Geldreich K., Hubwieser P. (2020) Programming in Primary Schools: Teaching on the Edge of Formal and Non-formal Learning. In: Giannakos M. (eds) Non-Formal and Informal Science Learning in the ICT Era. Lecture Notes in Educational Technology. Springer, Singapore. https://doi.org/10.1007/978-981-15-6747-6_6

Programming in Primary Schools: Teaching on the Edge of Formal and Non-Formal Learning

KATHARINA GELDREICH – School of Education, Technical University of Munich, Germany, katharina.geldreich@tum.de

PETER HUBWIESER - School of Education, Technical University of Munich, Germany, peter.hubwieser@tum.de

Abstract: While several countries have already introduced Computer Science or programming into their primary school curricula (e.g. the UK, Australia or Finland), Germany has not yet developed mandatory guidelines on how to deal with these matters. Although there is an agreement that students of all ages should gain insight into the recognition and formulation of algorithms, the focus in primary school is often still on the mere use of computers. Programming courses, on the other hand, are increasingly found in extracurricular activities. It is still open to what extent and in what form algorithms and programming can and should be introduced in primary schools in the longer term. To help answer this question, we trained 40 primary school teachers in algorithms and programming and examined how they implement the topics in their individual schools. Among these are teachers who teach programming in class (formal learning) as well as teachers who offer their students extra-curricular programming activities on a voluntary basis (non-formal learning). We interviewed all teachers about how they implemented the topics, what advantages they saw in the individual formats and what challenges they encountered. In this paper, we outline our didactical approach as well as the results of our interview study.

Keywords: Programming, Primary School, Teacher Training, Interviews

1. INTRODUCTION

In recent years, the discussion about the necessity of Computer Science (CS) and especially programming in primary education has grown steadily (Webb et al. 2017, Bell and Duncan 2018). The early development of key understanding, skills, and thinking approaches emerging from CS seems to have several positive effects on children. Learning to use computers not only as users but also as creators and gaining positive experiences in computing can strengthen their self-confidence in CS and technology in general (Duncan et al. 2014, Topi 2015). It may also prevent common misconceptions and prejudices towards CS regarding the nature of the subject and the role of gender (Moorman and Johnson 2003, Engeser et al. 2008, Funke et al. 2016). In addition, computational thinking – which is generally defined as the mental activity of abstracting problems and formulating automatable solutions (Wing 2006) – has the potential to improve students' problem-solving skills in other subjects as well (Yadav et al. 2014).

Several countries have already included aspects of CS in their primary school curricula, e.g. Australia (Falkner et al. 2014), Finland (Kwon and Schroderus 2017), the UK (Brown et al. 2013) and Switzerland (D-EDK 2016). Apart from these formal learning settings, there are numerous non-formal offerings aimed at promoting children's interest in and knowledge of CS. They offer the opportunity to deal with CS, even if it is not part of the curriculum. Many of them focus on programming or

coding, such as the website *code.org*[1] or the programming clubs *Code Club*[2] and *Coder Dojo*[3]. These extracurricular activities are voluntary and can provide experiences that are not anchored in the curriculum or not possible in regular classroom settings (Lunenburg 2010).

It is still unclear to what extent and in what form CS and programming can and should be introduced in primary education in the longer term, and what role extra-curricular offers should play in this context. To help answer these questions, we wanted to include the opinions and experiences of primary school practitioners. We trained 40 primary school teachers in algorithms and programming and examined how they implement the topics in their schools. We did not specify the setting of this implementation – both formal and non-formal formats were possible. In the course of a school year, we conducted exploratory interviews with all teachers. The focus was set on the following research questions:

- Which are the most common settings for introducing algorithms and programming?
- What advantages to the teachers see in the respective settings?
- What challenges and limitations do the teachers encounter?

In this article, we first give a brief introduction to CS in primary education and extra-curricular offerings on CS. We also give an insight into the teaching concept the teachers got to know as part of their teacher training. Afterwards, we will describe the research design and methods of our study as well as the results of the interviews. After discussing the results, we will give an outlook on our future research.

2. BACKGROUND AND RELATED WORK

One can see an increasing consensus in CS Education that beginning to learn CS in primary school is not only possible but also beneficial for learning as well as developing self-esteem and motivation (Webb et al. 2017). Besides, research is being conducted into how computational thinking and CS can and should be integrated into other subject matters (Yadav et al. 2014, Weng and Wong 2017, Friend et al. 2018).

Although Germany has not yet developed binding guidelines for dealing with the topics, the relevance of CS in primary school is becoming increasingly evident. In its strategy paper on education in the digital world, the German KMK[4] states that competencies on *recognizing and formulating algorithms* should be included in the curricula of all school types (KMK 2017). The German Informatics Society (GI) goes even further and formulates competencies in five different content areas that students should develop during primary school (Best et al. 2017). There are also various research efforts focusing on how we can allow children to acquire basic knowledge in the field of CS and which methods and contents are suitable for German primary schools (Diethelm and Schaumburg 2016; Gärtig-Daugs et al. 2016; Geldreich et al. 2016; Bergner et al. 2017, Goecke and Stiller 2018; Magenheim et al. 2018).

However, there is only little work on German primary school teacher's beliefs and opinions on CS. Funke et al. (2016) conducted an interview study with six primary school teachers without any previous experience in CS. In this study, they conclude that the interviewed teachers have no concrete picture of CS in primary school but do have some beneficial preconceptions and attitudes. Best (2019) conducted semi-structured interviews with eleven primary school teachers without any relevant prior knowledge about their views on CS as a discipline and subject in primary school. He repeated the interviews with three teachers after they had gained first teaching experience with Bee-Bots[5]. The findings show that the teachers consider CS education to be important for primary school, but also for the lives and future

[1] https://code.org/

[2] https://codeclub.org/

[3] https://coderdojo.com/

[4] Kultusministerkonferenz (literally "conference of ministers of education") is the assembly of ministers of education of the German states.

[5] https://www.tts-group.co.uk/bee-bot-programmable-floor-robot/1015268.html

careers of the students. The opinions where this education should take place were heterogeneous: as an independent subject, integrated into several subjects, integrated into one subject or as an extracurricular activity. They assumed that boys have a higher interest in CS than girls and are convinced that this must be counteracted already in primary education.

There are further international studies that focus on teachers' experiences and perspectives. Sentence et al. (2017) interviewed 15 teachers about their use and experience of the micro:bit[6], a physical computing device. They categorize different approaches and instructional styles to teaching with physical computing and identify teachers who can be classified as either *inspirers*, *providers* or *consumers*. Black et al. (2013) conducted a questionnaire-based study on teachers' perceptions of how to make computing interesting for students. Out of 115 responses from British CS teachers, several factors were identified as most important for engaging students. Based on the results, they give specific recommendations where teachers should be supported in this matter. Yadav et al. (2016) examine the experiences and challenges that novice CS high school teachers face in the classroom. They conducted 24 semi-structured interviews and identified several challenges, including isolation, lack of adequate CS background, and limited professional development resources. Duncan et al. (2017) analyzed the feedback of 13 teachers participating in a study that examines the implementation of new primary school topics based on computational thinking in New Zealand. The teachers had no previous experience in teaching CS and volunteered to take part in a program where they receive professional development and support to integrate computational thinking and CS in their teaching. They were asked to complete a feedback form each time they taught a session that focused on CS or computational thinking. Based on these feedback forms, they identified ways in which the teachers could integrate computational thinking into their current teaching, the key concepts they were able to engage students with, and their confidence in delivering the material.

3. CONTEXT

This study is part of a two-year project called *AlgoKids – Algorithmen für Kinder (in English: "Algorithms for Children"*), which is funded by the Bavarian Ministry of Education. The project investigates how primary school teachers can be prepared and supported to teach the topics *algorithms* and *programming* in Bavarian primary schools. In addition, both the implementations and experiences of the teachers are scientifically analyzed and evaluated. In two multi-day professional development trainings, the participating teachers received the opportunity to expand their computing knowledge (Geldreich et al. 2018). After the training, they were provided with additional online material as well as the possibility to seek further support if required.

The project is based on an already field-tested and evaluated programming course for primary school, which is aimed at third and fourth-grade students (Geldreich et al. 2019). We have tested it in practice with whole school classes as well as an extracurricular activity with children who have participated voluntarily. During the project, the teachers implemented the course at their school. They have not been told in which subject context this should take place and whether they should approach the topics in a formal or non-formal setting.

The course includes unplugged activities as well as working with the visual programming language Scratch (Maloney et al. 2010). At the end of the course, the students should understand that a device is following an algorithm that is implemented by programming the device. They should also get familiar with the process of testing and debugging a program and get to know the basic algorithmic structures *sequence*, *selection,* and *iteration*. At the same time, the course promotes the computational thinking skills of *algorithmic thinking* (e.g. follow algorithms, create algorithms to solve problems), *decomposition* (breaking down problems into

[6] https://microbit.org/

smaller steps), *logical reasoning* and *evaluation* (e.g. identifying possible solutions and choosing the best one) (Berry 2015). The course concept is described in the following.

3.1 What is an Algorithm?

Since most of the students do not have any prior knowledge in programming or CS in general, the first step is to give them a basic idea of how computer programs work. They initially work unplugged, i.e. without a computer, and program in everyday language. In the first step, they program the teacher – she or he plays a robot and is supposed to perform small tasks in the classroom, such as opening the window. Since the teacher only follows particular commands, the children quickly realize that each step in an algorithm must be formulated in an understandable, precise and unambiguous way. Larger actions must be broken down into sub-steps. It is also explored where they encounter algorithms in their everyday lives, for example in the form of handicraft instructions or recipes. In different tasks the students practice describing sequences in natural language, for example, they convert a pictorial instruction into unambiguous language-based commands (see Fig. 1).

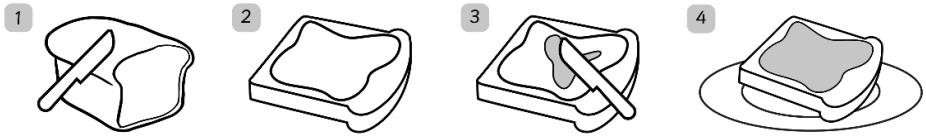

Figure 1: Pictorial representation of making a sandwich

3.2 Programming unplugged

Subsequently, the description of algorithms is further explored. The students use everyday language, symbols, and haptic Scratch blocks to program each other and solve different tasks (Fig. 2, left). As soon as a task has been solved, the solution can be executed in a grid and checked for mistakes (Fig. 2, right). This way, they can physically experience what later happens in a programming environment. We have designed the tasks in a way that allows them to be solved by using *selections* and *iterations*, but also by *sequences*.

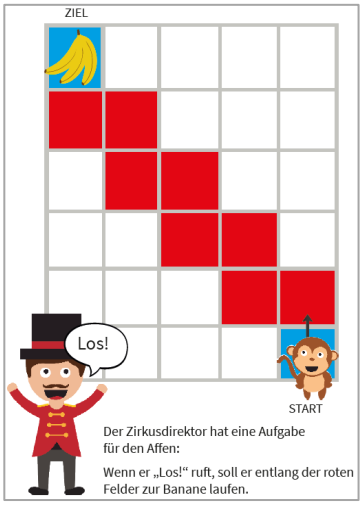

Figure 2: Task (left) and corresponding grid in the classroom (right)

3.3 Programming in Scratch

After these unplugged exercises, the students are introduced to the programming environment Scratch. To enable the students to concentrate on using the Scratch programming environment, they first work on some tasks they already have solved unplugged. Next, they work on a learning circle in which the core operations of Scratch are gradually introduced and which the students can master at their own pace (Fig. 3). Starting from questions regarding software handling, the stations lead from

simple *sequences* to the implementation of *selections* and *iterations*.

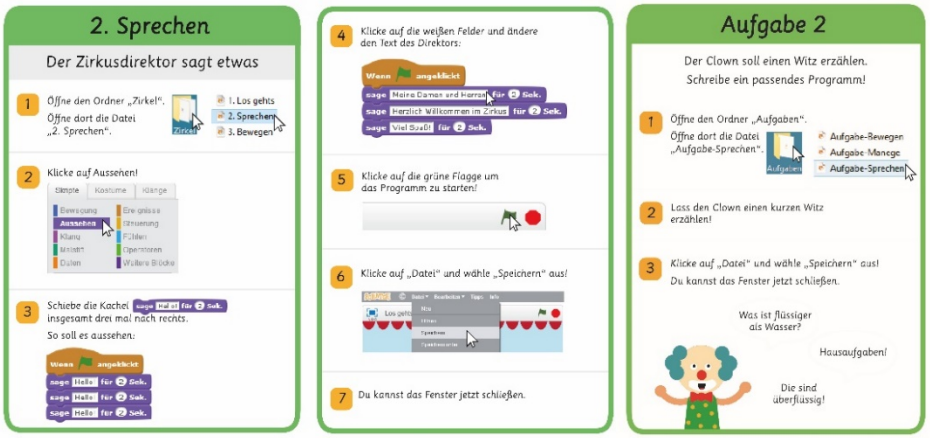

Figure 3: Station of the Scratch learning circle

3.4 Planning Programs

The last step in the teaching concept leads the students to plan and implement their own program ideas. In individual or partner work, the students think up their own Scratch story, write it down in a script (Fig. 4) and implement it in Scratch. To get comparable results, we set the following mandatory requirements for the students' projects. The programs should 1) work on more than one sprite 2) move the sprites during execution 3) comprise at least one loop and 4) include at least one conditional statement. After meeting these requirements, the students could continue their programming work without any further guidelines. The children present their programs in front of the class and are given the opportunity to comment on their projects.

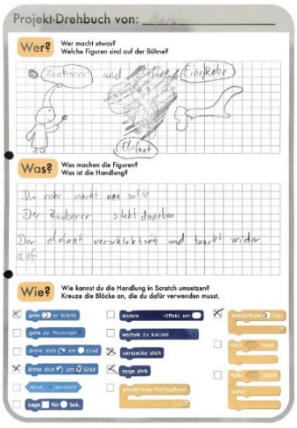

Figure 4: Project script

4. METHODS

4.1 Participants

The twenty schools which participate in the project were selected by the Bavarian Ministry of Education. In order to reflect the Bavarian school landscape, they chose primary schools from all government districts. To make a comprehensive selection, they also took into account the size, experience in digital education and technical equipment of the schools. In total, we worked with 40 teachers – two from each primary school (two males, 38 females). The age of the participants ranged from under 30 years to over 50 years, while the group of 30-40-year-olds made up the largest part (see table 1). Across both groups, 27 teachers had no previous experience in CS at all, 13 teachers had CS for 1-3 years as an elective or compulsory subject in school. We also assessed whether the teachers participated on their initiative or the initiative of their principal, or whether the initiative was evenly divided between the two. The answers were distributed almost equally among the three possible options.

Table 1: Age Distribution of participating Teachers

age	*number of teachers*
under 30 years	8
30-40 years	16
41-50 years	9
older than 50 years	7
	40

4.2 Data Collection

To get insights into the implementations and experiences of the teachers that took part in the project, we conducted exploratory interviews. The exploratory interview is not – like the classical interview – an asymmetrical form of communication. Although there is still a separation of roles between the *interviewer* and the *interviewee*, the interview situation is a quasi-normal conversation (Honer 2011). The exploratory interview does not follow any specific rules, the questions, however, should be asked as openly as possible. Nevertheless, the interviewer always has the possibility to follow up on interesting points or to steer the conversation in a certain direction with suitable questions (Ullrich 2006).

With few exceptions, the interviews in our study were conducted jointly with both teachers of each school. They were led by one researcher who has already given the teacher training and who knew the teachers well. They were asked to tell what they have done with the students so far and in which context they introduced *algorithms* and *programming*. In the course of the interviews, it was also discussed what learning gains they had observed among their students, what challenges they encountered and whether they were able to identify differences between boys and girls. The interviews were audio-recorded and transcribed. In the following, we present selected results from the exploratory interviews that relate to the setting the teachers introduced *algorithms* and *programming*. We conducted a total of 19 interviews, which lasted between 30 minutes and two hours.

4.3 Data Analysis

The data were analyzed within the qualitative data analysis software MAXQDA. Based on our research questions, we first categorized the transcripts regarding two main categories: formal and non-formal learning settings. Following grounded theory, we then started with open coding by attaching codes to the teacher statements (Corbin and Strauss 1990). In an inductive process, we searched for emerging patterns by grouping codes from both main categories (Glaser in Walsh et al. 2015). The overall objective at this point was to create themes that should lead to a structure for reporting our results.

5. RESULTS

Although the teachers were provided with all resources from our teaching approach, they were free to modify or expand the materials or develop own learning materials and scenarios. Even if all teachers have followed our teaching concept in general, there was considerable variation in the specific setting of the implementations and their experiences. Teachers from fourteen schools implemented programming exclusively in a formal setting, in three schools they offered programming clubs in a non-formal setting. Two schools collected experience in both settings. We report on the data in relation to five areas that emerged from the analysis – all areas contain results that refer to both formal (F) and non-formal (N-F) implementations:

- Implementation in school
- Student engagement
- Teachers' confidence
- Challenges and concerns
- Gender issues

All interviews were conducted in the German language. The anchor examples below were translated into English by the authors.

5.1 Implementation in school

Both the teachers who implemented programming in regular lessons and those who offered it as an extracurricular club considered it a useful activity for the students:

I think, on the one hand, it's very motivating, it's modern, it's simply a medium that children have to deal with in a meaningful way. On the other hand, with all these unplugged modules beforehand, we don't just place them in front of computers and let them do whatever they want. The precise formulation, bundling an idea and implementing it within Scratch as a program - this is highly complex. (F)

I think it helps the students to think in a more structured way. They have to make a plan in their heads – they can try things out, but they also have to think about it carefully. That's often not the case in regular lessons. (F)

For many teachers, it is a challenge that programming is not anchored in the curriculum. They would like to have more freedom in the timetable to allow them to implement such topics more flexibly. At the same time, however, many think that it could be problematic for a lot of teachers if it were required in future curricula:

It's a pity that it's not in the curriculum because there is so much potential in the children. You could really tap into that. They are so motivated and have no inhibitions and fears. (F)

On the one hand, it should be anchored in the curriculum, otherwise, nobody will do it. On the other hand, I also find it difficult to institutionalize it - how do you want to assess the performance of the students? (F)

I believe that interested teachers implement it, whether or not it's part of the curriculum. But many teachers have no affinity for it. And I don't think they would do it even if it was in the curriculum. (F)

I believe that programming could become a new cultural technique in the foreseeable future and that everyone should get insight. But the place for it in primary school has yet to be created. Finding a place in regular classes is difficult. (F)

Some teachers have opted for an extracurricular offer because they cannot provide enough time for programming during regular lessons. In addition, it was mentioned that only those children who are really interested in the topic sign up for a club:

We have outsourced programming into a club. It would be difficult for us to implement it in everyday school life. (N-F)

If you offer a programming club, you would always have a designated time for that. And you have children who are really interested in it. (F)

There were children in the club who made a conscious decision to participate. They find it cool to learn more about Computer Science. (N-F)

The majority of the teachers are in favor of programming being included in the curriculum of primary schools. However, there is disagreement about the context in which this should or could happen:

In mathematics, you could include sessions about giving precise instructions – because mathematics works similarly. You must follow a certain sequence of commands or rules when you do a calculation. German lessons would also be possible – they could write a recipe or other instructions. (N-F)

It's something interdisciplinary. It has something of mathematics, of language, of everything. In the curriculum there is the area „media education", but it is very vague and easy to avoid. It would have to be made much clearer in the curriculum how something like this can be linked with the other subjects. (F)

It is also nice when an expert comes from outside and offers an activity for the children. But that's always this one special project day – and it shouldn't be like that. You could do a lot during regular lessons. (N-F)

Although all teachers considered the unplugged activities in our teaching concept

to be necessary and have had positive experiences with them, they find it important to program on the computer as well:

> *I don‘t quite understand the idea of only doing the preliminary work for programming and to program unplugged exclusively. Of course, you can build understanding for the algorithmic structures – but isn't it like coffee without milk? (F)*

5.2 Student engagement

A recurring theme in the interviews – whether programming was implemented in a formal or non-formal setting – was the emphasis on the students' enjoyment of the sessions and the high level of engagement they demonstrated. Several teachers pointed out, that they were surprised about the engagement of individual students:

> *It is so nice when the students leave and say: "Wow, that was such an awesome lesson today!“ They have such great achievements. (N-F)*

> *All the students were very interested. Some children, who are otherwise very reserved, suddenly became really active. (F)*

> *I can see that children in the club are developing real enthusiasm. They've already bought Scratch books, registered in the online community and share their projects there. They even told me their older brothers and sisters started programming because they told them about it. (N-F)*

There were several comments from teachers who implemented programming in a non-formal setting where they stated that all students should have the experience of learning to program:

> *The motivation lasted the whole school year. If the club is canceled for any reason, the students asked me in the schoolyard: "Why is there no programming this week?” I would have wished that more students could have joined the programming club. (N-F)*

Several teachers started computing with the entire class and later thought about diving deeper with students that showed the most interest:

> *We programmed half a school year with the entire class. Then, we thought about offering a club in the second half of the year. We have a lot of children who are really interested and could explore it in depth. (F)*

Some teachers who have introduced programming in regular classes have expressed concerns about the seriousness of the activity or whether students are learning what was intended:

> *Everything was very simple and playful. I don't think they've realized yet that this is Computer Science – programming is a lot of fun for them. (F)*

> *On the next level, I want programming to become a little more serious. It's not just about coding funny things – I want them to think about how to program specific actions. But at the same time, I don't want to slow them down. They are so full of joy and imagination. (F)*

> *It must have added value. For sure, it’s good for motivation. They enjoy programming a lot. There is a benefit in that because if they enjoy coming to class, they learn something. But do they always learn what they are supposed to learn? (F)*

5.3 Teachers’ confidence

Many teachers were worried they would not be able to answer all the questions of the students. Some teachers first tried out certain contents and methods with a few students and only then ventured into a larger group:

> *At first, I tried out some exercises and methods with a few children from my*

class. We went to the computer room once a week for two months. After that, I felt comfortable to run the programming club on my own. Also, because I knew that I had your concept and material which I could stick to. A lot of things grew out of that. (N-F)

Some teachers noticed that they adopted a different teacher role than usual when programming with their students and felt quite comfortable with that. Despite some initial concerns, many even saw an advantage in not always knowing all the answers:

The role of the teacher is as it should be in exploratory learning. One can approach the individual children, respond to them, advise them. They decide what suits them best, think for themselves, become active and are not satisfied with ready-made solutions. They can bring in their ideas again and again. (F)

I was often clueless; stood by a student and had to admit that I had no idea. But that was also great because students realized that teachers aren't perfect either. And you grow together when you work on problems together. Sometimes the students came up with the solution – sometimes I came up with it. That was a great collaboration. (F)

Some teachers noted that before the project they were not at all interested in programming and now see it as a personal enrichment:

I am very grateful that I had the chance to participate in the project. It's so much fun and I've discovered hidden talents in myself. As a woman, I had the attitude that I wasn't interested in Computer Science. Well, I am now! (N-F)

Although some teachers have had positive experiences with programming as an extracurricular offer, they have reservations about programming with the whole class due to the high number of students:

Sometimes I wish there was a second person in the club with me. This person could help if the computer won't start or help the students when I'm busy. But the children are relaxed and know that sometimes it takes a while. They help each other a lot or just keep trying to solve the problem on their own. But in class, I have 29 children – that would be difficult to handle alone. (N-F)

5.4 Challenges and Concerns

The most frequent challenges for the teachers concerned the technical equipment of the school and not being able to respond adequately to all the needs and questions of the students:

We were always two teachers when we programmed in class – that was OK. It would've been hard if I had been alone. The organization, these adversities with the equipment, that's all difficult. (F)

Technical infrastructure and time are major problems. We don't have any system support at school and so I installed Scratch on all computers for a whole day. That's why it could fail – you save on staff and teachers are expected to do all the work voluntarily. (F)

For many teachers, it was a challenge to meet the different skills and knowledge levels of the students. Also, the use of a computer was a problem for many children:

One student left the club after a while. He was already very advanced and had already programmed in C – his father is a computer scientist. The other students had never heard of programming. (N-F)

I had students who already knew how to handle laptops, I had kids who knew Scratch and I had kids who never had any digital device in their hands. Balancing those differences was a big challenge in the beginning. (F)

The handling of a computer is a big problem. How do I scroll down? How do I make a double-click? I had the feeling that many students couldn't get

into the depth of programming because of this. (F)

When asked if they could imagine programming regularly with the whole class, they expressed conflicting concerns about the students' performance:

> *I'm a little worried that at some point we'll reach a level where I can't help the students anymore. That gives me a bit of a stomachache because that doesn't happen to me in any other subject. You reach your limits at some point. That's not a problem with single programming sessions - but if we were to program a whole school year regularly. (F)*

> *I believe there are children who, even in the fourth grade, are not yet so far advanced in their cognitive abilities. They have simply already reached the maximum of their development with the other subjects in fourth grade. (F)*

5.5 Gender issues

When talking about differences between girls and boys, the teachers were positively surprised that girls were also interested in programming:

> *I had already offered a computer club before. The girls didn't want to participate at all and said they were not capable of that. But with the programming club, it was different – many girls volunteered and wanted to take part. (N-F)*

> *Making positive experiences with Computer Science is important. I have noticed that many girls have discovered hidden abilities and got a sense of achievement — programming is not just for boys and isn't something they don't understand. (F)*

Several teachers reported that the boys had more experience with computers and were more involved with them at home:

> *I'd say the boys are better at handling the computer. Which is probably just because they have more contact with it at home. That doesn't mean they can do it better in general. But I think they just have more experience with it. Whereby girls have more patience when something doesn't work. (F)*

> *The boys in my club are the ones who are more involved with it at home. They sign up in the online community, download Scratch, get books and program at home. They approach me with specific project ideas they got at home and want to implement it in the club. I haven't heard that from the girls yet. (N-F)*

6. DISCUSSION

Returning to the research questions mentioned in Section 1, we first wanted to investigate in which setting the teachers in *AlgoKids* introduce the topics *algorithms* and *programming*. Out of a total of twenty schools, fourteen schools exclusively chose a formal setting during regular school days and three schools decided to offer an extracurricular programming club in an informal setting. Two schools decided to test both settings. It should be noted that in Bavaria the school administration must approve all extracurricular activities. These hours are then added to the teachers' working time. As there is currently a shortage of teachers at many primary schools, club lessons are often not approved.

As an advantage in favor of programming in a formal learning setting, it is mentioned that programming generally helps students to develop a more structured thinking and all children should be given this opportunity. At the same time, it could be an opportunity to reduce the gender gap regarding the students' interest in CS and the abilities in using the computer. Individual statements show that the family home can have a great influence on this previous knowledge. To ensure social justice, one would have the chance to take countermeasures in class. Another advantage of a formal setting was initially perceived as worrying by some teachers - the changing

teacher role. However, after gaining initial experience, teachers reported that they enjoyed the changing role and were even able to build a better connection with their students.

The missing legitimacy in the primary school curriculum and the associated lack of time is primarily cited as a challenge for programming in regular classes. Besides, there are often problems with technical equipment and rarely proper system administrators. The teachers, who programmed in a formal setting, were concerned about the seriousness of the lessons and wondered if the students would actually learn the things they intended to. They also wondered how they would assess the students' results. Some concerns were expressed that it would be a pity to force a creative activity like programming into the framework of a regular school subject.

The advantages of a non-formal setting result from the disadvantages of the formal one. There is a fixed time frame available and there is no need to link the lessons to the curriculum. One could focus on fun and motivation of the children without the pressure of achieving predetermined learning goals. Additionally, one can control the size of the group and encourage only suitable or interested students to join the club. As a major downside of implementing programming in a non-formal environment, teachers point out that not all students are given the opportunity to participate.

Concerning our methodology – the exploratory interview – we can say that it was well suited for our purpose. We wanted to create a pleasant atmosphere for the teachers in which they could freely share their opinions and views with us. The rather open interview situation was suitable for this. However, we also think that it is difficult to create this atmosphere if you don't know each other at all. It was helpful that we knew the teachers beforehand. It was only possible in some cases to interview the teachers of the individual schools separately. When analyzing the interviews, however, we determined that the speech proportions in the group interviews were balanced and that the respective teachers also expressed very different opinions.

7. CONCLUSIONS AND FUTURE DIRECTIONS

With the introduction of new curricula covering CS and computational thinking a and the growing market of out-of-school coding activities for children, it is important to include the opinions of experts in the field - primary school teachers.

In our interviews, teachers mentioned some concerns and challenges of implementing programming in a formal setting, but these were mostly of a more practical nature and related to the concrete implementation in individual schools. When it came to whether they found it useful for the students, almost all of them agreed that all students should have the opportunity to learn programming. The fact that programming is not included in the Bavarian primary school curriculum is a (mostly time-related) problem for many teachers and should not be underestimated.

When the project is finished, we will make recommendations to the Bavarian Ministry of Education on how programming could be implemented in primary schools and where teachers would draw the line between formal and informal education. In our future work, we will try to revise the course concept according to the teachers' remarks. For example, more programming units could be developed that relate directly to existing parts of the curriculum.

REFERENCES

Bell, T., & Duncan, C. (2018). Teaching Computing in Primary Schools. In S. Sentance, E. Barendsen, & C. Schulte (Eds.), *Computer science education*. Bloomsbury Academic.

Bergner, N., Köster, H., Magenheim, J., Müller, K., Romeike, R., Schroeder, U., & Schulte, C. (2017). Zieldimensionen für frühe informatische Bildung im Kindergarten und in der Grundschule. In I. Diethelm (Ed.), *Informatische Bildung zum Verstehen und Gestalten der digitalen Welt* (pp. 15–24). Gesellschaft für Informatik.

Berry, M. (2015). *QuickStart Primary Handbook*. BCS.

Best, A. (2019). Bild der Informatik von Grundschullehrpersonen: Ergebnisse eines mehrjährigen Projekts zu informatikbezogenen Vorstellungen. In A. Pasternak (Ed.), *Informatik für alle* (pp. 59–68).

Best, A., Borowski, C., Büttner, K., Freudenberg, R., Fricke, M., Haselmeier, K., Herper, H., Hinz, V., Humbert, L., Müller, D., & Thomas, M. (2019). Kompetenzen für informatische Bildung im Primarbereich. *LOG IN*, *38*(1), 1–36.

Black, J., Brodie, J., Curzon, P., Myketiak, C., McOwan, P. W., & Meagher, L. R. (2013). Making Computing Interesting to School Students: Teachers' Perspectives. In J. S. Downie (Ed.), *Proceedings of the 13th ACM/IEEE-CS Joint Conference on Digital Libraries* (pp. 255–260). ACM.

Brown, N. C. C., Sentance, S., Crick, T., & Humphreys, S. (2013). Restart: The Resurgence of Computer Science in UK Schools. *ACM Transactions on Computing Education*, *1*(1).

Corbin, J. M., & Strauss, A. (1990). Grounded theory research: Procedures, canons, and evaluative criteria. *Qualitative Sociology*, *13*(1), 3–21. https://doi.org/10.1007/BF00988593

Deutschschweizer Erziehungsdirektorenkonferenz. (2016). Medien und Informatik. In D-EDK (Ed.), *Lehrplan 21*. https://v-ef.lehrplan.ch/lehrplan_printout.php?e=1&k=1&fb_id=10

Diethelm, I., & Schaumburg, M. (2016). IT2School – Development of Teaching Materials for CS Through Design Thinking. In A. Brodnik & F. Tort (Eds.), *Informatics in Schools: Improvement of Informatics Knowledge and Perception* (Vol. 9973, pp. 193–198). Springer. https://doi.org/10.1007/978-3-319-46747-4_

Duncan, C., Bell, T., & Atlas, J. (2017). What do the Teachers Think? Introducing Computational Thinking in the Primary School Curriculum. In D. Teague & R. Mason (Eds.), *Proceedings of the Nineteenth Australasian Computing Education Conference (ACE 2017)* (pp. 65–74). The Association for Computing Machinery. https://doi.org/10.1145/3013499.3013506

Duncan, C., Bell, T., & Tanimoto, S. (2014). Should your 8-year-old learn Coding? *Proceedings of the 9th Workshop in Primary and Secondary Computing Education*, 60–69. https://doi.org/10.1145/2670757.2670774

Engeser, S., Limbert, N., & Kehr, H. (2008). *Studienwahl Informatik: Abschlussbericht zur Untersuchung*.

Falkner, K., Vivian, R., & Falkner, N. (2014). The Australian digital technologies curriculum: Challenge and opportunity. In J. Whalley & D. D'Souza (Eds.), *Proceedings of the Sixteenth Australasian Computing Education Conference*. ACM.

Friend, M., Matthews, M., Winter, V., Love, B., Moisset, D., & Goodwin, I. (2018). Bricklayer: Elementary Students Learn Math through Programming and Art. In T. Barnes, D. Garcia, E. K. Hawthorne, & M. A. Pérez-Quiñones (Eds.), *Proceedings of the 49th ACM Technical Symposium on Computer Science Education—SIGCSE '18* (pp. 628–633). ACM Press. https://doi.org/10.1145/3159450.3159515

Funke, A., Berges, M., & Hubwieser, P. (2016). Different Perceptions of Computer Science. In S. Iyer & N. Thota (Eds.), *2016 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)* (pp. 14–18). IEEE. https://doi.org/10.1109/LaTiCE.2016.1

Funke, A., Geldreich, K., & Hubwieser, P. (2016). Primary school teachers' opinions about early computer science education. *Proceedings of the 16th Koli Calling International Conference on Computing Education Research - Koli Calling '16*, 135–139. https://doi.org/10.1145/2999541.2999547

Gärtig-Daugs, A., Weitz, K., Wolking, M., & Schmid, U. (2016). Computer science experimenter's kit for use in preschool and primary school. In J. Vahrenhold & E. Barendsen (Eds.), *Proceedings of the 11th Workshop in Primary and Secondary Computing Education* (pp. 66–71). ACM. https://doi.org/10.1145/2978249.2978258

Geldreich, K., Funke, A., & Hubwieser, P. (2016). A Programming Circus for Primary Schools. In A. Brodnik & F. Tort (Eds.), *Informatics in Schools: Improvement of Informatics Knowledge and Perception* (pp. 46–47). Springer.

Geldreich, K., Simon, A., & Hubwieser, P. (2019). A Design-Based Research Approach for introducing Algorithmics and Programming to Bavarian Primary Schools. *MedienPädagogik: Zeitschrift Für Theorie Und Praxis Der Medienbildung*, *33*(Medienpädagogik und Didaktik der Informatik), 53–75.

Geldreich, K., Talbot, M., & Hubwieser, P. (2018). Off to new shores: Preparing primary school teachers for teaching algorithmics and programming. *Proceedings of the 13th Workshop in Primary and Secondary Computing Education on - WiPSCE '18*, 1–6. https://doi.org/10.1145/3265757.3265783

Goecke, L., & Stiller, J. (2018). Informatische Phänomene und Sachunterricht. Beispiele für vielperspektivischen Umgang mit einem Einplatinencomputer. In M. Thomas & M. Weigend (Eds.), *Informatik und Medien: 8. Münsteraner Workshop zur Schulinformatik*. Books on Demand.

Honer, A. (2011). Das explorative Interview: Zur Rekonstruktion der Relevanzen von Expertinnen und anderen Leuten. In A. Honer & R. Hitzler (Eds.), *Kleine Leiblichkeiten* (pp. 41–58). VS Verlag für Sozialwissenschaften / Springer Fachmedien Wiesbaden GmbH Wiesbaden.

Kultusministerkonferenz (Ed.). (2017). *Strategie der Kultusministerkonferenz „Bildung in der digitalen Welt". Beschluss der Kultusministerkonferenz vom 08.12.2016 in der Fassung vom 07.12.2017*. KMK. https://www.kmk.org/fileadmin/Dateien/veroeffentlichungen_beschluesse/2018/Strategie_Bildung_in_der_digitalen_Welt_idF._vom_07.12.2017.pdf

Kwon, S., & Schroderus, K. (2017). *Coding in Schools: Comparing Integration of Programming into Basic Education Curricula of Finland and South Korea*. Finnish Society on Media Education.

Lunenburg, F. C. (2010). Extracurricular Activities. *Schooling*, *1*(1).
Magenheim, J., Schulte, C., Schroeder, U., Humbert, L., Müller, K., Bergner, N., & Fricke, M. (2018). Das Projekt Informatik an Grundschulen. *LOG IN Informatische Bil-dung Und Computer in Der Schule*, *189/190*, 57–66.
Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The Scratch Programming Language and Environment. *ACM Transactions on Computing Education*, *10*(4), 1–15. https://doi.org/10.1145/1868358.1868363
Moorman, P., & Johnson, E. (2003). Still A Stranger Here: Attitudes Among Secondary School Students Towards Computer Science. *ACM SIGCSE Bulletin*, *35*(3), 193. https://doi.org/10.1145/961290.961564
Sentance, S., Waite, J., Yeomans, L., & MacLeod, E. (2017). Teaching with physical computing devices. *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, 87–96. https://doi.org/10.1145/3137065.3137083
Topi, H. (2015). Gender imbalance in computing. *ACM Inroads*, *6*(4), 22–23. https://doi.org/10.1145/2822904
Ullrich, P. (2006). *Das explorative ExpertInneninterview*. Technische Universität Berlin. https://doi.org/10.14279/DEPOSITONCE-4745
Walsh, I., Holton, J. A., Bailyn, L., Fernandez, W., Levina, N., & Glaser, B. (2015). What Grounded Theory Is…A Critically Reflective Conversation Among Scholars. *Organizational Research Methods*, *18*(4), 581–599. https://doi.org/10.1177/1094428114565028
Webb, M., Davis, N., Bell, T., Katz, Y. J., Reynolds, N., Chambers, D. P., & Sys\lo, M. M. (2017). Computer science in K-12 school curricula of the 2lst century: Why, what and when? *Education and Information Technologies*, *22*(2), 445–468. https://doi.org/10.1007/s10639-016-9493-x
Weng, X., & Wong, G. K. W. (2017). Integrating computational thinking into english dialogue learning through graphical programming tool. *2017 IEEE 6th International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, 320–325. https://doi.org/10.1109/TALE.2017.8252356
Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33. https://doi.org/10.1145/1118178.1118215
Yadav, A., Gretter, S., Hambrusch, S., & Sands, P. (2016). Expanding computer science education in schools: Understanding teacher experiences and challenges. *Computer Science Education*, *26*(4), 235–254. https://doi.org/10.1080/08993408.2016.1257418
Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational Thinking in Elementary and Secondary Teacher Education. *ACM Transactions on Computing Education*, *14*(1), 1–16. https://doi.org/10.1145/2576872

ACKNOWLEDGMENTS

We would like to express our special thanks to all teachers involved in the project for their openness, curiosity, and commitment.