

The teachers mainly used terms related to algorithms when describing programming (Table 1). Some of these terms were order, change formulas, connection, algorithms, changes, press two arrows, get from a to b, recipe, instructions for use, make, work, jump over a step, make a road/path, structure, build, step by step, instruction and construct. Some teachers explained their understanding of programming by describing examples from their classroom teaching. For instance, two teachers exemplified their thoughts of programming by describing the use of physical artefacts such as LEGO and microrobot. “We could actually say it’s a kind of LEGO...,” Teacher 7 (T-7) said.

Another teacher said, “We have these microbots [that] we can use, and we use them to see how this robot gets from a to b just by touching three arrows forward, two arrows up and one arrow left” (T-6).

For other aspects of programming, the teachers gave narrower descriptions. For logic, they talked about logical thinking, logic in practice or a logical setting. “In practice, programming is logic, I believe. It is like I have the number 1, then I have to [make] a function, then I get a result. It is about having an understanding of functions” (T-2). Furthermore, this teacher described the connection between music and programming by harmonic changes, accompaniment, and melody; thus, logic was described in a setting of music as the subject.

For other subskills related to programming—pattern, evaluation, abstraction and decomposition—the level of detail of their explanation was low. The only connection between programming and subject domains was made in the context of the subskill pattern (Table 1):

If we think about the geometry part in mathematics towards arts and crafts, where you should use patterns, rotation, mirroring, as a part of programming in arts and crafts, then
I think it strengthens the understanding and terminology in the mathematics, or vice versa. (T-1)

**Teachers’ understanding of CT**

The teachers were asked about how they understand CT. Note that the Norwegian term for CT is *algorithmic thinking*, as described in the Theoretical framework section. The teachers’ first reflections on CT showed hesitation and some variations in how they understood this concept. This is illustrated by the following first thoughts on CT by five of the teachers:

The algorithms are the codes […] behind [the program]. So, I have heard about [them]. (T-3).

Yes, that’s a typical word I’ve heard at the University; maybe I have to dig a little deeper. To me, CT, algorithm, I do associate [with] something linear, an equation, maybe restate equations, to me that’s CT. (T-2)

I have heard about it but do not remember what it involves. (T-6)

If you think like algorithmically, then I think of recipes … yes, it’s a little like programming, I think. (T-5)

My first thoughts are patterns and systems... I associate algorithms with a special recipe, which can have different designs and instructions. But putting it into [a] system in a given order is what I associate with CT. (T-1)

Teacher T-4 constructed her/his own definition of CT:

I have heard about the term, and I must admit, I have not had any training or any explanation of the term, but it has been used and then I have made myself an opinion. I know what an algorithm means, and I know what thinking means. So, I just maybe formed my own perception. (T-4)

Based on these first six explanations of CT, it is clear that the teachers did not define or understand CT well, and we can assume that even though the term is used in the curriculum, it was not commonly discussed in the schools or classrooms.

As shown in Table 1, algorithms and patterns were the most frequent subskills mentioned, as observed for programming. No further explanation or elaboration was provided.

The description of algorithms related to CT was similar to what was observed for programming. To illustrate the associations between CT and programming through the term *recipe*, T-5 said, “You make a recipe [for] how the car should drive from here to there, and you program it. It is the recipe [for] how [the car] should move from *a* to *b*…” (T-5).

Next, we presented the CT subskills (Figure 1) to the teachers to validate what we observed initially in the process providing more descriptions of the CT subskills and their possible connections to programming. However, even after
being presented with the CT model, the teachers had limitations in describing the CT subskills and CT, and their understanding of CT remained thin.

**Connecting programming to CT**

One of the teachers connected CT and problem-solving to programming (and this was before we asked the related question):

... well, through programming, one can definitely learn skills that are needed [in] mathematics, problem-solving, and CT is something they can learn by using programming, which can be useful in other parts of mathematics. (T-8)

Teacher T-8 also showed connections to different CT skills (Table 2). The other associations to CT that he mentioned were reasoning and reflection, which may be linked to problem-solving strategies.

**Table 2.** CT skills and associations with CT by respondent T-8

<table>
<thead>
<tr>
<th>CT skill</th>
<th>Quotations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm</td>
<td>“Solve the code.”</td>
</tr>
<tr>
<td></td>
<td>“Get the code to do what I want.”</td>
</tr>
<tr>
<td></td>
<td>“Be exact.”</td>
</tr>
<tr>
<td></td>
<td>“Work specifically on what they in fact want to achieve with the code.”</td>
</tr>
<tr>
<td>Decomposition</td>
<td>“A big problem they are solving, then they split it up into smaller and smaller parts which are manageable.”</td>
</tr>
<tr>
<td>Generalisation/</td>
<td>“Then they can try to solve the bigger problem which they meet, and this is something they meet in the work with [a] fraction or any theme.”</td>
</tr>
<tr>
<td>Pattern</td>
<td></td>
</tr>
</tbody>
</table>

This teacher highlighted personal interest as a main reason for his competence in programming and CT, as well as some training in programming during his education a while back and his teaching of these competences in the classroom. The rest of the teachers in our sample struggled to make this connection and generally repeated their understanding across algorithms related to programming and CT.

**Teaching Approaches**

The third main category in the interviews was concerned with how the teachers facilitated programming and CT in their classrooms. Table 3 shows an overview of the teaching approaches identified in the analysis. During the first readings, *collaboration* and *interdisciplinary* were the defined codes, whereas the remaining four were identified from the data.

**Table 3.** Overview of teaching approaches related to CT and programming

<table>
<thead>
<tr>
<th>Approaches</th>
<th>In-depth learning</th>
<th>Tinkering &amp; creating</th>
<th>Perseverance</th>
<th>Debugging</th>
<th>Collaboration</th>
<th>Interdisciplinary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No. of teach</td>
<td>Freq.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Teach</td>
<td>5</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PG</td>
<td>5</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Teach</td>
<td>2</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PG</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CC</td>
<td>6</td>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Note:* Teach = teacher, PG = programming, CC = curriculum, Freq. = how many times the approach is registered.
The interdisciplinary approach is related to programming activities that contribute to transferable skills between the subject domains and programming. Furthermore, CT tasks and activities are used across several subjects to create multidisciplinary or interdisciplinary tasks. Some examples in which the transferable skills were mentioned included reading and writing skills, construction of dance, graphic design, solving problems in mathematics and tools for the connection between mathematics and arts and crafts in geometry.

Because it is like this, we try to make the students think of Norwegian and Math, and English too. That is, what is really the content, and the coding language is relatively short, ehm, such that it is quite easy to see the solution regarding where you need to make changes to find the answer in a way. Ehm, so I think it is very valuable and has a transfer value to other subjects. (T-6)

Tinkering and creating are two different approaches in the literature; however, we combined them since they are related in terms of what they connote and because the mentions of these approaches were rather scarce during the interviews. Tinkering and creating were identified through the use of key terms such as strategies, to explore, learn skills, training strategies, understand why and testing out. Below are examples of tinkering and creating:

If you master the subject and start to become a decent coder, then it is the creativity you need for shaping and developing the things you want, and to get good ideas …. these lightning strokes, […] it was something I sat and tried to [fiddle] with [but] notably, without succeeding. (T-7)

Perseverance was a working approach that one teacher particularly mentioned as related to persistence, patience and “hanging in there”:

That is maybe why I sit with this at home and get muddy myself. But it requires that you hang in there. One feels it; it is a little interesting that it has a transference value for the kids too. When the adults are learning something new, that frustration to be patient and hang in there when you do not succeed at once. […] I think everyone benefits from such experience along the way, because this is how the kids experience it at school. (T-1)

Two teachers also touched upon debugging. They linked it to debugging processes, that is, to identifying and correcting errors and/or changing your approach to a task and starting over. In the debugging process, an assessment of the solution is essential to be able to evaluate what steps to take further. This emphasises the close connection between debugging and evaluation. Teacher 4 said: “How do your actions provide results, how do they give you the wrong result, and how do we debug? This can actually relate to surprisingly much” (T-4).

Collaboration, as mentioned in our data, had a teacher focus and not a student focus. Thus, collaboration was mainly described to illustrate how the teachers work together, help each other and share resources and materials. In general, this concerned how teachers shared ideas and materials and worked in teams. Surprisingly, collaboration, as a teaching approach that focuses on teaching
strategies such as students working in pairs or groups and not individually when programming, was not identified in our analysis. In-depth learning was identified in our analysis, though, and was mainly related to working holistically with the curriculum, and thus, focused on deep learning.

While the teachers’ approaches to teaching programming seemed fairly limited, several tools were specifically mentioned, including Python, Scratch, Minecraft, Picxelart, Micro:bit, Bit:bot, Microbots, Bluebot, Campus Inkrement, Minecraft, Cappelen Damn, Osmo, Salby, Lego, Blocks, kidsakoder.no and Kikora. Programming temperature sensors and calculators were also mentioned.

**Professional development (PD)**
One of the teachers who rated him/herself as holding high programming competence elaborated:

I have around 35 colleagues at work. I think I am […] unfortunately in many ways one of the 3 who can easily, with calmness and confidence, teach programming. We may be six to seven teachers who have tried many times in the lessons. (T-4)

We also observed large variations in programming skills among the teachers: “I do think it is more learning when the students teach me. It is said that we learn 90% from what we teach others” (T-2). Another teacher said:

We did not have any training or anything like that. I just ask the others, then they tell [me I] can try [something] out like this. Then, I just leave it to the students. I have done that a bit, yes. These program[s] are self-explanatory, right. (T-5)

Other reflections about competence and PD concerned personal interest, substituting classes and teaching programming, asking own students for help, and teachers with high skills training other teachers and serving as “teaching specialists” (teachers who take a lead role in the school for developing a specific competence, subject domains or basic skills, e.g., digital competence, reading literacy and numeracy). The teachers observed other interesting features that emphasised students’ competence and what the contribution of programming could be in this “new era” in the school. For example, “It’s especially boys who are more engaged in coding; they develop their own games and such things. It would be […] nice […] to include boys who don’t experience mastery in school in other ways” (T-2). Pedagogical skills in the classroom were also reflected on:

There absolutely could have been a larger focus on how you can use it [programming] in practice in teaching, but I believe the biggest challenge is the lack of adaptive teaching to fit the [students at] different levels at which they are taught. (T-4)

The teachers’ PD in programming skills and pedagogical skills related to programming varied significantly, from no training to a 40-hour training course. However, most teachers had experienced a one-day course. Personal interest and competence packages accessible through NDET were other sources of self-regulated PD in this field. The teachers emphasised a need for more training
related to programming competence and the development of pedagogical and didactical practices to facilitate teaching programming.

The teachers also accentuated the lack of time and teachers’ limited access to resources and training. These were associated with economies in the municipalities and schools, as well as with the few teachers and resources available for teaching programming. Resources for teaching programming were related to programming competence training and the development of pedagogical and didactical practices. Training could be offered by learning specialists at their respective schools and by external trainers and colleagues, and by assigning them tasks related to programming.

Discussion

Teachers’ understanding of CT and programming

The teachers in our study were generally positive towards the new curriculum and the integration of programming into the subjects. Positive attitudes to introducing programming and CT in the curriculum were also identified by Hijón-Neira et al. (2017). Furthermore, our results showed that the teachers’ understanding of programming and CT was narrow and limited, which is also in line with previous research (Umutlu, 2021). The teachers in this study provided a narrow description of CT, even though they rated themselves at the medium to upper competence levels. CT involves a broader perspective than programming (Bocconi et al., 2016); thus, we expected at least more diverse descriptions of CT. The teachers described CT and programming similarly and mainly focused on algorithms. However, algorithms, described as rules or step-by-step thinking (Csizmadia et al., 2015) constitute only one part of CT and programming. Previous studies support our findings that teachers tend to connect CT mainly to algorithms (Corradini et al., 2017; Yadav et al., 2018). Umutlu (2021) indicated that the teachers lacked a deep understanding of CT before the start of a seven-week PD course. Corradini et al. (2017) showed that only 10% (N = 972) of primary school teachers were able to completely define CT. As shown in the Theoretical framework section, the descriptions of CT in the Norwegian curriculum include more than algorithms. A possible explanation for teachers’ narrow understanding of CT may be, as Umutlu (2021) explained, their association of programming and CT mainly with coding, which leads to their frequent focus on CT as “thinking and calculations done by a computer” (p. 7). Another possible explanation is that the choice of terms in the Norwegian curriculum introduced another layer of confusion. In our data, we observed that the teachers hesitated to explain CT and often started by reflecting on the “everyday” words algorithm and algorithmic thinking rather than on the concept of algorithmic thinking.

In summary, given that the teachers in our study rated their programming competence at the middle or upper level yet were not able to explain the concept,
the teacher training with regard to CT must be considered behind schedule. Vinnervik and Bungum (2021) investigated, using text analysis, how and in what contexts Swedish and Norwegian curricula embody aspects of computational thinking and programming. They showed that a fragmented approach to the wider notion of computational thinking was taken in the Norwegian curriculum, and “only parts of the practices defined in the CT framework used could be explicitly identified” (p. 1). They concluded that “the time and teacher competences required for providing students with functional skills in programming as a tool [are] under-communicated and [leave] schools and teachers with major challenges in realising the curriculum” (p. 4).

We also observed that the teachers were not able to make a reasonable connection between programming and CT, except for one teacher. This teacher explained and described well the linkage between these concepts, and rated his/her own competence at an upper level compared to colleagues. Personal interest and formal education within these domains, combined with practicing teaching programming in the classroom, were factors that seemed to explain this comprehensive understanding of CT. These findings are different from those on the teachers in the study of Yadav et al. (2018), where the link between CT and programming was more solid.

Understanding of CT may also depend on the teacher’s professional background and subject domain. We expected teachers in mathematics to describe and understand algorithms differently from teachers in music. Learning goals related to programming are distributed across the four subject areas (i.e., mathematics, science, music, and arts and craft) in the Norwegian curriculum, and researchers have noted that “it appears unclear to what extent and in what subject children should learn the fundamentals of programming” (Vinnervik & Bungum, 2021, p. 1).

**Teaching approaches**

Csizmadia et al. (2015) provided an overview of the working methods or approaches contributing to learning CT in school. The teachers in our study highlighted only a few approaches, which could mean that they had not fully developed their pedagogical and didactical knowledge to teach programming. This assumption is supported by the teachers’ indication of their PD needs for this domain.

Moreover, the approaches to programming and CT used in the classroom most probably reflect the teachers’ knowledge of these concepts. This knowledge varied among the teachers in this study, and we clearly observed that the pedagogical and CT knowledge was not always present. One teacher, for instance, mentioned relying on students’ help to learn programming. Hijón-Neira et al. (2017) argued that teachers need training to develop sufficient knowledge to teach programming and CT. The general needs for training related to programming and pedagogical skills that the teachers described, could indicate that they are not
comfortable with their own level of knowledge. This is similar to what other studies found (Corradini et al., 2017; Hijón-Neira et al., 2017; Misfeldt et al., 2019).

An overall picture of working methods revealed a large variation in the programming tools mentioned by the teachers. Scratch and Python are tools described in competence packages provided by NDET. Many of the tools mentioned by the teachers are games, which are similar to what the teachers in the study of Hijón-Neira et al. (2017) used. This variety of tools offers many approaches to teach programming; however, it seemingly comes without training, and the teachers are left alone concerning their PD.

**Professional development**

Introducing CT in the curriculum significantly increases teachers’ need for PD in this domain (Bocconi et al., 2016). Large variations in CT training and programming skills, in addition to pedagogical aspects, were observed in this study. All the teachers agreed that more training is needed, which corresponds with the PD needs described by Bocconi et al. (2016) and “the need for a solid infrastructure scaffolding the curriculum changes and supporting teachers’ programming knowledge acquirement” (Stigberg & Stigberg, 2020, p. 494). However, the teachers also emphasised lack of time and resources as the main reasons for their low participation in PD courses and training. Resources include, for example, economic support and availability of substitute teachers while the main teachers attend such courses. Self-development during one’s leisure time and cultivation of personal interest in CT and programming were also described as possible ways to develop one’s own skills in these domains.

The current introduction of CT and programming in the national curriculum involves a wider range of teachers and subjects compared to the previous curriculum that merely offered electives in programming. Learning specialists teaching programming in all classes and subjects were mentioned as a solution; however, this does not seem to match the wide-ranging intentions of the curriculum.

While the teachers in this analysis categorised themselves as having a medium or upper level of programming skills compared to others, we believe there is a large group of teachers who possess a lower level of knowledge and pedagogical skills related to CT and programming. These views and the perspective that some teachers rely on their students to help them acquire programming and CT competence, certainly suggest that a more comprehensive, inclusive and detailed PD programme could be relevant for many teachers in Norway (and internationally), as these findings are reflected in other studies. Internationally, the curricular developments seem to be heading in the same direction, thus making our findings relevant for a broader audience.
Limitations and future directions

Although our study is one of the first to comprehensively investigate teachers’ understanding of CT and programming through in-depth interviews, some limitations need to be considered. First, the sample consisted of few teachers who represented the four subjects, especially music and arts and crafts. Hence, we suggest that future studies investigate a larger number of teachers and subject domains. Moreover, comparative studies of the teachers’ needs for PD across these subject domains could be valuable.

Our sample focused on teachers who viewed themselves as being at the intermediate and upper levels of programming competence. Further studies may investigate how teachers at low or high levels of programming competence understand and teach programming and CT. Future investigations should also focus on both teachers’ subject knowledge and programming competence to provide further insights into how PD programmes could be established. Moreover, in this study, the teachers volunteered to participate, which might have affected our results regarding their views or above-average interest in our topic. However, for the purposes of this study, we do not perceive this as a disadvantage, but we again recommend that future studies focus on a larger pool of teachers. For instance, a more quantitative study could investigate the relationships between teachers’ understanding of CT and programming, their didactical knowledge (teaching approaches) and their self-efficacy.

Conclusion

This study investigated teachers’ knowledge of programming and CT and their competence in teaching such subjects. Our findings showed limitations in the teachers’ understanding of programming and CT and in their pedagogical and didactical competence to teach these concepts. They clearly stated their need for more training to adequately teach the subjects and thus, gain more self-confidence. Indeed, in the new curriculum, there are some assumptions regarding what is required for teaching programming to enhance students’ CT skills, but such assumptions do not reflect the teachers’ reality and their PD needs to be able to fulfil the curricular goals. Thus, based on our findings, an increased focus on training for primary school teachers in this field is required.
Author note

The authors confirm that all relevant research ethical requirements have been met and received ethics committee approval from the NSD – the Norwegian Centre for Research Data.

This research was supported in part by a grant from the Research Council of Norway (NFR-FINNUT Grant 283520 “TEACH21st”).

About the authors

Reiar Kravik is assistant professor in teacher education at the University of South-Eastern Norway. His interests lie within the fields of teaching and learning of programming, computational thinking and 21st century skills in the teacher education.
Institutional affiliation: Department of Mathematics and Science Education, University of South-Eastern Norway, Lærerskoleveien 40, 3679 Notodden, Norway.
Email: Reiar.Kravik@usn.no

Tonje Karoliussen Berg is assistant professor in teacher education at the University of South-Eastern Norway. Her research includes teaching and learning of computational thinking skills in primary school.
Institutional affiliation: Department of Mathematics and Science Education, University of South-Eastern Norway, Lærerskoleveien 40, 3679 Notodden, Norway.
Email: tonje.k.berg@usn.no

Fazilat Siddiq is professor in the field of digitalisation of education at the University of South-Eastern Norway. Her research includes teaching and learning of digital competence and 21st century skills, curriculum developments and new technologies, and educational quality.
Institutional affiliation: Unit for Digitalisation and Educational Quality, University of South-Eastern Norway, Papirbredden – Drammen kunnskapspark, Grønland 58, 3045 Drammen, Norway.
Email: fazilat.siddiq@usn.no
References


Bocconi, S., Chiocchariello, A. & Earp, J. (2018). The Nordic approach to introducing computational thinking and programming in compulsory education. Report Prepared for the Nordic@BETT2018 SteeringGroup, 42. [https://doi.org/10.17471/54007]


NDET (2020). Læreplaner [Curricula]. Norwegian Directorate for Education and Training. [https://sokeresultat.udir.no/findlareplan.html?fltypefiltermulti%3D%26fltype%3D%252520Kunnskapsl%25C3%25B8fet%25202020]


https://doi.org/10.1080/08993408.2018.1560550