Computational Thinking for All – An Experience Report on Scaling up Teaching Computational Thinking to All Students in a Major City in Sweden

Fredrik Heintz
Linköping University, Sweden
fredrik.heintz@liu.se

Linda Mannila
Linköping University, Sweden
linda.mannila@liu.se

CCS CONCEPTS

• **Social and professional topics → Computing education**; **K-12 education**; **Computational thinking**;

KEYWORDS

K-9 education, teacher professional development, digital competence, programming, computational thinking

ACM Reference Format:
Fredrik Heintz and Linda Mannila. 2018. Computational Thinking for All – An Experience Report on Scaling up Teaching Computational Thinking to All Students in a Major City in Sweden. In *SIGCSE '18: SIGCSE '18: The 49th ACM Technical Symposium on Computer Science Education, February 21–24, 2018, Baltimore , MD, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3159450.3159586

1 INTRODUCTION

The increased exposure to technology raises a need for understanding how the digital world works, in the same manner as we get to know the physical world. Consequently, during recent years, we have witnessed an active discussion surrounding the role of programming and computer science (CS) for everyone (see e.g. [6, 9, 13]). As a result, an increasing number of countries have introduced or are in the process of introducing CS in their school curriculum. For instance in Europe, the majority of countries (17 out of 21) taking part in a survey conducted by the European Schoolnet in 2015 reported doing so [1]. The way in which this is accomplished varies. Some countries focus on K-12 as a whole, whereas others primarily address either K-9 or grades 10-12. Some countries have introduced CS as a subject of its own (e.g. Computing in England [3]) while others have decided to integrate it with other subjects, by for instance making programming an interdisciplinary element throughout the curriculum (e.g. Finland [5]). The role of CS and information technology in school curricula has – in general – varied over the years, placing focus on different areas, ranging from using technology as a tool to learning how the computer works and how to use it to create programs. This has also been the case in Sweden.

SIGCSE '18, February 21–24, 2018, Baltimore , MD, USA

ACM ISBN 978-1-4503-5103-4/18/02…$15.00
https://doi.org/10.1145/3159450.3159586

Introducing new content in curricula affects many teachers. When the content is new, such as programming and digital competence, most of the teachers affected have no prior experience in teaching the content. Consequently there is a large need for professional development and training initiatives. In this paper we present our experience from a three year long project, aiming at training Swedish teachers (grades 1-9) in teaching programming and computational thinking.

Although the Swedish government decided on including programming in the curriculum as late as in March 2017, the discussion on this had already been vivid since around 2014. To those involved it was more a question of when this would happen, rather than if. As a result, many projects focusing on programming and digital competence at primary and lower secondary school, were initiated already several years ago. For instance, Sweden's innovation agency Vinnova funded several such projects already in 2014. One of these projects was "A model for computational thinking in Swedish primary school", which received renewed funding under the new project name "Computational thinking for all" in 2016. Both are lead by the authors of this paper.

The first project aimed at introducing programming and computational thinking to a small group of teachers, who then were to implement the ideas and plans created in their own classroom. The later project, built on the previous one, now involving a larger group of teachers, who were not only to implement the ideas in their own classroom, but also to spread it to other teachers at their schools.

Both projects took place in Sweden's fifth largest city, Linköping, as a collaborative effort between the Computer Science Department at Linköping University and Linköping Municipality. The university was responsible for the project and all the project activities, while the municipality took care of the administration at the city level and the contact with the teachers.

The rest of the paper is structured as follows. We start in Section 2 with a background on digital competence, programming and computational thinking in Swedish education. In Section 3, we present the first part of our project, the pilot study. In Section 4, the second and larger part of the project is presented. In Section 5, we discuss the lessons learned from the projects and in Section 6, we draw conclusions and give some recommendations for the future.

2 DIGITAL COMPETENCE IN SWEDISH K-9 EDUCATION

The role of computer science and IT in Swedish schools has varied throughout the years [10]. In fall 2015, the Swedish government gave the National Agency for Education (Skolverket) the task of

preparing a proposal for K-12 education on how to better address the competences required in a digitalized society. In June 2016, Skolverket submitted a proposal putting a much stronger emphasis on digital competence and introducing both digital competence and programming as interdisciplinary traits. It also provides explicit formulations in subjects such as mathematics (programming, algorithmic thinking, and problem solving), technology (controlling physical artifacts) and social sciences (fostering aware and critical citizens in a digital society). In March 2017, the government accepted the proposal, which has to be implemented by fall 2018 at the latest. In October 2017, the government also decided on a National IT Strategy which was significantly weaker than the one Skolverket proposed in June 2016.

The Swedish school debate has in recent years circled around poor PISA results, difficulties in providing all children and youth with equal opportunities, and about modernizing the curriculum in order to meet future job market requirements. As "programmer" is the most common job in the capital Stockholm, and the need for software professionals is estimated to increase heavily both in Sweden and internationally, some have argued that the education system should teach programming in order to prepare young people for these jobs. Others, the authors of this paper included, believe that school should offer all students general preparation for any kind of work, and have therefore argued for digital competence as part of all-round-learning, including computational thinking as a set of general problem solving skill useful for all in the spirit of Jeannette Wing [14].

In 2012, the Swedish government established the Digitalization Committee (Digitaliseringskommissionen) with the task of providing guidelines for the future of work related to digitalization in Sweden. One of the committee's reports [4] highlights the need for the school system to put larger focus on digital competence. The report explicitly points out the need for including programming in the curriculum as part of existing subjects. As a result of the discussion around schools, programming and CS as part of all-round learning, persons representing school, universities and industry engaged in voluntary initiatives to help overcome the lack of CS in Swedish basic education. Teacherhack (http://teacherhack.com) is a non-profit organization aiming at inspiring "teachers to hack the current curriculum (Lgr 11) to include the essential skills that students need in a digital world". The Teacherhack website provides reviews of all subjects in Lgr 11 with practical advice on how the current texts can be interpreted in order to allow for a more active inclusion of content and practices related to CS, programming and the Internet, as well as security and integrity issues.

Extracurricular activities such as CoderDojos, code camps, after-school clubs and makerspace activities are organized to give children and youth access to informal learning opportunities. Teachers throughout the country are experimenting and sharing experiences from introducing programming in different subjects, ranging from languages to handicraft and music. Heintz et. al. [8] present an overview of ongoing activities related to CS and computational thinking in Sweden, highlighting several projects that show how one can introduce CS already within the current curriculum.

In September 2015, the Swedish government gave the National Agency for Education (Skolverket) the task of presenting a national IT strategy for the Swedish school system. As one part of this work, Skolverket was to update the curricula for primary (K-9) and upper secondary education (grades 10-12). The government explicitly stated that the curriculum should 1) strengthen students' digital competence and 2) introduce programming at K-9 level.

In March 2017, the Swedish government accepted Skolverket's proposal. The revised curriculum will be mandatory starting in fall 2018, but schools have the option to introduce it already in fall 2017.

The revision introduces a new general section on digital competence. Skolverket acknowledges that the meaning of digital competence changes over time due to changes in society, technology and available services [11]. Skolverket's definition is based on the set of key competences developed by the European commission [12] and the work by the Digitalization Committee [4]. In the Swedish curriculum, digital competence includes four aspects: 1) understanding how digitalization affects individuals and society, 2) understanding and knowing how to use digital tools and media, 3) critical and responsible usage of digital tools and resources, and 4) being able to solve problems and implement ideas in practice. In a supplemental material [11], Skolverket stresses that helping students develop their digital competence is a cross-curricular responsibility and aspects of digital competence should hence be covered in all subjects. Programming is considered part of this definition. The supplemental material clarifies that focus is not on coding skills, but on programming as a pedagogical tool and a problem solving process including many phases. Programming should also be seen in a wider context, including "creation, controlling and regulating, simulations and democratic dimensions" [11, p.10, freely translated]. Skolverket emphasizes the importance of seeing programming in this wider perspective both as a basis for teaching and as part of all four aspects of digital competence.

For a more detailed overview see [7].

3 THE PILOT STUDY

The first project was a pilot study in the spring of 2015. It involved 10 teachers in different subjects and from different schools, who had been selected by the municipality. Their background varied, from those that had been working with Bebras (http://bebras.org) and Hour of Code (http://code.org) to those that had absolutely no previous experience. The study took place during one semester and included three 3 hour workshops. The goal was for each teacher to carry out at least three activities with their students. In the end more than 300 students from more than 14 classes from grade 1 to grade 9 participated in computational thinking activities as part of the pilot study.

The first workshop was a lecture style introduction to computational thinking, which gave a motivation to why it is important, an introduction to what it is, concrete examples of computational thinking in different subjects, both with and without computers, and an introduction to programming using ScratchJr/Pyonkee. To introduce computational thinking, tasks from the international Bebras challenge were used. To introduce programming without a computer, material from CS Unplugged [2] was used. To introduce programming we used Hour of Code and ScratchJr/Pyonkee. The fact that these resources were easily available simplified our work notably.

The second workshop was a workshop style discussion around how to introduce computational thinking in the participating teachers' particular subjects. The teachers were divided into groups based on their subject. There were three groups: Swedish/language teachers, math teachers, and science/technology teachers. Each group was given the explicit task to come up with at least two activities related to computational thinking that they could carry out in their classes in the coming month: one unplugged activity and one involving a tablet or a computer. After the group discussions, the whole group discussed the suggested activities together. The workshop ended with each teacher having to commit to doing one unplugged activity, one Hour of Code session, and one programming activity using a digital device (tablets were more common than computers) before the third and final workshop.

At the third workshop the teachers presented what they had done in their classes and we discussed their experience and lessons learned. The teachers also filled out an evaluation form for each activity they had completed. In total we received information about 17 activities. One example of an activity that a teacher developed introduced a treasure hunt covering angles and fractions in mathematics. The students were given a grid map of the school yard and a sequence of instructions on the form "walk 1 3/4 squares forward", "turn 270 degrees to the right", etc. They then had to calculate where the treasure was hidden before actually executing the program to see if they could find the treasure. This is a good example of combining developing computational thinking skills with outdoor activities.

The final review of the pilot study revealed four major lessons learned:

(1) The teachers were in general positive and felt that the training made it possible for them to adapt the material provided and run an activity as part of their own teaching. Nevertheless, they only did this once and as far as we know the teachers did not continue to develop more activities after the pilot study was over.
(2) The teachers reported that other students than the usual suspects did best on the activities. Students that were usually quiet and low key were more excited and engaged in the activities than normally.
(3) According to teacher's experience, Scratch and ScratchJr worked well at lower grade levels, while students in grade 7-9 were not motivated by the cute graphics and cartoons. One teacher used Code Combat (http://codecombat.com/) instead together with grade 9 students, which he reported worked well.
(4) The students were in general positive towards the activities and engaged in them.

The conclusions from the pilot study are that it was definitely possible to get teachers to introduce computational thinking in their teaching with a limited amount of professional development, as long as it was directly connected to their subject. However, it did not seem to get a lasting effect. This seems to be a general observation, it is relatively easy to do one or a few activities, but it is much harder to make it part of the standard practice and integrate it into everyday teaching.

4 CT FOR ALL

As the conclusions from the pilot study were positive, while it was clear that this only affected a small number of students, the continuation project focused on addressing the question on scaling up. The basic question is: Now that we know how to get individual teachers to start including computational thinking in their classes, how can we scale this up in both the number of teachers/students and also in the regularity/longevity of the activities. The general plan for the project was to provide professional competence development to at least one teacher at each school and to build up a central support function within the municipal administration to support the teachers. The goal of the project was that 80% of all students in the municipality should have at least one activity per month related to computational thinking.

4.1 Organization

The municipality has 50 schools and about 14000 students in comprehensive education (grade 1-9) organized in 5 school districts. Due to the hierarchical organization, the communication from the central municipal administration is always through the school district, which can decide whether to forward information to the principals, or not. Each principal decides whether information should be forwarded to the teachers and, if so, whom to inform. All communication from the project to the teachers were taken care of by the project's municipality representative, who is also a teacher.

The original plan was to build an organization with one teacher from each school and a support group within the central administration consisting of one representative from each of the school districts and one representative from the central education development office (who would also be responsible for the whole project). We as the university representatives were outside the organization as our role was to bootstrap the change. Our responsibility was to provide teacher training, expert advice and to study the introduction of computational thinking in the schools.

Even though the central municipal administration has been very supportive and positive we have not managed to form a central support group and the representative from the municipality has been replaced three times during the project period of three years. Based on this experience, the school organization appears to be rather volatile with people constantly changing positions, but it could also be a coincidence that the municipality was in a shift of staff.

Luckily, we have managed to get representatives from more than 40 out of the 50 schools, which means that we actually reach at least 80% of the schools. Initially we got about 30 teachers in the spring of 2016, which was increased to about 70 teachers from the fall of 2016 after the head of education had sent out a request to all the schools. The group was mostly unchanged during fall 2016 and spring 2017, but the engagement and activity of these teachers varied substantially. Normally about 40 of the teachers showed up to the workshops. Finding suitable workshop times and getting teachers to commit is both hard and important.

4.2 Workshops and Activities

The workshops have been the backbone of the project. We have arranged three half-day workshops per semester at the university,

resulting in total 12 workshops during 2016 and 2017. Each workshop has had a theme and a program including both information or new material from us and discussions to activate the teachers. We have also given the teachers assignments to do between the workshops.

In addition to the workshops we have also encouraged participation in events such as Bebras and Hour of Code. We have also developed a handbook with computational thinking activities and an introductory material for teaching computational thinking. This material is freely available in Swedish as it is designed for Swedish teachers.

Workshop Program.

(1) Introduction to computational thinking, overview of the proposed new curriculum, introduction to Bebras, Hour of Code, and Scratch Jr. *Discussion:* What support do you need to implement the new curriculum?
(2) Assessment of computational thinking skills. *Discussion:* How to assess digital competence and programming in the new curriculum?
(3) Introduction to our handbook on computational thinking. *Discussion:* Now that you have learned the basics, how should you proceed? This was the first workshop with the full group, so we had a parallel session for the new teachers, where we summarized the content of the first two workshops to bring them up to speed.
(4) Bebras and models for introducing programming and computational thinking in K-9. Hands on programming exercises for those that were new to programming and a seminar on the computer science behind Scratch for the more experienced.
(5) The results from Bebras and introduction to Hour of Code. Programming in Python for those with programming experience and a hands on introduction to Hour of Code for those that were new to programming.
(6) Presentation of the introductory material for computational thinking. *Discussion:* Experiences from trying out Bebras and Hour of Code, how can these resources be used in teaching?
(7) Presentation of how others have worked with the new curriculum. Workshop on Micro:bit (http://microbit.org) and Swift Playgrounds (http://apple.com/swift/playground). *Discussion:* What do you think of the introductory material to computational thinking and how does it work in your class?
(8) From block programming to textual programming and programming and algorithms in mathematics. *Discussion:* How will you introduce programming in your teaching this fall?
(9) Progression and more on algorithms in mathematics. *Discussion:* What should students know after grade 3, grade 6 and grade 9? (The Swedish curriculum lacks details, so it is up to teachers to interpret the curriculum.)
(10) Spreading to other teachers at the same school. *Discussion:* How to spread the workshop contents and lessons learned to other teachers, get all teachers involved in order to have continuity at school level (instead of the level of digital competence teaching at a given school being dependent on a single teacher personally driving the change)?
(11) Lessons learned and moving forward. *Discussion:* How will you continue the work now that you have to work more independently?

4.3 Intro to Computational Thinking Package

To provide the teachers with a joint basic material on how to introduce computational thinking, we put together a small resource package. The material defines the main computational thinking concepts, provides examples of concrete activities and exercises that the teacher can use in his or her classroom, and presents a model for how to assess the attitude and maturity of the computational thinking of the students.

The material covers five concepts:

(1) step by step instructions (or how the computer works);
(2) detecting and finding patterns;
(3) breaking down a problem into smaller parts;
(4) abstraction and representation; and
(5) algorithms and programming.

The progression basically follows these concepts, so teachers start with the first and work through them one by one.

The material also considers seven attitudes:

(1) dealing with complexity;
(2) dealing with ambiguity and open problems;
(3) adapting solutions to new situations;
(4) evaluating own and others solutions;
(5) experimenting and troubleshooting;
(6) grit; and
(7) communication and collaboration.

The learning objectives of the material, when used in teaching, is for students to:

- know that a computer does things step-by-step;
- have experience working with different types of problems where he/she has benefited from or has developed concepts and attitudes related to computational thinking;
- recognize computational thinking as a problem solving process together with computers that are based on a set of concepts and attitudes; and
- be able to assess his/her own level of computational thinking.

The material consists of a set of slides presenting the concepts and attitudes, in addition to two matrices. The first matrix defines the concepts. In addition, concrete activities and examples related to math, technology and other subjects are provided for each concept. The activities are either Bebras tasks or activities from our handbook on computational thinking activities. Most activities can be carried out without a computer. The second matrix provides an assessment tool where each attitude progresses through three stages based on work by Phil Bagge, Mark Dorling, and Thomas Stephens (http://code-it.co.uk/attitudes). Each step is in the form of a concrete question for the student to answer.

4.4 Impact

Measuring the number of students that have participated in the project is challenging. The most specific figure we have is the number of students that participated in the Bebras contest. In the 2016

contest 3756 students from 47 different schools in Linköping participated. This corresponds to close to 30% of the students in grades 2-9, which is three times as many as the year before when the corresponding number was 1277 students. The Linköping students represented more than 40% of all students that participated in Bebras in grade 2-9 at a national level in Sweden. This shows that the project has had a large impact. Unexpectedly, the number of students that participated in Bebras in 2017 fell to 2209 students from 30 different schools. The comments we got indicates that the Bebras tasks are very popular but that the schools do not like to participate in the contest itself. In addition to Bebras, many students participated in the Hour of Code, but for that activity we lack concrete statistics.

The informal impact is very high as almost all schools in Linköping participate in the project and we have been invited to present the work to all school leaders in the municipality. It is very likely that this effort will be the main effort by the municipality to introduce the new curriculum.

5 LESSONS LEARNED

The project has provided many lessons learned.

As mentioned above, the project was organized around a municipality representative taking care of all direct contact with the teachers. This was challenging from two perspectives. First, teachers changing jobs or tasks at their work place led to the municipality representative changing several times throughout the project. This makes it difficult to have continuity in the project even if the representatives have all been very good and active. Second, having one person in charge of all teacher contacts complicated our communication with the teachers, as everything had to go through the middleman. One important lesson learned is hence to have an active coordinator at the municipality level, both for teachers who need somebody they can easily contact and for us as the university. Another, but closely related, observation is that it can be rather difficult to get communication across in large organizations.

Another lesson learned is related to teacher activity and the role played by school leadership. Regardless of the current and highly relevant topic of digitalization in schools, teachers were not able to prioritize the project. There is a need for a clear vision at the leadership level and resources that make it possible for teachers to not only take part in a limited number of workshops, but also to learn more and experiment on their own and together with colleagues. It is quite surprising to us that even if the new curriculum is decided and our program is available for free for the teachers the interest from school leaders is quite low (this could be a consequence of the difficulty of communication as we have no direct contact with school leaders either).

While introducing the basics of programming and algorithms can be considered rather easy, moving beyond unplugged programming, apps, Hour of Code and simple block based programming is not as straightforward. Questions such as what to use after Scratch or how to integrate programming in mathematics and other subjects in grades 7-9, are not as easy to answer. The suggestions made by us often were seemed as too advanced and hard. One lesson learned is hence that progression is important in many respects and that there is a need for teacher centric research in order to learn more about suitable ways to integrate programming at different grade levels. In Sweden, comprehensive school is organized in three main stages: K-3, grades 4-6 and grades 7-9, and it is between these stages that students shift schools and teachers. It is therefore crucial to have a joint progression, to make it easier for teachers at different levels both in order to know what to expect from students when they transition to their stage and to know what minimal level they have to reach before moving on to the following stage.

The question "what to do now that we know Scratch" is also not a straightforward one. Does it imply that the person knows how to use the tool Scratch or actually how to program in Scratch? For most teachers, the question seems to imply the former. They have used the tool quite extensively and feel that they know how to use it, therefore they believe they know how to program. This claim is based on our experience with assuming that the teachers actually had learned to program. For those teachers, who felt that they needed to learn more advanced topics, we tried to introduce text-based programming in a more university style manner. The first attempt was to take Scratch as the starting point, go through the different blocks and constructs and explain the computer science behind them. This turned out to be too complex. The second attempt was to take a subset of the introductory lecture in our university Python course, explaining the basics of programming in Python, and then do a hands on exercise writing a function which computes the maximum of two numbers. This went much better than the first attempt, but it was still clear that the step from block programming to text-based programming is quite big. The step was made even bigger by the fact that many schools in the region have opted for tablets, and teachers had to do all the programming on an iPad. From this we draw two conclusions. First, going from block-based programing to text-based programming is hard. Second, teachers who express that they are ready to move from Scratch probably do not need another tool or programming language, but rather more experience in developing interesting tasks and problems, where students can use Scratch, or some other block-based environment, as a tool solve the task or implement an idea.

The teachers participating in our projects had quite heterogeneous backgrounds, as some did not have any prior experience in programming, while others had already done quite a lot, both on their own and together with their students. This did not cause any problems during the workshops, as the participants were divided into groups with different program based on their background. However, making all activities suitable to everyone, regardless of subject taught and prior background, requires significant resources.

Most teachers in the project have done activities with their class and many have done activities with other classes at their school, but it is much harder to spread it to other teachers at their school. Some of the reasons expressed by the teachers are:

- lack of time;
- lack of mandate, they only have the mandate to participate in these activities not to take their own initiatives at their schools;
- lack of school leadership, in some cases they didn't even have a principal as a new was under recruitment; and
- lack of a clear idea on how to introduce programming and digital competence in a sustainable manner.

6 CONCLUSION

In this paper we have described our experience from introducing programming and computational thinking at a large-scale in a Swedish city. The goal was to reach at least 80% of all the schools and at least 80% of all the students. The plan was to provide professional training to the participating teachers the first semester, support them in carrying out a series of activities during the second semester and then support them to spread their knowledge to their colleagues during the third semester. The goal turned out to be too optimistic. First, it took longer to recruit teachers to the project so we had to restart the professional training the second semester. Second, getting the teachers to activate their colleagues requires both the mandate and the support from their local school leaders, which was outside of our control. Third, large organizations are constantly changing both in terms of directives and in terms of people, so finding a stable backbone is quite hard.

The main conclusions of the project are:

- it is possible to provide good teacher training with relatively modest efforts;
- it is possible to get these teachers to carry out activities in their own classrooms and usually also in other classes;
- the teachers are usually good at adapting the material we present and turn it into their own lessons; and
- it is much harder to get the teachers to do their own local teacher training and to get more local teachers at their schools to adopt the new material as it requires an explicit mandate from the local school leaders.

An observation that might be important is that it seems that the teachers who learn Scratch (or other similar languages), learn it as a tool not as a realization of common programming concepts. When they have learned the tool, they feel that they know programming, but when you start discussing the programming concepts or show how to do the same thing in another language, such as Python, they do not really follow. Scratch in all its greatness also seems to lure people into believing that they know more than they do, which is something we have to be aware of and try to mitigate.

If we were to carry out the project again, the most important thing we would do differently would be to explicitly get the commitment of the principals and the school leaders at the highest level. Having their explicit support would greatly empower the teachers to get more done at their schools and feel that they have a clear mandate and resources needed to inspire to change.

Overall we are satisfied with the project as we have gained valuable and important insights, and we know for a fact that we have reached more than 80% of the schools and at least 30% of all the students in city, probably significantly more.

7 ACKNOWLEDGMENTS

This work is partially supported by Vinnova. We would also like to thank Linköping municipality and all the teachers that have contributed to the project.

REFERENCES

[1] Anja Balanskat and Katja Engelhardt. 2015. Computing our future. Computer programming and coding. Priorities, school curricula and initiatives across Europe. (2015).
[2] Tim Bell, Ian H. Witten, and Mike Fellows. 2015. *CS Unplugged – An enrichment and extension programme for primary-aged students.*
[3] Department for Education. 2013. National Curriculum in England: Computing programmes of study. (2013). https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study.
[4] Digitaliseringskommissionen. 2014. *En digital agenda i människans tjänst : en ljusnande framtid kan bli vår : delbetänkande.* Technical Report SOU 2014:13.
[5] Finnish National Board of Education. 2014. Perusopetuksen opetussuunnitelman perusteet 2014. (2014).
[6] Fredrik Heintz, Linda Mannila, and Tommy Färnqvist. 2016. A Review of Models for Introducing Computational Thinking, Computer Science and Computing in K-12 Education. In *Proc. IEEE Frontiers in Education Conference (FIE).*
[7] Fredrik Heintz, Linda Mannila, Lars-Åke Nordén, Peter Parnes, and Björn Regnell. 2017. Introducing Programming and Digital Competence in Swedish KâĂŞ9 Education. In *Proc. ISSEP.*
[8] Fredrik Heintz, Linda Mannila, Karin Nygårds, Peter Parnes, and Björn Regnell. 2015. Computing at School in Sweden – Experiences from Introducing Computer Science within Existing Subjects. In *Proc. ISSEP.*
[9] Informatics Europe and ACM Europe. 2015. Informatics in Education: Europe Cannot Afford to Miss the Boat. (2015). Report of the joint Informatics Europe and ACM Europe Working Group on Informatics Education.
[10] Lennart Rolandsson and Inga-Britt Skogh. 2014. Programming in school: Look back to move forward. *Trans. Comput. Educ.* 2, 14 (2014), 12:1–12:25.
[11] Skolverket. 2017. Få syn på digitaliseringen på grundskolnivå. (June 2017).
[12] Stephanie Carretero Gomez and Riina Vuorikari and Yves Punie. 2017. DigComp 2.1: The Digital Competence Framework for Citizens with eight proficiency levels and examples of use. (2017).
[13] White House. 2016. Computer Science for All. (2016). https://www.whitehouse.gov/blog/2016/01/30/computer-science-all.
[14] Jeanette Wing. 2006. Computational thinking. *Commun. ACM* 49, 3 (2006), 33–35.