Design for learning programming
Approaches taken by novice learners

Abstract
Programming recently became mandatory in Swedish compulsory schools (age group 7-16); this article contributes to our understanding of novice learners’ different approaches when programming. The study builds on observations and informal conversations during programming lessons at three Swedish science centres. At the time of the study, science centres already had experience of programming education as they offered courses for pupils. This was used as a complement to teaching at compulsory school. We apply a design-for-learning perspective to help us understand how contextual aspects influence what novice pupils do and how they design their learning during programming lessons. During the analysis process, we combine thematic analysis with the Learning Design Sequence (LDS) model. We identify five qualitatively different approaches that pupils take to solve programming problems: mathematically, trial and error, step-by-step, routine as well as aesthetic. Each of these approaches allows pupils to use and practice different abilities that are important for programming. We discuss how these abilities can be compared to computational thinking (CT). The study provides an insight how pupils are involved in designing their own learning when using their abilities to solve programming assignments.
INTRODUCTION

Teaching programming has been mandatory in Swedish compulsory schools in the subjects of mathematics and technology since autumn 2018 (Regeringskansliet, 2017). This means that compulsory schools are supposed to provide all their pupils with the opportunity to learn programming. Learning programming is a wide field; many studies examine the needs and challenges of teaching and learning programming at different levels of the education system (e.g. Kalelioglu, 2015; Xia & Zhong, 2018). In this paper we analyse how pupils, when designing their learning in programming, use abilities which can be compared to those used in computational thinking (CT) (Cetin & Dubinsky, 2017; Wing, 2006). This term (CT) includes various types of abilities, such as problem-solving and abstract thinking, that may be necessary for a person who is going to program (Durak & Saritepeci, 2018).

Design for learning is used as a theoretical framework (Kress & Selander, 2012; Selander, 2008, 2017) to contribute to a systematic understanding of how contextual aspects such as physical surroundings, software and instructions influence the ways in which compulsory school pupils design their learning in programming. The design for learning perspective provides insights into what happens when pupils learn and how possible learning options, choices and decisions are constructed in certain contexts (Kress & Selander, 2012). Specifically, we apply a learning design sequence (LDS) model (Selander, 2008, 2017) to analyse programming lessons at Swedish science centres (SC). This model focuses on the components that influence a learning sequence and the processes that structure the learning (Insulander et al., 2017).

In the study it is assumed that learning is a consequence of the pupils learning design, as well as of the planning of programming lessons and the curriculum. The aim is to contribute knowledge about the learning process when novice compulsory school pupils are learning programming.

The research questions about compulsory school programming education at Swedish science centres (SC) reflect both a descriptive and an analytical perspective. The first question is purely descriptive; it aims at understanding what happens during the lessons observed: What are the characteristics of the learning processes when novice learners are programming?

The next questions are more analytical and follow up the first question:
What abilities important for programming, appear in the learning processes?
How can the abilities that pupils use be analysed and compared to concepts of computational thinking (CT)?

We analyse and describe the pupils’ learning process in the specific situation at SCs. In this way, the study contributes to an insight on how pupils are involved in designing their own learning when using their abilities to solve programming assignments. The results are contextualised within a historical situation where, for the first time, programming will be taught to all Swedish compulsory school pupils.

PROGRAMMING EDUCATION IN SWEDEN

In Sweden, programming became part of the mandatory learning content in mathematics and technology in 2018 (Skolverket, 2019); at the time of data collection for this study, the curriculum had not yet been revised. Programming was introduced in (some) Swedish schools as early as the 1970s and 1980s in the wake of computers and digital technology becoming a central tool in school education (Rolandsson, 2015). Samuelsson (2014) claims that there is digital inequality among Swedish young people at school, regarding both equipment and skills. Educational work in schools has not developed at the same pace as the purchase of digital equipment (Skolinspektionen, 2012). According to the Swedish Schools Inspectorate’s report of 2019, many teachers have expressed willingness to learn more about how they can develop their teaching of programming. However, there are still schools where only certain pupils are taught programming, depending on the teacher they have (Skolinspektionen, 2019).
Many teachers do not know where to start or what level of teaching in programming they should present. These teachers might need support, and in this context SCs are relevant because most of them offer courses in programming for compulsory school pupils (Svenska Science Centers, 2018). An SC is an interactive institution, whose purpose is to raise the public’s engagement and knowledge about relevant subjects related to science and technology (Achiam & Sølberg, 2016). To use SCs as a resource for teaching programming in compulsory school is in line with Dawson’s (2014) claim that utilising informal learning, such as that available at SCs, is an important opportunity for increased access to and equality in science education.

PROGRAMMING FOR NOVICE LEARNERS

Worldwide there is increasing interest in programming education at all levels of the education system (e.g. Qian & Lehman, 2019; Vieira et al., 2019). Learning programming at an early age can show pupils examples how they can use programming in the future when they will not only be consumers of digital technology but also possibly producers (Kalelioglu, 2015). Several studies have concentrated on how novices learn computational concepts and basic programming, in a research review Xia and Zhong (2018) compiled 22 empirical studies on teaching and learning robotics. A common focus in this research is the use of specific pedagogical tools such as LEGO robots. The review shows that studies of robotics in teaching are often quite limited; they also include different ages of pupils, from preschool to high school. It is therefore difficult to draw general conclusions about what this teaching provides (Xia & Zhong, 2018).

An example of this difficulty, to show the best way to teach and learn programming, emerges in a study by Lykke et al. (2015). Lykke et al. (2015) uses three different models: traditional teacher-led teaching with text-based code, problem-based teaching and problem-based teaching with visual programming using LEGO robots. The results show, among other things, that using the robots was engaging and motivating. However, robot teaching was also experienced as frustrating, partly due to practical problems. Above all, the students felt uncertain about whether they learned the right things when learning programming in other ways than the traditional ones. Teaching with text-based code, however, made the students feel more confident about theoretical understanding. At the same time, they experienced more stress, due to their limited experience of programming and the decreased possibility to collaborate with other students. Overall, the study showed that none of the teaching models were completely satisfactory, all three had advantages and disadvantages (Lykke et al., 2015).

Pupils may also affect their own learning, depending on how they communicate and how they handle the programming problems. An interesting study examines how ideas for problem-solving in mathematics are spread when 20 pupils in Grade three used programming (Harlow & Leak, 2014). Harlow and Leak (2014) analysed in several steps how the pupils talked about their programming solutions. The results show that when the pupils shared experiences and discussed different solutions, their problem-solving ability in mathematics developed and the programming solutions became more advanced (Harlow & Leak, 2014).

The advantage of giving pupils opportunity to experiment with programming problems in their own way was something Turkle and Papert identified as early as 1990. They argued that allowing different ways of thinking and permitting problem-solving in different ways can make programming more accessible to more people (Turkle & Papert, 1990). With a number of examples, they showed how pupils used different ways of solving programming problems. The problem-solving methods had distinctive features: abstract, systematic and distanced or concrete and with a sense of closeness (Turkle & Papert, 1990). Chao (2016) confirms that students can come up with adequate programming solutions using different strategies. When college students were given free rein to solve programming problems, their strategies differed as to how they chose to use programming blocks, for example to repeat code blocks or by trying out new combinations. In this way, Chao (2016) identified four strategies,
Sequent, Selective, Repetitious and Trial, all of which had different pros and cons for an effective and safe solution to programming problems. Solving problems in different ways is an important part of learning programming.

Computational thinking
An influential research trend regarding teaching and learning programming even for novices concerns computational thinking (CT). The term CT is not used in the Swedish curriculum (Skolverket, 2019a). It has however, been proposed as a way of developing teaching methodology in programming (Rolandsson, 2015), and is frequently used in the debate about programming education in Sweden (CETIS, 2017; Heintz et al., 2016). Wing (2006; 2008) has described that CT should be an integral part of educating young people, as important as reading and writing. CT is a way of thinking that involves the formulation of problems and how to find solutions (Cetin & Dubinsky, 2017). In a research review, Buitrago Flórez et al. (2017) compiled the results of 92 studies on programming education at different ages. These studies deal with different types of programming tools and identify challenges and benefits in several of the methods. Their conclusion is that teaching programming should focus on different types of abilities that can be related to CT. In their opinion pupils should be introduced to CT already in elementary school because it benefits programming on several levels (Buitrago Flórez et al., 2017).

Wing wrote in 2006 that CT “involves solving problems, designing systems, and understanding human behavior” (p. 33). Since then, researchers have made slightly different interpretations; CT is a field that is still being examined in both research and practice; it does not yet have a precise, coherent definition (Lockwood & Moony, 2018). However, CT can be summarised as a collection of concepts of abstraction, algorithmic thinking, problem-solving, decomposition, generalisation and debugging (Durak & Saritepeci, 2018). CT is used to break down problems, find patterns and create and use algorithms and abstractions (Heintz et al., 2016; Wing, 2006). CT can also involve debugging, collaborating, creating and preserving even when problems arise (Mannila, 2017). It can also mean learning in the presence of uncertainty (Wing, 2006).

THEORETICAL FRAMEWORK: DESIGN FOR LEARNING
Design in a learning context has many different functions. It can be for one’s own learning or for other people’s learning (Levinsen & Sørensen, 2019). Selander (2007) claims that teaching with digital technology involves a dual didactic challenge because the design needs to include both the digital tool and the subject content at the same time. Design for learning as a theory has proven useful in various contexts. The theory has previously been used in connection with teaching supported by digital equipment such as computers and tablets (Kjällander, 2011; Åkerfeldt, 2014) and in both formal education in schools and more informal learning environments (Insulander et al., 2017; Selander, 2017). In this study, the framework of design is used theoretically and analytically to primarily interpret the characteristics of novices’ learning processes. The focus will thus be on design as a process (Levinsen & Sørensen, 2019).

Within the theoretical framework, one way to analyse learning is to use a learning-design sequence (LDS) model (Kjällander, 2011; Selander, 2017; Åkerfeldt, 2014). This model offers an opportunity to analyse both the physical conditions, including learning resources, and the institutional norms of the teaching (see Figure 1). It provides a structure for analysing the activities in learning sequences. The LDS model consists of three elements: the setting, the primary transformation unit and the second transformation unit (see Figure 1). According to this model, learning takes place over time and is transformed and shaped by pupils within the context in which they are located (Selander, 2008, 2017).
The first part of the model, the setting, includes the conditions and purpose of the learning situation. The setting can be used to describe and analyse potential resources and how they link to the curriculum (when applicable), the context and the institutional norms. Resources in this context consist not only of material and digital equipment but also of teachers and peers and the specific physical environment. The norms may differ as the view of knowledge might be different depending on the context.

The primary transformation unit can be used to analyse the types of activities pupils undertake and what actual resources they use, including simple conversations, when transforming and shaping their knowledge. During this transformation, pupils process and combine the types of activities and information and form their own knowledge. The second transformation unit can be used to analyse the presentations and representations of knowledge. This unit includes meta-reflections and theoretical discussions about solving the problems posed. The process is not linear and can move back and forth between the primary and second transformation unit (Selander, 2008, 2017). The entire model can be used to analyse all types of learning and is also suitable for studying learning processes from the pupils’ perspective.

**METHODOLOGICAL CONSIDERATIONS AND ANALYTICAL PROCEDURES**

This study was conducted within a qualitative research tradition combining two ways of collecting data: observations of programming lessons (main method) and informal conversations (short interviews) during these lessons. By using observation as a data collection method, the observer gained insight into pupils’ behaviour during lessons (Loseke, 2017). Through conversations we added an opportunity to obtain explanations and clarifications from the participants themselves about the events that occurred during lessons (Ryen, 2004).

Empirically anchored descriptions provide a satisfactory foundation for a theoretical analysis (Larsen, 2005). Thus the study has a descriptive character; its research value lies in its detailed descriptions of the learning activities that took place during the lessons observed. In order to make an inno-
ative contribution in the research field of programming, the research questions are of various kind (Cohen et al., 2018). The first research question in the study is purely descriptive – to describe the pupils learning processes. The aims of the following research questions were to inductively analyse these processes. Three researchers analysed the collected data together. These analyses have also been tried and discussed with other researchers at seminars and scientific conferences and in that respect, they are well established.

Ethical considerations were carefully followed according to the recommendations of the Swedish Research Council (Vetenskapsrådet, 2017). An information letter was sent to the participating classes about the purpose of the study. All participants were asked in advance if they were willing to participate in the study and permission was granted by the pupils’ legal guardians. At the start of each observation, the participants were informed about the project and that they had the opportunity to cancel their participation at any time. The names appearing in published material are pseudonyms.

**Participants**
Invitations to participate in the study were sent by email to 19 SCs in Sweden. Swedish SCs vary in size and have different types of financing, such as municipally owned companies, a part of municipal administration or foundations. A significant part of the activities at SCs are aimed to the public at the same time as interdisciplinary programs are offered to schools (Skolverket, 2019b). Three SCs were chosen for this study based on the number of visitors per year and geographical location. Ten classes from compulsory schools were contacted and they agreed to participate when visiting SCs as part of the school day. 220 pupils ranging in age between 7 and 15 years, corresponding to Grades 1-8 in Swedish compulsory schools participated. The participants in Grade 1 are excluded from our analysis of the practical work because their workshops were explicitly adapted for young pupils.

The observations and conversation excerpts in the article are taken from lessons adapted for pupils in Grades 3–8. The passages are from sequences where pupils were working (usually in pairs) with LEGO robots and lessons where they were working with the screen-based software Scratch. In the LEGO robot lessons, the instructor gave the pupils a set of assignments, of various levels of difficulty, to choose from. The pupils were directed to solve the assignment at their own desk, and then test it on a large table (or the floor) where the instructor had prepared the assignments. In the Scratch lessons, the pupils were directed to create their own animated computer game. Everyone started with the same animated cat (avatar). There were several possibilities for animating the game, and there were also opportunities to create a game with different avatars, backgrounds and sounds. During the lessons at SC 3 the programming tools Makey-Makey, Arduino and Kojo, were also used as a complement.

**Procedures**
In order to ensure continuity and that all data collection was carried out in a similar manner, all observations and conversations were carried out by the same researcher. Each lesson lasted for about two hours, and in most cases the observations were followed up with informal conversations with pupils. Sometimes no conversations were conducted, because the pupils did not want to, or there was not enough time. All the pupils, with a few exceptions, were novices who had not worked with programming at school (see Table 1).
Table 1 Overview of the data

<table>
<thead>
<tr>
<th>SC</th>
<th>SC 1</th>
<th>SC 1</th>
<th>SC 1</th>
<th>SC 2</th>
<th>SC 2</th>
<th>SC 2</th>
<th>SC 3</th>
<th>SC 3</th>
<th>SC 3</th>
<th>SC 1</th>
<th>SC 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grade in compulsory school</td>
<td>8</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>8</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of conversations</td>
<td>4</td>
<td>5</td>
<td>10</td>
<td>6</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Active accompanying teacher</td>
<td>No</td>
<td>No</td>
<td>Part of lesson</td>
<td>Part of lesson</td>
<td>Part of lesson</td>
<td>No</td>
<td>No</td>
<td>Part of lesson</td>
<td>Part of lesson</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The main data collection was by observation. No predetermined observation scheme was used, so the observations can be said to be of low structure (Bryman, 2016). However, during the lessons, careful notes of what happened were taken. Throughout the introductions, the observer sat behind the pupils, noting the activities of both teachers and pupils. During the practical work, the observer moved around the classroom, concentrating observations on a small number of pupils at a time. In order to supplement the notes, parts of the lessons were filmed with a mobile camera. Using a mobile camera made it possible to be flexible in the classroom and get close to the pupils’ activities; the filming also allowed us to go back and study interesting parts of the observations. The conversations, with one or two pupils at a time, were audio-recorded using a portable device and lasted about 2-5 minutes. They were based on what happened during the lessons; no predetermined conversation guide was used. The conversations gave the observer an opportunity to contextualise and understand the activities during the lessons. They were used to capture the pupils’ perspectives and to understand how the pupils themselves reflected on their work. Field notes were edited and notes from the selected parts of the films and conversations were then transcribed.

Data analysis

Initially, a thematic analysis of the entire data material was carried out. Based on this thematic analysis, three themes were selected for this article. The themes were then further analysed with the LDS model (see Figure 2). Throughout the work process, memos were written to underpin the analytical work.

The thematic analysis included several steps, such as generating initial codes, searching for themes and defining and naming themes (Braun & Clarke, 2006). These themes were identified in an inductive way and are strongly linked to the data itself (Vaismoradi et al., 2013). The themes were defined not by quantitative occurrences but by qualitative differences. Five themes relating to activities for learning programming at SCs were identified in this way, three of which were selected for this article: introduction, instructions and approaches.1
The first three steps in the analysis process were thus implemented to formulate themes, using Braun and Clarke’s model (2006) for thematic analysis. From step four, the thematic analysis was supplemented with the LDS model (Selander, 2008, 2017) (see Figure 2).

We used the LDS model’s framework for settings to analyse the themes introduction and the instructions. In this way, we were able to describe and analyse the design for learning in the lessons. Relevant areas to study here were how the pupils were introduced to the work, what type of software and programming tools were offered and how the instructions were carried out.

The approaches, which became evident as our third theme, were analysed using the LDS model’s primary and second transformation unit. The different approaches were crystallised by comparing the resources the pupils used, what opportunities for reflection they applied and how the transformation units overlapped. The primary transformation unit gave us tools to analyse the actual resources, both practical and theoretical, that pupils used when they formed and transformed their knowledge of programming. The second transformation unit was used when analysing the pupils’ opportunities for meta-reflection and theoretical discussions during the learning process. The transformation units can be perceived as overlapping and sometimes the learning process may even be seen to move back and forth between the primary and second transformation units.

RESULTS
In this section we present the themes introduction, instructions and approaches identified by thematic analysis (Braun & Clarke, 2006) and analyse them using the LDS model (Selander, 2008, 2017).

Introduction to the lesson
The practical settings for programming lessons at SCs were planned by qualified teachers working at SCs and were all well prepared. The material to be used during the pre-booked lessons was already set up when pupils came to the SC. Potential resources such as technical and digital equipment were
readily available. Computers were prepared, and robots and tablets were loaded. Assignments were produced for each class, with all the necessary material prepared. This prefabricated and well-prepared arrangement illustrates an institutional norm whereby pupils’ activities are strongly framed by certain physical conditions that encourage specific activities in a spirit of learning by doing in a semi-formal learning environment.

Every group of pupils had at least one member of staff from the school present, either their regular class teacher or an assistant teacher. Thus the SC teacher and the accompanying teacher became two distinct potential resources, the SC teacher being an active resource and the accompanying teacher a more passive resource. In most cases, these accompanying teachers ceded all responsibility for the lesson to the SC teacher. If they were active, it was mostly to help individual pupils to concentrate.

All the lessons were introduced in the same way. An SC teacher initially introduced programming as a phenomenon in society and tried to make connections with the pupils’ everyday lives. The SC teachers were teasing out the pupils’ perceptions of programming while at the same time trying to broaden their views on it. The pupils’ variety of perceptions provided examples of everyday artefacts that the SC teachers used in their explanations. Some pupils associated programming exclusively with robots; other examples mentioned were computers and mobile phones, examples from their everyday lives. Using the pupils’ examples as a resource gave teachers the opportunity to create a form of collective understanding, which in turn became a potential resource for the rest of the lesson.

**Instructions**

At two of the SCs in the study, one programming tool was used per lesson and the entire lesson was led by an SC teacher. At the third SC, after the introduction, university students held workshops in smaller groups, giving pupils the opportunity to observe and try out three or four programming tools during the lesson (see Table 1).

Our observations of the instructions show that various methods were used by different instructors. In half of the observed lessons, the instructor’s method was to let the pupils interpret the instructions and possibilities by themselves. After a demonstration of the basic functions of the equipment, they were asked to follow written instructions on their own. On these occasions, it was hard for many pupils to get started with their own work. Some pupils asked the instructor or a friend for help. In other lessons, the instructors gave a presentation, with the aim that the pupils should carry out each stage at the same time. The instructions then became a powerful resource, guiding the pupils’ learning. However, on several occasions they were already trying other solutions during the presentation. Nevertheless, during these lessons all the pupils achieved a program that worked satisfactorily, allowing them to continue at their own pace with extended or new assignments.

The lessons in programming at SCs were relatively formally framed. That is, their structure consisted of verbal and sometimes written instructions, practical work and a short conclusion. Regardless of the number of programming tools used during the lesson, the instructions provided for the assignments had to be brief due to the limited lesson time. The settings for the instructions had a specific purpose: to enable the pupils to use the chosen programming tool. However, there was a risk that they would not get the help they needed to get started. This may be due both to the instructors not knowing the students’ personal ways of learning and to the limited time. Thus despite good resources there may be some risk associated with transferring a compulsory part of children’s education if SCs cannot meet the needs or provide support for all pupils. However, such support can be facilitated, for example, when the accompanying teacher from the school takes an active role in supporting individual pupils.

---

2 Hereafter, both SC teachers and university students are referred to as instructors.
Approaches
Each approach (compiled in Table 2) was observed a number of times, regardless of the programming tool and method used by the instructors. The fifth approach was only observed when the programming tool allowed this kind of activity.

Table 2 Overview of the approaches based on the results

<table>
<thead>
<tr>
<th>Approach</th>
<th>Learning process</th>
<th>Assignment solved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mathematical</td>
<td>problems are solved theoretically</td>
<td>in one piece</td>
</tr>
<tr>
<td>Trial-and-error</td>
<td>problems are solved practically</td>
<td>in one piece</td>
</tr>
<tr>
<td>Step-by-step</td>
<td>problems are solved utilising experience from previous steps</td>
<td>one step at a time</td>
</tr>
<tr>
<td>Routine</td>
<td>problems are solved practically and theoretically through given routines</td>
<td>one step at a time</td>
</tr>
<tr>
<td>Aesthetic</td>
<td>problem solving is subordinate to aesthetic expression</td>
<td>partly and with a focus on aesthetic expression</td>
</tr>
</tbody>
</table>

In the following, we describe the five different approaches using observation notes and conversation excerpts in order to contextualise the learning situation from the pupils’ point of view.

The mathematical approach
With the mathematical approach, pupils see the assignment as a problem that can be calculated and then solved as a whole. We first illustrate a problem concerning the programming of a game with the cat avatar in Scratch. The instructor asks the whole class about the distance the cat is supposed to go:

Instructor: How far does the cat move in ten steps?
Karl is the only pupil who puts up his hand and answers: It’s like 1/24th of the system.
The instructor confirms that it is like a coordinate system.

Observation SC 1, Grade 8

Karl perceives that the task can be solved by calculating how the cat should be programmed. This is confirmed by the instructor, and then the pupils start working by themselves. Karl tackles the assignment through calculation and does not take advantage of the opportunity to try the solution in practice. The same is true for Tilla and Tina, who are trying to solve an assignment involving robots.

Tilla and Tina have chosen an easy assignment and talk a lot about how they will do it. They try to figure out, by calculating in their heads, how far the robot should go. They use a measuring tape etc.

Observation SC 2, Grade 6

The calculations became difficult for them to handle and, when they tried their program on the large table, the distances were not correct. They started again by measuring the whole distance, then tried to calculate and compare how the robot should respond. They had problems seeing which part of the script they had to change and made several calculations before it worked fully.
When analysing the mathematical approach, we note that the pupils start discussing how to solve the problem. They then work in the first transformation unit by using some of the available resources, such as their mathematical knowledge, and if applicable the tape measure, in order to proceed with the programming. These are possible ways of solving the assignment. However, when working with the programming task according to the mathematical approach, they are not able to transform their mathematical knowledge automatically into knowledge of programming. They get stuck while trying to manage the entire assignment at the same time. They have difficulty in forming a representation (in the second transformation unit) making the game or robot work as intended. As they continue, they do not discuss the problem but start to calculate all over again.

The mathematical approach (using mathematical knowledge) is difficult for pupils to use as programming beginners; extensive support may be required from the instructor if the mathematical calculations are difficult. However, in the cases described, we wish to emphasise that pupils did not use the instructors (or other knowledgeable pupils) as a resource.

The trial-and-error approach
When using the trial-and-error approach, pupils work practically to solve the assignment. They try to solve the problem as a whole, trying again and again with persistence and determination to cope with the assignment.

Elin and Elsa want their robot to run in a square marked on the floor. They do a program several times and test it on the floor each time. When it results in errors, they change the program and test it again.

Observation SC 1, Grade 3
Without much thought, the pupils entered a program and tried it out using the robot on the floor. When it did not work, they started all over again. They had to try several times and make many adjustments before the robot followed the marked square.

During another observation, the observer saw that Erik waited for Eskil to start the robot they had just programmed once again after working for a long time with the same assignment.

Observer: What’s happening?
Erik: I’ve sort of dreamt of starting a game company with this kind of console. But if this doesn’t work, I’m giving up.
Eskil: But I’m not, so then you’re not either!
They put their robot on the test table again...
Erik: Now we’ll see victory!
The robot manages its task. Both pupils give a thumbs up and look very pleased.
Erik: At last!

Conversation SC 2, Grade 6
In contrast to the previous example (the mathematical approach), the pupils who attempt the trial-and-error approach start in an unplanned manner in the primary transformation unit by programming a code without much reflection. They enter the entire program and start from the beginning again when it does not work. When they fail, they quickly talk about how to change the program (second transformation) to make it work better. In this approach, they do not divide the assignment into partial problems. Their persistence and determination to succeed can be regarded as a resource for solving the programming assignments.

We can interpret this approach as pupils transforming their programming abilities by trying over and over again and that, when using this approach, they work mostly in the primary transformation unit. They form knowledge about programming by trial and error, and do not seem to reflect theoretically on their solutions. They use their endurance in not giving up.
The step-by-step approach
In this approach, pupils perceive that problems can be solved a bit at a time. Using the step-by-step approach includes finding a practical solution that also invites them to work methodically, whereby each part of the program is tested before it is assembled into a solution for the entire assignment.

Petra and Pia decide to start with an easy assignment. They test different parts of the program separately on their own desk several times before trying it on the table. There, they only need to make small adjustments to succeed. Petra and Pia are the first ones to complete their assignment. They spontaneously dance and demonstrate with their bodies that they are satisfied. They then proceed with a slightly harder assignment and tackle it in the same way.

Observation SC 2, Grade 6

When these pupils felt ready to run the entire assignment’s program on the large table, they noted that they only needed to make a few minor adjustments. Later on, Petra and Pia tested a new assignment and the observer asked them if it went well.

Petra: No, it didn’t work. We have to turn more and then go there.
Petra points to the big table and Pia nods.
Pia: Yes, I’ll do it.
Pia sits down in front of the computer and adjusts the last part of the program.
Observer: What are you thinking now – to make it work?
Petra: It has to turn.
The pupils take the robot and try again.

Conversation SC 2, Grade 6

Petra and Pia used the method they had developed during previous assignments they had already solved, by again checking their robot on the big table. The last assignment they attempted was highly advanced, with many factors to consider in the program.

In the step-by-step approach the pupils discuss which assignment they should attempt first, choosing an easy one. They then use their ability to divide problems into smaller units, taking one sequence at a time and making it work before tackling the next part. They work out how the program functions and think about the next step. Furthermore, they utilise the simpler assignments as a resource, utilising their experience of these programs to help them with increasingly advanced tasks. Thus, they are able to form knowledge of how to program. We notice there seems to be a movement between the transformation units, as the pupils program short sequences and then discuss and reflect over the next step.

The routine approach
With the routine approach, pupils use both practical and theoretical elements in their problem-solving, following the instructions they received initially. Linn programmed a robot together with Lisa, and they wanted to make sure that they did the right things. The instructor showed them how to get started, but when they continued to work on their own, they became uncertain again. The observer talks to Linn:

Observer: How do you think things have gone today?
Linn: Sort of OK, umm.
Observer: Sort of OK. Has it been difficult?
Linn: Yes, rather, when we didn’t know what to do. Then it was tricky.
Observer: So how did you solve the problem?
Linn: We tried various things, and when they went wrong, we redid what we’d changed.

Conversation SC 1, Grade 3
Linn says it is difficult when she does not know what she is expected to do. She is careful and wants to make sure they do the right thing while they are programming. When it does not work the way it was supposed to, she becomes uncertain. So the pupils erase the program they started with. They are hesitant about using their own ideas and return to following the given routine.

Anna and Alicia also carefully follow the instructions when programming their robot. They check each step of the programming at their own table before trying their solution on the assignments.

Anna and Alicia are helped by the instructor to measure and take the first step of the programming. They test the program by letting the robot run on the track. The robot can handle the first turn on the track. The girls follow the instructions that the following step is to measure the next distance.

Observation SC 3, Grade 5

At first, they were insecure, but when given a proposal for a routine to work with, they solved several assignments. They followed the same routines throughout the workshop.

Using the routine approach, the pupils start the assignment by discussing together with the instructor how to perform the assignments. We can note that they use several resources, for instance the ability to divide the assignment into parts, measuring with the tape measure, mathematical ability and testing each part before they complete the whole assignment. The approach is based on routines in two respects; - the pupils are accurate and at the same time cautious about trying out their own ideas. To do this, they follow the routine. Despite some uncertainty, the routine helps them solve the assignment.

The aesthetic approach

The aesthetic approach differs from the others insofar as the focus is not on the functional solutions to the assignment. Pupils do not attach equal importance to solving the whole assignment. They spend a lot of time choosing the avatar and changing the layout of the environment. At the same time, they use their creativity to design their representation of the programming assignment.

Aron and August are initially not satisfied with the scenes they can find in the computer library. They go online and search for pictures of snow, but they are still not satisfied so they go offline and choose a scene from the library, with snow. [...] The instructor notes that some have not yet begun programming but confirms to the pupils that it is OK to start in different ways.

Observation SC 2, Grade 4

These pupils focus on an aesthetic experience that in this example includes the appearance of the game they are going to create using the Scratch software. They do not focus on solutions that include other parts of the assignment and do not reflect on the fact that the main purpose, to create a computer game, also includes movement and animation. Instead, they concentrate only on the game's appearance.

The aesthetic approach also includes other aspects of pupils’ creativity, for example the use of special sound effects.

Filippa has exchanged the Scratch cat for an image of a horse. The sound has been changed to a neighing horse and the background sound to that of a galloping horse.

Observation SC 1, Grade 8

Although the assignment was solely to create a game with a number of functions using the Scratch cat avatar, Filippa chose to spend time replacing both the avatar and the sound and adding an extra background sound.
When using the aesthetic approach, pupils use their ability to create. Although aesthetic expression is important and a part of programming, it can be difficult for them to develop other abilities that are necessary to solve the whole programming assignment. If they do not get (or take) the opportunity to discuss and reflect upon the main purpose of the assignment, it may mean that they do not have the opportunity to practise more abilities that are important for programming.

**DISCUSSION**

The major contribution of this study is that we have identified and illustrated five qualitatively different approaches that novice learners take when programming. Our analysis, using the LDS model (Selander, 2017), shows that pupils within the approaches use different types of abilities. It is important to acknowledge that we do not consider these approaches to be stable learning styles or fixed representations of pupils’ knowledge of programming but rather strategies that they use in certain situations (see also Kirschner, 2017). The approaches may partially overlap; the examples have been chosen to highlight the unique qualities. The abilities the pupils use remind us to a certain degree of what has been described in relation to the concept of computational thinking (CT) (Wing, 2006), as will be discussed below.

**Approaches, abilities and computational thinking**

What pupils learn is dependent on both the curriculum and the educational design (Popat & Starkey, 2019). Pupils understand programming in different ways and deal with programming assignments based on the experiences they have from other contexts. In line with Turkle and Papert (1990), we identified both abstract and concrete approaches. There is also a similarity with the strategies that Chao (2016) identified, both the approaches that repeat what one previously programmed and the ones that try over again. We see an added usefulness in these approaches when we compare with CT.

In the mathematical approach, pupils use abstract thinking and mathematical procedures, and in the trial-and-error approach they practise determination to succeed and endurance in not giving up. This is similar to what have been described as CT concepts, “using abstractions” (Cetin & Dubinsky, 2016) and “persevering by trying out new solutions” (Wing, 2006), both important abilities for problem-solving. The opportunity to divide a problem into smaller segments, used in the step-by-step and routine approaches, makes it possible to solve it without understanding the whole assignment at once (Cetin & Dubinsky, 2017; Wing, 2006). In the step-by-step approach, pupils utilised their experience from previous steps, which indicates the CT ability to generalise (Durak & Saritepeci, 2018). In the routine approach, they solved all the assignments in the proposed way. This ability resembles a mechanical routine (Wing, 2006). Problem-solving within CT means being creative and able to find solutions (Durak & Saritepeci, 2018). Furthermore, the aesthetic approach actualises, in a concrete way, how pupils use and practise their creativity. That is to say, the settings described in this study enable them to use their own ideas when forming their approach.

Wing (2006) argues that CT should be seen as a concept that includes thinking and skills that will soon be as important as reading and writing. We do not claim that the abilities used are equal to CT. However, we can see that each of the approaches we have described builds on certain abilities that are similar to CT. These abilities are used and probably also developed further during the programming practice. We argue that this is a strength – the fact that pupils can and actually do act differently. We consequently claim that pupils do not necessarily need to be introduced to CT as early as possible (cf. Buitrago Flórez et al., 2017), since pupils appear to develop CT spontaneously by programming. These results can contribute important knowledge in the ongoing implementation of programming in the compulsory school.

**Design for learning programming at SC**

Dawson (2014) suggests that informal learning environments, such as SCs, should be utilised in order to provide a good starting point for increased access and equality in science education. Through the
introduction and instructions given by staff, we show that the digital equipment available at SCs and the nature of the assignments motivate pupils to do programming. They move between computers and there are good opportunities to collaborate with others. This can be compared with the informal educational tradition at SCs, which urges pupils to discover on their own (Pedretti, 2004). However, since programming is a complex competence, pupils may need support to help them develop more skills than those possible in the approach they use by themselves. One way could be for instructors to allow pupils to demonstrate some form of “know-how” relating to the programming assignments. In the next step, they can then be encouraged to use a different approach.

One aspect of learning is being able to show how you understand something (Kress & Selander, 2012). Being able to explain things to others is an important process for understanding the subject you are learning (Vieira et al., 2019). However, our results show that only a small part of the lessons was devoted to discussion and reflection, which may be because the traditional norm at SC is “learning by doing”. When the lessons follow a style of “SC settings”, this can mean that there is not always time for reflection. Therefore, in addition to the opportunities that exist in each approach the pupils’ opportunities to learn from each other could be strengthened by organising time for joint discussions. By explaining to others, pupils with limited experience of programming can participate in a reflective process, thereby understanding and providing explanations for the programming tasks they have performed (Vieira et al., 2019).

It is by discussing and reflecting that learning can become meaningful in a context that is larger than the separate learning sequences. However, we can see that pupils do not reach the second transformation unit in the LDS model (Selander, 2008, 2017) to a particularly high degree. Programming is an important field of knowledge, not only because it is now mandatory in Swedish compulsory schools but also because, in order to cope with everyday challenges, there is a need for digital understanding. Accordingly, when designing opportunities for learning, time for joint discussions is required.

CONCLUSION
This study has implications not only for those interested in the enactment of programming education at Swedish SCs but also for research on learning programming in compulsory schools. The results show how programming can be implemented in a context where all the participants are novices. In the present context, pupils were given the opportunity to use their abilities to solve the programming assignments they were given. In this way they were involved in designing their own learning.

Owing to the small-scale basis of this study, generalisation of the approaches taken by the novice learners is limited. Further studies are needed to explore novices’ learning in programming. Pupils design their own learning based on their personal experience. Thus it would be worthwhile to investigate how pupils’ engagement relates to their learning process. This is especially true now that programming is a mandatory element in compulsory schools, leading to renewed interest in introducing programming to young pupils.

REFERENCES


