International Journal of Technology and Design Education (2024) 34:691–710
https://doi.org/10.1007/s10798-023-09841-x

What strategies do students use when they are programming a robot to follow a curved line?

Per Anderhag[1] · Niklas Salomonsson[2] · Andre Bürgers[3] · Cesar Estay Espinola[4] · Birgit Fahrman[5] · Dana Seifeddine Ehdwall[6] · Maria Sundler[5]

Accepted: 20 June 2023 / Published online: 14 July 2023

Abstract
During a relatively short period of time, programming has been implemented in the national curriculum of the compulsory school in Sweden. Since 2018, programming is a new content in the technology subject and the research field has discussed some of the challenges teachers and students, who generally have little experiences of programming, face when programming is introduced in teaching. In this study, we have explored what strategies lower secondary school students (ages 13–15) use when they are programming a robot to follow a curved line in technology education class. Data consists of screen recorded films when students are pair programming a robot. Student talks were transcribed verbatim and analysed using Practical Epistemological Analysis. The analysis revealed three different strategies that the students used when programming the robot: (1) sensor—follow the line, searching for a code that automatically would make the robot to follow the route, (2) sensor—wheels, using codes to create a feedback system between sensor and wheels, and (3) rotations—degrees–wheels, using the position of the robot to stepwise fine tune the movement of the wheels. In line with previous research, the students in our study spent much time discussing, testing, and debugging their code, and our findings contribute by showing how these discussions were aligned with the strategy used. Depending on the strategy, students actively looked for and tested codes affecting different aspects of the sensor-wheel system, such as for example sensor input, power, rotations or turning. Implications for teaching is discussed.

Keywords Programming · Robots · Follow a line · Strategies · Technology education · Lower secondary school

Introduction

In a relatively short period of time, programming has become content in the compulsory school curricula of many Western countries (Balanskat & Engelhardt, 2015; Raptopoulou, 2021). For teachers, this change has been challenging due to for example lack of formal and non-formal experiences in programming, time for transforming the content of

Cesar Estay Espinola previously worked at Education Administration, Botkyrka Municipality.

Extended author information available on the last page of the article the curriculum into teaching, and a great variation in opportunities for in-service training (Vinnervik, 2022a). The present practice-based study focuses on a programming task that was designed and implemented in two technology classrooms in lower secondary schools in Sweden. The study, conducted in collaboration between researchers and two technology teachers, presents and discusses findings showing how students use different strategies for programming a robot to follow a line. Our study, and the analytical procedures used to capture learning as it is constituted in classroom action, comply with the call for studies investigating consequences of teaching for student learning in technology more closely (Hallström, 2018).

Since 2018, programming is part of the curriculum of the Swedish compulsory school. The implementation was preceded by an extensive discussion among various stakeholders arguing for its potential role in the Swedish educational system (Raptopoulou, 2021). Programming did not become a separate school subject but is now stated in the introductory chapters of the national curriculum as essential in the digital infrastructure of today's society (Swedish National Agency for Education, 2018). Knowledge about programming and its role in everyday life is contextualized as a form of literacy that students should develop through the teaching in every school subject. Programming was also implemented in the mathematic and technology subjects and in the technology syllabus, programming is described as a tool that the students should learn to use for controlling objects. As in many other countries, the technology subject in the Swedish compulsory school has emerged from several different traditions, focusing on technology content in relation to workshop technology, civic education, technology's social consequences and applications in the natural sciences (Hallström et al., 2014; Jones et al., 2013). In relation to this background, it is hardly surprising that there is disagreement among policymakers, teachers, and educational researchers about the content and character of the subject (Danielsson et al., 2018; Fahrman et al., 2020; Hallström et al., 2014; Norström, 2014) and what role programming may play in the technology subject (Raptopoulou, 2021; Vinnervik, 2022a). Teaching and learning programming in relation to the Swedish compulsory school technology subject is therefore little explored and there is yet little experience among the technology teachers concerning the use of programming when teaching technology (Vinnervik, 2022a).

In Sweden, as in many other countries, a simple robot is often the object that the students first meet and potentially will learn to control through programming (Anderhag et al., 2021; Misirli & Komis, 2014). Several studies have described programmable robots as suitable tools for introducing programming concepts but also for developing various skills (Green et al., 2018; Komm et al., 2020; Misirli & Komis, 2014; Xia & Zhong, 2018). The use of robots is often motivated by the fact that programming is difficult to learn (Misirli & Komis, 2014; Sáez-Lopéz et al., 2016) where especially syntax and the logic required to formulate a solution to a specific problem is difficult for beginners (Robins et al., 2003). Also, the concrete and sometimes playful context is considered to be motivating and thus able to support students' learning (Highfield, 2010; Komm et al., 2020; Newhouse et al., 2017). In addition to programming skills and learning specific programming concepts, programmable robots are suggested to support students' social and generic skills such as collaboration competencies and logical thinking (Bers & Horn, 2010). Studies have also shown that programming robots can be important for students' understanding of technology in society and thus also for their technical literacy (Komm et al., 2020; Kurebayashi et al., 2007). Research on younger children's (6–8 years) learning has been of particular interest to the relatively well-known Beebot and Bluebot (Misirli & Komis, 2014) and Beraza et al. (2010) argue for their usefulness, although they also conclude that there are limitations in how they can support a more advanced understanding of programming. Beebots have only five commands, which is a limit to the type of tasks they can be programmed to perform (Beraza et al., 2010; Kazakoff et al., 2013). In a previous study, we examined primary school students' perception of functionality in their spontaneous programming language for steering a Blue bot. The findings showed that the students primarily perceived a code's functionality as a question of readability, rather than how well it fit the purpose of controlling the robot, which we suggested—in line with Beraza et al. (2010)—may be connected to the limitations of the programming language that can be formulated (Anderhag et al., 2021). Using more advanced robots with sensors opens up for more complex programming activities by using for example input—output feedback processes. Komm et al., (2020, pp. 259), using Lego EV3 robots, have also shown that such physical—digital systems "can help the students form a viable model of the machine to be controlled: the notional machine finds a tangible manifestation of its basic properties and capabilities". Although not studying students working with robots but rather Micro:bits, similar conclusions are forwarded by Cederqvist (2022) who explored the constitution of technological knowledge as students addressed aspects of construction and functionality of a technical solution when programming. Activities that create opportunities for the students to design and adjust different parts of a technical system, such as a programable robot, therefore have the potential to expand students´ understanding of technology.

A relatively well-known programming activity for supporting students´ understanding of input–output-feedback processes and the relation between the code and the physical object is the programming of a robot to follow a line (e.g. Cesaretti et al., 2017; Francis et al., 2016; Merkouris & Konstantinos, 2018; Xia & Zhong, 2018). By using sensors, the robot is programmed to register the surface and depending on the direction of the line, adjust its position and movement by how the wheels rotate. In the study of Gonçalves et al. (2019) the students (age 15–17) who had little prior experiences of programming worked with a task where they were supposed to make a robot follow a curved line. After approximately an hour they were able to solve the task by testing and adjusting the demo code they started from. Although the authors do not discuss in detail the programming strategies the students used, the findings may be in line with Merkouris and Konstantinos (2020) study where it was shown that students with different experiences of programming used different strategies to solve the programming tasks, one of the tasks being to follow a line. More advanced students reused and remixed large parts of code and removed unnecessary blocks while novice students tended to reuse small parts and tested them step by step in cycles. This incremental—iterative approach was not evident among the more advanced students who rather were shown to abstract and modularize different strategies. Both groups were struggling with testing and debugging (Merkouris and Konstantinos, 2020). Some strategies and associated debugging practices may be related to the misconceptions students have when working with programable robots. The students in Komms et al. (2020) study were working with the follow line—problem and two other programming tasks and misconceptions were shown to revolve around recurrent difficulties concerning for example ending the program (understanding the stop() command), multitasking, using delay() commands, and using loops in a proper way. Other difficulties students may encounter may be coupled to the misconception of a one-to-one correspondence between for example the input value of rotating a wheel and the degree measure of a turn (Francis et al., 2016).

In summary, previous research has demonstrated that programmable robots may serve as meaningful and functional tools for teaching and learning technology. Technical systems, being a key component in a technological knowledge, are scrutinized when students debug and readjust their code because of how the parts of the object respond to the program (Cederqvist, 2022). The code itself can also be understood and operationalized as a form of a technical solution, evaluated, and revised due to its functionality (Anderhag et al., 2021). Research has however also shown that students often encounter difficulties when programming feed-back dependent robot movements such as following a line. For example, it has been demonstrated that there are differences in how novice and more experienced students debug their programs. Much of this important knowledge on encountered difficulties and the strategies students apply for overcoming them have been generated from interviews, open-ended questions and pre-, post-test and written representation and the knowledge about how such strategies are constituted in talk and action when students are programming is to our knowledge sparse. The aim of the study is therefore to address this gap on how learning to program can come about in the technology classroom. Our primary interest is how students approach the task of controlling an object through programming and we ask:

- What strategies do students use when they program a robot to follow a curved line?

Methods

Study context and data collecting

In the Swedish technology syllabus, programming is since 2018 part of the core content *methods for developing technological solutions* and in years 1–3 (age 7–9) the students are supposed to learn to control objects, such as a robot, using programming. In years 4–6 (age 10–12) the students should learn to control their own constructions or other objects by using programming, and in years 7–9 (age 13–15), the students are supposed to use programming for controlling and regulating their own constructions. Programming is thus primarily a tool for controlling objects and a progression in terms of knowledge in programming is not formulated in the technology syllabus. Neither are there any specific rationales, methods, or programming concepts that the students are supposed to learn or be assessed on. Progression is thus a question of the object programmed, that is whether it is constructed by the students or not.

The data for this study comes from two lower secondary technology classrooms (year 9, ages 15–16, School A and School B) in Stockholm, Sweden. Estay Espinola was the teacher of the class in which the study was conducted and thus knew the students well. Salomonsson worked together with a colleague, and they jointly decided that the study should collect data in the class of the colleague, Salomonsson knew his colleague's students well. A lesson was planned in which the students should program their robot to follow a curved line. The activity addressed an overarching learning outcome stated in the technology syllabus, learning to control an object through programming, but also a genuine pedagogical question, that is, what challenges do students encounter when they are programming a robot? Estay Espinola and Salomonsson used green and black tape respectively to create a curved line on the working desks in their technology classrooms, a red tape formed the end of the track. Due to contingencies the curvature of the tracks come to differ somewhat between School A and School B, see Fig. 1.

The students were given written and oral instructions about the task, namely, to program the robot to follow the black/green line and stop at the red line. The teachers also told them that they had access to demo codes that they could use if they wanted to, see Figs. 2 and 3. During the lesson, the teachers walked around among the student groups checking how

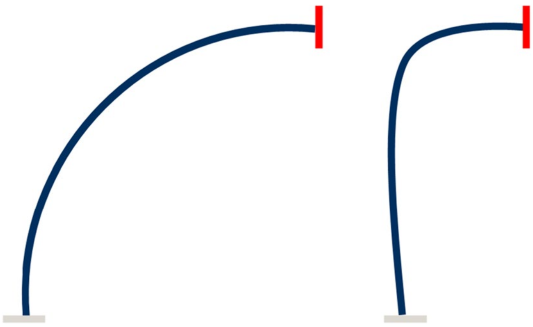
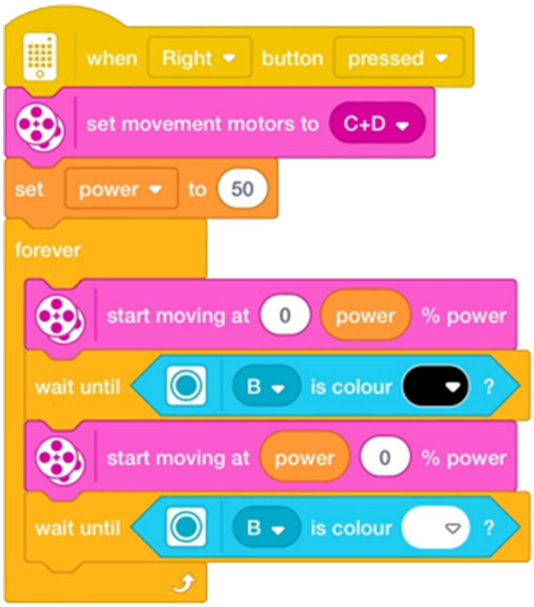
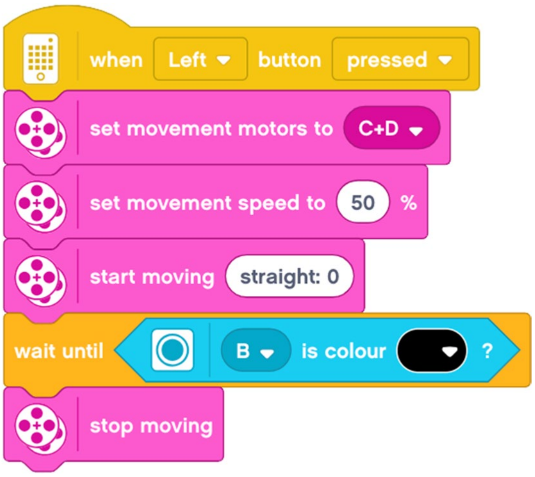


Fig. 1 The routes that the robots should follow; School A (left) and School B (right)


```
when Right button pressed
set movement motors to C+D
set power to 50
forever
    start moving at 0 power % power
    wait until B is colour ?
    start moving at power 0 % power
    wait until B is colour ?
```
Fig. 2 Demo code 1, register the surface and adjust movement

their programming progressed, while doing that they asked and answered questions and encouraged the students to discuss and articulate what they wanted their code to do.

The students used Lego Spike Prime robots with two motors and one light sensor for colours mounted on them. iPads with the Lego package were used to program the robots, codes were executed via Bluetooth. The students in School A had little previous experience of programming and it was the first time they used Lego robots in a school setting. Most of the students had however encountered block programming through Scratch in previous years. The students in School B had programmed Lego Mindstorm robots and was thus experienced and familiar with the robot and programming language. They had, however, never used Lego Spike Prime and the code blocks differed a bit from Lego Mindstorm's. In School A, the Follow a curved line-task was one of many programming tasks that the students were working on, the programming activity were part of a unit on sustainability and technology in which the teacher and the students discussed and modelled an automated recycling station. In School B the students only worked on the curved line-task.



```
when Left button pressed
set movement motors to C+D
set movement speed to 50 %
start moving straight: 0
wait until B is colour ?
stop moving
```

Fig. 3 Demo code 2, register the surface and stop movement

The students worked in pairs using the Lego Spike-package on an iPad when they pair-programmed their robots. Every group screen recorded with sound while they were coding which resulted in films showing how the program gradually emerged. This in situ programming activity and associated student conversations constitutes the data of this study. Video recordings of the movements of the robots were also collected from both schools, this data was however primarily used for visualizing the setup. Estay Espinola and Salomonsson collected the data. In total 7 screen recorded films, 4 from School A and 3 from school B, were transcribed verbatim and analysed. Due to the time the students needed to complete the activity, the length of the films varied from 30 to 60 min.

Ethical considerations

The study follows the ethical guideline of the Swedish Research Council (2017). The participating students were guaranteed anonymity, that their participation was voluntary, that the decision to/to not participate would not affect their grades or learning opportunities, and that they could whenever they wanted to withdraw their participation. Participating students and their guardians took part in and signed a letter of consent in which the purpose and structure of the study were presented. The data of the study is handled according to the General Data Protection Regulation (GDPR).

Data analysis

Initially the transcribed material was read and the discussed by the research group. In this phase Estay Espinola and Salomonsson, being the students´ teachers, made professional assessments on ambiguous situations in the material and could also clarify student utterances. In the next step, Practical Epistemological Analysis (PEA) (Wickman & Östman, 2002) was used to identify instances during the activity where student talk indicated different programming strategies for making the robot to follow the line. PEA builds on the later Ludvig Wittgenstein's writings and John Dewey's pragmatism (Wickman & Östman, 2002) and is used to operationalize meaning making and learning as discourse change as part of an activity. Its purpose is "to understand what people say and do during authentic classroom work and what this tells us about what and how students learn by participating in the specific interactions of a certain curricular setting" (Wickman, 2004, p. 326). PEA has been adopted in numerous studies exploring learning as it unfolds in classroom action (see Kelly et al., 2012).

Here we use three analytical concepts of PEA: *stand fast*, *gap* and *relations*. What stands fast in a situation are things, phenomena, actions, words that the interlocutors do not question in talk or action. For example, in the utterance "Let's use the sensor" made by imaginary Student A the word sensor stands fast. That is, in the situation Student A knows what a sensor is and another student, Student B, who responds, "Great idea, I mount it on the robot" also knows what a sensor is. However, Student B could also say "Sensor? Which one is the sensor?" or "Why should we use a sensor?". In both utterances a gap is noticed analytically, that is, in the situation it is not clear to Student B what a sensor is or looks like or why they should use it. In order to proceed in the activity, students A and B need to agree on what a sensor is or what it should be used for. Analytically this is described as that they are filling a gap by establishing a relation to what stands fast. In the example above this could happen by Student B saying, "Is this a sensor?" while holding a sensor or Student A saying, "We need a sensor for the robot to read the surface". In both cases relations are presented that may potentially fill the gap: What is a sensor—this is a sensor, why use a sensor—for the robot to read the surface. If the gap is filled the activity can proceed. In this view, words such as sensor are thus approached as gaining their meaning through their use and consequences as part of an activity, rather than representing some hidden or mental entity that are ready-made once for all (Wickman, 2004). If the gap is filled in the example above, meanings regarding what a sensor looks like or what it can be used for is constructed in the situation. The short example does thus not report what the two students know with certainty about sensors, or whether they know about sensors in the same way but is a report of the knowledge they construe in talk and action in order to proceed with the classroom activity.

The PEA procedure started with an additional reading of the material guided by the analytical question: what gaps occur when the students discuss how they should program the robot to follow the line? Initially, Anderhag was responsible for making a first PEA on situations in the material that the research group in the previous step had identified as possibly interesting, in that the students seemed to talk about different ways of proceeding with the task. This analysis provided the research group with a few examples that was discussed. In this step, we thus focused on situations where the activity halted, and where the students were discussing different ways to solve the specific problem they encountered. In this process we extracted preliminary strategies that appeared to be recurrent in the material. These strategies were discussed and internally validated within the research group before we made closer analyses.

Results

There are several possible ways in which the robots in the task could be programmed to follow a curved line. As described above, the students were expected to use the robots' light sensor and they had access to a demo code (Demo code 1, Fig. 2) that they could use as a starting point. In the demo code, the sensor was programmed to continuously register the surface and depending on the colour registered (in School A a black line and white table surface, in School B a green line and a black table surface), the movement was adjusted by the power which the two wheels mounted in parallel rotated. Through the interaction between the sensor and the wheels, the demo code resulted in a zigzag movement of the robot. In this way, the robot moved jerkily forward across the line. The idea of this first strategy, Sensor-Wheels, is thus that algorithms are used to make the two parts, sensor and wheels, work together. It turned out, however, that the students did not use the demo code to any great extent, but instead tinkered with another demo code (Demo code 2, Fig. 2) that they could use as a starting point to make the robot stop when its sensor detected a certain colour. The students that were using Demo code 2 as a starting point were shown to use a second strategy, Sensor-Follow the line. The student groups from school B had access to the same demo codes, but the task differed in that the line did not have the same curved shape (Fig. 1) and the activity of getting the robot to follow a curved line was the whole task for these students. In school A, the activity of following the line was the first in a series of programming tasks. The analysis revealed a third strategy among the student groups in school B, Rotations-Degrees-Wheels. The three strategies are presented below.

Sensor—follow the line

The Sensor—Follow the line strategy was characterized by the students looking for and trying to implement a code that automatically controlled the robot based on the information coming from the sensor. Adjustments of the movement of the robot in relation to the bending of the line were thus something that was expected to happen automatically. As described above, most students initially started with Demo code 2 (Fig. 3) and example 1 below showed this. The students (School A, Group 1) had just started with the task (Bold and italic utterances represent instances when the students read out load or says the name of code blocks or variables).

Example 1 It should move along the black one.

1. S1: What do we need?
2. S2: It should move along the black one [the line]
3. S1: Yes, exactly
4. S1: Should it start on black? There, there's the sensor
5. S2: We can start with, to…
6. S1: Because then we must change, uh, we need to make it follow the black line like that
7. S2: It´s supposed to. Let's check out ***Movements***. ***Go***, or no, it is ***Events***. ***When***. [scrolling in the menu]
8. S2: Er, yes. When the colour is [inserts a code block]
9. S1: Shouldn´t we have ***when the colour is***?
10. S1: Here we have it
11. S2: Shall we start by checking what these are doing?
12. S1: Yes, okay
13. S2: Left, should move
14. S1: We test it over there
15. [They place the robot on the track and starts the program]
16. S2: It stopped

17. S1: Now it stops when it is black, instead it should follow black [commenting demo code 2 and removes the command "Stop"]

In turn 6 the students noticed a gap; the demo code was designed so that the robot stopped when the sensor registers the colour black, and this was something that they needed to fix. Before they started making changes to the code, they tested it and could then see that the robot really stopped when the sensor registered black (turn 15). In turn 16 they concluded that, "instead it should follow black" and removed the command Stop. They continued working on their program and tested it continuously and when doing that they discovered that it did not work as they had intended. After about 10 min, they started looking for a command to follow the line:

Example 2 Is there any follow command?

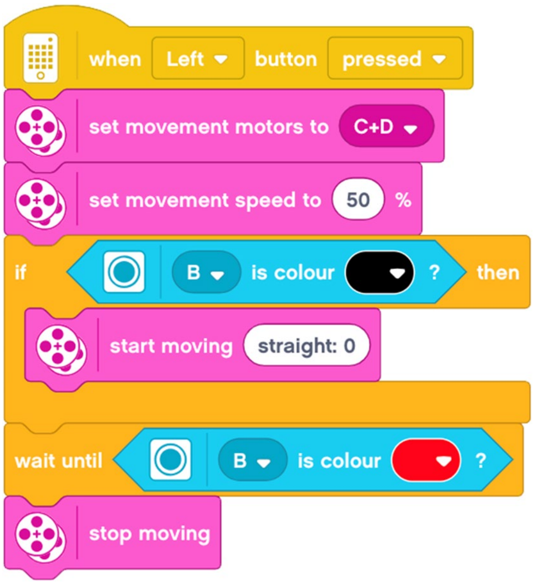
1. S2: But?
2. S1: Oh!
3. S1: Maybe we can
4. S2: It should follow that one
5. S1: Mm
6. S2: Follow, is there any follow command? There should be
7. S1: Yeah, I´ll try to check that

How do we get the robot to follow the line? was a recurring gap in all student conversations and as the above example shows, a relation that can be described as *by using a follow-command* (turns 6–7) was suggested for filling this gap.

The next example comes from another group (School A, group 2) who were very quick to construct their program. At the very beginning of their discussion, they concluded that they needed to remove the Stop command in relation to the colour black. A gap was thus noticed, like previous examples, concerning the question of how the robot should follow black (Example 3, turn 1). A relation was suggested to fill this gap: *how should the robot follow black—it should turn, not stop on black* (turn 2) and in the conversation they talked about and looked for commands that could be used to make it turn. The students did not test the code while they were programming and when they in turn 28 concluded that the first part of the program was ready, they did not yet know that it did not work (Fig. 4).

Example 3 We need to have something so it follows the black colour.

1. S1: We need to have something so it follows the black colour, sort of
2. S1: So, it stays like this, turns, not like this, so it only stops when it feels the black
3. [Searching among the code blocks in the menu]
4. S1: Um
5. S1: ***Control***, um
6. S1: Exactly, on all
7. S1: We should, sort of, if ***Control***
8. S2: Um
9. S1: Or ***Event***
10. S1: No, they must be on ***Control***



```
when Left button pressed
set movement motors to C+D
set movement speed to 50 %
if B is colour ? then
    start moving straight: 0
wait until B is colour ?
stop moving
```

Fig. 4 Example 3: We need to have something so it follows the black colour

11. S1: Um you, could you take something like that [if clause] maybe?
12. S1: Puts it there
13. S1: And like that, sort of
14. S1: And then it intends to move
15. S2: It starts to move
16. S1: Yes exactly
17. S2: Darn, it has to, have this
18. S1: Like that, right?
19. S1: Yes exactly, and then ***Wait until***
20. S1: It should be one of these [sensor X register colour Y]
21. S1: There
22. S1: Is ***Colour***
23. S1: It is **red**, yes
24. S2: Yes
25. S1: And then he should, like that
26. S1: And then stop
27. […]
28. S1: So, now the first part is finished
29. S2: Hm
30. S1: Stops at read

The students searched among the different categories of code-blocks (turns 5–10) and finally chose to use the algorithms If–then and Wait until (turns 11, 19–20). The students judged these changes as appropriate for making the robot to follow the black line and the gap was filled in turn 28, *So, now is the first part done*. As mentioned above, this code did not work as the program only made the robot move straight forward and not turn according to the curved line. In the beginning of their coding, the student groups only rarely talked about what the code Start moving (straight: 0) (Demo code 2, Fig. 3) meant in relation to the curved line that the robot should follow. In accordance with the Sensor-Follow Strategy, the construed relation *if colour is black—then go forward* aligns with how we in everyday situations talk about steering a vehicle. The instruction "follow the road straight ahead", for example, means following the road regardless of how curved it is, the thing is that you should not make an exit. In the programming context this analogy does not work. Later in the activity, however, the students began to pay attention to the code Start moving and what it meant for the robot's movement toward the target, as example 4 below showed.

Sensor–wheels

Most student groups needed help from the teacher to use the strategy Sensor–wheels, that is to program the robot so that the movement was gradually adjusted to the sensor's continuous reading of the surface. The teacher encouraged them to use variables regulating the power on the motors. By controlling these variables through the input of the sensors, the wheels of the robot could be rotated with different power and so creating a zigzagged movement over the line. Several students understood that the scanning of the surface was important but had difficulties in how they should program the robot to perform such a movement. Even if they understood that the relation between the sensor and the wheels was the key, they had great difficulties in converting this relation to a code. The scanning of the robot was often described as "searching", in example 4 the students had just tested their program and now they tried to understand how the code worked.

Example 4 It moved like this, it turned, it searched for colours.

1. S2: Look, it [the robot] was searching because it moved like this, searching for what?
2. S1: Look
3. S2: Look at what?
4. S1: They shouldn't, okay, when the right button is pressed it sets the ***movement motors C plus D to 50% always***, it starts running ***zero strength*** and waits until black and continues to move with **power**, and wait until the **colour b** [the light sensor] …
5. S2: …Is white, although this is the white colour, the white colour on this table
6. S1: Yes but
7. S2: It moved like this, it turned, it searched for colours

Later the students asked the teacher for support who then made them pay attention to the logic of programming and what that means in relation to having the robot scanning the surface. In turn 1 the teacher asked them about their program which they had made changes to.

Example 5 You have to tell it what not to do and then you have to tell it what to do.

1. Teacher: But it needs to have, in order to run it, it still needs to have some kind of starting block
2. S1: Yes, it's there, but now we are trying to figure out what is going to happen
3. T: Okay
4. S1: So, it shouldn't be like this
5. T: No exactly, it is not activated now

6. S1: Exactly
7. T: No. Good.
8. S2: It still needs, it moved around like this. It won't find the thing, it's searching
9. T: You have to think like this, either or, it can't look for black if it doesn't know that it should look for white. And it can't look for red if it doesn't know to look for white, black, or red
10. S2: Mm. Okay.
11. T: It can't just look for black, it also needs to learn that it shouldn't look for white
12. S1: Hm
13. S2: That's the hard thing about programming, you have to tell it what not to do and then you have to tell it what to do

One group of students came to use the strategy without direct support from the teacher. Before turn 1, the students had worked for about 30 min in accordance with the Sensor—Follow the line strategy (see examples 1–2 above):

Example 6 It must search somehow.

1. S2: If the colour is black. Then it should set ***movement motors***, it should move straight ahead. So this program should run forever [inserts a conditional block]
2. S1: Or, or
3. S2: It´s supposed to search
4. S2: Or
5. S2: It must move in all possible directions to look for, to find
6. S1: Yes, how will it do that??
7. S1: It must search somehow. Can it, sort of, of shake or something?
8. […]
9. S2: They´re using this [referring to another student group]. Let's remove a few things. There is so much crap [removes blocks of code from the workspace]
10. S2: It should do this immediately. It should do that first
11. S1: Mm
12. S2: ***For ever***, ***if B***
13. S2: Allright! This one. I think [discovers Demo code 1 and inserts parts from it in their program]

In turn 3, S1 suggested that the robot should search, a relation was thus suggested to fill the gap of *how should the robot follow black?—it should search*. In turn 5, further relations were suggested: *moving in all possible directions to search for, to find* which S1 agreed with (turn 6) and a new gap was noticed: *what should the robot do to search*? Maybe because S1 saw the movement of another student group's robot, the Sensor–Wheels strategy results in a jerkily forward movement of the robot, he suggested that the robot should shake (turn 8). A short time passed, the students then found and started using Demo code 1 (Fig. 2) in turn 14. They made some changes to the demo code and relatively quickly got the robot to follow the line the way they wanted, in example 5 they tested the adjusted program. When they succeeded in making the robot following the curved line (turns 4–5), they started talking about the next task (line 7).

Example 7 Perfect movement.

1. S1: Here it comes [the robot]
2. S2: Damn [giggles], what, what movements it makes
3. S1: Perfect movement
4. S2: Check it out, check it out, stop then. Like that. Yeeeees
5. S1: Yeeeees
6. S2: Yeeeees
7. S1: And then, then we must have, then we make a new one like this. **Sensor A** [starts working with the next task]

Rotations–degrees–wheels

This strategy was characterized by the students' using rotations and degrees to program the robot to follow the line. This strategy was used exclusively by the students in school B, perhaps because the line did not have the same curved shape as in school A. The students in school B had experience of programming robots and it is also likely that the strategy that they used was one they have used before and were familiar with. No student group in School B applied the Sensor-Follow line strategy which also may be related to their previous experiences of programming robots. As example 8 showed, the students (School B, Group 1) understanding of the problem, how to program the robot to follow the line, was a question of making it register the surface and adjust its movement accordingly. The students had just started discussing the task before turn 1 (the taped line was green which was registered as blue by the sensors, the surface of the table on which the line was taped on was black).

Example 8 When it sees black it should go back to blue.

1. S1: Yes we can try, when the colour is black
2. S2: Exactly, black, perfect
3. S1: When it sees black it should go back to blue
4. S2: Exactly

The next example showed another student group (School B, Group 2) making similar conclusions. Before turn 1 the students in example 9 had controlled that the sensor read the surface, and just like in school A they started from Demo code 2 when they started programming. They stated that this code meant that the robot would move along the line and that it would stop when the sensor sensed black (turns 1–4).

Example 9 If it senses black it moves along the line.

1. S1: If it senses black it moves…
2. S2: …Exactly
3. S1: …along the line, then it has to stop every time it senses black
4. S2: Exactly
5. S1: And…
6. S2: …and then it moves the other way
7. S1: Shall we test like this?

The two students quickly made changes to Demo code 2 by picking it apart and inserted code blocks that they thought would make the robot to move towards the end of the curved line. They did so by programming the robot to move forward when its sensor registered blue (they placed the robot on the line of green tape) and stop when it registered black (the surface of the table). This code made the robot follow the line up to the curve (Fig. 1) where it stopped. Instead of adjusting the movement forward by using the Sensor-Wheel strategy, they programmed the robot's further movements by estimating, testing, and revising code blocks using Rotation and Start moving left or right with different degrees.

Example 10 Go!

1. S1: Have we fixed that, so it turns?
2. S2: I don´t know, wait. We use forty instead. We use two, no wait. Two rotations [make changes to the code]
3. [Testing]
4. S2: We use fifty rotations [make changes to the code]. Vi use forty-three
5. S1: Go!
6. S2: Because forty, that is, redo it
7. S1: Yes, it needed a little bit more
8. S2: Redo it
9. S1: Go
10. S1: No, that was better, forty-seven [changes to – 47]
11. S1: Go

Several gaps were noticed while they were working with the program, for example regarding what direction the robot would turn and how many rotations the wheels should turn to make the robot reach its destination. The analysis showed that the work of the three groups in School B followed the same pattern. They started by identifying the problem (making the robot adjust its movement as a result of the reading of the surface, examples 8 and 9) and then agreeing on and testing different variations of how the robot could adjust its position and further movement. The gaps noticed and the relations suggested to fill these gaps therefore usually concerned small adjustments such as shown in example 10. Example 11 showed how group 2 discussed whether they should use degrees or rotations to make the robot change its position.

Example 11 Shall we use degrees or rotation?

1. S1: Are you sure, yes, yes, should we use degree or rotation, it will be easier, we will use 90 degrees, okay 90 degrees, when it turns 90 degrees, it should continue
2. S2: When it sees the colour black
3. S1: When it turns it should continue on the green line, I think
4. S2: Should we do this, what it´s called, **repeat if**?

In summary, the observed Rotations-Degrees strategy, the robot was coded to follow the line by using the sensor to check the robot's position at certain destinations along the line, rather than continuously reading the surface.

Summary

The three strategies can be described and visualized as: (1) smooth and continuous movement forward (Sensor-Follow the line), (2) zigzagging movement forward (Sensor–Wheels), and (3) Stepwise movement forward with position adjustments (Rotations–Degrees–Wheels) (Fig. 5). In the beginning of the activity there were a notable difference in the type of gaps that emerged in the student conversations. I School A, the gaps concerned the strategy or how the problem should be addressed, that is how should the robot follow the line? In school B, such gaps were not evident in the beginning of the activity, instead gaps concerned adjustments of the code to realize the chosen strategy.

Both strategy 2 and 3 was successful in that it resulted in the robots reaching their destination. Whether they were equal in regard of how well the code fits the purpose of the task, that is whether the strategy is good or effective will be elaborated on in the Discussion section. It can be noted that strategy (1), Sensor-follow the line, was never realized in a functional code but the smooth movement envisioned could actually be a fine-adjusted adaption of strategy 2.

Discussion

At the same time as policymakers argue that knowledge about programming is important in current society (Raptopoulou, 2021), relatively little is known about how programming can be taught by teachers and how it is learnt by students (Anderhag et al., 2021; Vinnervik, 2022a, 2022b). In the Swedish technology subject syllabus, programming is a mean for learning aspects of technology rather than a content and little guidance is provided for teachers regarding what and how programming could be implemented (Vinnervik, 2022a). In this study we have addressed aspects of teaching and learning programming by investigating what strategies students use when they are programming a robot to follow a curved line in the technology class.

Programming an object to follow a line is a rather conventional teaching activity and in accordance with the Swedish syllabus of the technology subject, which states that students should learn to steer an object by using programming (Swedish National Agency for

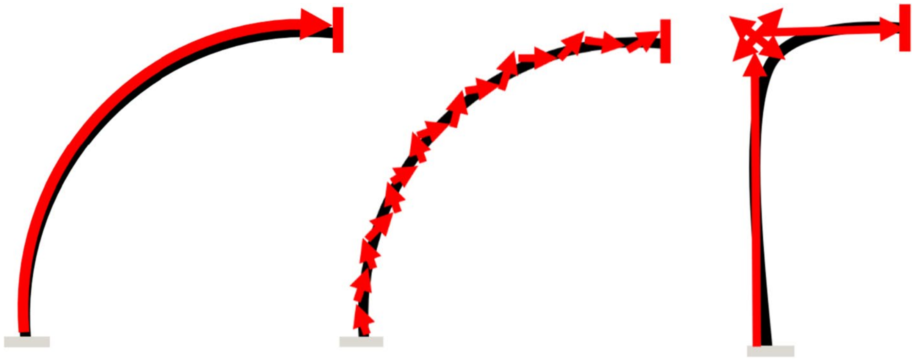


Fig. 5 Showing a simplified visualization of the movements of the robots following strategy Sensor-Follow the line (1), Sensor-Wheels (2), and Rotations-Degrees-Wheels (3). The Sensor-Follow the line strategy was never realized in a functional code

Education, 2018). The study showed that the students used three different strategies for steering the robot: (1) *sensor-follow the line*, where students discuss and try to implement a code that makes the robot follow the line without communicating with the wheels of the robot, (2) *sensor–wheel*, where codes were used to create a feedback system in which a light sensor continuously register the surface and where changes in colour resulted in different movements of the wheels, and (3) *rotations-degrees-wheel*, where students tested and revised the movement along the line by changing the number of rotations of the wheels or how many degrees the wheels should turn. The first strategy could not be realised in a functional program. Our findings support previous studies showing that programmable robots can be usable tools for novice learners (Misirli & Komis, 2014), also students with little or no experiences of programming could participate fruitfully in the activity. The direct feedback on the code by the movement of the robot seemed to be important for student engagement (e.g. Komm et al., 2020) during the activity the students almost exclusively talked about the code in relation to anticipated and observed robot movement. The students thus continuously identified and made judgements on the functionality of the technical solution they worked with, which has been argued to be an important aspect of what constitutes technological knowledge (Anderhag et al., 2021; Björkholm, 2014; Cederqvist, 2022). In line with the findings of Gonçalves et al. (2019), the students in our study spent much time testing and debugging code, our findings contribute by showing how these discussions were aligned with the three strategies. Depending on the strategy, students actively looked for and tested codes affecting different aspects of the sensor-wheel system, such as for example power, rotations or turning. The study did not find the differences between novice and advanced programmers as reported by Merkouris and Konstantinos (2020). In their study, advanced programmers were shown to use and remix large parts of code and while novice students tended to reuse and test small parts of code in iterative cycles. The most salient difference between experienced and novice students in our study was that the advanced students were faster to articulate a reasonable solution to the problem that they worked with, which is in line with the findings of Merkouris and Konstantinos (2020). The more advanced students also spent less time navigating and searching for code blocks, which is expected as they are familiar with the programming environment.

The strategies presented have implications for teaching technology and could be used by teachers when planning and designing learning activities. All student groups succeeded in programming a robot to follow the line and stop at the designated place. However, as the study has showed, "follow" is constituted differently in the three strategies. In strategy 2, follow amounts to a code that adjusts the movement of the robot, so the sensor always is placed above the line. In strategy 3, follow is realized through a code that is fine tuned to a specific route. In this strategy, the independent movement of the robot is not guided by the line but rather the programmer's evaluation of the movement along the specific line the robot is following. Strategy 1, finally, differs from the other two in that follow is supposed to be realized through an explicit follow code, thus automatically constructing the necessary feedback system between sensor and wheels. Although more studies are needed to see whether the strategies presented here also is evident in other contexts, a starting point for teaching could thus be discussing what follows mean in relation to a self-driving vehicle. The everyday understanding of self-driving vehicles, where following a line or a road means just that, could thus be problematized, and discussed in relation to what kind of feedback system that is necessary for that to happen. It is not surprising that students may look for a code block named "When sensor register black—follow", and so ignoring that following needs to be continuously orchestrated through the input from the sensor and the output of the wheels.

The everyday understanding of programming is likely to support such a position and the code block for stopping the movement of the robot is actually constructed this way in the programming environment used in the study, "When sensor register red—stop", thus not being transparent of the relation between the sensor and the wheels. Another aspect that teaching could address is the functionality of the code in strategies 2 and 3. Both strategies were successful as they did what they were supposed to do, that is making the robot follow a line. However, as discussed above, the meaning of follow and the code that realizes it can be discussed in relation to different scenarios and settings, such as static and dynamic worlds (e.g. Parson & Sklar, 2004). Strategy 3, for example, only works if the robot always starts at the same place—what are the advantages and disadvantages of such a program? Strategy 2 have the potential of being functional even if the route changes, may there be scenarios when you still want to use another strategy? Would the code work if the line was curved to the left rather than to the right? How could the program—movement of the robot be optimized further? Although not being the scope of the article, it can be noted that the students were occupied with an important aspect of what constitutes technology knowledge, namely how feedback processes operate in a technical system (see Cederqvist, 2022). We can only speculate but some aspects of the strategies here extracted, such as how the students handle the relations between the components of the system or whether they actually make use of feedback processes, may be of relevance also in other contexts than programming.

Finally, the findings of the study are extracted through micro-analysis of student and teacher conversations and other methodologies could provide fruitful avenues for further deepen the understanding on the strategies students use when programming feed-back dependent robot movements. The empirical material of the study comes from two technology classrooms, and we cannot make any claims regarding how prevalent the strategies are. A more quantitative set up could thus be used to explore whether the strategies here presented also are evident in other contexts. Moreover, the fact that the students differed in their previous experiences of programming and also encountered slightly different routes that they should adapt the movements of their robots to, any suggestions to why some students use one strategy before another would be speculations. Further studies are thus needed and an interesting set up would be to explore how we as teachers can plan for activities that better support students in testing and evaluating the functionality of the strategies presented here.

Acknowledgements We would like to thank the research environment Stockholm Teaching & Learning Studies (STLS) that is financed by school organizers and Stockholm University.

Funding Open access funding provided by Stockholm University.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

Anderhag, P., Björn, M., Fahrman, B., Lundholm-Bergström, A., Weiland, M., & Wållberg, T. (2021). Kod som teknisk lösning: en studie om grundskoleelevers uppfattningar av ändamålsenlighet i deras spontana programspråk. [Code as technical solution: A study on primary school students´ perception of fitness of purpose in their spontaneous programming language] *Nordic Studies in Science Education, 17*(1), 113–129. https://doi.org/10.5617/nordina.7020.

Beraza, I., Pina, A. & Demo, B. (2010). Soft & hard ideas to improve interaction with robots for kids & teachers. In *Workshop proceedings of SIMPAR 2010 Intl. Conference on simulation, modelling and programming for autonomous robots*, (pp. 549–557).

Bers, M. U., & Horn, M. S. (2010). Tangible programming in early childhood. In I. R. Berson & M. J. Berson (Eds.), *High-tech tots: Childhood in a digital world* (pp. 49–70). Greenwich, CT: Information Age Publishing.

Björkholm, E. (2014). Exploring the capability of evaluating technical solutions: A collaborative study into the primary technology classroom. *International Journal of Technology and Design Education, 24*(1), 1–18. https://doi.org/10.1007/s10798-013-9240-1

Cederqvist, A.-M. (2022). An exploratory study of technological knowledge when pupils are designing A programmed technological solution using BBC Micro:bit. *International Journal of Technology and Design Education, 32*, 355–381. https://doi.org/10.1007/s10798-020-09618-6

Cesaretti, L., Storti, M., Mazzieri, E., Screpanti, L., Paesani, A., & Scaradozzi, D. (2017). An innovative approach to school-work turnover programme with educational robotics. *Mondo Digitale* 16, 2017–2015. Available online at: https://mondodigitale.aicanet.net/2017-5/best_papers_didamatica_2017/MD72_03_paper_64.pdf.

Danielsson, A. T., Berge, M., & Lidar, M. (2018). Knowledge and power in the technology classroom: A framework for studying teachers and students in action. *Culture Studies of Science Education, 13*, 163–184. https://doi.org/10.1007/s11422-016-9782-0

Fahrman, B., Norström, P., & Gumaelius, L. (2020). Experienced technology teachers' teaching practices. *International Journal of Technology and Design Education, 30*, 163–186. https://doi.org/10.1007/s10798-019-09494-9

Francis, K., Khan, S., & Davis, B. (2016). Enactivism, spatial reasoning and coding. *Digital Experiences in Mathematics Education, 2*, 1–20. https://doi.org/10.1007/s40751-015-0010-4

Gonçalves, J., Lima, J., Brito, T., Brancalião, L., Camargo, C., Oliveira, V., & Conde, M. A. (2019). Educational robotics summer camp at IPB: A challenge based learning case study. In *Proceedings of the Seventh International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'19)* (pp. 36–43). Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3362789.3362910

Green, T., Wagner, R., & Green, J. (2018). A look at robots and programmable devices for the K-12 classroom. *TechTrends*. https://doi.org/10.1007/s11528-018-0297-2

Hallström, J. (2018). Ett forskningsfält i tillväxt. Teman i svensk teknikdidaktisk forskning. In K. Stolpe, G. Höst & J. Hallström (Eds.), *Teknikdidaktisk forskning för lärare. Bidrag från en forskningsmiljö* (pp.77–93). Naturvetenskapernas och teknikens didaktik, 2

Hallström, J., Hultén, M., & Lövheim, D. (2014). The study of technology as a field of knowledge in general education: Historical insights and methodological considerations from a Swedish case study, 1842–2010. *International Journal of Technology and Design Education, 24*, 121–139. https://doi.org/10.1007/s10798-013-9252-x

Highfield, K. (2010). Robotic toys as a catalyst for mathematical problem solving. *Australian Primary Mathematics Classroom, 15*, 22–27.

Jones, A., Buntting, C., & de Vries, M. J. (2013). The developing field of technology education: A review to look forward. *International Journal of Technology and Design Education, 23*(2), 191–212. https://doi.org/10.1007/s10798-011-9174-4

Kazakoff, E. R., Sullivan, A., & Bers, M. U. (2013). The effect of a classroom-based intensive robotics and programming workshop on sequencing ability in early childhood. *Early Childhood Education Journal, 41*(4), 245–255. https://doi.org/10.1007/s10643-012-0554-5

Kelly, G. J., McDonald, S., & Wickman, P.-O. (2012). Science learning and epistemology. In K. Tobin, B. J. Fraser, & C. J. McRobbie (Eds.), *Second international handbook of science education* (pp. 281–291). Dordrecht: Springer, Netherlands. https://doi.org/10.1007/978-1-4020-9041-7_20

Komm, D., Regez, A., Hauser, U., Gassner, M., Lütscher, P., Puchegger, R., & Kohn, T. (2020). Problem Solving and Creativity: Complementing Programming Education with Robotics. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE*

'20) (pp. 259–265). Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3341525.3387420

Kurebayashi, S., Kamada, T., Kanemune, S., & Kuno, Y. (2007). The effect of learning programming with autonomous robots for elementary school students. In *11th European Logo Conference, Bratislava*, (pp. 1–9). Comenius University Press, Bratislava.

Merkouris, A. & Konstantinos, C. (2018). Programming touch and full full-body interaction with a remotely controlled robot in a secondary education STEM course. In *Proceedings of 22nd Pan Pan-Hellenic Conference on Informatics (PCI '18)* (pp. 5). ACM, New York, NY, USA. https://doi.org/10.1145/3291533.3291537

Misirli, A., & Komis, V. (2014). Robotics and programming concepts in early childhood education: A conceptual framework for designing educational scenarios. In C. Karagiannidis, P. Politis, & I. Karasavvidis (Eds.), *Research on e-learning and ICT in education* (pp. 99–118). New York, NY: Springer. https://doi.org/10.1007/978-1-4614-6501-0_8

Newhouse, C. P., Cooper, M., & Cordery, Z. (2017). Programmable toys and free play in early child-hood classrooms. *Australian Educational Computing, 32*(1), 199–212.

Norström, P. (2014). How technology teachers understand technological knowledge. *International Journal of Technology and Design Education, 24*(1), 19–38. https://doi.org/10.1007/s10798-013-9243-y

Parsons, S, & Sklar, E. (2004). Teaching AI using LEGO Mindstorms. In *Proceedings of AAAI Spring Symposium*, (pp. 1–6).

Raptopoulou, A. (2021). *Politics of contemporary education policy: the case of programming in the Swedish curriculum*. Diss. Stockholm: Stockholm University.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education, 13*(2), 137–172. https://doi.org/10.1076/csed.13.2.137.14200

Swedish Research Council (2017). God forskningssed [Good Research Practice]. Stockholm: Vetenskapsrådet [Swedish Research Council].

Swedish National Agency for Education. (2018). Curriculum for the compulsory school, preschoolclass and school-age educare.

Vinnervik, P. (2022a). Implementing programming in school mathematics and technology: Teachers' intrinsic and extrinsic challenges. *International Journal of Technology and Design Education, 32*, 213–242. https://doi.org/10.1007/s10798-020-09602-0

Vinnervik, P. (2022b). An in-depth analysis of programming in the Swedish school curriculum—rationale, knowledge content and teacher guidance. *Journal of Computers in Education*. https://doi.org/10.1007/s40692-022-00230-2

Wickman, P.-O. (2004). The practical epistemologies of the classroom: A study of laboratory work. *Science Education, 88*, 325–344. https://doi.org/10.1002/sce.10129

Wickman, P.-O., & Östman, L. (2002). Learning as discourse change: A sociocultural mechanism. *Science Education, 86*, 601–623. https://doi.org/10.1002/sce.10036

Xia, L., & Zhong, B. (2018). A systematic review on teaching and learning robotics content knowledge in K-12. *Computers & Education, 127*, 267–282. https://doi.org/10.1016/j.compedu.2018.09.007

Authors and Affiliations

Per Anderhag[1] · Niklas Salomonsson[2] · Andre Bürgers[3] · Cesar Estay Espinola[4] · Birgit Fahrman[5] · Dana Seifeddine Ehdwall[6] · Maria Sundler[5]

✉ Per Anderhag
per.anderhag@su.se

[1] Department of Teaching and Learning, Stockholm University, Stockholm, Sweden

[2] Education and Administration, Stockholm Municipality, Stockholm, Sweden

[3] Nacka Gymnasium, Nacka Municipality, Nacka, Sweden

[4] Children and Education Administration, Huddinge Municipality, Huddinge, Sweden

[5] Department of Learning, Royal Institute of Technology, Stockholm, Sweden

[6] Education Administration, Botkyrka Municipality, Botkyrka, Sweden