PHD DISSERTATION

ELISA NADIRE CAELI

Computational Thinking in Compulsory Education: Why, What, and How?
A Societal and Democratic Perspective

AARHUS
UNIVERSITY
DANISH SCHOOL OF EDUCATION

UNIVERSITY
COLLEGE
COPENHAGEN

Computational Thinking in Compulsory Education: Why, What, and How?

A Societal and Democratic Perspective

Elisa Nadire Caeli

PhD Dissertation

2021

AARHUS UNIVERSITY
DANISH SCHOOL OF EDUCATION

UNIVERSITY COLLEGE COPENHAGEN KP

Computational Thinking in Compulsory Education – Why, What, and How? A Societal and Democratic Perspective

PhD dissertation submitted to the Graduate School at the Faculty of Arts, Aarhus University

© Elisa Nadire Caeli

Submission date: November 29, 2021

Principal supervisor: Jeppe Bundsgaard, PhD, Professor, Danish School of Education (DPU), Aarhus University

Co-supervisor: Simon Skov Fougt, PhD., Associate Professor, Danish School of Education (DPU), Aarhus University

Normal pages (of 2,400 characters and rounded, excluding bibliography, lists, appendices, etc.):

Overview article:	88 pages
Article A:	18 pages
Article B:	22 pages
Article C:	13 pages
Article D:	21 pages
Article E:	23 pages
Total:	185 pages

This dissertation is designed as "a collection of several academic texts that are related in content and/or methodology and where the results obtained in course of the PhD programme are presented and possibly published, either by the PhD student alone or by the student together with other authors. In addition, the dissertation must include a separate presentation by the PhD student that takes the form of a large-scale overview article", as per "Rules for the PhD Programme at the Graduate School, Arts", chapter 5.2: https://phd.arts.au.dk/fileadmin/phd.arts.au.dk/AR/Generelle_retningslinjer_UK_1-11-2012.pdf

Table of Contents

A. Caeli, E. N. & Yadav, A. (2019). Unplugged Approaches to Computational Thinking: a Historical Perspective. *TechTrends*. AECT, Springer. https://doi.org/10.1007/s11528-019-00410-5

B. Caeli, E. N. & Bundsgaard, J. (2019a). Computational Thinking and Technology Comprehension in K-9 schools: A Round Trip.
Translated from: Caeli, E. N. & Bundsgaard, J. (2019). Datalogisk tænkning og teknologiforståelse i folkeskolen tur-retur. *Læring og Medier (LOM)*, 11(19). https://doi.org/10.7146/lom.v11i19.110919

C. Caeli, E. N. & Bundsgaard, J. (2020). Technology Criticism in Schools – a Democratic Perspective on Technology Comprehension.
Translated from: Caeli, E. N. & Bundsgaard, J. (2020). Teknologikritik i skolen – et demokratisk perspektiv på teknologiforståelse. In Haas, C. & Matthiesen, C. (Eds.): *Fagdidaktik og demokrati*. Samfundslitteratur.

D. Caeli, E. N. & Bundsgaard, J. (2019b). Computational thinking in compulsory education: a survey study on initiatives and conceptions. *Educational Technology Research and Development*. AECT, Springer. https://doi.org/10.1007/s11423-019-09694-z

E. Article E: Caeli, E. N. & Dybdal, M. (2020). Technology Comprehension in Schools. Computational Design for Solving Authentic Problems.
Translated from: Caeli, E. N. & Dybdal, M. (2020). Teknologiforståelse i skolens praksis. Datalogisk design til autentisk problemløsning. *Læring og Medier (LOM)*, 12(22). https://doi.org/10.7146/lom.v12i22.115613

Acknowledgments

It has been a privilege to be able to examine the field of computational thinking and technology comprehension in K-9 in-depth and during my PhD studies I have met a bunch of great people. My studies have been three years of fun and exciting work because of them. Here, I want to acknowledge and thank the ones who have inspired and supported me the most.

First and foremost, I could not have wished for a better principal supervisor than **Jeppe Bundsgaard** who has strengthened my confidence and courage as to the directions I have wanted to go with my research. He has received a jillion emails from me with both good and less good (also very bad) ideas, and I have always known that I could trust an honest answer: That he would encourage me to "go crazy" on good ideas but also made me reflect on less good ideas. A big thank you!

Also, I want to thank my co-supervisor **Simon Skov Fougt**, who has been great at asking me questions that I thought had obvious answers. Thank you for making me explain and reflect more on why this and that was relevant and related.

When I was applying for my PhD position, I came across **Aman Yadav**, a productive professor in East Lansing, Michigan State University whose work on computational thinking in education I admired (and still do). When I suggested that I come to Michigan State University for six weeks before even knowing if I would get the PhD position, he immediately welcomed me. As if he was not busy enough with his own students, he has taken the (non-paid) time to discuss, offer advice on, and write articles with me as well. Also, he patiently waited for me at the airport for a couple of hours when I was detained and questioned about my travel plans by immigration officers and almost not allowed into the country, one of their reasons being: "No one goes to East Lansing for six weeks". Luckily I did! Thank you for your time and support, and I look forward to collaborating more in the future.

No one who reads this dissertation will doubt that it is based on and inspired by historical perspectives. It was a paper by another professor, **Peter J. Denning**, that caused me to survey history within the field of computational thinking. Peter J. Denning has also helped me understand computing as a discipline as well as his perspectives on what he has labelled the new computational thinking movement, compared to the historical one. Thank you for your time, effort, and patience in explaining heavy subjects to a non-computer scientist who has asked a lot of questions.

In addition, Peter J. Denning put me in touch with a Finnish professor, **Matti Tedre**, with whom I have had some collaboration on work on computational thinking, and who also took the time to discuss some of the issues related to the subject. Thank you, and I hope to collaborate more in the future.

With regard to Danish history within the field, a bunch of people have inspired and talked to me. One of my biggest heroes within the field is Peter Naur, who I unfortunately never got to meet. Fortunately, though, I met his son, **Thorkil Naur**, who (in the same honest and straightforward manner as his father) has discussed issues about computer science (datalogy), as well as provided me with books by his father. Thank you.

In addition, Associate Professor Emeritus **Erik Frøkjær** was one of the persons who knew Peter Naur very well. They were colleagues at the Computer Science Department at the University of Copenhagen, and in several of his publications Naur thanked Erik Frøkjær for his feedback. I wrote to Erik Frøkjær to find out more about Peter Naur's thoughts and theories, and he welcomed me at his office for discussions – and provided me with books as well. Thank you for encouraging me and helping me understand the course of history – and for pointing out when I was writing nonsense. I appreciate your feedback.

I also want to thank the Danish Data History Association (*Dansk Datahistorisk Forening*). Especially, I want to thank the former chairperson, **Finn Verner Nielsen**, for inviting me to their time warp of a basement filled with fascinating history: old machines and computer stuff, books etc. Thank you.

Driven by Finn Verner Nielsen, I also met two former pioneers within data education/computer science in practice: the former teachers **Gerd Belhage** and **Bjarne Belhage**. Thank you for explaining and illustrating how you implemented data education in your K-9 classes already in the 1970s. It has been very inspiring.

When speaking about practice, I want to thank PhD in computer science **Martin Dybdal,** who among other things has collaborated with me on a design experiment conducted in a Danish eight-grade class. With your computer science specialist knowledge, I learned a lot from our planning, conducting and evaluation of this experiment together, as well as from our other discussions on computational thinking.

The abovementioned experiment, as well as other parts of my studies, have taken place in practice, and I could of course not have done these studies with no practices to examine. Therefore, a big thank you to the **students**, **teachers** and **school principals** who took part in surveys, observations, experiments, pilot testing, etc. I know you are all very busy doing what it is all about – learning and teaching – and I appreciate your time very much.

Also a bunch of thank yous to my great colleagues at The Department of Educational Theory and Curriculum Studies at The Danish School of Education, Aarhus University. I especially want to mention

Louise Rosendal Bang, **Marie-Louise Molbæk** and **Lise Baun**: My workdays have been much more fun because of you.

Furthermore, I want to thank **Mikala Hansbøl** and **Ole Sejer Iversen,** who discussed my preliminary project at my work-in-progress seminar. I deleted my whole overview article and started all over after our discussions, and I am thankful that I did. When looking back at it, I do not know what I thought I was doing; however, that version was a stepping stone towards this final and (would I say) much better version.

And for the actual presence of any versions at all, I would like to thank my friend **Leik,** who encouraged me to apply for a PhD position. I am not sure I would have thought about it myself. Thank you for helping me realize that this was the right thing for me to do at a time when I was rethinking my career.

Last, but most of all, I want to thank my lovely daughter **Nanna,** who came into my life in the middle of my PhD studies. She luckily made sure that I had a life besides my studies by puncturing my writing bubble to sing and laugh (sometimes cry as well) and dance and play with her. She also made sure that I did not sleep too long in the morning. **Vera**, who is the sweetest bonus daughter that I could have ever wished for. She is visibly also the best big sister Nanna could have ever wished for (wherever Vera goes, Nanna crawls). And **Peter**, who made sure that I could focus on finishing my studies by taking care of practically everything at home in busy periods.

For the sake of good order, I must mention that everything in this dissertation is at my own expense. Although I have been inspired by the many competent people mentioned above, it is not certain that they agree with the understanding of the subject-area and the theoretical position I present in this dissertation.

Elisa Nadire Caeli

Copenhagen, Denmark, November 2021

Summary

Over the past decade, *computational thinking* has attracted increasing interest within K-12 education. The subject area is being implemented in curricula around the world, as our societies have become increasingly digitalized, thus it seems to be regarded as a fundamental general competence in students' present and future lives. In continuation of this development, the purpose with this research project was to dig into the reasons why, including if, computational thinking is a central competence to develop in compulsory education (K-9), and what aspects of it that are important.

Specifically, the project was primarily centered on answering the following research question: *Why* is computational thinking essential to teach in compulsory education, and *what* aspects of it are essential to teach? Additionally, I wanted to examine what teaching could look like in practice, based on the answers from my primary research question. Therefore, my secondary research question was: *How* can computational thinking be implemented in the classroom?

Methodologically, this is a didactic study based on German-Scandinavian critical-constructive Bildung-centered theories. It stands on the understanding that compulsory education should aim for students to develop general competencies that prepare them for life in a free and democratic society with rights and responsibilities. Compulsory education should therefore not aim at developing specific career competencies, in this case within computer science.

Although a number of schools worldwide are already teaching computational thinking, it was my hypothesis that arguments as to why it is important to teach this subject area in compulsory education are often either lacking or only superficially described in literature in the field. This hypothesis was prompted by Professor Peter J. Denning, who in an essay (2017b), among other things discussed today's lack of consensus regarding what computational thinking is, and whether it is indeed a general competence that everyone can benefit from acquiring. He pointed out that the today's new understanding of computational thinking is poorer and narrower than the historical understanding of the field. Therefore, he believes that researchers in the field should build on the foundation that has already been laid historically and thereby create progress – rather than begin theorizing and discussing all over again without historical insight.

I have examined *why*, *what,* and *how* from three perspectives: literature (state-of-the-art), historical perspectives and present perspectives.

A systematic review and analysis of literature in the field confirmed that arguments concerning why everyone should be taught computational thinking in compulsory education were either lacking or

superficially described. In addition, I found that today computational thinking is internationally regarded as predominantly mathematical problem-solving strategies or concepts, such as abstraction, decomposition of problems, and automation.

In my historical analyses, I focused mainly on Danish history within the field, as Danish compulsory education (The Danish *Folkeskole*), in accordance with my theoretical position, has a tradition of being oriented towards Bildung rather than towards college and career. I found that around 50 years ago, a subject that contained elements of what today is called computational thinking was discussed. The subject was, however, much broader. Professor Peter Naur in particular argued that everyone in a digitalized society should learn *datalogy* (a term he coined instead of *computer science*) as an essential interdisciplinary tool in the same way that everyone learns to master other essential tools, especially language (including reading and writing) and mathematics. Naur had a societal and democratic perspective on education; thus, he did not want to introduce this new subject in order for everyone to become computer scientists, but rather so everyone would learn to understand data, their nature and use – including how computers are programmed. Naur believed that failure to educate the general population would result in experts in the field gaining power and determining the direction of society, which in turn would dismantle democracy.

In line with Naur's arguments, the subject *data education* (*datalære*) was formulated in the 1970s. A committee set up by the Danish Ministry of Education described in a report what the subject should contain, and from their work I saw that merely focusing on mathematical problem solving was considered inadequate. The committee recommended a broader focus that included, for example, the use of data processing and the societal aspects of this, as well as a focus on understanding what a computer was and how it could be programmed.

Computers were, however, much simpler then than they are today. Today, user interfaces are designed so intuitively that understanding how they actually work does not seem to be relevant. It may also seem irrelevant to look at simple historical examples of how the subject data education was taught around 50 years ago, because of the obvious excessive developments in the digital field since then. In this dissertation, I discuss that this simplicity is precisely one of the strengths of history in the way that these simple examples can help students understand how computers basically work, and what they actually can (and cannot) do.

At the same time, I conclude that historical perspectives on *why*, *what*, and *how* are insufficient in a present and future society, and that contemporary analysis should supplement such historical perspectives. Thus, in a present perspective, I discuss that Naur's arguments, as well as the content description of the subject data education, are fundamentally still valid – partly because of Naur's anticipation concerning the use of computers in the future and the power of experts. But the fact that society has changed drastically in the field of digitization since the 1970s calls for *specific* arguments for teaching a similar subject (*why*) to change as well. Computers are not used in the same ways today, and the opportunities and consequences of digitization today are different than they were back then. Furthermore, it is not the same *specific* content (*what*) that should be taught today. Today, for example, many computer programs are based on machine learning models with dynamic algorithms. Historically, data models were much simpler. Finally, teaching (*how*) should not take place in the same ways as historically. For example, digital equipment and programs are to a much greater extent available today, and the changes with regard to *why* and *what* also change *how*. I exemplify *how* with a specific teaching example, where I in collaboration with a computer scientist planned and conducted a design experiment in an eighth-grade class where students were to design a prototype of, and program, a product that could reduce energy consumption.

Overall, in this dissertation, I argue that historical theories and discussions can serve as a good foundation for theories and reflections on the present didactic questions *why*, *what*, and *how*. I conclude that it is not computational thinking in the narrow common mathematical understanding that is needed in general education, but a broader coherent subject – in Denmark called technology comprehension – which focuses on developing competencies within datalogy, design, and technology criticism.

1 Introduction

This PhD dissertation communicates the result of a three-year-long research project on computational thinking in compulsory education (grades K-9). In this project, I examined whether computational thinking is essential to teach in compulsory education and why, as well as what aspects of computational thinking all humans need to learn in compulsory education to be able to live and work in a society with democratic rights and responsibilities. In addition, I conducted an experiment to bring some of my theories to practice.

Over the last decade, there has been an increasing interest in teaching computational thinking to primary and secondary students (grades K-12) (see for example Bocconi et al. 2016; EMU 2021; ISTE 2016; NGSS 2013). The need to teach aspects of computer science[1] in primary and secondary classrooms has, however, been discussed by researchers and educators worldwide since the 1960s. One of these researchers, who still actively participates in discussions on this topic today, is professor in computer science Peter J. Denning. In recent years, Denning has presented some trouble spots with the new movement on teaching computational thinking in schools. He says:

> Around 2006 the promoters of the CS-for-all K-12 education movement claimed all people could benefit from thinking like computer scientists. Unfortunately, in attempts to appeal to other fields besides CS [computer science], they offered vague and confusing definitions of computational thinking. As a result today's teachers and education researchers struggle with three main questions: What is computational thinking? How can it be assessed? Is it good for everyone? (Denning 2017b)

Denning finds it concerning that teachers are still unsettled about these basic issues: “How can they be effective if not sure about what they are teaching and how to assess it?” he asks, and argues that there is no need for vagueness regarding what computational thinking is. The meaning of it, which has evolved since the 1950s, is clear, he states. The claims that it benefits everyone beyond computational designers are, though, as yet unsubstantiated, he points out (Denning 2017b).

In the article, Denning explains how the present new notion of computational thinking and the historical traditional notion of computational thinking differ. First and foremost, he argues that the new version of computational thinking is independent of the past history: “One of the important differences is that in Traditional CT[2] programming ability produces CT, and in the New CT learning

[1] In the 1960s, the term *computational thinking* was not used; instead, other terms related to computer science such as datalogy and data education were used. I will elaborate later on these terms.

[2] CT is short for computational thinking.

certain concepts produces programming ability. The direction of causality is reversed" (Denning 2017b). For example, in traditional computational thinking "algorithms are directed to control a computational model (abstract machine) to perform a task," whereas in new computational thinking, "algorithms are expressions of recipes for carrying out tasks; no awareness of computational models is needed" (Denning 2017b). The absence of any mention of computational models is a mistake, he argues: "We engage with abstraction, decomposition, data representation, and so forth, in order to get a model to accomplish certain work" (Denning 2017b). Moreover, in new computational thinking, an algorithm is described as any sequence of steps and for all kinds of information processors including humans. A step that requires human judgment is, however, not an algorithmic step, he points out. Therefore, the new notion of an algorithm and the possible absence of a machine could cause students to miss the most basic idea of an algorithm.

Another issue is how new computational thinking in the frameworks Denning has reviewed is described as a set of mathematical problem-solving concepts, whereas in traditional computational thinking it involves skills of design and software crafting. "Computational thinking includes designing the model, not just the steps to control it." It is "loosely defined as the habits of mind developed from designing programs, software packages, and computations performed by machines," he argues (Denning 2017b).

Denning and one of his colleagues, Professor Matti Tedre, suggest that a lack of knowledge about the long and rich history of computational thinking may lead to weaker and less ambitious versions of the subject area than we have seen in the past, causing computational thinking to diminish and not progress (Tedre & Denning 2016).

They argue:

> When researchers do their homework well, they know what previous generations of scientists have tried and done, and where they have succeeded and failed. They avoid 'reinventing the wheel' by acknowledging predecessors who built the foundations on which the current generation of researchers is now working. (Tedre & Denning 2016)

Similarly, Denning, Tedre, and chief academic officer for the non-profit organization code.org Pat Yongpradit point out that early work on computer science has shown us what works and what does not. It is our history that has shaped today's world. For example, equating computer science with programming is problematic and illustrates how we repeat history if we forget it (Denning, Tedre, & Yongpradit 2017).

In order to understand the development up until today and be able to benefit from early work and past successes and failures, I decided to examine historical perspectives on and practices within computational thinking in education. I wanted to examine whether history could help answer some of today's central questions when embedding computational thinking in compulsory education.

1.1 Research Questions: Why, What, and How

Specifically, I examined two of the abovementioned three trouble spots identified by Denning, namely what computational thinking is and if it is good for everyone (why).

With this regard, my primary research question was:

> ***Why* is computational thinking essential to teach in compulsory education, and *what* aspects of it are essential to teach?**

Moreover, I wanted to examine what my primary studies could mean for practice. Therefore, my secondary research question was:

> ***How* can computational thinking be implemented in the classroom?**

In this dissertation, I look at *compulsory education*, understood as kindergarten to grade-nine (K-9). I focus on what all human beings need to learn to be able to live and work in a democratic society; that is, what they need to learn in a compulsory subject in compulsory education. Consequently, my purpose is not to determine what computational thinking in computer science is, but specifically to examine what computational thinking in a context of compulsory education is in order to suggest answers to what knowledge and skills *all humans* need to develop.

1.2 Structure of Dissertation

This dissertation is designed as:

> a collection of several academic texts that are related in content and/or methodology and where the results obtained in course of the PhD programme are presented and possibly published, either by the PhD student alone or by the student together with other authors. In addition, the dissertation must include a separate presentation by the PhD student that takes the form of a large-scale overview article.[3]

The overview article consists of six chapters (chapters 1-6), and the appendix contains five published and peer-reviewed articles, referred to as sub-studies. In the present overview article, I discuss how the sub-studies are related to and contribute to answering my overall research questions on *why*, *what,* and *how* – three keywords that are repeated throughout the dissertation.

[3] As per "Rules for the PhD Programme at the Graduate School, Arts", chapter 5.2: https://phd.arts.au.dk/fileadmin/phd.arts.au.dk/AR/Generelle_retningslinjer_UK_1-11-2012.pdf

In the present chapter, *Chapter 1,* I introduce my research project and its overall purpose.

In *Chapter 2*, I describe my research design, including my intended research design since it led to my actual research design. I discuss the didactic[4] theoretical position on which my studies are based. Moreover, I briefly describe each of my sub-studies (five published articles, and two studies that only appear in this overview article).

I have examined the research questions from three perspectives: State-of-the-Art, Historical Perspectives, and Present Perspectives. In the three following chapters, I present my results from each of these three perspectives.

In *Chapter 3*, State-of-the-Art, I communicate the results from a review of international computational thinking research literature. From this review, I analyze what computational thinking is regarded as today, as well as the arguments for embedding it in K-12.

Chapter 4 is centered on historical perspectives. In this chapter, I present three sub-studies: The first sub-study is centered on Peter Naur's perspectives on *datalogy* (a term he used instead of *computer science*). The next two sub-studies are included in this dissertation in the form of articles (A and B). One article is about unplugged[5] approaches to computational thinking inspired by historical examples, and the other article is about the development from the 1960s and up until today with regard to computer science in Danish education. In the latter, I additionally discuss shortcomings of historical examples.

This leads me to present perspectives in *Chapter 5*. In this chapter, I discuss three sub-studies that are all included as articles (C, D, and E). In the first sub-study, I present and discuss the findings from a survey of Danish school principals' conception of *why* and *what*. In the second sub-study, I analyze how digital technologies influence our lives today. I mainly discuss why we need to teach technology comprehension (as a broader concept than computational thinking) today, and what it is. The final sub-study illustrates a possible *how*. I present a design experiment that I co-planned, co-conducted, and co-assessed based on the *whys* and *whats* from previous chapters.

In *Chapter 6*, I conclude on and discuss the overall findings of my project in terms of *why*, *what,* and *how*. I discuss the contribution as well as limitations of my work, and what future work is needed.

[4] In this context, the term *didactic* is based on Nordic educational traditions, e.g., professional factors concerning *what* students should learn, *why,* and *how*. I elaborate on the term in the next chapter and argue for my didactic position.

[5] Without a computer.

The five articles that together with my overview article constitute my dissertation are included in the appendix (page 95 ff.), articles A-E.

2 Research Design

Methodologically, this project is a didactic study. It contains five sub-studies, and the primary purpose was to examine *why* and *what* aspects of computational thinking are essential to teach in compulsory education. The secondary purpose of this project was to examine *how* computational thinking can be implemented in the classroom.

I chose to first and foremost focus on the *why* and the *what* before the *how*. This choice was in part based on a hypothesis that arguments for why students need to learn the kind of computational thinking that researchers, educators, and politicians find they do is often missing in literature, and in part based on two preliminary studies that I did in collaboration with Professor Jeppe Bundsgaard from January to June 2018. The purpose of these studies was to identify characteristics of effective computational thinking teaching. Originally, these were the two main studies in my project, but the results of these studies made me change my research design. This is why I find them worth mentioning here. Therefore, in the following I present my intended research design and method. I discuss why it did not work out the way I was expecting, and how the results of it led me to my actual research design and method.

2.1 Intended Research Design

Table 1 illustrates the two studies (Phases 1 and 2) and the progression of my *intended* research design; that is, what I planned to do when I started my studies. At that time, the purpose of my studies was to identify characteristics of effective computational thinking teaching through a three-phased design:

Phase 1. Preliminary survey study	**Phase 2. Observation study**	**Phase 3. Secondary analysis of ICILS 2018 scores**
Digital questionnaire to 145 principals of schools participating in the ICILS[6] 2018 assessment	Six weeks of structured observation in six different eighth-grade classes with ICILS 2018 participation	Comparison of observation study results to secondary analysis of ICILS data

Table 1. Intended Research Design

[6] *International Computer and Information Literacy Study* that every fifth year measures eight-grade students' computer and information literacy as well as their computational thinking skills

2.1.1 A Preliminary Survey Study

Phase 1 of my intended research design was a survey study conducted in January and February 2018. We sent out a digital questionnaire to 145 Danish school principals. A total of 83 principals completed the questionnaire. The purpose of the study was to identify to what extent Danish schools in compulsory education offer IT[7] initiatives that involve computational thinking, as well as educate teachers to teach computational thinking, and to what extent Danish school principals are familiar with computational thinking, as well as what their perspectives on embedding it in compulsory education are. We wanted to analyze and report the results as a stand-alone study, the results of which were interesting in themselves, but primarily, we wanted to use the results to select six schools for Phase 2: an observation study.

2.1.2 An Observation Study and Secondary Analysis of ICILS Scores

The original intention of the preliminary survey study was to select three schools that had a self-reported focus on and knowledge about computational thinking and three schools who reported low focus on computational thinking and limited knowledge about it. I wanted to do structured observations of and video record all teaching in all subjects in one eighth-grade class at each of the six schools, every day for a week. The observations would allow me to categorize and subsequently quantify my qualitative observations of what happened every minute in all of the observed lessons with regard to teaching methods, student activities, content activities, learning resources, use of technologies, concentration level, etc.

The reason for only observing schools that had participated in ICILS 2018 was to identify differences between schools regarding to their scores. In other words, I wanted to analyze the results of ICILS 2018 that measured eighth-grade students' computational thinking skills and thereby be able to identify characteristics of the teaching in schools with high and low student scores, respectively.

The observation study was conducted from April to June 2018. I ended up observing only five of the six classes since I gradually realized that there was no focused computational thinking teaching in the teaching I was observing. That also meant that the "high focus/low-focus" classes were not different enough with regard to computational thinking to be able to differentiate between them as being focused or not. I did, though, finish the observations in the first five classes, which gave me 149 observed lessons, categorized into 706 different sequences of what happened every minute in detail, including notes and video recordings.

[7] Information Technology

Since I did not see any focused computational thinking teaching in practice, it was not possible to move on to Phase 3 or to analyze and present any theories on how to teach computational thinking efficiently based on these observations. I only saw a few sequences containing aspects of computational thinking, such as *data collection* in math. The observation study is, therefore, not included in this dissertation because it resulted in non-findings towards answering my research questions. I do, however, briefly describe it here, since it explains my need to 'step back' and answer *why* and *what* before examining *how* any further. Moreover, the non-findings from these observations caused me to subsequently co-plan and co-conduct a design experiment[8] to identify didactic opportunities and challenges based on my studies on *why* and *what*. I will explain in the following what my actual research design and method came to look like.

2.2 Actual Research Design

As I stated in the introductory part of this chapter, this research project is methodologically a didactic study that is based on the study of three fundamental didactic questions: *why*, *what,* and, subsequently, *how*, illustrated in Figure 1.

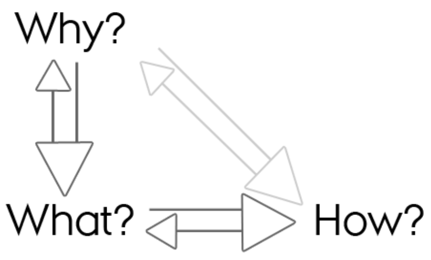


Figure 1. Why, what, and how to teach computational thinking

The figure illustrates a dialectic relationship between *why*, *what*, and *how*. It shows that, in my understanding, there is a linear process from *why* to *what* to *how* (big arrows): *Why* dominates *what* to teach, and *what* to teach dominates *how* to teach it.

There are, however, also interactions in the opposite direction. *What* affects *why* since there is no why to discuss without a what (small arrow). And *how* affects *what* to teach in terms of what is actually possible (small arrow)[9].

[8] Included as Article E in this dissertation.

[9] The dialectic arrow is adapted to this model from Bang, Døør, Steffensen & Nash (2007). They describe how two parts are both individualities (stable parts in a whole), interdependent and interconnected (one does not exist without the other), unequal (one dominates the other), interactional (one dominates but does not determine the other), and historical phenomena (their results are impermanent = subjects of change)

Moreover, the faded arrows show an interaction between *why* and *how*. *Why* affects *how* (big faded arrow), and *how* affects *why*, since it is hardly possible to think of why without thinking pre-existing thoughts on how (small arrow). For example, in one study that I co-authored (Article C), we discuss *how* in terms of whether technology comprehension should be embedded in the school curriculum as its own subject and/or integrated in existing subjects; hence, our *why* was indirectly affected by *how* (in this case organizing the school day with subjects on a timetable). And *why* directly affects *how* when we conclude it should be embedded as both its own subject and integrated in other subjects.

In its essence, it is illogical to teach computational thinking in the classroom without any clear idea about *why* one teaches *what*. *How* does not exist without *what* and *why*, and *what* does not exist without *why*. Therefore, the purpose of this present project was primarily to examine the dialectic relation between *why* and *what* and, secondary, exemplify with *how* ,which means that I follow the dominating process in the model (big arrows), while keeping in mind that they all counter-interact as well, and that each of them is, therefore, subject to change.

While, empirically, my actual research design originated from the studies from my intended research design, theoretically it is based on German-Scandinavian didactic theories. I will discuss my theoretical position in the following section.

2.3 Didactic Theoretical Position

In this section, I will argue why I have chosen to examine the didactic questions *why*, *what*, and *how*, how these questions are linked together, and what I, thereby, chose not to examine. To clarify what the term *didactics* means in a German-Scandinavian context, I begin with a description of German-Scandinavian notions of this term, and discuss how it differs from Anglo-Saxon notions of the same term.

2.3.1 German-Scandinavian Notions of Didactics

When looking the term *didactics* up in the dictionary, it is clear that in German-Scandinavian countries notions of the term are very different from notions in Anglo-Saxon countries. With regard to Anglo-Saxon countries, Collins Dictionary, as an example, provides the following definition: “something that is didactic is intended to teach people something, especially a moral lesson”, and a didactic person is someone who “tells people things rather than letting them find things out or discussing things”. Similarly, in Cambridge Dictionary, didactic is explained as “intended to teach, especially in a way that is too determined or eager, and often fixed and unwilling to change”.

In contrast to this, in German-Scandinavian countries, didactics is far from the specific way of teaching that is understood in common Anglo-Saxon conceptions. It is an open and much broader concept. Professor Bjørg B. Gundem and Professor Stefan Hopmann describe it as an independent discipline (Gundem and Hopmann 2002). In their work, they compare two sets of attitudes: the Anglo-Saxon tradition of curriculum studies and the Central/North European tradition of didactics.

One of the problems, they say, is how many concepts, terms and words of the German-Scandinavian language area lack counterparts in English:

> Indeed the term *Didaktik*[10] itself with its comprehensive intertwining of action and reflection, practice and theory is one such untranslatable concept. The most obvious translation of Didaktik, didactics, is generally avoided in Anglo-Saxon educational contexts, and refers to practical and methodological problems of mediation and does not aim at being an independent discipline, let alone a scientific or research program. (Gundem and Hopmann 2002)

Gundem and Hopmann (2002) discuss how different attitudes towards curriculum planning and implementation are, therefore, important obstacles to address when it comes to cooperation across national borders in curriculum research and development. They argue that the problems with transferring senses of meaning of central concepts make comparative research, cooperation, and mutual exchange of traditions and approaches important.

In this project, I aim to share Scandinavian-German traditions of and approaches to education in the context of teaching computational thinking. Despite the different meanings, I have chosen to use the translated word *didactics* to make the language more fluent and easier to use in different word classes[11] and inflectional forms. Therefore, it is important to emphasize that throughout this dissertation, *didactics* refers to German-Scandinavian notions of the term.

Another key term, related to the German-Scandinavian notions of didactics that I will explain in the following, is the German term *Bildung,* which also lacks a counterpart in English and is therefore impossible to translate directly into English. Gundem and Hopmann point out that: "No term in English conveys the meaning of this concept which refers to the process and product of personal development guided by reason" (Gundem & Hopmann 2002).

[10] The German-Scandinavian word for didactics.

[11] Additionally to using *didactics* as a noun, I use *didactician* when referring to a person who professionally does research in the field of didactic theories, engages in didactic discussions and/or conducts didactic actions in the classroom, and *didactic* when using the term as an adjective.

In the absence of an associated English word, I decided to use the German word *Bildung* throughout this dissertation. In the following, I will further explain what Bildung and didactics, as key terms in my studies, involve.

2.3.2 Critical-Constructive Didactics and Bildung-Centered Theory

My position in the didactic field is based on the critical-constructive theory of didactics of Professor Wolfgang Klafki and his Bildung-centered theory. Klafki (2016) perceives Bildung as the *self-determination*, *co-determination* and *solidarity competence* of the individual: It encompasses *competencies to self-determine* one's personal living conditions and views of life of humane, professional, and religious nature; it encompasses *competencies to co-determine* opportunities and take responsibility to form our common societal and political conditions; and it encompasses *solidarity competencies* in the sense of the individual's efforts to help other individuals whose opportunities for self-determination and co-determination are hindered or limited by societal conditions, underprivilege, political constraints or oppression.

General Bildung is the kind of Bildung that *all* humans in a society should develop. It involves learning about and being confronted with things that concern all humans collectively, for example common tasks and problems, human experiences and suggested solutions developed throughout history, as well as changes, risks, and opportunities in the future in order to liberate and activate new generations so they can understand and form the historical present and the future in free self-determination (Klafki 2016).

Critical-constructive didactics is a Bildung-theoretical didactics; that is, it is based on theories of Bildung. To Klafki, Bildung as fundament is necessary and possible for systematical and historical reasons.

From a systematical perspective, Klafki argues that a central, informative category such as the concept of Bildung is required so that the practical-pedagogical efforts – and the studies and reflections that clarify and theoretically substantiate these efforts – do not fall apart in an incoherent chaos of co-existing individual activities. The concepts of self-determination, co-determination, and solidarity competence act as central, general informative criterions and assessment criterions for the many pedagogic and didactic individual activities and plans.

From a historical perspective, Klafki says that the concept of Bildung was developed around 1770 to 1830 as a central critical-progressive and societal-critical concept in pedagogical thinking. Klafki refers to Kant and his thoughts on every individual's right to and possibilities for developing self-determination – to become empowered – and Pestalozzi's and Humboldt's thoughts on every individual's right to development in all ways possible.

Klafki states:

> the conception that humans through reasonable conversation and discussion and reflective processing of experiences can arrive at a continued humanization of common living conditions and an increasingly reasonable design of the social policy conditions, i.e. that humans can break down an unsubstantiated domination of power and increase the space for their freedom. (Klafki 2016, my translation)

In his view, didactics is not only a theoretical discipline but a science *of* practice *for* practice. Therefore, he argues that the didactic tasks are, with help from scientific methods, to work towards the views that cause didactic decisions, developments, discussions, and arrangements, as well as their often hidden ideas about the future and philosophical implications. Subsequently, didactics must make it possible to test and discuss these views and thereby help didacticians and decision-makers such as curriculum planners and teachers, as well as students, to become aware of what they are actually doing and deciding, and under what historical circumstances they are acting. They must be aware of the underlying reasons for their decisions, considerations, and actions, he says.

Professor Frede V. Nielsen has elaborated on how didactics serves as a science of practice for practice with what he calls *the theory/practice problem*.

2.3.3 The Theory/Practice Problem

When planning a lesson, it is insufficient to theorize, Nielsen states. He says that to realize one's theories, it is necessary to plan and make decisions about content and other aspects of teaching. In that sense, didactics usually refers to both theory and practice planning.

Nielsen distinguishes between (a) didactics as theory that is scientifically oriented, and (b) didactics as planning and decisions that can be scientifically based on scientifically oriented didactics. He argues that his differentiation does not mean that the two kinds of didactics are not interconnected. He explains that (a) directly or indirectly acts as a servant for (b). In that sense, theoretical didactics is directed towards pedagogical practice. As is the case with Nielsen (1998), I primarily operate in the field of (a) in this present research project, but with the intention of making a more conscious and reflective basis for the field of (b) possible.

Nielsen states that didactic theories give opportunity to examine, understand and reflect on real situations. Though theories simplify reality, they make reality more apparent and easier to understand. Moreover, he argues that the critical distance associated with theoretical activity is a good basis for changing one's own pedagogical decisions and actions in well-considered, constructive, and progressive ways. However, since theories are less detailed than practice, no didactic theory and no theoretical awareness can encompass all aspects of a practice. The very purpose of theories is to examine and simplify, he says. In that way, theories can enrich practice, but practice can also enrich theories through the experiences one makes in a practice. It is hardly possible to decide what is essential in pedagogical situations without engaging in practice. Thus, Nielsen points out that scientifically oriented didactics can also include empirical studies of teaching realities. For example, if empirical studies and analysis relate to and examine specific theories.

My secondary studies of *how*, communicated in Article E, build on Nielsen's thoughts on this interconnection between didactic theories and practice. As is explained briefly earlier and in details later in this dissertation, my empirical study took the form of a course (a didactic practice) based on my studies of *why* and *what* (didactic theories). Moreover, the empirical study (the course) could possibly enrich these theories through the experiences made in practice.

2.3.4 Didactic Questions and Different Focuses

Nielsen (1998) distinguishes between narrow and broad understandings of didactics. In a narrow understanding, didactics first and foremost addresses the questions *what* (contents), *where to* (objectives), and *why* (arguments and purposes). In a broader understanding, it also addresses questions regarding *how* (method, planning), *with what* (teaching resources), and *where* (institutions, classrooms etc.). In both the narrow understanding and the broad understanding, *who* is also addressed, but in different ways. In the narrow understanding, addressing who means that reflections about content etc. must take into account student qualifications, whereas in the broader understanding it means to involve students in reflecting and choosing content etc. Table 2 illustrates didactic questions according to Nielsen.

Narrow understanding	**Broad understanding**
What (content)	What (content)
Where to (goal)	Where to (goal)
Why (argument and purpose)	Why (argument and purpose)
Who (students)	Who (students)
	How (method, planning)
	With what (teaching resource)
	Where (institution, design of room, and more)
	Wh ...

Table 2. Didactic questions according to Nielsen

Nielsen (1998) points out that didactic questions in a historical perspective have been more focused on methods (*how*), but when the knowledge society replaced the industrial society during the late 20th century, didactics started to focus on content (*what*) and arguments on content (*why*).

With an increase of the amount of knowledge and new academic areas of knowledge (such as computational thinking), it was no longer given what should be taught in school and why, Nielsen states. Therefore, he argues, subject-specific reflections on teaching as ways to accomplish the general purpose of compulsory education are much more important than earlier. Subject-specific didactics depends on general didactics, which is also an argument for why selecting content (*what*) is closely related to arguments (*why*). I will elaborate on this in the following section on subject-specific didactics in a Bildung perspective.

2.3.5 Subject-Specific Didactics in a Bildung Perspective

With reference to Professor Karsten Schnack, Nielsen describes how subjects and their didactics can be based on deciding current problems in society. Schnack refers to this didactic paradigm as *didactics of challenges*. He says:

> In didactics of challenges, questions are asked about what understanding and concepts that are important for us and the next generation in order to handle the challenges that we face as a society and as humans. If one considers that some of the crises mentioned before are threatening to humanity in a whole new way, it must mean that the next generation needs opportunities to comprehend the world in a way that enables them to handle some of these crises. To a large extent, the content of Bildung must be decided on the basis of what competences that are necessary to develop to act in a world where democracy is threatened from all sides. (Schnack 1993 as cited in Nielsen 1998, my translation)

This position is described as being societal-/ideological-critical with an aim to change (improve) the world, human awareness, and future possibilities by students developing responsibility and competences to act on big problems in society. In addition, moral is important in this didactic position, meaning that education and Bildung are committed to dealing with societal challenges.

My didactic approach and position in this project are critical-constructive as described by Klafki and are based on the didactics of challenges. I examine possible *whys*, *whats,* and *hows* of computational thinking in compulsory education in a societal, democratic perspective in order for students to be able to handle societal challenges related to computing and data processing.

For example, in Article C, co-written with Bundsgaard, we, in line with these didactic theories, define the concept of Bildung to encompass the individual, social, and cultural development of the whole child. This is also reflected in the legislation relating to what Danish compulsory education should be concerned with, for example students' development into active citizens with social competences and the ability to understand and take part in the democratic processes as well as their individual overall development as human beings (Article D; see also Chapter 2.3.7 for the formal aims of Danish compulsory education).

2.3.6 Levels of Didactic Discussions

In his PhD dissertation, Bundsgaard (2005) distinguishes between different levels of didactic discussions. General didacticians and subject-specific didacticians, for example work at different levels, he says. The general didactic researcher examines and discusses the overall purpose of education, whereas the subject-specific researcher examines and discusses specific content based on professional competencies within her or his subject – but with a general view and the overall purpose in mind. Bundsgaard puts it this way:

> Subject-specific and general didactics are not two fundamentally different areas of research or practice; they are two perspectives on the same thing. Subject-specific didactics is equivalent to general didactics but with a focus on a subject. Or more accurately, with interest in how the worldview and perspective on the world, the methods and areas of knowledge that one or more related subjects represent, can contribute to students' education. (Bundsgaard 2005, my translation)

In the cited work, Bundsgaard focuses on information technology (IT) didactics in relation to Danish as an L1 subject (language one), and in that context he exemplifies that:

> IT didactics in Danish as an L1 subject is, thus, general didactics with a focus on how Danish can contribute to students becoming competent citizens and humans in the information and network society of today and the near future, and on how IT can be used when organizing teaching (Bundsgaard 2005, my translation)

Similarly, in this present project I examined the specific subject area *computational thinking,* focusing on why this subject area is relevant to teach in compulsory education today to contribute to students becoming competent citizens and humans in the digitalized society of today and the near future. Therefore, the project contributes to subject-specific didactics on teaching computational thinking.

Bundsgaard further points out that Nielsen as well as Professor Carl Aage Larsen, limit subject-specific didactics to the selection of content. However, in Bundsgaard's view

> A subject-specific didactician must work on a basis of an understanding of the situation's constituents (participants, relationships, resources, etc.) and in the light of that describe:
>
> * the society and the world,

> * the purpose of education,
>
> * the teaching objectives,
>
> * the teaching content,
>
> * the methods and organizations of settings, collaboration and activities, and:
>
> * the evaluation.
>
> (Bundsgaard 2005, my translation)

To Bundsgaard, there is no hierarchical order in these tasks. This is in line with Klafki who says that: "We are dealing with a structure of relations, where each individual decision and the holistic context mutually address each other" (Klafki 2002, my translation).

I agree with this non-hierarchical view as I illustrated in my model of a dialectic relationship between *why*, *what,* and *how* (Figure 1). For example, *how* also affects *why* and *what* in terms of what is actually possible in practice. We cannot think about *why* and *what* without considering questions regarding *how*, for example questions about how we as a society have chosen to run schools (timetable, a teacher and xx students, a classroom etc.) Even when we plan to think freely and break the framework, we do so with the existing framework in mind. There are, however, some questions that I find more dominating than others, as illustrated in Figure 1.

2.3.7 My Didactic Choice of *Why*, *What,* and *How*

Bundsgaard (2005) discusses how Nielsen (1998) believes that one ideally needs to argue and decide what is important to learn and therefore to teach before one can consider how to teach it. In contrast, Bundsgaard, argues for dialectic relations between, for example, content and methods, that is questions related to what I in this present project refer to as *why*, *what,* and *how*. He states that it does not make sense to single out and deal with just one of these aspects as if it were autonomous:

> On the other hand, it is completely acceptable to be more interested in one aspect than in the others, but the didactician who is primarily interested in determining relevant content must consider teaching practice and organization in order for the students to develop not only information and skills, but also understanding, experience, insight, ability and will; i.e. in order for them to develop competence. (Bundsgaard 2005, my translation)

Bundsgaard defines didactics as reflections about the teaching situation, reflections on planning before the teaching situation, and evaluations after the teaching situation that form the basis of further planning. In his conception, didactics is a reflection on three separate but interdependent phases: *The situation*, *Before the situation*, and *After the situation*. He argues that planning involves

“2) what the students should learn, 3) how they should learn it, and especially 1) why they should learn it”.

My original intention with the survey study and observation study, described in Chapter 2.1, was to examine *The situation;* however, the results of these studies made me realize a need to examine *Before the situation* to be able to move on to *The situation*. In my understanding, reflecting on *Before the situation* involves reflecting on *why* and *what* prior to *how*.

At an abstract level, *why* and *what* are reflected in the formal aims of Danish compulsory education (The *Folkeskole*, K-9) as declared by the Government:

> The Folkeskole is, in cooperation with the parents, to provide students with the knowledge and skills that will prepare them for further education and training and instil in them the desire to learn more; familiarise them with Danish culture and history; give them an understanding of other countries and cultures; contribute to their understanding of the interrelationship between human beings and the environment; and promote the well-rounded development of the individual student.
>
> The Folkeskole is to endeavour to develop the working methods and create a framework that provides opportunities for experience, in-depth study and allows for initiative so that students develop awareness and imagination and a confidence in their own possibilities and backgrounds such that they are able to commit themselves and are willing to take action.
>
> The Folkeskole is to prepare the students to be able to participate, demonstrate mutual responsibility and understand their rights and duties in a free and democratic society. The daily activities of the school must, therefore, be conducted in a spirit of intellectual freedom, equality and democracy. (Ministry of Children and Education 2018)

As such, the purpose of Danish compulsory education is not only to prepare students for further education or specific careers but also to promote a well-rounded development in a democracy. It is central that students develop confidence in their own possibilities and willingness to take action. And, perhaps most importantly in the context of developing computational thinking skills, that compulsory education prepare them to be able to participate, demonstrate mutual responsibility, and understand their rights and duties in a free and democratic society.

Consequently, examining *why* and *what* must involve examining societal perspectives on what all individuals need to learn to be prepared not only for college and career but also for life in a free and democratic society. Moreover, Bundsgaard points out that we are educating for the future, not the past. Therefore, examining *why* includes examining what aspects of the society we live in and will be living in in the near future that education should prepare students for. On the basis of the characteristics of today's society and of the society of the future, one can come up with suggestions on *what* to learn, he argues.

With reference to Figure 1, in my view *why*, *what,* and *how* are interdependent. They all affect each other. Keeping this in mind – for example that *how* already affects *why* and *what* in terms of our pre-existing knowledge – I do, however, find it relevant to ask *why* and *what* before asking *how*. This is primarily because *how* is not an interesting question to answer if an analysis of the purpose of education, society, and the world shows that no reasons exist for teaching a specific subject. Therefore, my approach in the present project was to examine *why* and *what* before exemplifying with *how*.

When looking at Nielsen's list of didactic questions, I place myself within a broad understanding of didactics even though I have chosen to focus on some didactic questions and not others. For example, I include didactic reflections on *where to*, *who*, *with what,* and *where,* which are necessary when planning teaching. I have included these reflections in my research of *how,* which I explain further in Chapter 5.3 and in detail in Article E. In addition, I touch on some of these other question words in my overall reflections on *where to* (prepare for life in a democracy) for *who* (all individuals, every student).

In the following section, I briefly describe the methods of my sub-studies and their contribution.

2.4 How My Sub-Studies Contribute

I have included five sub-studies in the form of published articles in this dissertation. The articles do not necessarily have the same purpose in this dissertation as they do as journal articles, as they are written as independent studies. Results from each of them do, however, contribute to this project. Below, I explain the purpose of each of these studies in terms of why I conducted them as well as how they contribute to answering my research questions. In addition, I have conducted two additional sub-studies that have not been published as articles (chapters 3.1 and 4.1). In chapters 3, 0 and 5, I elaborate on the method, analysis and results of each of these seven sub-studies.

2.4.1 State-of-the-Art (Chapter 3)

Common Whats and Whys in Literature (Chapter 3.1)
This study is a state-of-the-art in regard to my primary research questions on *why* and *what*.

Denning's discussion about trouble spots on computational thinking (2017b) made me hypothesize that common understandings of computational thinking (*what*) would be rather narrow, and that arguments for why we need to teach it would be weak (*why*).

In this study, I surveyed these questions. The purpose was to examine what other researchers have said computational thinking is and their arguments for why it is important to learn.

Specifically, I based my analysis on the work of PhD student Tauno Palts and Professor Margus Pedaste (2020). Based on a systematic literature review, they present an overview of the dimensions of computational thinking defined in scientific papers. I build on their presentation of definitions to analyze *what* and examine *why* – which is also the reason that *what* and *why* are examined in reverse order in this chapter (*what* before *why*).

2.4.2 Historical Perspectives (Chapter 4)

Societal and Democratic Perspectives by Peter Naur (Chapter 4.1)

The didactic position in my studies builds on German-Scandinavian traditions within education and Bildung, with a special focus on Denmark. Therefore, I wanted to examine Danish history within the field of computational thinking in K-9 to see what has been discussed earlier, and if this could provide answers to my research questions.

During my studies, I found out that especially one Danish researcher, Peter Naur, had already in the 1960s discussed the importance of teaching datalogy in compulsory education. Because of the main position Naur has as a theoretician in my field, I discussed his thoughts and theories in several of my sub-studies. Therefore, I decided to address his theories in more detail in an initial chapter compared with how I approached them in the following journal articles.

Specifically, the purpose of the present study was to present and discuss Naur's perspectives on datalogy for all with a special focus on *why,* in his view, everybody should learn to understand computers and their use, and *what* they should learn. Furthermore, I illustrate his theories with historical examples of *how* this could be taught.

Historical Approaches (Chapter 4.2)

Article A: *Unplugged Approaches to Computational Thinking: a Historical Perspective*. Co-author: Professor Aman Yadav

My historical studies were first and foremost motivated by Denning's considerations. As I discussed in the introduction, Denning argues that there is no need for vagueness with regard to what computational thinking is since the meaning of this concept has evolved since the 1950s and is, according to him, clear (Denning 2017b).

The study resulted in a published article, which was written with Professor Aman Yadav at the beginning of my studies when I first began examining the historical routes of computational thinking. I have included the article here to illustrate how historical examples of teaching unplugged (without a computer) can inspire and complement today's focus on teaching plugged (with a computer, often focused on programming).

From Historical to Present Perspectives (Chapter 4.3)

Article B: *Computational Thinking and Technology Comprehension in K-9 Schools: A Round Trip*. Co-author: Professor Jeppe Bundsgaard

In collaboration with Bundsgaard, I further examined historical aspects of computer science education, but this time with a specific focus on Danish discussions and activities. As was the case with the previous article, this article was written at the beginning of my studies to gain an understanding of developments in the field. At that time I had become aware that, in Denmark, researchers and educators had discussed a subject called datalogy in the 1960s, the purpose of which was similar to that being discussed today. I wanted to examine what had happened from then and until today (50-60 years) to see whether history can provide us with lessons learned and keep us from making similar mistakes and reinventing the wheel.

Society has, however, changed since the 1960s. I therefore also wanted to discuss the shortcomings of the historical discussions. Thus the purpose of including this study in my dissertation is to present the developments up until today and discuss that our historical perspectives are only stepping stones. The world looks very different today than it did in the 1960s; for this reason we need to add to history regarding *why* students need to learn *what*, and *how*.

2.4.3 Present Perspectives (Chapter 5)

Societal and Democratic Perspectives (Chapter 5.1)

Article C: *Technology Criticism in Schools: a Democratic Perspective on Technology Comprehension*. Co-author: Professor Jeppe Bundsgaard

In this study, I analyzed present and future societal and democratic discussions and perspectives. In collaboration with Bundsgaard, I discussed how digital technologies influence our society, our common lives and our personal life, and how it will pose a threat to humanity if the next generation does not develop competencies within computational thinking to be able to "to handle the challenges that we face as a society and as humans" (Schnack 1993 as cited in Nielsen 1998, my translation). Thereby, we based our study on a societal-/ideological-critical position that aims for students to develop the responsibility and competences to act on big problems in society and in that way improve the world in the future.

Originally, we wrote the article in Danish for a book on democracy and subject-specific didactics. In this dissertation, it especially contributes to answering my research question with regard to *why*, in continuation of Naur's historical theories on *why*. In that sense, the main purpose of the study was to analyze what consequences digital technologies have for society *today*, in order to discuss *why* students need to develop *what* kind of computational thinking skills today.

Conceptions in Schools (Chapter 5.2)

Article D: *Computational Thinking in Compulsory Education: A Survey Study on Initiatives and Conceptions.* Co-author: Professor Jeppe Bundsgaard

This study was published as an article that communicates the results of a survey study on initiatives and conceptions of computational thinking in Danish compulsory education. The study was designed as a digital questionnaire that was sent to 145 Danish school principals and was completed by 83 of them. The survey was originally Phase 1 of my intended research design, described in Chapter 2.1, and therefore, it was conducted at the beginning of my studies as well (beginning of 2018).

The study was conducted, and the article was written, in collaboration with Bundsgaard. As I described in my presentation of my intended research design, the survey was originally a preliminary study to select three schools that thought they had a focus on and knowledge about computational thinking and three schools that thought they had low focus on computational thinking and limited knowledge about it. In this dissertation, however, I wanted to analyze their answers regarding *what* computational thinking is according to the school principals, and *why/*if we need to teach this subject area in compulsory education. The idea was to illustrate how the principals' answers reflect Danish traditions within compulsory education, for example by focusing on Bildung rather than on educating students for the labor market.

As is the case with my literature review, I looked at *what* before *why*. The reason for this is that in our survey, we first examined a specific *what* (computational thinking). Afterwards, we wanted to examine the participating school principals' conceptions of *why*.

A Computational Design Experiment (Chapter 5.3)

Article E: *Technology Comprehension in Schools: Computational Design for Solving Authentic Problems*. Co-author: PhD Martin Dybdal

Finally, based on theories from my studies of *why* and *what*, I conducted a design experiment in an eighth-grade class with PhD Martin Dybdal to examine *how.*

The purpose was to examine and combine a way of teaching computational thinking and design thinking by developing a computational design to solve an authentic problem.

Originally, the study was planned and conducted because of my knowledge from my observation studies in my intended research design where I did not see any focused computational thinking teaching in practice. Therefore, I wanted to plan and conduct an experiment to experiment with the theories in practice. With reference to my didactic position, I found theorizing insufficient.

In the following chapters 3, 0 and 5, I present and discuss the sub-studies in terms of method, analysis and outcomes relevant to my dissertation, as well as a sub-conclusions of each chapter. In Chapter 6, I conclude on my overall findings.

3 State-of-the-Art

In this chapter, I present and discuss a review of literature on computational thinking in compulsory education. My analysis is based on an existing literature review on common definitions (*what*) of computational thinking. Based on the results of that study, I examined common arguments (*why*) for embedding computational thinking in compulsory education; that is, arguments for why it is relevant for all children – all individuals – to learn.

3.1 Literature Review

3.1.1 Method

This analysis was centered on two questions: 1) What are the common definitions of computational thinking in compulsory education? and 2) What are the common arguments for embedding computational thinking in compulsory education; that is, why is it relevant for all children to learn?

To begin with, I searched for a complete sample of articles on computational thinking in education based on a number of criterions, for example that *computational thinking* was included in the title, and that the article was relevant to K-9. As I was reading the abstracts of the returned articles, however, I became aware that reviews on *what* already existed. Therefore, I changed my method. Instead of making another review on common definitions, I based my review of *why* on an existing new and comprehensive review from 2020.

In this review, PhD student Tauno Palts and Professor Margus Pedaste (2020) identified six fundamental articles: Wing (2011), Barr and Stephenson (2011), CSTA and ISTE (2011), Brennan and Resnick (2012), Selby and Woollard (2013), and Moreno-León (2015). In the following section, I describe the method and outcomes of the study by Palts and Pedaste, and discuss the findings of their review regarding *what*.

3.1.2 Analysis of What

The purpose of the systematic review study by Palts and Pedaste (2020) was to develop a model for developing computational thinking skills. Their argument was that:

> as many authors have published various ways of defining and approaching CT, this leads us to the problem that not much attention has been dedicated to finding a common understanding of the dimensions of CT skills that would help us focus on developing and accessing CT skills. (Palts & Pedaste 2020)

As such, in their study, they aimed to find a common understanding, and to do that, they systematically reviewed and presented an overview of the dimensions of computational thinking defined in scientific articles. They based their work on two research questions: "1. Which dimensions of CT skills can be identified in articles on developing CT? 2. How can these dimensions from different articles be combined into a new theoretical model for developing CT?"

In my analysis, I focused on their findings regarding their first research question, which is the one relevant to my studies. Based on their results, I continued my examination of why researchers think that exactly these dimensions are important in order to develop computational thinking.

For their systematic review, Palts and Pedaste (2020) used the search engines EBSCO Discovery Service and the ACM Digital Library. For EBSCO they set the criteria set: (1) search term "computational thinking" in the abstract; (2) full text available; (3) peer reviewed, and (4) in English. As the ACM Digital Library search engine has slightly different search options, the following search criteria were set in the ACM Digital Library search engine: (1) search term "computational thinking" in the abstract, and (2) full text available.

They describe:

> The search was carried out on 1 January 2018 and returned 541 matches, including 228 in the ACM Digital Library and 313 in the EBSCO Discovery Service search results.
>
> The next step was to filter out duplicate results (13), only 1–3 pages long texts (127) and those not written in the context of computer science education (32). Then, articles that did not include a clear list of CT skills (313) were excluded. Eventually, 9 articles were added based on references in selected articles. In total, 65 articles were included in qualitative analysis. (Palts & Pedaste 2020)

Their comparative analysis of the data from these 65 articles showed that the articles were often based on each other. Palts and Pedaste (2020) categorized the articles "based on the theoretical framework, definition and dimensions of CT used in each article", and to characterize the descendancy of the articles, they took into account the year of publication. This led them to six clusters of computational thinking dimensions that they identified as originating from six different articles:

* Wing (2006)
* Barr and Stephenson (2011)

* CSTA[12] and ISTE[13] (2011)
* Brennan and Resnick (2012)
* Selby and Woollard (2013)
* Moreno-León et al. (2015).

Figure 2 shows the relationship between the articles as identified from Palts and Pedaste's systematic literature search. The authors of the study explain:

> [...] the arrows show how previously published articles have been used as the theoretical rationale for the new articles; the blue font of the references with a surrounding square indicates key articles that have been further used in the synthesis of the current study. (Palts & Pedaste 2020)

[12] Computer Science Teachers Association

[13] International Society for Technology in Education

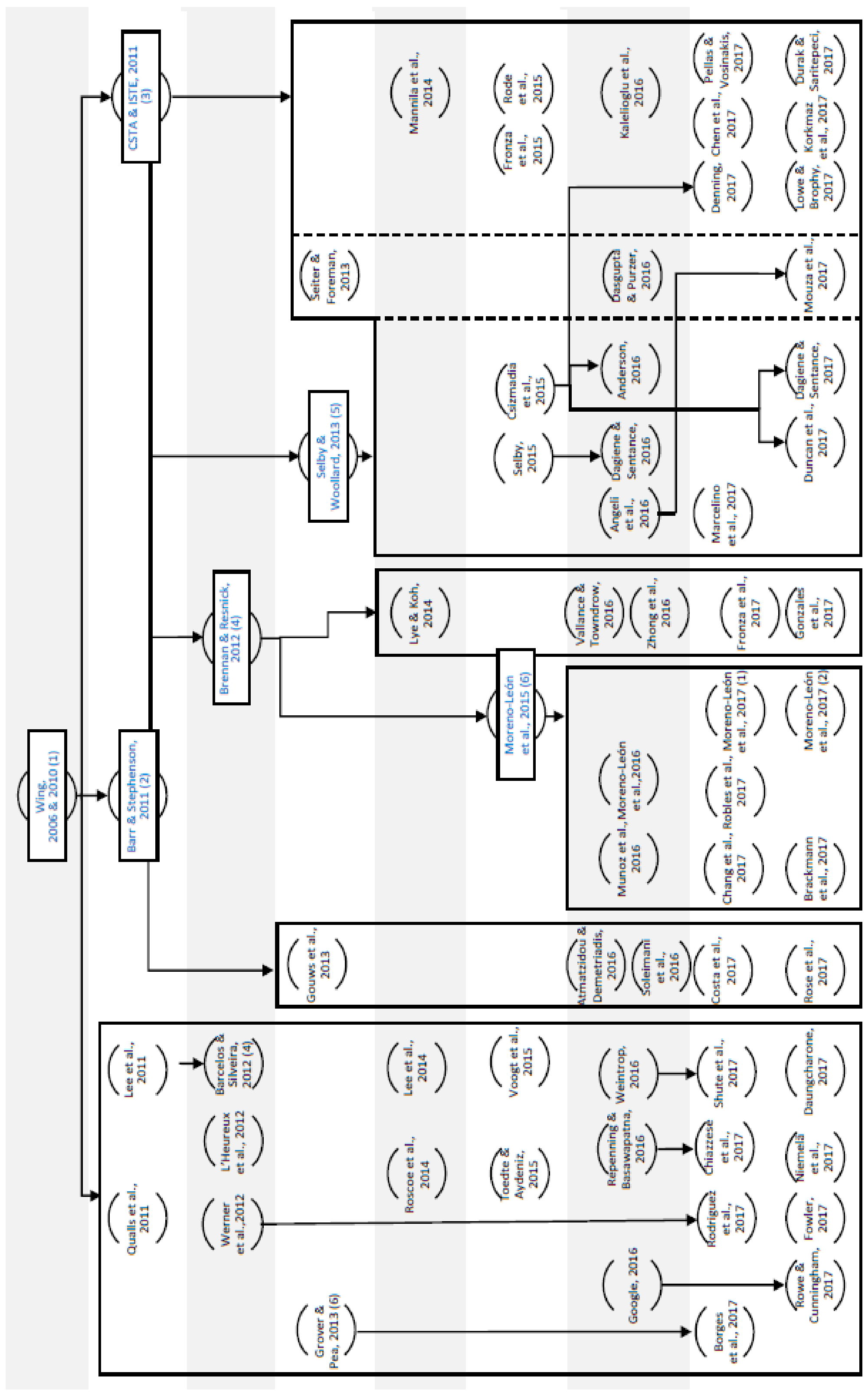


Figure 2. Relationship between articles, identified from systematic literature search by Palts and Pedaste (2020)

As seen in the figure, the articles all refer to Wing as the main driver of the present computational thinking movement. Later articles build on her work as the theoretical rationale.

Below, I specify the definitions that Palts and Pedaste found in each of the six key articles in a chronological order.

3.1.2.1 Wing (2006)

Characteristics of computational thinking: abstraction, problem decomposition, problem reformulation, automation, and systematic testing.

3.1.2.2 Barr and Stephenson (2011)

Three concepts of data manipulation: data collection, analysis, and representation.

Six problem solving concepts: decomposition, abstraction, algorithms and procedures, automation, parallelization, and simulation.

Greater focus on data manipulation and algorithms, and parallelization and simulation are added as separate concepts of computational thinking.

3.1.2.3 CSTA and ISTE (2011)

Six concepts: formulating problems, organizing and analyzing data, abstractions, automation through algorithmic thinking, evaluation for efficiency and correctness, and generalizing.

Main focus: solving problems using algorithms.

Evaluation for efficiency and correctness and are added as dimensions of computational thinking.

3.1.2.4 Brennan and Resnick (2012)

Four practices to assess computational thinking projects: abstracting and modularizing, reusing and remixing, being incremental and iterative, and testing and debugging.

Focus is on project analysis.

New dimensions of iteration and reuse to be used in coding project analysis in several articles.

3.1.2.5 Selby and Woollard (2013)

Identified the terms mostly associated with computational thinking in literature → Computational thinking includes: abstractions, decomposition, algorithmic thinking, generalization, and evaluation.

Data manipulation terms left out for being either too broad, not-well defined or not considered a skill. Generalization and evaluation added from CSTA and ISTE.

3.1.2.6 Moreno-León et al. (2015)

Based on connecting computational thinking dimensions with automatic project analysis. Computational thinking aspects: abstraction in creating functions and clones, parallelism in starting several processes at the same time, logic in using logical operations, synchronization in sending messages, flow control in creating reasonable loops, user interactivity in using interaction, and data representation in using variables and lists in programs.

This approach has opened up algorithmic thinking as a demonstration of usage of parallelism, synchronization, logical thinking, and flow control. Furthermore, data manipulation has been emphasized by data representation and user interactivity.

To sum up on what, Palts and Pedaste (2020) conclude that most articles agree that computational thinking involves solving algorithmic problems, and that core concepts are often described "starting with defining the problem and ending with testing and evaluation". Based on that knowledge, they created a framework for problem solving as a cyclic process with three main problem-solving stages: Defining the problem, Solving the problem, and Analyzing the solution. I will not go into further details with their model since it does not contribute to answering my research questions.

In the following, I examine common arguments for embedding computational thinking in compulsory education; that is, why it is relevant for all children to learn computational thinking. I base my analysis on the six main articles found by Palts and Pedaste.

3.1.3 Analysis of Why

To analyze arguments for teaching computational thinking, I studied the same six main articles as Palts and Pedaste. Below, I list my findings of each article regarding *why*.

3.1.3.1 Wing (2006)

In her essay, Wing states that every human must be able to think computationally to function in today's society. She argues that all of us use computational concepts "to approach and solve problems, manage our daily lives, and communicate and interact with other people". As for that, she gives examples such as packing a backpack or considering what line to stand in in the supermarket.

3.1.3.2 Barr and Stephenson (2011)

Professor Valerie Barr and Professor Chris Stephenson, who is the founding director of the CSTA, claim that it is no longer sufficient to wait to introduce these concepts until students are in college. They argue that: "All of today's students will go on to live a life heavily influenced by computing, and many will work in fields that involve or are influenced by computing. They must begin to work with algorithmic problem solving and computational methods and tools in K-12".

3.1.3.3 CSTA and ISTE (2011)

This article contains a list of an operational definition of computational thinking, and there are no arguments for why it is important. However, the ISTE website introduces a definition: "Advances in computing have expanded our capacity to solve problems at a scale never before imagined, using strategies that have not been available to us before now. Students will need to learn and practice new skills – computational thinking – to take full advantage of these revolutionary changes brought about by rapid changes in technology" (ISTE 2014).

3.1.3.4 Brennan and Resnick (2012)

Associate Professor Karen Brennan and Professor Mitchel Resnick argue that when young people engage in programming, for example through Scratch, which is a program they both contributed to developing at Massachusetts Institute of Technology (MIT), this provides a valuable setting for developing capacities for computational thinking. As for arguments regarding why to teach computational thinking, they refer to the fact that computational thinking has received considerable attention over the past several years. Therefore, their article does not present any arguments for why all children need to learn computational thinking. Rather, it exemplifies how children can learn computational thinking through programming, using Scratch as a programming environment.

3.1.3.5 Selby and Woollard (2013)

In their article, the two senior teaching fellows Cynthia C. Selby and John Woollard focus on contributing to the development of a definition. They justify their inclusion or exclusion of terms based on consistency of usage and consistency of interpretation across the literature and not on arguments. Therefore, arguments for why computational thinking needs to be taught are missing. The authors believe and conclude that: "There is a genuine need for a robust and agreed definition of computational thinking. The definition can facilitate the development of computer science curriculums in line with Wing's original vision to encourage computational thinking for all".

3.1.3.6 Moreno-León (2015)

Professor Jesús Moreno-León, Associate Professor Gregorio Robles, and Associate Professor Marcos Román-González focus on assessment of computational thinking and present a web tool that "allows analyzing Scratch projects to automatically assign a CT score as well as to detect potential errors or bad programming habits, aiming to help learners to develop their coding and CT skills as well as to support educators in the evaluation tasks". They base their work on the fact that they, with reference to Wing, in the last decade have witnessed a resurgence of programming and computational thinking in schools. Thereby, they leave out any questions regarding why students need to develop programming and computational thinking skills.

To sum up on *why*, two of the articles (Barr & Stephenson and CSTA & ISTE) do at some point discuss why students should learn computational thinking skills, and four do not at all discuss why. However, neither Barr & Stephenson nor CSTA & ISTE reflect on this didactical question in any deeper way. Barr and Stephenson argue that computational skills are important as students' lives will be heavily influenced by computing, and many will work in fields that involve or are influenced by computing. CSTA and ISTE point out that students need to develop computational thinking in order to take advantage of the revolutionary changes that computing has brought in terms of large-scale problem solving. But neither article provides any evidence or analysis of *why* this is so.

3.2 Sub-Conclusion of State-of-the-Art

In this chapter, I examined common *whys* and *whats* of computational thinking today.

With regard to *what*, the analysis showed that definitions on computational thinking, as discussed in the introduction with reference to Denning, emphasize skills within mathematical problem solving and thereby underestimate or leave out design and software crafting.

With regard to *why*, the analysis showed that evidence in the examined articles is missing regarding why there is a need to teach computational thinking in compulsory education.

As discussed in Chapter 2, *why* and *what* depend on each other. With reference to Klafki, it is essential that didacticians, decision-makers and students are aware of the underlying reasons for deciding, considering, and acting the way they do. Since many countries are embedding computational thinking in the curriculum, and the arguments for doing so are weak, it is important to further address issues relating to *what* aspects of computational thinking are essential to teach in compulsory education, and *why*. I begin by looking at historical perspectives since I found out that *why* and *what* have already been discussed by former researchers and educators and with broader conceptions of the subject than the common conceptions in literature today described in the above.

4 Historical Perspectives

To find out what computational thinking is, Denning suggests to look in our history since "computational thinking has a rich pedigree from the beginning of the computing field in the 1940s". Therefore, he says, there is no need for vagueness on what computational thinking is (Denning 2017). Moreover, as stated in the introduction, he says that the claims that it benefits everyone are unsubstantiated.

From my analysis of definitions and arguments in literature, I found similar results (Chapter 3). I argued that today's computational thinking movement is in general characterized by mathematical problem-solving concepts, and that in-depth arguments and evidence of why we need to teach computational thinking in compulsory education (and *if* we need to teach it at all) are missing. If arguments exist, they are general and superficial.

Motivated by Denning's suggestion, my analysis, and my didactic position that emphasizes the importance of knowing *why* and *what*, I decided to examine the historical roots of computational thinking, specifically with a focus on discussions and theories concerning compulsory education. These historical analyses of what computational thinking is, and what we have already done, discussed, and found out in the past, have significantly contributed to my overall research questions concerning *why* computational thinking is essential to teach in compulsory education, and *what* aspects of it are essential to teach.

In the following Chapter 4.1, I present and discuss the Danish professor Peter Naur's perspectives on datalogy for all people and especially *why* everybody in his view should learn to understand computers and the use of them. In addition, I will present what Naur broadly discussed about what students should learn. Naur's thoughts laid the foundation for a K-9 subject named *data education* (*datalære*), and in order to describe in more detail the historical thoughts about what students should learn, I present further works from that time that focus on especially *what* and *how*. That Naur's arguments are valuable in regard to answering my research questions in a contemporary context is partly due to the fact that he made a number of – in many ways accurate – assumptions about how datalogy could and probably would be used in the society of the future (our present) and would affect all people. His arguments that everybody should learn to understand datalogy must be seen in the light of these predictions – that datalogy is a relevant tool for all of us.

Next, in Chapter 4.2, I present the results of a study, predominantly based on Naur's perspectives, which illustrates what we can learn today from history with regard to *how* in the classroom. In this study, I specifically discuss merging historical ideas with the present for students to gain a deeper understanding of computers as tools for people, for example to solve key problems in our society.

Finally, in Chapter 4.3, I present results from a study on the development from the 1960s and up until today. Concerning my research questions, I discuss what parts of historical computer science education are not sufficient today; that is, *what* children in addition to historical perspectives need to know in the world we live in today. This will lead me to Chapter 5 on present perspectives.

4.1 Societal and Democratic Perspectives by Peter Naur

In my view, one of the most significant researchers within the field in a Danish context was professor of datalogy Peter Naur, who as early as in the 1960s argued that everyone should learn to understand data, their nature and their use, and thus also learn to understand computer programming. He introduced the term "*datalogi" (datalogy)* as the Danish translation of the term *computer science* as a protest against computer science that he found was a misleading name since it emphasizes computers instead of what the science is about, namely humans. "Datalogy has in it the human aspect", he explained (Naur 2005), and further: "Data is a matter of human understanding". In 1966, Naur argued that a datalogy subject for everyone was necessary in a democratic society, even though not everyone was to become a computer scientist.

The purpose of this study was to examine how Naur's historical theories on *why*, *what,* and *how* with regard to the historical K-9 subject of data education can contribute to the *why*, *what,* and *how* of computational thinking education in compulsory education today. Throughout this chapter, I use *datalogy* as a term instead of computer science because of Naur's specific distinction between these two terms.

4.1.1 Method

Naur's societal and democratic perspectives on datalogy have been central in my research. In this study, I present and discuss his thoughts and theories with regard to compulsory education as a subject-specific didactic contribution to the field concerning my research questions: *Why* we need to teach computational thinking today, *what* aspects of it we need to teach, and *how*.

The study is based on a narrative review of his work. I searched for literature by Peter Naur in library databases and I followed references from the literature found; that is, I used a snowballing technique. I have only included works that were relevant to answer my overall research questions as well as relevant to compulsory education.

4.1.2 Analysis of Why and What

While society with regard to digitalization has gone through enormous changes over the last six decades, Naur was already in the 1960s aware that all people should learn to understand digital technologies. He explained the need to learn datalogy by comparing it with learning language and mathematics, as he found significant similarities between the three subjects. He said:

> When I say language learning, I mean the phenomenon in the broadest sense. I want to include very basic things, reading, writing and spelling, and you can continue with grammar, foreign languages, literature, stylistics, linguistics, and probably even more. When I say mathematics, I also think of the whole spectrum of this subject from simple arithmetic in primary school to applications of many kinds and up to advanced pure mathematics.
>
> Both language and mathematics are deeply influenced by the fact that they are the most important aids, tools, for a large number of other activities. It is hardly too much to say that language is the most important human aid of all. This circumstance demands the placement of these subjects in the teaching; mathematics and language are both incorporated in two completely different ways, namely partly as in-depth special disciplines side by side with many other special disciplines, and partly as transversal auxiliary subjects; that is, as subjects whose more elementary basic principles and aspects are taught to students who later specialize in completely other disciplines. We all learn to write, read, and calculate, whether we end up being artists, doctors, lawyers, or whatever.
>
> It is my belief that in the long run it will be recognized that a subject exists
>
> > <u>datalogy</u> – the science of data, their nature and use –
>
> which must be included in our education in a way that is very similar to learning language and mathematics.
>
> (...)
>
> From this point of view, I predict that datalogy will be placed as a subject in education, and that its basic concepts will become common possession and not just a matter for specialists. In parallel, in-depth studies for the education of specialists will develop, quite analogous to what is the case in language and mathematics. (Naur 1966b, my translation)

Naur defined datalogy as the science of data, their nature and their use. He believed that the elementary parts of datalogy should be included in compulsory school teaching. A good basis would be the concepts of data representation and formal data processes as well as exercises in or demonstrations of realizing data processes with concrete equipment, he said (1966a). He explained how data, data representations and data processes contain a completely new view on many human forms of expression. "Through their intimate connection with these universal human things, the societal consequences of computers become far more understandable" (Naur 1967, my translation).

Naur said that we introduce data as an aid, a tool, to do things in this world. He explained a data process by saying that you have a reality that you, with a specific purpose in mind, are interested in changing. But instead of changing it directly, it is often better to make a transition to data and make a data process, for example to see the possible consequence of a change before changing it in reality. To illustrate how computers affect society, Naur explained that data models allow one to experiment *with* reality without having to experiment *in* reality. For example, an engineer can perform calculations over a bridge they are building before building the bridge, instead of experimenting with how it behaves in different situations.

Overall, he described what at the time were contemporary, applications of datalogy as *information search*, for example by means of keywords: *language translation* of, for example, scientific or technical literature; *language analysis*, where a computer, for example, can systematically search for ambiguities of sentences; *nuclear reactors*, with the use of data models to study the consequences of certain constructions before making them; *road construction*, where a computer can, for example, include a number of different considerations such as reducing the amount of soil to be moved during road construction or experiments with the visibility of road users; *money and stock administration*, where a computer program can, for example, keep accounts of ordering production parts; *production planning*, for example in relation to the order in which production parts are to be produced to avoid waiting times for the machines; as well as *direct datamatic control* of, for example, chemical productions. Overall, he emphasized, "that we are already long past the stage when calculations were the most important work for them [the computers]" (Naur 1967, my translation).

With the argument that notions of the future are important for present dispositions, he presented his thoughts on the societal important possibilities of computers, as well. Specifically, he anticipated developments in communications, including an expansion of the telephone service as well as terminals with a television screen. He envisaged a system for keeping track of people's financial circumstances, such as calculating taxes and advising on favorable loan options. For such tasks, it was crucial that services were linked to a common public network – a central computer system – he said. Completely private computers would not have similar benefits, as many beneficial calculations would depend on external conditions, such as prices and legislation, which had to be kept up to date centrally. With such a central system, payments could also "be made without the transfer of money or paper, simply through an internal data process in the system, which causes the payer's account to shrink and the payee's to increase with the amount" (Naur 1967).

Another future use of datalogy could be private counseling, he said. For example, "contact about positions and about private purchase and sale of property which now takes place through ads in the daily press could be taken over by a central computer system". With such a system, one could also provide more detailed information and one could make specific demands so that both parties only saw relevant options. At the same time, the system could keep track of whether certain positions were posted or a particular movie would hit a particular cinema.

These were some of Naur's thoughts on future good use of data – to help people. However, he also thought about future problems. For example, the possibility of *datamatic*[14] surveillance of the society with the aim to analyze and discover complicated connections in the society much more thoroughly. In a way, such surveillance of, for example, the health status of the population could help trace sources of disease. The danger of datamatic surveillance of citizens was, however, that the information could be misused, that is, that data about us could be used for undesirable purposes. He said:

> Using computers, however, will make it possible to keep track of more information about more citizens, and it will be easier to search for specific information. Therein lies a temptation for the administration to accumulate information, in that case that they would later be useful. (Naur 1967, my translation)

Based on his thoughts about the use of data and the consequent awareness that it would require an understanding of datalogy to influence decision-making processes in the computerized system of the future, Naur stated that: “There is no way around it, we all need to understand computers” (Naur 1968, my translation). He said:

> This fact is the reason why many of us who are close to computers, and who think about their societal consequences, feel that we at every favorable opportunity must emphasize that the understanding of computer programming must be brought into general education and thus become common possession. (Naur 1968, my translation)

By that he meant that those who understand how computers work will have the power of the system, and therefore, in a democratic society we should not give up and leave the understanding (and power) to programmers.

> All of us have learned significant amounts of language, arithmetic and mathematics in school, regardless of the fact that only quite a few of us have become linguists or mathematicians. Similarly, datalogy must be brought into school and prepare us all for life

[14] Adjective form of datalogy.

in the age of computers, just as reading and writing are considered a necessary precondition for life in a society characterized by print. (Naur 1967, my translation)

Naur did not think that his perspectives would be realized, though, when he looked at what he described as the inertia that characterized the development of educational institutions (Naur 1966a; 1968). Despite this, he relentlessly shared his perspectives because he believed it was the right thing to do – because it was of great societal importance when it came to central decisions (Naur 1966a).

As presented in Chapter 2.3, Klafki's general critical-constructive didactics that I position myself within focuses on people being able to increase their freedom and "break down an unsubstantiated domination of power". This is in line with Naur's subject-specific arguments on why everybody should learn to understand computerized systems in a society dominated by computerized systems. Those who understand the system will have the power. It is also in line with Schnack who argues for the need to determine content based on analysis of the competencies that are needed in a democracy.

Naur argued how the subject could be included in the compulsory education curriculum as an independent subject or integrated in a subject such as mathematics, but he did not find organization central. "The important thing is the content," he stated (Naur 1967).

Naur discussed that a subject should focus on fundamental concepts regarding data and their use, for example how content can be represented very differently, and how we with formalized descriptions of a problem can become aware of opportunities we have not seen before. The subject should mention computers, but he pointed out that it was not the most central part. He believed, however, that the subject should deal with data representations that could be processed automatically or computerized, and that working with automatic data processes would require exercises with machines. Only after this foundation was taught, should a language for programming be introduced. He emphasized that the purpose of compulsory education was to build such a foundation for understanding and not to merely prepare the students to become, for example, programmers (Naur 1966a; 1967; 1968).

Naur's perspectives were central to the Danish subject that in the 1970s was called *data education* (*datalære*). In 1972, the Danish Ministry of Education published a report on the subject (Ministry of Education 1972), formulated by a committee set up by the Ministry. Their descriptions and recommendations of the subject in compulsory education were, among other things, based on Naur's descriptions of datalogy and his thoughts on education. The committee highlighted general, interdisciplinary concepts, such as data, problem formulation, model, algorithmization and process to be part of practical and application-oriented problem-solving processes. They warned against orienting the teaching towards programming or coding, as problem-solving would then be too specific rather than open and creative. Specifically, they formulated the following purpose for the subject:

* To give insight into fundamental, interdisciplinary datalogical issues and concepts
* To communicate knowledge about possibilities and limitations of computers
* To inform about applied data processing and societal advantages and disadvantages associated with extensive use of automatic data processing.

(Undervisningsministeriet 1972, my translation)

More detailed, the students should learn to be able to:

* recognize different types of data carriers, data representations, data structures, etc. from everyday life.
* give an algorithmic description of various basic processes, such as simple sorting problems, searching problems, and updating problems.
* assess the suitability of simple algorithms.
* program and run smaller tasks on a computer in a simple language.
* describe the main components of a computer.
* give an overview of fundamental phases in a data processing project.
* describe the main areas of applied data processing, and consider related societal aspects.

(Undervisningsministeriet 1972, my translation)

The committee suggested the following topics as a starting point for later preparation of a detailed content description:

1. The concept of data
    * *Data representations and notation forms*
    * *Data structures and data processes*
    * *Data Organization*
2. Problem formulation and task structuring

3. The concept of models and model types
4. The concept of algorithms
    * *Algorithmization*
    * *Algorithm descriptions*
    * *Problem-oriented programming language*
    * *Reading, writing and running programs*
5. The basic structure of a computer
6. Data processing systems, data processing applications and societal aspects.

(Undervisningsministeriet 1972, my translation).

Their content description was based on the formal aims of Danish compulsory education, which they were aware were to come into force three years later, in 1975. What was new in these aims compared with earlier formal aims were, among other things, that compulsory education should focus on preparing for life in a democracy, on equal opportunities for all, as well as on students' personal development (AU 2018). These aims are visible when looking at how the committee emphasized everyday life, areas of applied data processing as well as societal aspects. Also, they emphasized awareness of computers as tools in order to counter the bureaucratization and technocratization in society caused by the increasing use of computers. They directly pointed out that data education would help fulfill the new aims of compulsory education (Ministry of Education 1972).

In line with Naur, the committee stated that compulsory education should not focus on teaching students specific professional competences, and that programming was not the core of data education. In addition, since programming languages were at that time hampered by strict formal structures and technical details that could limit creativity, they suggested using flowcharts as an alternative to describe the nature of algorithms. However, while flowcharts worked well for communication between people, they found it problematic that students would not be confronted with the consequences of their solutions through contact with a computer. Therefore, despite the risk that the subject would be centered on machines and not people, they believed that using computers was necessary – both as a motivating factor and as a valuable tool for testing solutions. This was an issue that they suggested be analyzed further.

Seemingly, the subject was ahead of its time in both Denmark and other countries. In fact, it was never fully implemented – even though the committee argued for the importance of data education for all and recommended a curriculum, and even though many teachers and schools initiated initiatives on their own. Few seemed to have realized at that time the importance of everyone learning to understand and apply datalogy as a tool for human activities.

4.1.3 Analysis of How

As stated, Naur believed that a datalogy subject should be included in compulsory education in a similar way as language and mathematics to learn about processing of data with automatic tools. He argued that the impact of computers would affect our inner selves – that is, our way of thinking and our way of dealing with problems. Therefore, his argument for a subject built on the fact that datalogy is a tool for people. He called such tools *datamates* (*datamater*) – machines that process data (Naur 1966b). Datalogy did not necessarily involve the use of computers, though. Naur's focus on people as central is illustrated in the way he described the relationship between people, problems and tools in problem-solving processes. One of his basic theories was that human understanding and problem formulation are closely related to the tools at their disposal at any given time, whether digital or not, as depicted in Figure 3.

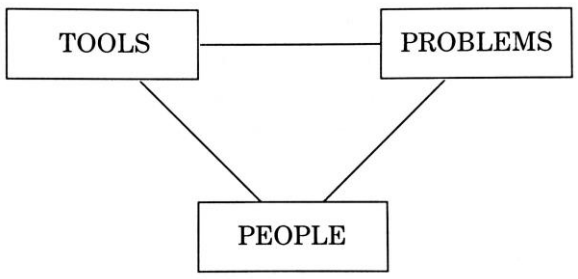


Figure 3. People, problems, and tools as fundamental elements in problem solving (Naur 1965)

The figure illustrates a symmetrical relationship between people, problems, and tools. One can view problems from any of the three elements and discover a relationship with the two remaining ones. Naur explains that:

> We can view it from the position of either of the three elements and discover that the two remaining have a subtle relationship: People can only understand the problems on the background of an assumed set of tools, while a tool designed to solve a problem which is understood by nobody is meaningless. Problems exist only in the minds of people and only relative to understood tools. Tools only exist as such in so far as some people think of them as the proper things with which to solve some problems. (Naur 1965, translated in Sveinsdottir & Frøkjær 1988)

Illustrating his point with programming languages as tools, he continued:

> For good or bad, the characteristics of the programming language will shape the thinking of people and their conception of problems. The dangers of this are at once apparent when we realize that programming languages are designed to be, as it is said, problem-oriented. Now, in the view I put forward here there is no problem without an understanding of the tool available for solving it. Being problem-oriented must therefore imply an understanding of a tool also. What crops up here is of course the conventional view of the problem, the view based on older tools. The great danger of problem-oriented languages is therefore that they will tend to perpetuate a view of the problem which is appropriate to an obsolete tool. (Naur 1965, translated in Sveinsdottir & Frøkjær 1988)

In line with Klafki, Naur was against the usual way of teaching a curriculum of highly structured knowledge. He argued that "successful project work depends on the skilled application of certain techniques", and that the "difficulty lies not in the mastery of each of these, but rather in their interplay and in their consistent, yet flexible, adaption to the actual project at hand" (Naur 1970).

4.1.3.1 Historical Experiments with How

Though the committee recommended implementing data education in compulsory education, they also stated a number of challenges in implementing the subject because of a lack of resources. Too few teachers was educated to teach the subject, and therefore, it was important to include data education in teacher education as soon as possible for both pre-service and in-service teachers. In addition, there were no teacher and student resources (textbooks) for the subject, and the committee recommended to set up working groups that could prepare suitable resources. And finally, there was a lack of computer equipment, that is, computers, as an aid to illustrate automatic processing of algorithms and to demonstrate the iterative nature of problem-solving processes (Undervisningsministeriet 1972).

One of these challenges was dealt with by computer scientist Erik Frøkjær in collaboration with Carsten Fischer and Lisbeth Gedsø. In 1972, they published a student textbook called "Datalære i skolen" (*Datalogy Education in School*) with assignments and activities and an accompanying teacher's book (Fischer, Frøkjær & Gedsø 1972a; 1972b). Frøkjær was at that time studying mathematics and electrical technology at the Technical University of Denmark and was later, in 1985, employed at the Computer Science Department, University of Copenhagen, where he worked together with Naur. The books were written on the basis of the committee's recommendation to introduce the subject in compulsory education that basically was in line with Naur's theories. This is why it is relevant in continuation of his theories.

The three authors concretized data education to include:

1) Basic concepts such as data, data representation, algorithms, and data processing.
2) Description of a computer, its structure, operations, possibilities and limitations, as well as techniques used to solve tasks on computers.
3) Description of areas of application for data processing. The societal consequences of widespread use of computer technology.

Below, I present and discuss some of the activities within each of the three topics to illustrate examples of what teaching looked like in practice historically, and discuss whether historical *hows* can benefit the *hows* of a subject today.

The purpose of *the first topic regarding basic concepts* is for students to understand that a lot of what we do in our everyday life can be described as data processing, and that data processing takes place as part of solutions to tasks or problems. As a first foundation, the concept of data can be explained using examples of different forms of representation, such as mathematical expressions, sentences, numbers, letters, and drawings. Conversely, it can also be exemplified that random consequences of dots and dashes or 'an apple' are not data in themselves. Next, students can work on tasks where, for example, they have to select an item (such as an apple) and find different data about the apple as well as work on how data can be transformed or processed into something else (for example, a sentence can be translated into English without losing its content, and numbers can be described with matches instead of regular numbers).

Other types of tasks could be to consider situations with problem solving from everyday life as data processing, for example having to cross a street with traffic lights, where data is represented by the color of the traffic light that the brain processes based on the knowledge of the meaning of the colors of a traffic light. When we are on one side of the traffic light (old reality) and want to cross to the other side of the road (new reality) without being hit by a car, we use data to make a data process in our heads before we start walking. In this way, students can learn that with data processing, one can consider the consequences of an action without actually performing it.

Teachers can introduce the concept of algorithm with flowcharts. With flowcharts, students can work on reading and describing different data processes themselves – for example, how to pump a bike as shown in Figure 4 below. The authors explain: "Once you have developed an algorithm to deal with a problem, you can save it and retrieve it every time you need it" (Fischer, Frøkjær & Gedsø 1972a, my translation).

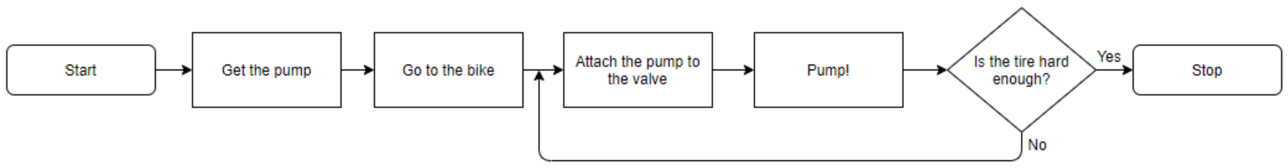


Figure 4. Instructions on how to pump a bike, described by a flow chart (reconstructed and translated from Fischer, Frøkjær & Gedsø 1972a)

Moreover, the authors suggest working with arranging data in data registers (for example, an alphabetically arranged list of goods).

The purpose of *the second topic regarding the description of a computer and techniques* is that students come to understand how computers can be used to solve problems by their programming. Students can work on understanding a program as an algorithm that is described in a way so that a computer understands it; that is, a series of very precise orders to be executed in a specific order.

The authors suggest to illustrate a computer's execution of orders by comparing it to an authentic situation at a factory that produces a product – for example liver pate – controlled by a worker who follows a program with the following orders: order lard, order liver, transfer lard, mix liver, save liver pate, deliver liver pate. Here, the task is to produce liver pate, the input is liver and lard, and the output is liver pate – if the worker only does what the program orders him to do.

Similarly, a program on a computer with the task to add two numbers could be: read a, read b, take a, add b, save sum, write sum. Input data are the two numbers that in the program are called a and b, while output is the sum of the two numbers. In order to get the computer – or worker – to execute the same program over and over again, teachers can introduce the command "jump to n", so that the control unit jumps to a given order and follows the program from there. For example, if the designer of a program asked the worker to jump to order 1 after the last order ("deliver liver pate"), he would start all over again, which means he would automatically produce a new portion of liver pate. This is a way to work with an understanding of programming, but without necessarily (or in the first place) using a computer.

The third topic regarding areas of application and societal consequences is quite simple when compared to the possibilities and use of computers today, nevertheless, it is still worth studying. The authors present examples such as calculating salaries, ordering airline tickets, managing factory productions, and monitoring hospital patients. They discuss that we can use a computer to perform many different tasks, but emphasize that "we must know exactly how the tasks are to be solved" in order to be able to program the computer to execute the orders (Fischer, Frøkjær & Gedsø 1972, my translation). They differentiate between problem solving (the more creative part) and programming or coding (the more technical part).

Discussions with students can focus on what a computer is good at and why. Teachers can compare it to another tool – for example a hammer that is good at hammering, whereas the computer is good at working quickly and accurately and can keep track of large amounts of data. As humans, we can use the computer as a tool to do things faster than we can do as humans, and to solve tasks with economic advantages.

Understanding the human-machine relationship is essential. The authors point out that in principle humans can solve the tasks we solve with the help of computers. But only in principle: "In practice it often turns out that the amounts of data are so large that only computers can manage to solve them" (Fischer, Frøkjær & Gedsø 1972, my translation). They give an example of how launching space rockets would be impossible without computers to control their course. A launch takes a few minutes, and "during that time the computer makes so many calculations that it would take a scientist months to do the same" (Fischer, Frøkjær & Gedsø 1972, my translation). In addition, a computer never makes mistakes or gets tired like humans do, as long as it is programmed correctly.

The authors also point out the importance of discussing how the computer is not a substitute for humans. It falls short in social situations with human contact. However, some of the examples mentioned in their book have actually been replaced with computers today. For example, self-service using computers in supermarkets and so-called 'social robots' that are aimed at replacing parts of previous human contact. In a contemporary context, students can discuss what we as humans have won and lost by letting computers take over some social situations as well.

The authors suggest that students can discuss what types of task computers are good at solving and what they are not good at solving. They suggest working with punched cards, which in a contemporary context should be supplemented with contemporary programming methods. However, teachers could possibly use punched cards as a starting point for students to understand some of the basic principles of early programming.

Examples and tasks from the book need a contemporary adaption, but many historical discussions are still relevant today, which is why I have chosen to include them here. For example, the authors mention tax calculation, banks, statistics, research, weather forecast, literature search, language translation, teaching, process management, hospitals, the military, and the central personal register (CPR). Students can discuss the types of data processing in the mentioned areas. With regard to teaching, students can also consider and discuss advantages and disadvantages of automatic control of what they have learned, as well as issues concerning computers replacing human contact between teachers and students.

Misuse of data was also highlighted at the time. The authors illustrated the problem with examples such as secret data (for example a baker's unique recipe that competitors should not see) and the information collected about all of us in personal data registers when clear rules are lacking regarding what information can be collected and to whom the information can be given. The authors point out:

> Once the data has been collected, there will always be a risk that the wrong persons will gain access to it. (Fischer, Frøkjær & Gedsø 1972, my translation)

They aim for students to become aware of the risk that data storage in personal registers can affect people unreasonably, and that by comparing a whole series of data about a particular person in a data register, one can get a detailed picture of a person's behavior.

> This was not possible in the past, since it was simply unmanageable to keep track of so much data. With computers, it is manageable, and if it is done, there is a risk that information that is currently described as private can be misused. (Fischer, Frøkjær & Gedsø 1972, my translation)

They suggest that students discuss ways to make reassuring rules for using data registers that contain personal information. These issues are still very relevant today. Collection, storage, processing, resale and leakage of personal data are often headlines in today’s news.

The historical *how* examples presented in this chapter illustrate a value in transferring didactic reflections on *why*, *what* and *how* from data education to a present subject. I discuss this further in the following chapters.

4.1.4 The Value of Naur’s Perspectives Today

As stated in the introduction, many countries are currently implementing subjects and initiatives aimed at students developing computational thinking. This new development began with an essay by Wing (2006) in which she reintroduced computational thinking as an important skill in line with reading, writing, and arithmetic, similar to what Naur and other pioneers discussed in the 1960s.

In Denmark in 2018, the former Danish Minister of Education launched the experimental subject technology comprehension. The purpose of this subject is in many ways similar to the purpose Naur and the Johnsen Committee formulated for a subject, for example to be able to understand and critically reflect on the possibilities and consequences of digital technologies in a society where digital technologies play a major role (EMU 2019).

The formal objectives for the subject include four competence areas: Digital Empowerment, Digital Design and Design Processes, Computational Thinking and Technological Knowledge and Skills (EMU 2019), and the purpose of the subject is:

> In the subject technology comprehension, students must develop competencies and acquire skills and knowledge so that they constructively and critically can participate in developing digital artifacts and understand their impact.
>
> Paragraph 2. Students' mastery of the subject requires mastering digital design processes and the language and principles of digital technologies in order to be able to iteratively and collaboratively analyze, design, construct, modify and evaluate digital artifacts in order to recognize them and to be able to solve complex problems.
>
> Paragraph 3. In technology comprehension students gain competences to understand the capabilities of digital technologies and the implications of digital artifacts in order to strengthen students' capacity for understanding, creating and acting meaningfully in a society where digital technologies and digital artifacts increasingly are catalysts for change. (EMU 2019, partly my translation, partly translated by Smith et al. 2020)

The critical-constructive Bildung-centered tradition is reflected in the purpose of technology comprehension that states that children should be able to constructively and critically understand the impact of digital technologies in order to understand and act in our society. Therefore, it is relevant to bring up in this context.

Even though there are clear similarities between then and today, there are, however, also clear differences. Though we still face many of the same problems and challenges as we did 50-60 years ago, we do have easier access to computer devices and software, and hence have much better opportunities to move from *thinking* and *designing* algorithms to (also) *programming* and *testing* the solutions in practice. Excessive development of software has made it easier for children to create plugged designs, that is, where the solutions can be programmed and tested in reality. Therefore, they can go a step further than merely *orientate* themselves about the application of their solutions.

For example, in technology comprehension as a subject, Digital Design and Design Processes is regarded as one of the competence areas that aims for students to understand the possibilities and consequences of data processing. This has solved one of the challenges that the Johnsen Committee pointed out in the 1972 report: that students would not be confronted with the consequences of their solutions when they 'plug' them to a computer.

At the same time, the Johnsen Committee was aware of the risk that a subject would be centered on machines and not on humans, for example that use of a programming language with strict formal structures and technical details could limit the creativity of students. Naur's model (Figure 3) can remind educators to reflect on this when planning their teaching by remembering the symmetrical relation between people, problems, and tools, and not, for example uncritically focusing on using a specific tool that the students do not understand as a tool, or digitalize a solution for a problem that students do not regard as a problem.

In Figure 5, I developed Naur's model into a contemporary perspective that can serve as a didactic framework when planning, conducting, and evaluating teaching in a subject today with the intention to use tools to solve problems for people:

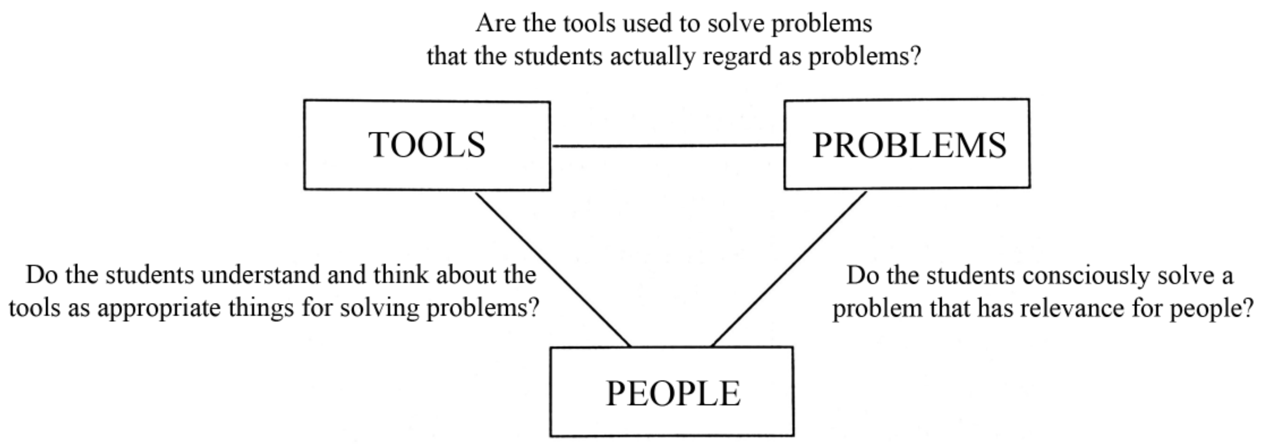

Are the tools used to solve problems that the students actually regard as problems?
TOOLS
PROBLEMS
Do the students understand and think about the tools as appropriate things for solving problems?
Do the students consciously solve a problem that has relevance for people?
PEOPLE

Figure 5. Subject-specific didactic framework for planning and evaluating teaching in technology comprehension – based on Naur 1965

Educators can, for example, use the framework to plan their course based on the three perspectives by asking:

* What is the relationship between *problems* and *people*? Do the students consciously solve a problem that has relevance for people? The questions allow the teacher to see whether the students are working with authentic problems, or whether they risk working with digitalization simply 'because we can'.
* What is the relationship between *people* and *tools*? Do the students understand and think about the tools as appropriate things for solving problems? These questions will help the teacher become aware of whether the students have what it takes to solve problems, and whether they understand the tools used for open problem solving, or whether there is a risk that students' limited functional user skills will limit their use of tools.
* What is the relationship between *tools* and *problems*? Are the tools used to solve problems that the students actually regard as problems? Here, the teacher can become aware of

whether the tools are used to solve problems, or whether there is a tendency for using the tool to become the goal in itself.

Educators can ask themselves the same questions during and after a course in their formative evaluation of their own or others' teaching. In this way, they can become aware of implementing technology comprehension in meaningful ways when solving authentic problems by using tools. And although tools play a significant role, they can avoid operational uncreative mastery of tools to be the ultimate goal, as well as remember that the importance of the problems students should solve should be relevant to people. In Chapter 5.3, I demonstrate a way of using the framework in practice today when presenting an experiment that I co-designed.

4.2 Historical Approaches

This study was conducted in collaboration with Professor Aman Yadav. We surveyed the historical routes of computational thinking, and how history can inspire and inform initiatives and progression today. Our results were published as an article, which is included in this dissertation as Article A.

Our views on *what* and *how* are especially based on perspectives by the Danish professor Peter Naur from the 1960s. With regard to *how*, we add to today's focus on using computers and learning to program ('plugged') by introducing ways of teaching computational thinking without a computer as well ('unplugged'), and we illustrate how historical approaches to teaching in the field can serve as inspiration today.

4.2.1 Method

This study was motivated by Denning's paper on remaining trouble spots with computational thinking (2017b). In this paper, he asks, "What Is Computational Thinking?" As I have described in the introduction to this dissertation, Denning states that a "good place to look for an answer is in our history". In a chronological order, Denning briefly presents historical discussions among researchers regarding the field of computational thinking. Our study took these references as a starting point, continued with Danish perspectives based on Naur's theories, and finally analyzed what these historical theories and thoughts mean with regard to teaching computational thinking in K-9 classrooms today.

In this manner, we used a snowballing technique to collect parts of our empirical material, combined and compared this material with the empirical material, described in detail in Chapter 4.1 (Danish historical theories), and then analyzed how these theories would impact practice. In this regard, the main contribution of this article is historical perspectives on how, combined with present perspectives.

4.2.2 Analysis of Why and What

In this article, we presented historical discussions on why everyone should develop computational thinking skills, what skills they need to learn, and how, with a specific focus on Danish historical discussions in the field, especially discussions by Naur. In Chapter 4.1, I presented and discussed Naur's perspectives in more detail, and I will therefore refer to that chapter regarding his thoughts and theories and not repeat them here. To briefly sum up, though, what is important in this context is Naur's focus on his term *datalogy* – the science of data, their nature and use – rather than the American term computer science that emphasizes computers and therefore indicates that machines are the most central part.

Naur focused on human thinking and understanding rather than on just learning to program. In our article, we explained how he "saw programming as a tool that can influence students' thinking in ways where they see problems and possible solutions based on a tool's perspective and what that specific tool is capable of" (Article A). Based on Naur's perspectives, we argue how teaching programming from a tool's perspective with closed-ended solutions "does not alone develop deep understanding and wicked problem-solving skills" (Article A). We argued that human creativity and innovation must be at the center and that computing tools play an ancillary role.

In the following, I discuss how to move beyond merely focusing on computers by introducing alternative ways of working with students' thinking rather than working with specific tools.

4.2.3 Analysis of How

Based on the historical discussions, we advocated for combining unplugged and plugged approaches when teaching computational thinking. In our view, this provides students with a better understanding of what computer science is and that what happens in the machines is not magic. A machine cannot think – it "simply runs programs, designed by human brains" (Article A).

For example, instead of – or combined with – using programs with intuitive interfaces that are easy to use without understanding, for example to program a robot, the students could work with basic flowcharts to visualize and come to understand how traditional algorithms that execute instructions step by step work. I elaborated on that idea in Chapter 4.1.3.

Working with unplugged flowcharts is not sufficient, though. Instructions must be so precise that machines can understand them. An algorithm does not require human judgment (Denning 2017b), and therefore we argued that students eventually need to implement their algorithms in a machine to test their computational ideas and solutions to gain an understanding of automation processes and what computers are capable of (and not capable of). For example, Denning argues that the present computational thinking movement presents algorithms as any sequence of steps, such as the procedures we follow in our daily life and that this makes fuzzy and overreaching claims about what computers can actually do. With this in mind, we argued that examples from daily life, like packing a backpack, tying shoelaces, or baking cakes "might be useful to illustrate and connect the idea of algorithms with well-known activities – but they should not stand alone" (Article A).

Another unplugged way of learning how computers work is inspired from early work on computer science in the classroom from the 1980s. Figure 6 shows a group of grade 5 students using matchboxes to build a model of a computer to understand its different parts and how it works.

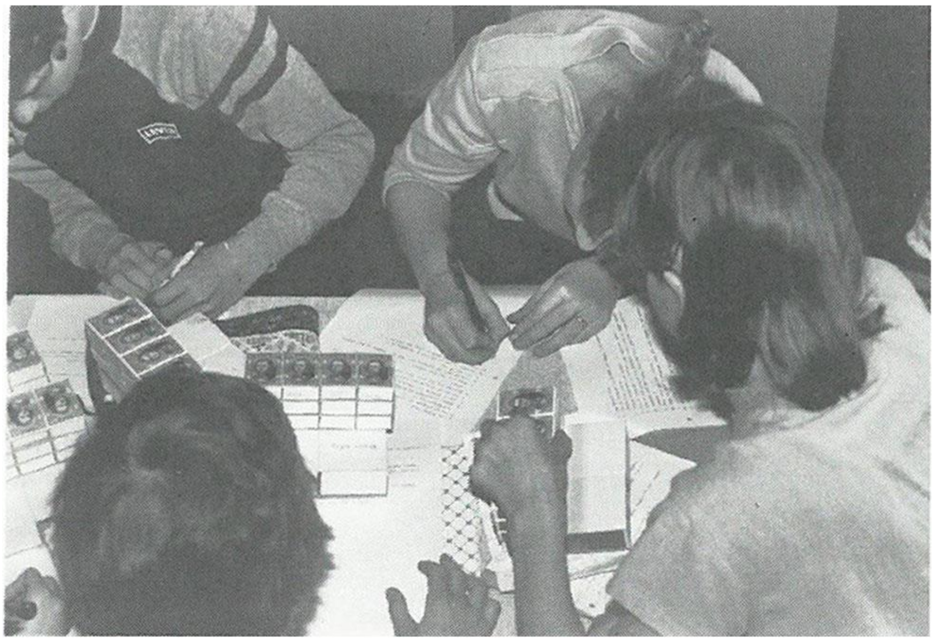


Figure 6. Grade 5 students are building a model of a computer out of matchboxes. Photo: Dansk Skolelederforening (Frandsen 1983)

The teachers explained:

> In principle, the model was built as a computer; i.e. it consisted of an input unit, a control unit, a calculation unit, a unit for internal storage and a readout unit. The storage unit consisted of 12 small boxes, while the other units consisted of large boxes. After the students had assembled a model, they reviewed several examples of instructions for use (programs). Through this, the students got to know the different units on a computer by something tangible and concrete. (Frandsen 1983, my translation)

The students worked with automatic control, for example of a crane or a traffic light, and registers, for example statistics of book lending at the library.

The purpose of teaching the students to understand computers and how they worked was:

> to help demystify and dedramatize the use of microprocessors and computers. Dedramatization should not be used to eliminate critical attitudes to the use of computers and new technology. On the contrary, the teaching should help replace mysteries, myths, misunderstanding, and rumors with knowledge, understanding, skills, experience and attitudes. (Frandsen 1983, my translation)

Our main point was to illustrate how historical ways of learning about computer science might help today's students understand the development up until today, and make it easier to understand how computers work today.

Sixty years ago at NASA, humans were calculating and analyzing the trajectory of space flights by hand with the use of calculating machines, and these equations were later programmed into a computer. Such people were referred to as 'human computers'. This is another example that illustrates how a computer runs programs that are designed by human brains. When NASA switched from 'human computers' to 'automatic computers', astronauts were in fact "wary of putting their lives in the care of the electronic calculating machines". For example, the American astronaut John Glenn wanted a human "to run the same numbers through the same equations that had been programmed into the computer, but by hand, on her desktop mechanical calculating machine" before he wanted to go[15].

Historical ways and approaches to teaching are not sufficient today, though. For example, today there is a need to understand how machine-learning algorithms work and not only step-by-step algorithms. I will elaborate on that in Chapter 4.3 and Chapter 5.1.

4.3 From Historical to Present Perspectives

This study was completed in collaboration with Bundsgaard and was published in an article, which is attached to this dissertation (Article B). We examined computational thinking and technology comprehension in a historical perspective from the 1960s and up until today with a focus on Danish educational developments in the field. We presented discussions and initiatives chronologically and described the development as being a *round-trip*, because foresighted discussions and experiments from around 1960 to 1980 looked very similar to the ones we see in Denmark today. For example, the previously mentioned subject data education focused on computational problem solving and included understanding digital technology in a societal perspective. Therefore, we believed that experiences and lessons learned could serve as inspiration today with regard to *why* we need to teach *what* aspects of computational thinking in compulsory education, and *how*, in order to prepare

[15] This story is told on NASA's website: https://www.nasa.gov/content/katherine-johnson-biography.

students to live in today's society and the society we will be living in in the near future. Hence, we explored how the subject had developed, declined, and now re-developed over time and up until today in order to derive inspirational perspectives.

As was the case for the previous study, we based this study on Naur's theories. In previous chapters, I have discussed how his perspectives can serve as inspiration today. In continuation of this, the purpose of this study was to present the development up until today and discuss how our historical perspectives are only stepping stones. The world looks very different today than it did in the 1960s, 1970s, and 1980s, and therefore we need to add to history regarding *why* students need to learn *what*, and *how*.

4.3.1 Method

Methodologically, the study was based on an explorative literature search. Specifically, we searched for literature that included concepts such as *"datalogi"* (datalogy) and *"datalære"* (data education), we searched for literature and key people who have been mentioned in the public debate, participated in legislative work, and designed teaching resources, and we talked to stakeholders from that time.

Our empirical material is limited to initiatives within, and discussions about, computer science as well as digital technology in K-9 schools. As stated in the article, it includes:

> statutory provisions in the form of factual presentations of educational policy initiatives in the area, including official subject descriptions and decision-making processes; academic theory in the form of descriptions and analyses of what researchers within computer science and education have, over time, emphasized as important in a general education school; and events in practice and in the societal debate in the form of descriptions and analyses of how society and schools in practice have acted within the individual periods. (Article B)

We analyzed historical events, and identified four periods from 1966 until the present. We prepared a timeline (see Appendix 1 of Article B) to illustrate trends, central initiatives and discussions to thereby be able to compare the key initiatives and debates in the various periods. And, finally, we discussed how the periods have affected the development in the field.

Additionally, we had conversations with Denning about the development to elaborate on his theories about what he has entitled *new computational thinking* – in contrast to historical computational thinking. Here, I bring his perspectives up front to discuss how the world looks different today, and what this means for education.

4.3.2 Analysis of Why and What

Our analysis of the development of computational thinking and technology comprehension in Danish compulsory education, illustrates four periods from the 1960s and up until today.

Period one (1966-1990) was focused on data education, based on Naur's ideas about teaching datalogy in K-9 schools. The purpose basically was to enable students to think critically about the role of computers in society and to think computationally.

Period two (1990-2000) was focused on operational user competences and infrastructure. Focus shifted to enabling students to use the machines, for example how to turn it on and off, how to save documents, and how to use a word processor. In Denmark, computers were on the verge of being integrated into all subjects. Therefore, focus was also on getting a large number of devices and connecting them to the burgeoning internet.

Period three (2000-2016) was focused on procurement of hardware and development of teaching resources. Schools bought not only traditional computers, but also interactive whiteboards and tablets, and later on also robots, 3D printers, etc. There was also focus on developing teaching resources and integrating computers in daily teaching. This generated a big market for digital learning resources.

Period four (2016-present) is focused on computational thinking and, in a broader perspective, on what in Denmark is called technology comprehension. In Denmark, this period began with initiatives such as Coding Pirates and FabLab@SCHOOLdk and debates about IT didactics. Worldwide, this period began with Wing who re-introduced computational thinking as an important competence in line with reading, writing and arithmetic similar to what Peter Naur did in the 1966. Wing's and Naur's perspectives are different, though. Naur emphasized that all children should learn to understand and use the machines, whereas Wing proposed they learned to think like computer scientists, highlighting programming concepts such as computational steps and algorithms.

In this regard, Denning argues that emphasis on programming largely excludes other areas within computer science. Though, he is positive about the efforts to make computational thinking more accessible, he warns that a lack of insight into the long and comprehensive history of computational thinking causes a poorer and narrower version of it than has been the case historically.

First, he points out that the concept of algorithms in the new computational thinking movement is based on outdated notions about algorithms as step-by-step procedures, which is not true today due to the massive use of, for example, machine learning in society. Also, as I have discussed in the above, the new movement presents algorithms as any step-by-step instruction (for example, instructions on tying a shoelace, baking a cake or brushing one's teeth). He foresees "a risk that students will believe that computers can do more than they actually can, and that we will fail to distinguish between what people can do that computers cannot. [...] This ignores the essential requirement that the algorithm has to be accurate enough for a machine to execute it without human assessment or interpretation" (Article B). Therefore, Denning argues that computers are important in the formulation of algorithms, since algorithms are designed to direct computers. Computational thinking "involves mental habits and methods to find out *how to get computers to do a job for us*" (Article B, my italics).

Another issue that arises from new computational thinking being focused on algorithms is its focus on formal evidence when deciding whether a program works or not. Denning says that evidence is useful when it is possible to obtain, but that a system depends on other methods for reliability. "Much of computing is not about programming but design of computations, and much of design draws from engineering rather than mathematics". In addition: "Users – not programmers – decide whether a solution successfully performs a job and is useful. Therefore, students should learn to listen to users and incorporate what users say in their design" (Article B).

Second, he argues that computer scientists not only think about programming and that this emphasis on programming excludes other areas such as "artificial intelligence, data analysis, neural networks, quantum mechanics and so on, all of which depend on computational thinking with hardware, computer systems, networks, simulation and design" (Article B).

This discussion encouraged me to further analyze *why* computational thinking is essential to teach in compulsory education, and *what* aspects of it are essential to teach in our present and future society.

4.4 Sub-Conclusion of Historical Perspectives

In this chapter, I have examined *whys*, *whats* and *hows* of computational thinking in a historical perspective. The purpose with looking at history was, motivated by Denning, to learn what researchers and educators have already discussed in the past and what conceptions of the subject they had to see whether their perspectives could possibly have a positive impact on today's discussions and conceptions.

With regard to *why*, I found that 50-60 years ago, Peter Naur argued that everybody should learn to understand data, their nature and use in a democratic society. His reasoning was that datalogy, similar to language learning and mathematics, is an important aid for a number of general activities that are relevant to our general life and to all disciplines. He stated that people would need to understand datalogy in order to be able to influence decisions in our society, and therefore – illustrated by examples of how datalogy affects the lives of all individuals from his own time and the future as he anticipated it – he said that this understanding should be taught in compulsory education.

In continuation of Naur's thoughts, a committee established by the Ministry of Education described that the purpose of the subject data education in compulsory education should be to give insight into fundamental, interdisciplinary datalogical issues and concepts, to communicate knowledge about possibilities and limitations of computers, and to inform about applied data processing and societal advantages and disadvantages associated with extensive use of automatic data processing.

With regard to *what*, I found that datalogy – data, their nature and their use – was at the heart of a subject, according to Naur. He believed that a subject should focus on the elementary parts of datalogy, such as the concept of data, representations and formal data processes. He did not consider the computer to be the most central part of a subject, but he considered it a good idea to illustrate and do exercises with concrete equipment to realize data processes. For example, make a computer program perform certain actions in the form of algorithms.

Whereas Naur formulated his thoughts in more general terms, yet in detail, a committee set up by the Danish Ministry of Education described more specifically what students should learn. They suggested the following overall topics: the concept of data; problem formulation and task structuring; the concept of models and model types; the concept of algorithms; the basic structure of a computer; and data processing systems, data processing applications, and societal aspects.

In the previous review of present conceptions (state-of-the-art), the analysis showed that today computational thinking is mainly regarded as concepts related to mathematics, which is in contrast to – or is more limited than – these broader historical perspectives on what students should learn. Even though the historical perspectives were broader than the perspectives of today on what students should learn about such concepts, they are, nevertheless, still not sufficient today. In my studies, I saw that, historically, contemporary data model types that used, for example machine learning were – for obvious reasons – not included, which is essential today.

With regard to *how*, I found that a substantial part of historical *how*-examples in the classroom were unplugged. I argue that such unplugged ways of learning computational thinking skills could also help today's students develop a deeper understanding of how machines work and that what happens in them is not magic; rather that computers run programs that are designed by human brains. For example, by working with the concept of data, by using flowcharts to describe primitive algorithms, or by constructing a computer, students could gain a better idea of what they are dealing with. Therefore, I argue that unplugged ways can complement a present subject and push it beyond merely focusing on a programming environment.

In addition, using a triangular model of people, problems, and tools, Naur illustrated how the three elements interact, and that focusing on, for example, programming as a tool, could limit our creativity. Therefore, rather than just learning specific tools plugged, students can benefit from unplugged ways of learning to develop their thinking in creative ways that are detached from specific tools.

The analysis showed that working unplugged is not sufficient, though. As was also discussed in the past, there is a need to implement algorithms in a machine, for example to test whether the algorithm is precise enough for a computer program to follow the instruction, and to understand what computers can and cannot do. This was a challenge 50-60 years ago when computer programs – and computers in general – were much more limited than they are today. Today, there are greater opportunities to work with computers as well.

The majority of my historical studies centers on Naur's subject-specific educational theories. These theories can be seen in continuation of Klafki's general educational theories in that they both emphasize societal and democratic reasons for subjects to be included in compulsory education in a general Bildung perspective and not for specific career purposes. Moreover, both Klafki and Naur emphasize project activity in terms of developing competences in an application-oriented context instead of acquiring knowledge or learning concepts out of their context. This is one of the reasons why Naur's subject-specific didactic theories are well-suited to and valuable in a critical-constructive Bildung perspective.

Naur expected that the changes required for a subject in datalogy to be included in compulsory education would take decades. Even though pioneers prepared the foundation for the subject data education, and described possible solutions to challenges in detail, politicians and decision-makers in Denmark decided that they would not implement the new subject. Today, though, there is a renewed interest in a similar subject – technology comprehension – that aims for everyone to understand digital technologies. Data is an inevitable topic, and in this regard, Naur's thinking is valuable and inspirational.

Though I argue that we can learn from history and should build on it instead of reinventing the wheel, I do, however, argue as well that we need to improve that wheel. Society has gone through massive changes concerning digital technologies and the use of data. Therefore, with reference to my didactic position and Schnack's arguments on the importance of determining current problems in society when embedding subjects and their didactics, historical perspectives are not enough for students today to learn to understand computers. For example, students need to learn to understand machine learning as well so they can participate in – that is, influence as well as navigate – today's society.

Moreover, when Naur discussed how datalogy affects our inner way of thinking, he also said that he did not expect that datalogy would come to affect the external form of our daily lives in the same way that, for example, cars or television had done (Naur 1967). In my view, this expectation has proven to be wrong. Though datalogy is a way of thinking and dealing with problems, it has also affected the way physical products are designed today (for example things that are referred to as 'smart': apps in smartphones, programs in smartwatches, etc.). And even though datalogy as a tool for thinking about problem solving has made such products possible, I would argue that such products highly affect the external form of our daily life today. The next chapter focuses on present time with a view to a possible future.

5 Present Perspectives

This chapter is centered on present time. With reference to my didactic position, described in Chapter 2.3, I elaborate further on my studies regarding *why* and *what* and their consequences for *how* from a critical-constructive Bildung-centered position. Central parts of these studies discuss conceptions of computational thinking that are broader than the conceptions held by the new movement, in that they include an awareness of the history of computational thinking as well as the principles of computing that resemble the highly-digitalized world that children are part of today. Even though history can inspire our thinking today, the world has changed since the 1960s. To answer my research questions, it is therefore essential to analyze *why what* is important to learn and *how* in the present and a future society in order to be able to deal with the challenges that we face as a society and as humans.

In Chapter 5.1, I present the results of a study I co-conducted with Bundsgaard. We examined present societal perspectives on *why* and *what* in terms of how technologies influence our democracy, how digital technologies influence our personal and social life, and how digital technologies influence our work life.

In Chapter 5.2, I present and discuss a survey study from 2018 that aimed to examine *what* computational thinking is according to Danish school principals, and *why/* if it should be taught in compulsory education. The study contributes to my overall project by demonstrating present discussions in the field from a German-Scandinavian Bildung perspective, exemplified by Denmark that differs from the Anglo-Saxon perspectives that seem to be the most common in literature on this topic[16].

Next, in Chapter 5.3 I present a design experiment performed in collaboration with PhD Martin Dybdal. The experiment is based on the didactic position as well as theories on *why* and *what* that are described in this dissertation, and from that perspective, it illustrates a possible *how* in the classroom.

5.1 Societal and Democratic Perspectives Today

In 1968, Peter Naur argued that it was absolutely untenable for an area that affects key decisions to give up and leave everything up to the 'experts'. Although he discussed data use and data processing in the future, he referred to examples from the society of his time. For example, back then the internet was not even invented. In this study, we examined what consequences digital

[16] Five of the six articles in the review by Palts and Pedaste are American (see Chapter 3.1)

technologies have for society today and could have in the future, in order to discuss what kind of computational thinking skills students need today. The analysis and results of our study are attached to this dissertation as Article C.

5.1.1 Method

Our analysis was based on a societal-/ideological-critical position with the aim to "change (improve) the world, human awareness, and future possibilities by students developing responsibility and competences to act on big problems in society" as discussed in Chapter 2.3.5.

As argued in the same chapter, moral is important in this didactic position: "education and Bildung are committed to dealing with societal challenges". Thus, we need to examine societal challenges to determine *why* we need to teach *what* and, next, *how*. This is what we have done in this study that takes the form of didactic theoretical reflections and use empirical examples.

5.1.2 Analysis of Why and What

Early pioneers such as Naur argued 50-60 years ago that everybody should understand computer programming and be able to influence the development of society. Even though arguments on teaching datalogy in compulsory education are basically still the same, society has developed. Therefore, we need to examine and develop our thoughts on what is needed today for students to understand computer programming and be able to influence the development of society.

In our article, we reflect on how humans have "achieved enormous progress for individuals and society as a whole, but we have also seen an abundance of examples of how secret algorithms increasingly mislead us rather than guide us" (Article C). Exploring the use of digital technologies in society today, we see that many computer programs are designed as neural networks built up of parallel algorithms. Therefore, it is not enough to let history guide us on what to teach in compulsory education, and it is not enough to teach computational thinking concepts such as the ones described in my analysis of what computational thinking is regarded as in literature today (Chapter 3.1), for example algorithms as-step-by-step procedures.

In this societal analysis, we argued for a need to add *technology criticism* to *computer science* and *design* (see Figure 7). The name of the subject in the figure, *technology comprehension*, emphasizes the need to be able to comprehend (digital) technologies. Therefore, in a K-9 context, I find this term more suitable than, for example, computing, computer science, or (only) computational thinking in the new common conception that emphasizes computational steps and concepts from mathematics.

Technology comprehension as a subject in K-9 schools		
Computer science	Design	Technology criticism
↕	↕	↕
Subject-specific didactics	Subject-specific didactics	Subject-specific didactics

Figure 7. Three aspects of technology comprehension as a subject in K-9 schools: What should schools teach, why, and how?

In our analysis, we explain that:

> In general, computer science can be said to refer to a basic understanding of what data are, their characteristics and use; design refers to the implementation of design processes and development of digital technologies that solve relevant issues; and technology criticism refers to an understanding of the consequences (risks and possibilities) of digital technologies. (Article C)

The fact that many algorithms today are built as neural networks that 'learn' new behaviors through continuous input has indeed made it much more difficult to understand consequences, risks, and possibilities of using and designing digital technologies. Remembering what Danish compulsory schools in a critical-constructive Bildung perspective are about – preparing students to be able to participate, demonstrate mutual responsibility, and understand rights and duties in a free and democratic society – there is a need to develop skills within all three aspects of technology comprehension and not only focus on, for example, technical or mathematical aspects.

In 1968, Naur argued that if "this broad understanding of programming is not effectuated, expert programmers will gain a power position that can lead to the end of democracy" (Naur 1968). Thus, we need to implement a subject – today named technology comprehension – in the school curriculum. In our article, we elaborated on what the students need to learn in such a subject.

First, we presented a number of present and future moral dilemmas with far-reaching consequences to illustrate why technology comprehension – including technology criticism – must be integrated as a subject in compulsory education. For example, when government institutions investigate crime based on historical data on who, when and where, they 'repeat history' so to speak:

> If the police exclusively or predominantly patrol the areas with previous cases of crime, they will exclusively or predominantly arrest individuals in these areas, and this in turn will intensify the algorithm. (Article C)

If historical information about our behavior is registered and processed to assess, for example, how much a customer should pay for a flight ticket or health insurance, this affects equality and our rights in a democracy. In addition, who should decide what constitutes desirable behavior?

There are also problematic issues related to our democratic rights to speak freely when algorithms are designed to show different people different news or to decide what content is appropriate (and thus visible) to others. More examples and more detailed discussions are included in the article.

Thereby, we argued that students must learn to comprehend technology in terms of data, their characteristics and use, implementation of design processes and development of digital technologies that solve relevant issues, and understanding of the consequences (computer science, design and technology criticism). More specifically, we argued that they "must develop an understanding of how systems work, and they must be able to discuss in a broader perspective the moral dilemmas and consequences of the choices a designer makes when designing an algorithm".

5.1.3 Analysis of How

From an organizational perspective, we discussed issues related to *how* as well. We argued that it is more likely that students can acquire basic knowledge and skills if they are taught an independent subject by teachers with specialist knowledge and education.

We believe that teachers with another academic focus cannot be expected to take on this responsibility; however, all teachers should consider the aims of their teaching subjects and how technology comprehension as a subject area is useful or even necessary to achieve these aims. For example, in Denmark in *history* as a subject a purpose is that students develop competencies to "relate changes in everyday life and living conditions over time to their own lives" (Undervisningsministeriet 2019, our translation). In our present society, this includes being able to comprehend how digital technologies have changed everyday life and living conditions.

5.2 Conceptions in Schools

This study was conducted in collaboration with Bundsgaard and it is included as Article C. In the context of this dissertation, the purpose was to examine *what* computational thinking is according to school principals, and *why/* if we need to teach it in compulsory education.

This study was conducted in a German-Scandinavian context, more specifically Denmark. As stated in the formal aims of Danish compulsory education (see Chapter 2.3.7), compulsory school in

Denmark does not only aim to prepare children and youth for college and a career, but also aims to prepare them "to be able to participate, demonstrate mutual responsibility and understand their rights and duties in a free and democratic society" (Ministry of Children and Education 2018). This background is important to know since it is reflected in the answers given by the Danish school principals.

5.2.1 Method

The study was designed as a survey, more specifically a digital questionnaire (see Appendix 1 in Article D). The questionnaire was sent to school principals of Danish K-9 schools (compulsory education). It consisted of three parts: Part 1 on *technology initiatives in schools*, Part 2 *on the teachers' professional development*, and Part 3 on *conceptions of computational thinking*. In the context of my research questions concerning *why* computational thinking is essential to teach in compulsory education, and *what* it involves, Part 3 of the questionnaire is relevant to look at.

In the introduction of Part 3, the purpose of computational thinking was specified:

> There is no common definition or understanding of what computational thinking involves. In the following questions, we are interested in the way you understand the concept and your perspective on the importance of teaching K-9 students CT[17]. (Article D)

We asked the following six questions about familiarity, definition, importance, relevance, challenges, and advantages:

- Familiarity: To what extent are you familiar with the term computational thinking?
- Definition: How strongly do you agree or disagree with the following statements about what computational thinking involves?
- Importance: Do you think K-9 school students should be taught computational thinking?
- Relevance: How strongly do you agree or disagree with the following statements about the relevance of teaching computational thinking in K-9?
- ~~Challenges: How strongly do you agree or disagree with the following statements about the potential challenges of implementing computational thinking in K-9?~~

[17] In our article, we used *CT* as an abbreviation for *computational thinking*. However, we also used the term "*datalogisk tænkning"* (*DT, datalogical thinking*), derived from the Danish term "*datalogi"* that emphasizes data and human understanding rather than computations, as is discussed in Chapter 4.1.

* ~~Advantages: How strongly do you agree or disagree with the following statements about the potential advantages of implementing computational thinking in K-9?~~

In this context, responses to the question categories Familiarity and Definitions are relevant with regard to analyzing *what*, and responses to the question categories Importance and Relevance are relevant with regard to analyzing *why*. The question categories Challenges and Advantages have been stroked through to indicate that they have been left out in the analysis in this chapter.

The full questionnaire including all possible statements is attached to Article C. The questionnaire was sent to 145 Danish school principals and was completed by 83 principals. As stated in the article, "the participants who completed the survey included principals from all Danish regions with no significant differences in terms of gender, school size, or achievement scores" (Article D)[18].

5.2.2 Analysis of What

To analyze *what*, I examined answers to the questions regarding Familiarity and Definitions. First of all, to be able to analyze whether the principals' responses were based on knowledge or thoughts, it was important to ask to what degree respondents had heard about and had knowledge about what computational thinking is. Therefore, we asked them a nominal closed-ended question about to what extent they were familiar with the term. They could choose between the following four options:

* I have never heard of it.
* I have heard of it but have limited knowledge.
* I have read/watched videos about it and/or had it explained by colleagues/at conferences.
* I feel familiar with the concept and national/international discussions on it.

As is seen in Figure 8, 73 percent of the principals assessed themselves to have some degree of knowledge about computational thinking, and 27 percent said they had never heard of it.

In the following analysis of definitions (Figure 9) and relevance (Figure 11), I only include responses from principals who reported they have some degree of knowledge.

[18] The full analysis as well as methodological approach is available in Article D.

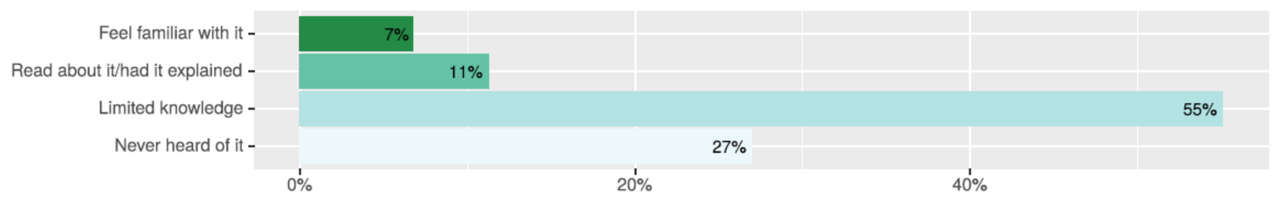


Figure 8. Principals' familiarity with computational thinking

Next, principals were asked about how they define computational thinking. More specifically we asked them a closed-ended Likert scale question to measure how strongly they agreed or disagreed with eight different statements about what computational thinking involves. We designed the statements "to cover the most commonly heard arguments. Specifically, policy documents, research articles and debates regarding the idea of teaching computer science to everyone have informed our statements" (Article D).

The eight statements were

* Computational thinking involves understanding and formulating algorithms.
* Computational thinking involves working with data (collecting, processing, communicating).
* Computational thinking involves knowledge of computers to solve tasks using a computer.
* Computational thinking involves critical thinking regarding the role of digital technologies in our life and society.
* Computational thinking is more or less the same as programming and coding.
* Computational thinking involves solving problems in systematic and logical ways.
* Computational thinking involves using computers to solve problems and perform tasks in easier ways.
* Computational thinking involves a number of general principles of problem-solving with or without the use of computers.

Figure 9 illustrates their answers sorted in an increasing order, with blue and green representing agreement, and orange and red representing disagreement.

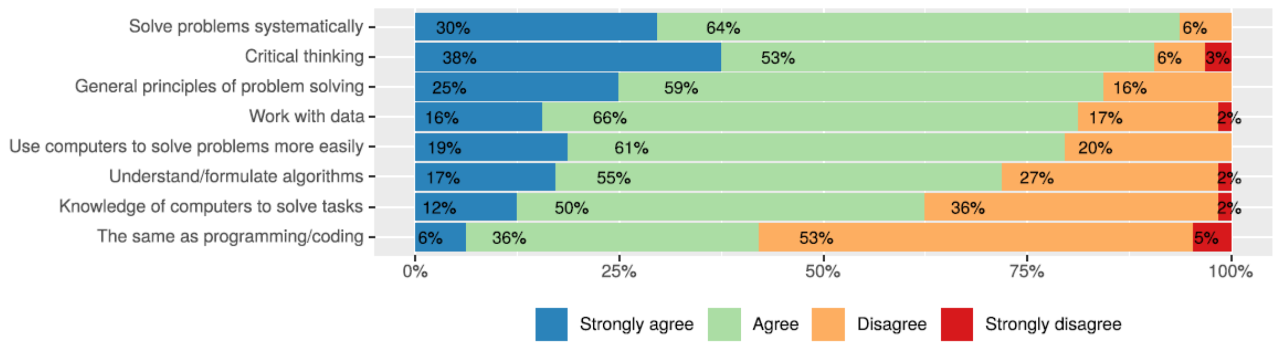


Figure 9. Principals' definition of computational thinking

The answers were in general characterized by a high degree of agreement. Our analysis showed that the principals in our survey to a high extent agreed that computational thinking involves general problem-solving skills. Specifically, 94 percent agreed that computational thinking involves solving problems in systematic and logical ways, and 84 percent thought it involves a number of general principles of problem-solving with or without the use of computers. In addition, as many as 91 percent agreed that it involves critical thinking regarding the role of digital technologies in our lives and in society. They agreed the least that computational thinking more or less involves the same as programming and coding (42 percent), that it involves knowledge of computers to solve tasks using a computer (62 percent), and that it involves understanding and formulating algorithms (72 percent). 80 percent believed it involves using computers to solve problems and perform tasks in easier ways, and 82 percent said it involves working with data.

A general pattern in our data was a high degree of agreement, which suggests that respondents from our survey had broad views on what computational thinking involves. When we further analyzed the categories they agreed the least on, however, we found that "principals agreed the least on computational thinking as being equal to basic technical skills and knowledge (the same as programming/coding and knowledge of computers)" and that "they agreed less on subject-specific definitions (understand/develop algorithms, using computers to solve problems more easily and work with data) than on definitions including more demanding mental skills such as critical thinking and problem-solving approaches (general principles of problem solving, critical considerations and solve problems systematically)" (Article D).

5.2.3 Analysis of Why

To analyze *why*, I examined the responses from the question categories Importance and Relevance. With regard to importance, we asked principals if they thought K-9 school students should be taught computational thinking. In Figure 10, their answers are broken down by principals who assessed they had some degree of knowledge of computational thinking and principals who said they had never heard of it. They had three options to choose from:

* Yes, as a separate subject (and perhaps also integrated in other subjects)
* Yes, integrated in other subjects
* No

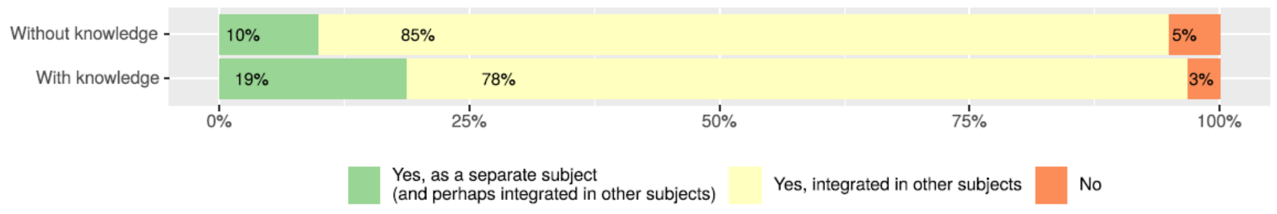


Figure 10. Principals' views on the importance of teaching computational thinking as a subject

As is seen in the figure, the majority of principals regarded computational thinking as important to teach in compulsory education (95 percent and 97 percent). Most of these principals thought it should only be integrated in other subjects and not as a separate subject. The principals who assessed themselves to have some degree of knowledge of computational thinking were, however, more positive towards a separate subject (19 percent compared with 10 percent).

With regard to the relevance of teaching computational thinking, principals were asked how strongly they agreed or disagreed with eight different common statements about the relevance of teaching computational thinking in K-9:

* Computational thinking only prepares students for the labor market, and K-9 should not focus on developing this skill.
* K-9 students should learn to understand principles of data in order to understand and act in an increasingly digitalized everyday life.
* Developing computational thinking is as important as developing reading, writing and math skills.
* Computational thinking should be an optional subject for those interested in the field, and everyone should not be forced to learn computational thinking principles.
* The future job market has a need for more digitally skilled employees.
* Computational thinking is an important part of students' digital Bildung.

* From their experience with digital devices in pre-school, students are already digitally Gebildet[19] when they start school.
* Teachers are important when facilitating technology training (e.g. discussing danger/potentials of technology or setting challenging tasks).

Figure 11 illustrates their responses. Again, the responses are sorted in an increasing order, with blue and green representing agreement, and orange and red representing disagreement.

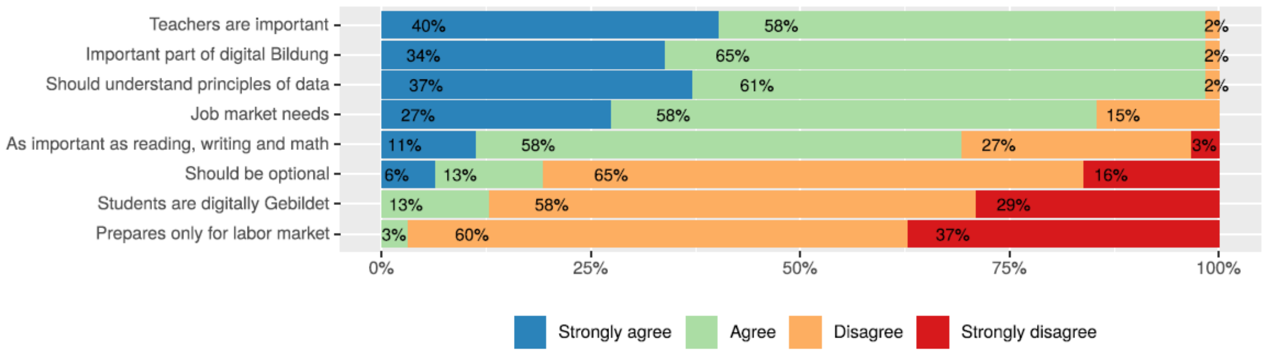


Figure 11. Principals' attitudes towards the relevance of teaching computational thinking

The results show that whereas a small percentage of the principals think that computational thinking is important with regard to preparing students for their future jobs (3 percent), by far the majority of the principals think that it is an important part of students' digital Bildung (98 percent). Children are, in the views of 87 percent of the participating principals, not digitally Gebildet when they start school just because they have already had experience with digital devices in pre-school, and as many as 98 percent find teachers important with regard to facilitating technology training, which indicates that computational thinking is not something they think children learn "by themselves" by using digital technologies. A total of 98 percent think compulsory education should introduce the principles of data in order for children to understand and act in an increasingly digitalized everyday life, 81 percent think that computational thinking should not be optional to learn in school, and 69 percent actually think developing computational thinking is as important as developing reading, writing and math skills.

As was the case with their responses in the category Definitions, responses in the category Relevance suggested that the principals who took part in the survey had broad views on *why* computational thinking is important to teach in compulsory education.

[19] *Gebildet* is the adjective form of *Bildung.*

5.2.4 Subsequent Reflections on Methodological Approach

This study was planned and conducted in January-February 2018 when I had just started my PhD. At that time, the description of the Danish subject technology comprehension and its four competence areas (one being computational thinking) had not yet been published, and computational thinking was not a well-known term. The survey data shows that only seven percent of the participating school principals *felt familiar* with the term. Moreover, my own understanding of the term has developed since 2018, and the statements designed to cover the most common definitions and common arguments are not consistent with how I would have designed them today.

The primary purpose of the questionnaire was to select 'focus' and 'non-focus' schools that I could observe[20]. Therefore I wanted to assess the school principals' conceptions of the term to examine a kind of 'state of the art' in schools as a starting point for my studies. Specifically, I wanted the questions to reflect some of the discussions I found present at that time. Concerning the category Relevance (if/*why*) I wanted to know whether the principals thought of computational thinking as a competence that prepares students for the labor market or as Bildung in a digitalized society; if they thought it involved computers, if they thought that students are already able to think computationally when they begin school and therefore did not consider teachers and education to be important in this regard, rather optional to teach and learn; if they found it important for students to understand the nature of data; and if they saw computational thinking as important as reading, writing and math. Concerning the category Definitions (*what*), I wanted to know whether they saw computational thinking as a machine-focused technical competence, for example as being equal to programming, rather than as a more general problem-solving approach to working with data; and whether they thought it involved critical thinking.

When asking these questions and formulating the statements, my own thoughts on computational thinking were colored by international discussions on computational thinking as a way of thinking that did not necessarily involve a computer. During my studies, though, I have developed my own thoughts on what computational thinking is. I used to think computers were not that necessary in order to become a competent computational thinker. However, based on my analyses, it is today my belief that computational thinking is a way of (human) thinking about getting *computers* to solve tasks (for humans), and that working with computers is required to develop as a competent computational thinker.

[20] See Chapter 2.1 concerning my intended research design.

Today, I would have used my knowledge on present versus historical *whats* and *whys* of computational thinking, and asked the school principals whether they agreed more with one or the other, whether they thought it was essential to teach, and why. I would have asked whether they thought computational thinking could prepare students to live and work in a democracy and whether they additionally found design thinking and critical thinking skills essential, and why. Thereby, the statements would have been consistent with my understanding today, and I would have a better foundation for determining whether the understanding of Danish school principals reflects the German-Scandinavian Bildung traditions of schools and their purpose.

Furthermore, it would be interesting to ask American school principals the same questions to see whether their answers differ and reflect the Anglo-Saxon traditions more. As Gundem and Hopmann argue[21], comparative research and exchange of traditions are important when collaborating and discussing central concepts and their meanings.

In the following, I present a design experiment on teaching computer science today; that is, I illustrate a possible *how* inside the classroom.

5.3 A Computational Design Experiment

Until now, the majority of my studies has been concentrated on theoretical didactic analysis and visionary perspectives with regard to *why* we need to teach computational thinking in compulsory education today, and *what* aspects of it are important to teach. Didactics, however, refers to both theory and practice planning. As argued by Nielsen (see 2.3.3), theorizing is insufficient.

This experiment contribute to the didactical question *how*, and is scientifically based on my theoretical studies concerning the questions *why* and *what*. In that sense, the didactical theories from my studies have enriched practice, and I anticipated that practice could possibly enrich the theories as well through practical experiences. Therefore, I conducted a design experiment in collaboration with PhD Martin Dybdal. A description as well as the analysis and results of the experiment are attached to this dissertation as Article E.

The experiment is a didactic contribution to the subject area technology comprehension. In this context, we regard technology comprehension as a discipline that involves developing competences in the fields of computational thinking, design thinking, and critical thinking concerning the use of computer science in society. With that, we refer back to the analysis presented

[21] See Chapter 2.3.1: German-Scandinavian Notions of Didactics.

in Chapter 5.1 on the three aspects of technology comprehension: computer science, design and technology criticism.

5.3.1 Method

While Dybdal was responsible for teaching, I was responsible for observing the teaching, for writing field notes and for having conversations with the students. In addition, a female computer-science student participated as an assistant teacher. We conducted our experiment in a Danish eighth-grade class with the purpose to examine a way of teaching computational thinking and design thinking by developing a *computational design* to solve an authentic problem.

Methodologically, we based our experiment on design-based research to explore our theories in practice and to analyze and discuss improvements based on our experiments. We did not conduct our experiments with iterations, though, as is common in design-based research, but we present our course and analysis as a contribution to subject-specific didactic theory. In addition, by presenting our course in-depth, it is possible for others to test and improve it.

5.3.2 Analysis of Why and What

As stated, this is an example of *how* based on my conclusions regarding *why* and *what* that have been presented in previous chapters of this dissertation. In short, I have argued for the need (*why*) to teach technology comprehension in K-9, and that such teaching, at an abstract level, should involve competencies within computer science, design, and technology criticism (*what*). In this presentation of our experiment, I therefore concentrate on *how*. This *how* is based on Klafki's critical-constructive understanding of Bildung as also being a societal issue. Specifically, it is based on an environmental key problem in modern-day lives: how to reduce CO_2 emissions by reducing electricity consumption.

5.3.3 Analysis of How

There is not one single best practice – not one single *how* – but many good practices. Specifically, in this experiment we mainly concentrated on computational thinking and design thinking, and only secondarily on technology criticism.

The students were to work with a problem-oriented authentic project that we had selected: They were to develop a prototype of a physical, programmed design that they thought could help people reduce their electricity consumption and thereby CO_2 emissions. The students were to retrieve data about Danish real-time CO_2 emissions from the data provider electricitymap.org, and were to program LED lights to show (for example, with specific colors and/or patterns) when electricity consumption in Denmark is high and low, respectively; that is when it is most and least harmful to the environment. They were to integrate their programmed LED lights in a physical design that they developed themselves based on reflections on what it should look like and where it would be best placed for its users to see it:

> If, for example, consumers can see that CO_2 emissions are high at a given time of day, they may be encouraged to delay turning on their dishwasher or charging their mobile phone until emissions are lower, because this would entail that the energy being used is greener. (Article E)

More specifically, we based the course on Naur's theories of *datalogy* (see Chapter 4.1) as an interplay between humans and computers. When planning, conducting, and assessing the course, we used Naur's people-problem-tools model (Figure 3 and Figure 5). We "explained to the students that they were to design solutions for people for a current problem in the world, and that they needed some tools to solve the problem" (Article E).

Moreover, to make the process visible for the students in terms of what was going to happen in the course of the next three days, we introduced a design model (see Figure 12).

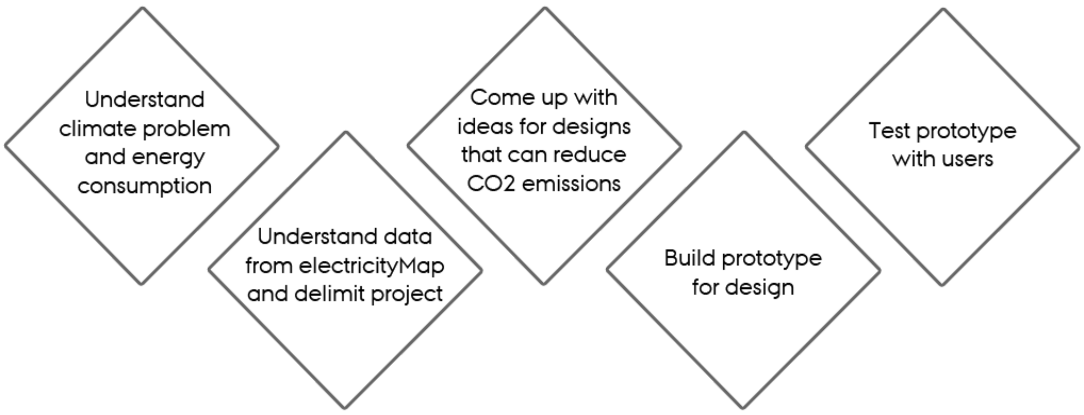

Understand climate problem and energy consumption
Understand data from electricityMap and delimit project
Come up with ideas for designs that can reduce CO2 emissions
Build prototype for design
Test prototype with users

Figure 12. Our design model[22] *with the experiment's specific content area: "the climate problem and energy consumption"*

Our course is described in detail in the attached article, and our teaching plan, teaching slides, and copy sheets are freely available. Therefore, I will not repeat the course in detail here, but just briefly describe the five phases to illustrate how we designed it based on the didactic and subject-specific theories and visions presented in previous chapters.

* *Understand climate problem and energy consumption*: introducing our design model, open dialogue about climate change, learning about power consumption and power production.

[22] The model is inspired by *Værktøjskassen* (*The Toolbox*) (Katapult/TEACH 2013) and a design model from the School of Design, Stanford University.

* *Understand data from electricityMap and delimit project*: introducing electricityMap.org, introducing Naur's people-problem-tools model (Figure 3) with examples of why well-known tools are designed the way they are (what problems do they solve for people?), introducing programming of LEDs, pair programming of LEDs.
* *Come up with ideas for designs that can reduce CO_2 emissions*: discussing good and bad designs, planning own designs.
* *Build prototype for design*: build prototype for design (alternating between programming the LEDs and building).
* *Test prototype with users*: peer feedback ('user feedback').

The last phase led the students to improve their designs (which meant moving two steps back in the model to planning and then building) with the intention that they experienced the iterative nature of working as a 'computational designer'.

Computational design is a term used by Denning who argues that:

> Clearly, designers are a subset of thinkers because you need to be a computational thinker to design computational tools; and not every thinker is a designer. Also, designers are tool users, but not all tool users are designers or thinkers. (Denning 2017a)

We found that this term suited our experiment well in that the students constantly alternated between developing/using *computational* competencies (programming of the LEDs) and *design* competencies (building their prototype, testing it with users).

As seen in Figure 7 with the three aspect of technology comprehension (computer science, design, and technology criticism), the students worked directly with computer science (understood mainly as a basic understanding of what data are, their characteristics and use) and design (understood mainly as the implementation of design processes and development of digital technologies that solve relevant issues). We did not articulate technology criticism (understood as mainly understanding of the consequences (risks and possibilities) of digital technologies). However, when discussing the people-problem-tools model as well as what a good design is, we did talk about the importance of designs as being relevant to people, and we did discuss the possibility of the digital technology they were designing to help reduce CO_2 emissions.

A shortcoming of this project was that students only learned about sequential algorithms; that is, they did not learn about machine learning which I previously described as being important in our present society. It would be interesting to progress with learning about machine learning models – for example computational designs that use neural networks to counteract climate change.

5.4 Sub-Conclusion of Present Perspectives

In this chapter, I examined *whys* and *whats* and *hows* of computational thinking in a present perspective. Even though I have argued that historical perspectives are important to know and take into account when looking at present time, I also argue that historical perspectives are not enough. The world has changed in many ways since the 1960s – especially in the field of digital technologies and use of data. When discussing a subject today, it is, from the didactic position that I have built my work on, necessary to examine what the world that children today live in, and will live in in the future, looks like. Therefore, I have examined *why what* is important to learn in the society of today and of the future in order to be able to tackle the challenges that we face as a society and as humans, as well as *how*.

With regard to *why*, my analysis showed that Naur's arguments in favor of establishing a specific subject are still relevant: All people need to develop an understanding of computers and programming to be able to influence decisions in society. Contrary to the past, our present society is now to a high extent dominated by those who understand how computer systems work, for example by so-called tech-giants such as Google and Facebook that implement models that, based on data about us, determine what we as users of their systems can see on our screens. And smaller companies use services that collect and process our data through machine-learning models to predict and affect how we act. Such models threaten our democracy and freedom. Thereby, in a sense we are already too late with a subject – even though had we paid heed to the foresighted theories of pioneers, we could have acted in time. This does not mean that we should give up now and abandon implementing a subject; however, our route to taking back the power is longer than it could have been.

I have included a survey that illustrates Danish conceptions of computational thinking. It shows that Danish school principals emphasize and prioritize computational thinking as being important for Bildung for life, for example "in order for children to understand and act in an increasingly digitalized everyday life" (see Chapter 5.2) rather than just educating them for the labor market. Their arguments illustrate the critical-constructive Bildung traditions that the school systems of Danish and other Scandinavian countries are based on.

With regard to *what*, I found that today, we need to add to historical theories on what to learn – at least when we look at specific content. Historical examples and specific *whats* are only stepping stones towards teaching computational thinking today. For example, historically a committee set up by the Danish Ministry of Education argued that students should learn about data models in our society – which is still relevant – but many of today's data models are quite different than they were back then. For example, learning to understand traditional sequential algorithms, executed step by step, is not enough today. Many systems are designed to 'learn' continuously by new input, new data. Our input as users – our data – often determines the behavior of a system. Therefore, students also need to learn about, for example, machine learning in order to be able to comprehend the potentials and consequences of digital technologies today, and the way this technology is used.

Not only do students need to develop computational thinking skills. The analysis from the state-of-the-art showed how present conceptions of computational thinking tend to refer back to Wing as the main driver. She characterizes computational thinking in terms of abstraction, problem decomposition, problem reformulation, automation, and systematic testing; that is, she emphasizes mathematical concepts. Based on the present analysis from a critical-constructive Bildung perspective, this *what* is too narrow. Students need to learn both computer science aspects, design aspects, and aspects regarding technology criticism.

A broader view on *what* is also illustrated in the conceptions of Danish school principals in the abovementioned survey. They think of computational thinking in more comprehensive ways than learning specific concepts and programming. Generally, they had a high degree of agreement with the statements presented to them which suggests inclusive views on what computational thinking involves. They agreed the most that it involves problem-solving and critical thinking, and they agreed least that it involves knowledge of computers and that it means the same as programming or coding.

With regard to *how*, through a design experiment, I found that the general and subject-specific didactic theories from *why* and *what* were able to serve as a good fundament for *how.* Klafki emphasized that Bildung encompasses learning and being confronted with things that concern us as humans collectively, for example common key problems. In the course described, students were confronted with the environment, focusing on climate change as a key problem (problem) that concerns all humans (people). They were to learn a way of coming up with solutions by developing computational thinking and design thinking competencies (tools). The course was planned, conducted and assessed based on Naur's model on people, problem, and tools. The aim of using his model was for the students to consciously solve a problem that had relevance for people, to understand and think about the tools as appropriate things for solving problems, and to use the tools to solve problems that they actually regarded as problems.

As argued, technology comprehension encompasses both computer science aspects, design aspects, and technology criticism aspects, and there are many other ways, more comprehensive ways and additional ways, for students to learn competencies within these areas. The design experiment was an example of one way, and a first step, of acting like a computational designer (inspired by Denning's term computational design). Thus, it was a possible step towards educating students to be able to participate and demonstrate responsibility in terms of being able to design solutions to a key problem in the world.

6 Conclusion

The primary purpose of this project was to examine *why* computational thinking is essential to teach in compulsory education, *what* aspects of it are essential to teach, and, secondary, to examine *how* computational thinking can be implemented in the classroom.

Theoretically, the studies were based on critical-constructive Bildung-centered didactic theories as well as subject-specific theories and perspectives by Naur and Denning. As the subtitle of this dissertation states, I have taken a societal and democratic perspective on computational thinking in compulsory education. By that I mean that I have examined the field in relation to what all people need to learn in a democratic society to be prepared and empowered to live a free life with rights and duties – contrary to what they need to learn to be prepared for college and specific careers.

6.1 *Why* is computational thinking essential to teach in compulsory education, and *what* aspects of it are essential to teach?

An implicit question here is *if* computational thinking is essential to teach in compulsory education. By reviewing common present conceptions of computational thinking, I found that computational thinking is today regarded as a number of concepts that are mostly related to math. In a review of existing literature, the authors found that main articles within the field refer to Wing as the main driver of the present computational thinking movement. In her understanding, computational thinking involves abstraction, problem decomposition, problem reformulation, automation, and systematic testing. They conclude that in general, the articles perceive computational thinking as something that involves solving algorithmic problems. When I in the articles found arguments on *why* the authors of these articles saw it as essential to teach computational thinking in compulsory education, they were only weak and superficial. For example, arguments were that students will live in a world and work in fields that are influenced by computing, and that students will need to take advantage of the changes caused by technology, but they do not elaborate or present any analysis of why.

This does not mean that computational thinking should not be taught; rather that it involves much more than mathematical concepts. In this dissertation, I have discussed how simply learning such concepts from a critical-constructive Bildung perspective is not desirable. Already in the 1960s, Naur argued that calculations were no longer the most important task for computers. Naur specifically articulated *datalogy* as important in order to maintain a democracy with human rights and possibilities, and he emphasized that experts in the field would gain power over all of us if we were not educated to understand how computers were programmed. He stated that it would require an understanding of computer science to be able to influence decision-making processes, and that we

– in a democracy – should not give up and let experts decide such important processes. Hence, he stressed that data education needed to be implemented in the curriculum of compulsory school.

Arguments on *why* are today still similar to the historical ones. The analysis of present perspectives does, however, illustrate that implementing a subject is much more urgent today. For example, experts have now to a great extent gained the power over the system, which already now undermines and threatens our democracy. This might be one of the major consequences of not having implemented a subject in general education aimed at teaching all students to understand digital technologies. Decision-makers ignored historical recommendations and warnings about the power of 'tech-giants', threats to democracy and interconnected personal data that can affect our lives in terrible ways.

A critical-constructive Bildung-centered perspective is illustrated in a survey in which Danish school principals were asked if and *why* they saw computational thinking as important to teach in compulsory education. The principals to a greater extent emphasized computational thinking skills as being important for Bildung for life; that is, that children develop competencies to understand and act in everyday life, rather than education (only) preparing children for the labor market. Concerning *what* computational thinking involved in their view, the principals had more inclusive views. For example, they agreed more that computational thinking involves critical thinking and problem-solving approaches rather than agreeing on subject-specific definitions such as understanding and developing algorithms, using computers to solve problems more easily and to work with data. In addition, they agreed the least that computational thinking is equal to basic technical skills and knowledge.

The historical analysis illustrated that *what*, in a Danish context, has been centered on learning to understand how computers work(ed). Emphasis was on practical and application-oriented problem-solving processes, and that students in compulsory education should not be educated for the labor market, rather for life in a democracy. Specifically, the recommendation from a committee that discussed and described a subject in the 1970s was that the subject area comprise the concept of data; problem formulation and task structuring; the concept of models and model types; the concept of algorithms; the basic structure of a computer; and data processing systems, data processing applications, and societal aspects.

Concerning *what* to teach from a present perspective, there is a need to combine computer science aspects, design aspects and technology criticism in order "to prepare the students to be able to participate, demonstrate mutual responsibility and understand their rights and duties in a free and democratic society", as is stated in the formal aims of Danish compulsory education (Ministry of

Children and Education 2018). Therefore, I regard the present understanding from literature to be insufficient. The computational thinking movement that originated from Wing's essay does not take into account the importance of teaching students design thinking as well as critical thinking about digital technologies, and at the same time it highlights algorithms presented as step-by-step procedures; thereby, the movement does not take into account the excessive use of machine learning models in our society today, some of which can be solutions to key problems in the world, and some of which pose a threat to our democracy. We should rather and in a broader sense aim for students to develop what in a Danish context has been named technology comprehension.

6.2 *How* can computational thinking be implemented in the classroom?

In continuation of my conclusion above, the wording of this question should rather be how *technology comprehension* can be implemented in the classroom.

With reference to Naur, learning to master individual techniques is not what students need; rather they need to be able to skillfully apply techniques in real projects. At a visionary level, Naur formulated a model on the relationship between people, problems, and tools in problem-solving processes; that is, he saw problem-solving as a process with a conscious relationship between people, problems, and tools. Exemplified by programming, programming is only a tool to solve some problems for people, and a way of learning to use such a tool is in an authentic context.

When planning a course, teachers can maintain awareness on the relationship between problems and people; people and tools; and tools and problems. For example, are the tools (such as programming) used to solve problems that the students regard as problems? During their teaching, teachers can show the students the model for them to become aware of these relations, and when assessing their teaching, the teacher can use the model to assess whether the students understood these relations.

Through planning and conducting a design experiment, I have illustrated what teaching students to deal with key problems in the world, using computational thinking, design thinking and technology criticism as tools, could look like in practice. Because of analysis from *why* and *what*, the experiment aimed for students to develop not only computational thinking skills, but also broader skills within technology comprehension. There are, however, many ways of teaching technology comprehension. This experiment is not meant to illustrate a best practice, but to illustrate one good way of doing this.

Other historical *hows*, that were found to be able to serve as inspiration today are unplugged approaches to understanding data, computers and areas of application, for example by using flowcharts to develop algorithms to deal with problems. I have emphasized that historical unplugged ways of teaching technology understanding (including computational thinking) cannot stand alone. At some point we need to plug our ideas into the computer to understand how it works; to understand what machines are capable of – and not capable of.

6.3 Contribution, Limitations and Future Work

When I selected to focus on the three didactic questions *why*, *what* and, secondarily, *how*, I automatically deselected focusing on other didactic questions. In this paragraph, I discuss the contribution as well as the limitations of my project.

In Chapter 2, I have discussed my didactic position as a subject-specific didactic researcher with reference to theories by Klafki, Nielsen, Schnack, and Bundsgaard. In short, Klafki sees didactics as not only a theoretical discipline but also as a science of practice for practice. Nielsen elaborates on that and argues that didactics do include theory but also planning and decisions, which can be scientifically based on scientifically oriented didactics. He distinguishes between a narrow understanding that involves the didactic questions *what*, *where to*, *why* and *who*, and a broader understanding that also involves *how*, *with what,* and *where*.

Bundsgaard argues that subject-specific didactic researchers need to examine and discuss their subject with general reflections on the overall purpose of education. To him, subject-specific didactics is not limited to reflections on content, as in some didactic theories. It is a much broader discipline that involves reflections on the society and the world, the purpose of education, teaching objectives, teaching content, methods and organizations of settings, collaboration and activities, and evaluation. He points out that it does not make sense to reflect on only one of these aspects as if it was autonomous. Taking into account how didactics is far from only a theoretical matter of why and what, but in a broader sense also includes how, where to, who, with what, and where, and since all of these aspects are interdependent and interconnected (see Figure 1), research is needed on other didactic questions.

Bundsgaard does, however, argue that it is completely acceptable to be more interested in one aspect than in the others. In this project, I primarily chose to focus on examining theories about *why* and *wha*t. As a subject-specific didactic researcher in the scientific field of computational thinking and technology comprehension education, I first and foremost discuss general questions about the purpose of education with regard to what Schnack refers to as a didactic of challenges; that is, questions about what content is important for us and the next generation in order to handle the challenges that we face as a society and as humans. This involves reflections about society and the world. I connected results from my sub-studies to conclude *why what* content is relevant in compulsory education. Based on the literature that I reviewed as well as my theoretical didactical position, the project contributes with evidence on *why* we need to teach computational thinking and – in a broader sense – what in Denmark has been named technology comprehension, as well as *what* aspects of computational thinking we need to teach. On that note, computational thinking as a subject-area is, in the light of my didactic analysis too narrow to capture what students need to learn today.

The theories from my studies on *why* and *what* can enrich practice. But in my view, it is, as Nielsen puts it, insufficient to theorize. Therefore, I decided to plan and conduct a design experiment in practice based on my theories (scientifically based practice), thereby connecting my results regarding *why* and *what* with *how*. My examination of *how* includes reflections on objectives, methods, organizations of settings, collaboration and activities, as well as evaluation. It is, however, not an in-depth study. For example, the method was *inspired* by design-based research, but we did not do iterations and improve our practice to generate new theories. In-depth studies on *how* are needed, as are other approaches to teaching technology comprehension.

In addition, the field of technology comprehension is developing rapidly and constantly, and present literature might include more in-depth arguments on *why* to teach *what* – as well as more comprehensive views on what computational thinking in compulsory education should be about. Moreover, the school principals who took part in the survey might feel that they have a better understanding of *why* and *what* today.

Emphasizing the discussion on present perspectives and the analysis of the use of digital technologies in society today, it is evident that students need not only to learn to understand programming in terms of step-by-step algorithms and mathematical concepts, but also to understand machine-learning models and to critically reflect on such models. Research on this area is lacking and much needed.

References

AU (2018). Lov om folkeskolen, 26. juni 1975. *Danmarkshistorien.dk*, Aarhus Universitet. www.danmarkshistorien.dk/leksikon-og-kilder/vis/materiale/lov-om-folkeskolen-26-juni-1975

Bang, J. Chr., Døør, J., Steffensen, S. V. & Nash, J. (2007). *Language, ecology, and society: A dialectical approach*. Continuum

Barr, V. & Stephenson, C. (2011). Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community? *ACM Inroads*, 2(1): 48-54.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A. & Engelhardt, K. (2016). Developing computational thinking in compulsory education – Implications for policy and practice. *Joint Research Center, European Commission*. https://doi.org/10.2791/792158

Brennan, K. & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association*, Vancouver, Canada: Pp. 1-25.

Bundsgaard, J. (2005). *Bidrag til danskfagets didaktik*. PhD Dissertation.

CSTA & ISTE (2011). *Operational Definition of Computational Thinking for K-12 Education*. https://cdn.iste.org/www-root/Computational_Thinking_Operational_Definition_ISTE.pdf

Denning, P. J.; Tedre, M. & Yongpradit, P. (2017). Misconceptions about computer science. *Communications of the ACM*, February 2017. https://doi.org/10.1145/3041047

Denning, P. J. (2017a). Computational Design. *ACM Ubiquity*. Volume 2017, August: 1-9. https://dl.acm.org/doi/10.1145/3132087

Denning, P. J. (2017b). Viewpoint. Remaining Trouble Spots with Computational Thinking. *Communications of the ACM*, 60(6): 33:39.

EMU (2021). *Technology Comprehension*. https://emu.dk/grundskole/teknologiforstaaelse/technology-comprehension

EMU (2019). *Teknologiforståelse*. https://www.emu.dk/grundskole/forsogsfag-teknologiforstaelse/formal. Børne- og Undervisningsministeriet.

Fischer, C.; Frøkjær, E. & Gedsø, L. (1792a). *Datalære i skolen. Om data og edb i samfundet*. Textbook. Gads Forlag.

Fischer, C.; Frøkjær, E. & Gedsø, L. (1792b). *Datalære i skolen. Om data og edb i samfundet*. Teacher's Resource. Gads Forlag.

Frandsen, K. (1983) (ed.). *EDB i skolens undervisning*. Danmarks Skolelederforening.

Gundem, B. B. & Hopmann, S. (2002). *Didaktik and/or Curriculum: An International Dialogue*. Peter Lang.

ISTE (2014). Computational Thinking for All. https://www.iste.org/explore/Solutions/Computational-thinking-for-all

ISTE (2016). *ISTE Standards: Students*. https://www.iste.org/standards/iste-standards-for-students

Katapult/TEACH (2013). *Værktøjskassen: Model for designtænkning*. Projektet Next Generation. Københavns Universitet. https://innovation.sites.ku.dk/model/design-thinking

Klafki, W. (2016). *Dannelsesteori og didaktik – nye studier*. 3. ed. Forlaget Klim.

Ministry of Children and Education (2018). *The Aims of the Folkeskole*. https://eng.uvm.dk/primary-and-lower-secondary-education/the-folkeskole/the-aims-of-the-folkeskole

Moreno-León, J., Robles, G. & Román-González, M. (2015). Dr. Scratch: Automatic analysis of scratch projects to assess and foster computational thinking. RED. *Revista de Educación a Distancia*, (46): 1-23.

Naur, P. (2005). *Computing Versus Human Thinking. ACM Turing Award Lecture Video*. ACM. https://amturing.acm.org/vp/naur_1024454.cfm

Naur, P. (1966a). Datalogi og datamatik og deres placering i uddannelsen. *Magisterbladet*, May 15, 1966.

Naur, P. (1967). *Datamaskinerne og samfundet*. Søndagsuniversitetet – Bind 85. Munksgaard.

Naur, P. (1968). Demokrati i datamatiseringens tidsalder. *Kriterium*, 3. årg., nr. 5, June, 1968. Nyt Nordisk Forlag Arnold Busck.

Naur, P. (1966b). *Plan for et kursus i datalogi og datamatik*. Regnecentralen.

Naur, P. (1970). Project activity in computer science education. *Calcolo*, 7(1). https://doi.org/10.1007/BF02575555

Naur, P. (1965). The Place of Programming in a World of Problems, Tools, and People. *Proc. IFIP Congress*, 65: 165-199.

NGSS (2013). *Next Generation Science Standards. For States, By States*. National Academies Press.

Nielsen, F. V. (1998). *Almen Musikdidaktik*. 2. ed. Akademisk Forlag.

Palts, T. & Pedaste, M. (2020). A Model for Developing Computational Thinking Skills. *Informatics in Education*, (19)1: 113-128. https://doi.org/10.15388/infedu.2020.06

Selby, C. & Woollard, J. (2013). Computational thinking: the developing definition. Monograph (Project Report). https://eprints.soton.ac.uk/356481/

Smith, R.C.; Bossen, C.; Dindler, C. & Iversen, O.S. (2020). When Participatory Design Becomes Policy: Technology Comprehension in Danish Education. *Proceedings of the 16th Participatory Design Conference 2020*: 148-158. ACM.

Sveinsdottir, E. & Frøkjær, E. (1988). Datalogy – The copenhagen tradition of computer science. *BIT Numerical Mathematics*, 28(3), 450-472. https://doi.org/10.1007/BF01941128

Tedre, M. & Denning, P. J. (2016). The long quest for computational thinking. *Proceedings of the 16th Koli Calling Conference on Computing Education Research*, November 24-27, 2016, Koli, Finland: 120-129. https://doi.org/10.1145/2999541.2999542

Undervisningsministeriet (1972). *Betænkning om edb-undervisning i det offentlige uddannelsessystem*. Betænkning nr. 666. Undervisningsministeriet.

Undervisningsministeriet (2019). *Historie. Faghæfte 2019*. Undervisningsministeriet. https://emu.dk/sites/default/files/2020-09/Gsk_fagh%C3%A6fte_historie.pdf

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33-35. https://doi.org/10.1145/1118178.1118215

Wing, J. M. (2011). Research notebook: Computational thinking – What and why. *The Link Magazine*: 20-23.

All links have been checked on November 29, 2021.

Danish Summary

Gennem det seneste årti har interessen for datalogisk tænkning (*computational thinking*) i obligatorisk uddannelse (folkeskolen) været stigende. Fagområdet implementeres i læreplaner rundt om i verden, i takt med at vores samfund bliver mere og mere digitaliseret. Datalogisk tænkning ser altså ud til at blive betragtet som en grundlæggende almen kompetence i elevers nuværende og fremtidige liv.

I forlængelse af denne udvikling har formålet med dette forskningsprojekt været at undersøge årsagerne til, hvorfor (og om) datalogisk tænkning er en central kompetence at udvikle i obligatorisk uddannelse, og hvad det er vigtigt at undervise i.

Projektet var primært centreret om følgende forskningsspørgsmål: *Hvorfor* er det essentielt at undervise i datalogisk tænkning i obligatorisk uddannelse, og *hvad* er det essentielt at undervise i?

Derudover ønskede jeg at undersøge, hvordan undervisning kunne se ud i praksis, baseret på svarene fra mine primære forskningsspørgsmål. Derfor var mit sekundære forskningsspørgsmål: *Hvordan* kan datalogisk tænkning implementeres i undervisningen?

Metodisk set er dette et didaktisk forskningsprojekt baseret på kritisk-konstruktiv dannelsescentreret teori. Forståelsen er, at obligatorisk uddannelse bør sigte mod, at eleverne udvikler almene kompetencer, der forbereder dem til livet i et frit og demokratisk samfund med rettigheder og ansvar. Obligatorisk uddannelse skal altså ikke sigte mod udvikling af specifikke erhvervskompetencer, i dette tilfælde inden for datalogi.

Selvom der undervises i datalogisk tænkning, var det min hypotese, at argumenter for, hvorfor det er vigtigt at undervise i det i obligatorisk uddannelse, ofte enten mangler eller kun er overfladisk beskrevet i litteraturen på området. Denne hypotese var foranlediget af professor Peter J. Denning, der i et essay (2017b) blandt andet diskuterede, at der i dag er i tvivl om, hvad datalogisk tænkning er, og om det er en kompetence for alle – altså om datalogisk tænkning er en almendannende kompetence. Ligeledes diskuterede han, at der er tvivl om, hvad datalogisk tænkning indebærer. Han påpegede, at den nuværende nye forståelse af datalogisk tænkning er fattigere og mere snæver end en historisk forståelse af feltet, og han mener derfor, at man som forsker på området bør bygge oven på det fundament, som allerede er lagt historisk set og dermed skabe fremskridt – frem for tilbageskridt ved at starte forfra uden historisk indsigt.

I forhold til min teoretiske position og forståelse er *hvorfor* et af de vigtigste spørgsmål at kunne besvare. Det påvirker i høj grad, *hvad* vi underviser i, og *hvordan* vi bedst underviser i det. Derfor anser jeg det for centralt at være i stand til at argumentere for at kunne svare på, hvad der er det

væsentlige at undervise i, samt for kunne gøre sig refleksioner over hvordan man bedst underviser i det.

Jeg har undersøgt *hvorfor*, *hvad* og *hvordan* fra tre perspektiver: litteratur (state-of-the-art), historiske perspektiver og nuværende perspektiver.

Gennem et systematisk review og analyser af litteratur på området blev jeg bekræftet i, at argumenter for, hvorfor alle skal undervises i datalogisk tænkning i obligatorisk uddannelse, enten manglede eller var overfladisk formuleret. Derudover fandt jeg ud af, at datalogisk tænkning i nyere tid internationalt anses som overvejende matematiske problemløsningsstrategier, for eksempel abstraktion, nedbrydning af problemer i enkeltdele og automatisering.

I mine historiske analyser fokuserede jeg overvejende på dansk historie inden for området, da dansk obligatorisk uddannelse (folkeskolen) i overensstemmelse med min teoretiske position har tradition for at være dannelsesorienteret frem for erhvervsrettet. Her fandt jeg ud af, at der siden 1960'erne har været diskuteret et fag, der godt nok indeholdt elementer af det, der i en snæver forståelse i dag kaldes datalogisk tænkning, men som var langt bredere. Særligt professor Peter Naur argumenterede for, at alle i et samfund præget af digitalisering bør lære datalogi som et væsentligt tværfagligt værktøj på samme måde, som alle lærer at beherske andre væsentlige værktøjer, særligt sproget (herunder læsning og skrivning) samt matematik. Naur havde et samfundsmæssigt og demokratisk perspektiv på almen uddannelse, og således ønskede han netop ikke at indføre et fag med henblik på, at alle skulle blive dataloger, men med henblik på, at alle lærte at forstå data, deres natur og brug – herunder hvordan computere er programmeret. I modsat fald var han af den overbevisning, at eksperter på feltet ville få magten og altså bestemme samfundsretningen, hvilket ville afvikle demokratiet.

I halen af Naurs argumenter blev faget datalære formuleret i 1970'erne. Et udvalg nedsat af Undervisningsministeriet beskrev i en betænkning, hvad faget skulle indeholde, og af deres arbejde ses det, at matematisk problemløsning blev anset som utilstrækkelig. Historisk var der således et bredere fokus på blandt andet anvendelse af databehandling samt samfundsmæssige aspekter. Der var fokus på, at man forstod, hvad en computer var, og hvordan den kunne programmeres.

Computere var dengang meget simplere, end hvad det er tilfældet i dag. I dag er brugerflader designet så intuitive, at det kan virke ligegyldigt at sætte sig ind i, hvordan de egentlig virker. Det kan også virke ligegyldigt at kigge på simple historiske eksempler på undervisning i datalære, eftersom udviklingen på det digitale området har været enorm siden dengang. I denne afhandling diskuterer jeg, at det netop er en af historiens styrker på den måde, at sådanne simple eksempler kan fremme forståelsen for, hvordan computere rent basalt virker, og hvad de egentlig kan (og ikke kan).

Samtidig kommer jeg frem til, at hverken historiske perspektiver på *hvorfor*, *hvad* eller *hvordan* er tilstrækkeligt i et nutidigt og fremtidigt samfund. Nutidige analyser bør supplere. I et nutidigt perspektiv diskuterer jeg således, at Naurs argumenter samt indholdsbeskrivelsen af faget datalære grundlæggende stadig er gyldigt – delvist på grund af hans fremsynethed om fremtidig brug af computere samt eksperternes magt. Men det faktum, at samfundet har ændret sig drastisk inden for digitalisering siden 1970'erne, gør, at det ikke er samme *specifikke* argumenter (*hvorfor*). Der er ikke samme brug af computere samt muligheder og konsekvenser ved digitalisering i dag som dengang. Det er heller ikke samme *specifikke* indhold (*hvad*), der bør undervises i. I dag gør mange computerprogrammer for eksempel brug af maskinlæringsmodeller med dynamiske algoritmer, hvor datamodeller historisk set var noget simplere. Endelig bør undervisning heller ikke foregå fuldstændig (*hvordan*) som historisk set. For eksempel har vi i dag i langt højere grad digitalt udstyr og programmer til rådighed, ligesom ændringerne af *hvorfor* og *hvad* også ændrer *hvordan*. Jeg eksemplificerer med et specifikt undervisningseksempel, hvor jeg i samarbejde med en datalog har planlagt og gennemført et designeksperiment i en 8. klasse, hvor eleverne skulle designe en prototype på samt programmere et produkt, der kunne minimere energiforbrug.

Samlet set argumenterer jeg i denne afhandling for, at historiens diskussioner kan fungere som et godt fundament for nutidige teorier inden for og refleksioner over de didaktiske spørgsmål *hvorfor*, *hvad* og *hvordan*. Jeg konkluderer, at det ikke er datalogisk tænkning i den snævre gængse matematisk forståelse, der er brug for i almen uddannelse, men et bredere sammenhængende fag, i Danmark kaldet teknologiforståelse, der både fokuserer på udvikling af datalogiske kompetencer, designkompetencer samt teknologikritiske kompetencer.

Appendix: Articles A-E

This appendix includes a collection of the five academic articles A-E that along with the overview article (chapters 1-6) constitute my dissertation. All articles have been published in peer-reviewed national or international journals in 2019 and 2020.

Article A: Caeli, E. N. & Yadav, A. (2019). Unplugged Approaches to Computational Thinking: a Historical Perspective. TechTrends. AECT, Springer. https://doi.org/10.1007/s11528-019-00410-5

Article B: Caeli, E. N. & Bundsgaard, J. (2019a). Computational Thinking and Technology Comprehension in K-9 schools: A Round Trip.

Translated from:

Caeli, E. N. & Bundsgaard, J. (2019). Datalogisk tænkning og teknologiforståelse i folkeskolen tur-retur. *Læring og Medier (LOM)*, 11(19). https://doi.org/10.7146/lom.v11i19.110919

Article C: Caeli, E. N. & Bundsgaard, J. (2020). Technology Criticism in Schools – a Democratic Perspective on Technology Comprehension.

Translated from:

Caeli, E. N. & Bundsgaard, J. (2020). Teknologikritik i skolen – et demokratisk perspektiv på teknologiforståelse. In Haas, C. & Matthiesen, C. (Eds.): *Fagdidaktik og demokrati*. Samfundslitteratur.

Article D: Caeli, E. N. & Bundsgaard, J. (2019b). Computational thinking in compulsory education: a survey study on initiatives and conceptions. Educational Technology Research and Development. AECT, Springer. https://doi.org/10.1007/s11423-019-09694-z

Article E: Caeli, E. N. & Dybdal, M. (2020). Technology Comprehension in Schools. Computational Design for Solving Authentic Problems.

Translated from:

Caeli, E. N. & Dybdal, M. (2020). Teknologiforståelse i skolens praksis. Datalogisk design til autentisk problemløsning. *Læring og Medier (LOM)*, 12(22). https://doi.org/10.7146/lom.v12i22.115613

Two articles have been published in English (A and D), whereas three articles (B, C and E) were originally published in Danish. The three Danish articles have been translated into English for this dissertation.

TechTrends
https://doi.org/10.1007/s11528-019-00410-5

ORIGINAL PAPER

Unplugged Approaches to Computational Thinking: a Historical Perspective

Elisa Nadire Caeli[1] · Aman Yadav[2]

Abstract
In the recent years, there has been a push to engage primary and secondary students in computer science to prepare them to live and work in a world influenced by computation. One of the efforts involves getting primary and secondary students to think computationally by introducing computational ideas such as, algorithms and abstraction. Majority of this work around computational thinking has focused on the use of digital technologies, in particular programming environments (Yadav, Stephenson, and Hong 2017). In today's highly digitalized world, we often associate computational problem-solving processes with the use of computers. Yet, solving problems computationally by designing solutions and processing data is not a digital skill, rather a mental skill. Humans have solved problems for eons and before anyone even thought about the types of digital technologies and devices we know today. The purpose of this article is to examine the historical route of computational thinking and how history can inspire and inform initiatives today. We introduce how computational thinking skills are rooted in non-digital (unplugged) human approaches to problem solving, and discuss how mainstream focus changed to digital (plugged) computer approaches, particularly on programming. In addition, we connect past research with current work in computer science education to argue that computational thinking skills and computing principles need to be taught in both unplugged and plugged ways for learners to develop deeper understanding of computational thinking ideas and their relevance in today's society.

Keywords Computational Thinking Unplugged · Problem-Solving · Primary and Secondary Education · Datalogy · Algorithms

Historical discussions on computers in education

The need of computer science in primary and secondary classrooms has been discussed by computer pioneers worldwide for around 60 years. At a conference in 1960, computer science professor Alan Perlis suggested that students should be taught to understand computers as general tools for problem solving rather than specific tools to solve specific problems (Katz 1960). Though Perlis was discussing this issue with undergraduate students in mind, he argued that designing algorithms to solve problems involved basic thought processes that everyone should eventually learn. However, as Katz (1960) pointed out " the pedagogy for computers had not yet been developed properly" and was a barrier to implement these ideas in formal schooling. MIT Professor, Peter Elias also suggested that humans should generally understand algorithms, but that it did not necessarily need to involve a computer and that everyone did not need to go into details on how a computer worked, stating: "I think it is quite possible to justify learning these topics, computer or no computer. That is, I don't think this is a matter of matching people to machines. I think it is simply a matter of being able to talk intelligently about a class of interesting problems that we have discovered" (as cited in Greenberger 1962, p. 202).

Around that same time, a Danish computer science professor, Peter Naur, argued that children should learn *datalogy* (Danish translation of computer science) as part of their general education. He invented this term as a protest to computer science to stress that computer science is not only the science of computations, rather of all types of data and data processes,

✉ Elisa Nadire Caeli
elisa@edu.au.dk

Aman Yadav
ayadav@msu.edu

1 Danish School of Education, Aarhus University, Tuborgvej 164, 2400 Copenhagen NV, Denmark

2 College of Education, Michigan State University, Erickson Hall, 620 Farm Lane, East Lansing, MI 48824, USA

Published online: 06 July 2019

and to move the focus away from computers as being central to this endeavor. Datalogy has a human aspect, he explained; and, data is a matter of human understanding (Naur 1966, 2005). Specifically, Naur argued that fundamental principles of datalogy should play a role in children's education stating, "All of us have had to learn a considerable amount of languages, arithmetic and mathematics in school, without many of us subsequently becoming linguists or mathematicians. In the same way we must bring computer science into the school and prepare ourselves for life in the era of the computers, just as reading and writing are regarded as necessary prerequisites for life in a society characterized by the printed word" (as cited in Sveinsdottir & Frøkjær 1988, p. 456–7). Hence, Naur saw datalogy as a cross-curricular problem-solving skill at the same levels as mathematics and language learning; but he acknowledged that while it might take decades to implement the necessary changes given the structural issues within educational systems, key stakeholders would eventually come to understand the need for such subject (Naur 1966, p. 7).

In addition, Naur underlined that people's understanding and formulation of problems are related to the tools they have at their disposal and that solving problems involves an understanding of those tools. He considered the relation between tools, problems, and people as a unified whole as illustrated in Fig. 1. He explained that problems exist only if someone come to think of them as problems – i.e., they only exist in the mind of people. In the same way, tools only become tools when someone thinks of them as things to solve some problems with. "The problem and the tool", he said, "is nothing if they are not recognized as such by a person" (Naur 1965).

Naur argued that characteristics of tools to solve problems, for good (and bad) shape the thinking of people and their conception of problems; therefore, problem-solving requires an understanding of tools (Naur 1965; Sveinsdottir & Frøkjær 1988, p. 463). He saw programming as a tool that can influence students' thinking in ways where they see problems and possible solutions based from a tool's perspective and what that specific tool is capable of, and he asserted that a deep understanding of programming and its capabilities is a precursor to developing programming skills. Thus, programming as a tool, problems, and people are interconnected to form key elements in computational problem-solving. This means that teaching programming from a specific tool's perspective, for example "learning Scratch" or "learning Python" with closed-ended solutions, does not alone develop deep understanding and wicked problem-solving skills. Teachers should engage students in learning to program by focusing on relevant problems that students could solve using programming as a tool when planning, conducting and assessing a broad and inclusive computer science education.

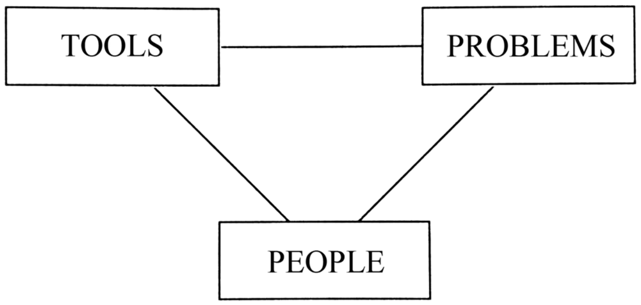


Fig. 1 The fundamental components in problem solving according to Naur (1965)

Understanding and Design of Algorithms

While programming can be an important tool for solving problems, discussions on how computer science should be taught have been going on since the 1960s, with some arguing for everyone to learn programming while others arguing for understanding the concepts and expressing algorithms as being the most important part (Malmberg 1970; Katz *1960*; Greenberger 1962; Naur 1966). For example, in 1972 the Danish Ministry of Education proposed a new datalogy subject for primary and lower secondary school students stressing that this subject had the ability to foster creativity and imagination since solutions of problems have many different forms. However, they addressed the need to engage students' in problem-solving activities that go beyond specific forms tied to programming, which are not generally applicable. (Undervisningsministeriet 1972). The intention was not to develop programming skills, rather to understand how to design algorithms as an iterative process. The argument against programming languages leading the learning process was that the formal and technical details of programming could possibly hinder understanding of designing algorithms. The alternative approach was using a tool, such as a flowchart, that was not restrained by formal structures and details and allowed the ability to communicate solutions as algorithms. Some have even argued that programming skills are not the most interesting part of problem-solving processes rather it was designing and expressing algorithms as Malmberg (1970) argued, "the actual – and often highly demanding – task is preparing the algorithm" (p. 274). He emphasized the importance of learning how to design and express algorithms in ways that *a machine* would be able to understand.

Similar, Knuth (1974) argued that learning computer science concepts have "educational side-effects" and described computer science concepts as general-purpose mental tools that develop deeper understanding in other subject areas. He argued that being able to express something as an algorithm prepares a person for much more than the formal act of programming, stating, "a person well-trained in computer science knows how to deal with algorithms: how to construct them, manipulate them, understand them, analyze them" (p. 326). Knuth further pointed out that computer machines and algorithms do not only compute with numbers, but "with information of any kind, once it is represented in a precise way" (Knuth, p. 323); thus, he liked how Naur's Danish term datalogy indicated that computer science deals with more than solutions to numerical equations.

During the 1980s, when computers started to enter schools and computer usage grew, there was still no pedagogy for using computers, which made educators anxious and uncertain as Pea and Kurland (1984) pointed out, stating, "Now that this admittedly powerful symbolic device is in our schools, what should we do with it?" (p. 137). Seymour Papert was one of the pioneers, who had a far-reaching influence on possibilities of how computers could be used in schools in powerful ways by having children program the computer rather than the computer programming the children (Papert 1980).

A Constructionist Approach to Learning to Think Computationally

Papert described what he called a schizophrenic split between "humanities" and "science" built into our language, worldview, and educational systems. He suggested that the computer could help break down the line between these cultures by bringing children into a more humanistic and humane relationship with mathematical ideas. In 1980, he introduced the first programming language for children, LOGO, to bridge this gap and provide an environment for children to communicate with computers (Papert 1980).

Papert's approach to learning were highly influenced by Jean Piaget, who throughout his career emphasized the importance of children's ability to reflect on their own thinking as "builders of their own intellectual structures" (Papert 1980, p. 7). Papert (1993) stated that while many "computer-aided instructions" at the time were designed for computers to program the child, LOGO could support development of new ways of thinking and learning. In particular, he stressed that the relationship between the child and computer when using LOGO was different as "the child programs the computer" (Papert 1980, p. 5). LOGO was designed as a tool to allow children to build intellectual structures and prepare them for a future where computers would become a significant part of their lives.

While Wing (2006, 2010) popularized the term computational thinking, Papert (1980) had already introduced the idea when discussing computational environments for children, which he found were too weak at that time. He asserted that the vision of these environments on "how to integrate computational thinking into everyday life was insufficiently developed...but there will be more tries, and more and more. And eventually, somewhere, all the pieces will come together and it will 'catch'" (Papert 1980, p. 182). The idea computational thinking might even have been proposed before Papert when Naur (1970) used the Danish version of the term computational thinking (*datalogisk tænkning*) to describe analysis of data representations, data processing, data medias, etc.

During this period, researchers were also examining whether learning to program led to the development of general higher mental functions (Pea & Kurland 1984). Pea and Kurland argued that claims for the cognitive benefits of programming were not supported by empirical evidence even though their presumed validity influenced decisions in public education. With this climate of uncritical optimism about computer science, they saw a risk of having naive "techno-romantic" ideas embedded in the curriculum that were not supported by empirical evidence .

With the recent interest in bringing coding to primary and secondary education, we need to ask ourselves: How do we know that specific digital technologies and programming can enhance learning? What are cognitive benefits of learning to program and how do students transfer and apply skills from learning to program to other areas? If the goal of computational thinking is to expose students to skills specific to computer science ideas, do we need programming environments? And could we engage students in those skills using unplugged approaches? How do students transfer learning from unplugged environments to plugged environment? In the following section, we look at examples of unplugged approaches to teaching computer science to examine some of these questions.

Unplugged Approaches to Computer Science

Understanding Fundamental Principles

Based on arguments laid out by Naur and other early computer science pioneers, we believe that in order for learners to conceptually understand computer science ideas and practices, we need to add or even begin with unplugged approaches. Since Wing (2006) reintroduced computational thinking as a cross curricular twenty-first century skill, the idea of getting children to think computationally rather than just learning to program has been discussed widely (Yadav et al. 2016). Countries worldwide are launching reforms or arguing for the need to include the integration of computational thinking into compulsory education (Bocconi et al. 2016; ISTE 2016; NGSS 2013).

Wing argued that "computational thinking builds on the power and limits of computing processes, whether they are executed by a human or by a machine" (p. 33). Along the same lines, Weizenbaum (1976) argued that human abilities are more important and a computer is not indispensable or a prerequisite to solve managerial, technological, or scientific problems. Rather, computers provide processing power for tasks that would otherwise take too long to be completed by humans (Fischer et al. 1972). To add to this viewpoint, Naur

(1954) argued that the machine lacks both initiative and originality as it executes only mechanical processes that a human brain has planned for it.

To illustrate how Naur saw humans using tools (such, as programming) available to them when solving problems and that computational tools can be both plugged or unplugged, consider this example. If a person has 400 dollars at their disposal for food and drinks for a birthday party, and they want to get the most out of their money, they could go from one store to another and calculate prices with a pen and paper or with a calculator, they could search the internet for sales and create a spreadsheet for automating calculating and comparison processes, they could use an already automated spreadsheet, and so on. However, they would need an understanding of how spreadsheets work in order to adapt it to his needs. This is a simple example of identifying and solving a simple problem; yet, it illustrates the central position of humans and the objective position of tools.

Another example is NASA's transition from "human computers" to "automatic computers" that illustrates how computers simply execute computing processes made by humans. Until 1962, humans did the calculations and trajectory analysis of spaceflights, using other tools than programming. For example, a pioneer in that field, Katherine Johnson, worked as a "human computer" with equations that would control the trajectories by hand, on her desktop mechanical calculating machine (NASA 2018) – equations, which were later programmed into a computer. What happens in the computer is not magic. The computer simply runs programs, designed by human brains.

Thus, human creativity and innovation is at the center and computing tools serve an ancillary role. While plugged tools play an important role in today's highly digitized world and have remarkably expanded and enriched our toolboxes, we need to examine how unplugged activities support plugged activities given that humans are central in computing. To foster innovation and open solutions, mental human skills and understanding of the underlying principles of computing are required. Thereby, we argue that we need to combine unplugged and plugged approaches to engage students in computer science ideas.

Thinking and Designing before Writing the Code

Today, there seems to be a growing interest in examining unplugged approaches to computer science education. For example, computer scientist and founder of csunplugged.com, Tim Bell has asserted that computational thinking is more about humans than computers (Bell & Roberts 2016). Bell suggested that computational thinking is a useful toolkit in the *process* of problem-solving rather than a *product* and it can be applied to all sorts of situations, not necessarily involving a computer. One needs to start with good computational thinking before programming. But what does unplugged look like in action? History provides some suggestions.

Examples of Unplugged Approaches

Using computers in the classroom have been explored for around 50 years. Back then, computer science in education was approached in unplugged ways and there was a visible need to understand how computers worked. One reason for understanding how computers worked was because there were no user-friendly interfaces. As an example, Danish schools implemented *datalogy* activities in primary and secondary education many of which did not involve the machine. Figure 2 shows an example of students designing a computer with matchboxes to understand how a computer was actually built and how different parts of it worked, e.g. how programs were expressed as algorithms.

In general, Denmark implemented and experimented with a number of visionary datalogy initiatives in primary and lower secondary school already in the 1970s and 1980s even before computers were common in the classroom. They foresaw the risk of tech-giants shaping our future in non-democratic ways and wanted children to understand how these new automatic machines worked to prepare all humans to participate in shaping the world (for example Naur 1968). One example of this was datalogy being implemented into Language Arts and Mathematics for Grade 5–6 students in four different topics: The newspaper, Communication, The Shop, and The Travelling Agency (Holt 1988). For example, the Travelling Agency project was aimed to develop skills on how to use databases. The students visited Travelling Agencies and built a travelling catalogue – first manually and later using a computer. Fictitious families could order their travels both manually and with the use of a computer. The projects were used to engage students in problem identification and design of

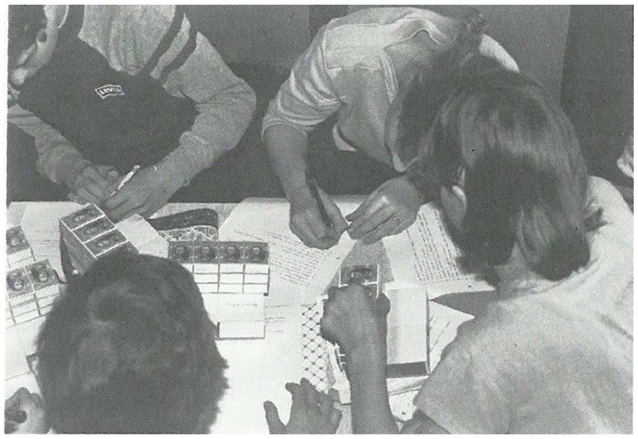


Fig. 2 Danish Grade 5 students building a model of a computer out of matchboxes to understand the different parts and their functions (Frandsen 1983). Photo: Dansk Skolelederforening

algorithms and solutions as their starting points rather than focusing on writing programs.

Today the prevalence of digital devices and user-friendly interfaces are so intuitive that we might not feel the need to look "under the hood" and understand how computing tools work. However, today it is more important than ever to understand how digital technologies work in order to understand how our data is collected with every single click we make and step we take and used by companies to deliver and even manipulate our online experiences. A better understanding of digital tools would also allow us to participate in designing our world by solving today's and tomorrow's problems. We must teach kids to learn to think computationally in creative and open-ended ways and not just to follow closed-ended instructions.

In a students' book on datalogy, Fischer et al. (1972) focused on teaching children a basic understanding of computer science concepts using unplugged tasks, in particular getting them to understand what data and data processes are (and are not) and what automation means. For instance, the authors suggested that the students work with the concept of data by thinking of any kind of item or product (for instance, an apple) and suggesting the data describing that product. They suggested to discuss how data processing works – for instance, how we can transform written text (old data representation) into sound (new data representation) by reading out loud – to get a better fundamental understanding of how data can be represented in different ways. The authors also provided examples on how many of the things we do in our daily life are data processes. For instance, when crossing a street with a traffic light, we process data in our mind:

1. We look at the color of the light. Data (the color) is represented in a way that our brains can directly work with.
2. In our minds, we compare the color of the light with the knowledge we already have about what these colors mean, that is: Red means "wait" and green means "walk".
3. The result of our comparison quickly transforms into new data in our minds that contain information on whether to walk or wait (Fischer et al. 1972, p. 15).

Old data transforms into new data (new information) when we add meaning to it. The fact that this is new data, might be easier to understand by imagining ourselves writing the results on a piece of paper (e.g. "wait" or "walk"), the authors explained. Such daily life examples can help students understand the nature of data and how humans process data automatically all the time.

If we want to express the traffic light data process as an algorithm, we could write a flowchart, see Fig. 3. Fischer et al. (1972) explained how this example transforms into many other actions, such as counting our money before buying a new bike or the way engineers calculate on the weight of things that are going to cross a bridge before building the bridge. This shows the role of designing algorithms that can be used to plan and test solutions to problems, which can then be automated by machines.

This example illustrates how data processing is independent of computers. In reality, however, when we solve big scale problems – such as data analysis of trajectories for spaceflights as illustrated above – we often deal with an amount of data and computations that are so difficult that computers are useful or even needed in order to solve the problem efficiently and in time. That is one of the reasons why we need to engage learners in plugged computational activities and unplugged should only be first step to develop deeper understanding of data and data processing and to showcase the important role humans play in the problem-solving process.

In summary, before engaging students in learning how to program, it is important for them to learn how to decompose problems into smaller more manageable parts (*decomposition*) and designing precise steps to solve those problems (*algorithms*), and then representing the solutions into code that can be *automated* by the computer. It is humans that solve problems, not code or computer, which are just tools at our disposal.

Evidence is also starting to emerge that unplugged approach can be effective in developing computational thinking skills and helping students translate those skills into coding. For example, a recent study compared plugged approaches to unplugged when students learned programming (Hermans and Aivaloglou 2017). The controlled experimental study included 35 elementary school children with half of them receiving plugged lessons first and the other half unplugged lessons first lasting four weeks. Afterwards both groups received plugged lessons using Scratch lasting four weeks. Results from this study suggested that after eight weeks there were no difference in how the two groups mastered concepts of programming; however, the unplugged group was more confident in understanding these concepts and used a wider selection of Scratch-blocks. Similarly, another quasi-experiment on plugged versus unplugged approaches to computational thinking in primary school, measured computational thinking skills of grade 5–6 students ($N = 73$) defined as decomposition, pattern recognition, abstraction and algorithmic design (Brackmann et al. 2017). Results from this study suggested that students, who participated in unplugged activities, increased those specific skills more than the students in the control groups who did not take part in these activities.

Combining Unplugged and Plugged Approaches

Today, a number of lessons on unplugged activities are also starting to emerge. For example, CS Unplugged provides "a collection of free learning activities that teach Computer Science through engaging games and puzzles" (CS Unplugged 2019) and Code.org compiled "a list of all of our

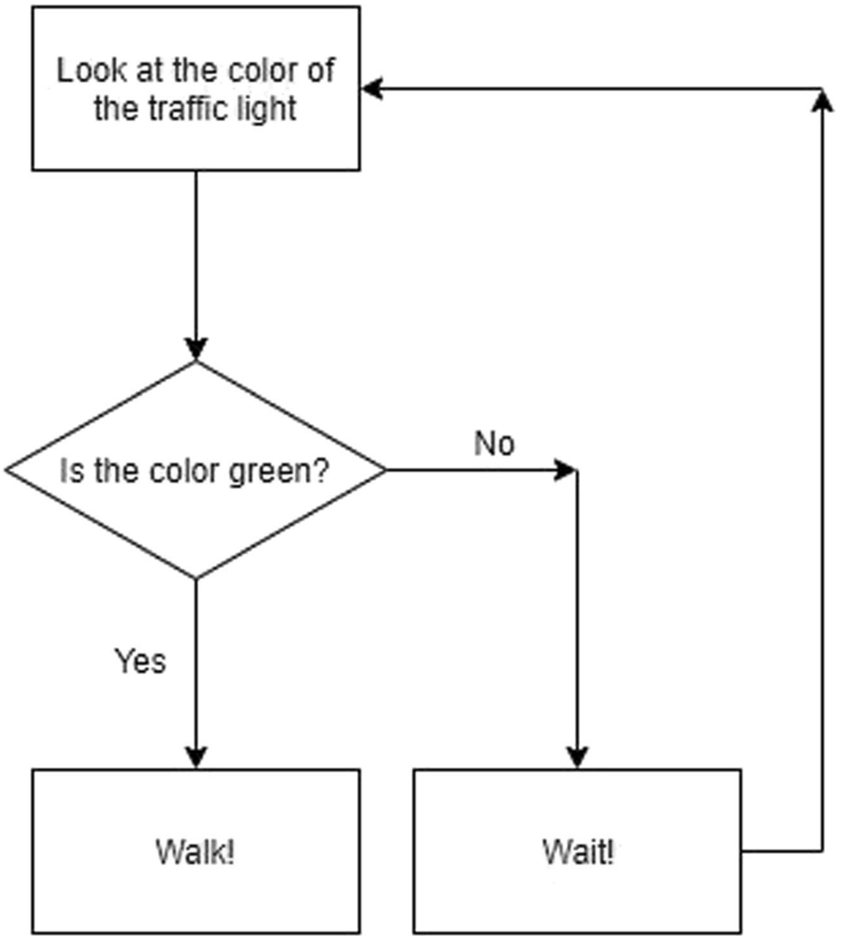


Fig. 3 Flowchart example, translated from Fischer, Frøkjær & Gedsø 1972, p. 18.

unplugged lessons for you to use in your classroom". These activities can "either be used alone or with other computer science lessons on related concepts" (code.org 2019). However, though we applaud such initiatives, there is a risk that if we teach concepts as stand-alone and out of context, they do not transfer into real situations and authentic problem solving.

Moreover, one cannot understand the potential of computational thinking using only unplugged approaches. To gain an understanding of automation processes and what computers are capable of as tools, students need to work with plugged approaches as well. Eventually, we need to implement the algorithms with a machine to test our computational ideas and solutions. Professor Peter J. Denning points this out by stating that, "an algorithm is not any sequence of steps, but a series of steps that control some abstract machine or computational model without requiring human judgment" (Denning 2017). The new movement with computational thinking, he says, makes fuzzy and overreaching claims by presenting algorithms as any sequence of steps, such as the procedures we follow in our daily life. This aligns with our previous discussion: When learning about the nature and use of algorithms, it is key that they are designed in such precise ways that a machine can understand. Daily life examples, like backpack packing, tying shoelaces, or baking cakes, might be useful to illustrate and connect the idea of algorithms with well-known activities – but they should not stand alone.

As we bring unplugged and plugged approached to computational thinking in primary and secondary classrooms and create curricula that leverages both to develop students' capacity to use computing tools to solve problems and enhance creativity, we need teachers who think computationally and are prepared to embed computational thinking ideas into their classrooms.

Educating teachers to teach computer science concepts has also been discussed since early 1970s. For example, Malmberg (1970) argued that lack of preparing teachers trained in datalogy remained a choke point to embed it in the primary and secondary education. Even back then, researchers argued for the need for computer scientists and educators to work together to bring relevant topics and terminology into primary and secondary classrooms (Naur 1966) given that teachers (rather than computers) are the ones that teach programming (Pea & Kurland 1984).

Even today, there is a problem of qualified teachers, who are prepared to teach computer science concepts. For example, CSTA (2013) report that computer science teacher certification in the United States was deeply flawed and that the importance of computing in our daily lives had not translated to preparing teachers to teach it. Another study also found that while Danish school principals found it important to teach computational thinking in primary and lower secondary schools, they saw insufficient teacher training as one of the biggest challenges to bring computational thinking ideas into classrooms (Caeli & Bundsgaard forthcoming).

While national curriculums are pushing computer science as a stand-alone subject and integration of computational thinking in subject areas, we lack significant number of teachers with adequate knowledge and skills. From pre-service to in-service teachers, there is a lack of understanding of computational thinking and typically, they have simplified views of computational thinking, including seeing it as mathematics or rudimentary uses of computers (Yadav et al. 2014; Sands et al. 2018; Yadav et al. 2018). We believe that a key component of building teacher capacity is to show relevance of computational thinking to their classroom using unplugged approaches combined with plugged approaches.

Conclusion

In this article, we have examined the historical route of computational thinking to illustrate similarities between ongoing discussions today and those from as early as 1960s. The purpose was to discuss how history can inspire and inform us today when teaching computational thinking in compulsory education.

Specifically, we introduced how computational thinking skills are rooted in non-digital (unplugged) human approaches to problem solving, and we discussed how mainstream focus changed to digital (plugged) computer approaches with a focus on programming. As early as the 1960s, Peter Naur argued for the need to bring computer science into school at the same level as reading and writing. He considered the relation between tools (such as, programming), problems, and people as a unified whole in problem solving processes and underlined the importance of human brains – not machines – in solving problems.

To gain an understanding of the potential of computational thinking and what computers are capable of as tools, we need to use both unplugged and plugged approaches in the classroom. Thus, we advocate for combining unplugged and plugged activities to provide students with an opportunity to fully understand and take advantage of the power of computing and prepare them to thrive in today's society.

Many of the current discussions around computational thinking described in our article were as lively in the 1960s, 1970s, and 1980s as they are today. Scholars in the field were engaged in similar discussions as we are having today: What is computational thinking? Does it require a computer? Is computational thinking also relevant in non-STEM subjects like humanities and arts or does it apply only to math and science? Does computational thinking prepare for workforce, life or both? Is it necessary for children in compulsory education to develop computational thinking skills, and why? How do we prepare teachers for teaching computational thinking? How is it possible for educators and computer scientists to successfully collaborate to consider both pedagogical concerns and computing as a domain? What evidence supports ways to teach computational thinking? These are all questions that need to be examined further and worthy of consideration as we develop and integrate computational thinking activities for primary and secondary classrooms.

Compliance with Ethical Standards

Ethical Approval This article does not contain any studies with human participants or animals performed by any of the authors.

Conflict of Interest Author A declares that she has no conflict of interest. Author B declares that he has no conflict of interest.

References

Bell, T & Roberts, J. (2016). Computational thinking is more about humans than computers, *set* 2016: no. 1, p. 3–7. https://doi.org/10.18296/set.0030

Bocconi, S.; Chioccariello, A.; Dettori, G.; Ferrari, A.; & Engelhardt, K. (2016). Computational thinking in compulsory education. Joint Research Center. European Commission.

Brackmann, C. P.; Román-González, M.; Robles, G.; Moreno-León, J.;Casali, A.; & Barone, D. (2017). *Development of Computational Thinking Skills through Unplugged Activities in Primary School.* WIPSCE 2017. Proceedings of the 12th workshop in primary and secondary computing education: 65–72.

Caeli, E. N. & Bundsgaard, J. (forthcoming). *Computational Thinking in Compulsory Education: A Survey Study on Initiatives and Conceptions.* Manuscript submitted.https://code.org/curriculum/unplugged

CS Unplugged (2019). *Computer science without a computer.* https://csunplugged.org/en/

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM, 60*(6), 33–39. https://doi.org/10.1145/2998438.

Fischer, C.; Frøkjær, E.; & Gedsø, L. (1972). *Datalære i skolen. Om data og edb i samfundet.* Gads Forlag.

Frandsen, K. (1983) (ed.) EDB i skolens undervisning. Danmarks Skolelederforening.

Greenberger, M. (1962) (ed.). Management and the Computer of the Future. The M.I.T. Press and John Wiley & Sons, inc.

Hermans, F., & Aivaloglou, E. (2017). *To Scratch or not to Scratch? A controlled experiment comparing plugged first and unplugged first programming lessons. WIPSCE 2017. Proceedings of the 12th workshop in primary and secondary computing education* (pp. 49–56).

Holt, Lone H. (1988). Datalære integreret i dansk og matematik. *Datalære*, Årg. 12, nr. 5, pp. 16–19.

ISTE (2016). ISTE standards for students. https://id.iste.org/docs/Standards-Resources/iste-standards_students-2016_one-sheet_final.pdf?sfvrsn=0.23432948779836327

Katz, D.L. (1960). *The Use of Computers in Engineering Classroom Instruction. Conference Report. College of Engineering.* The University of Michigan. The Ford Foundation Computer Project.

Knuth, D. E. (1974). Computer science and its relation to mathematics. *The American Mathematical Monthly, 81*, 323–343.

Malmberg, A. C. (1970). *Datalogi i skolen: Læreruddannelsen – en flaskehals. I Undervisningsministeriets tidsskrift* (pp. 272–276).

NASA (2018). Katherine Johnson biography. https://www.nasa.gov/content/katherine-johnson-biography

Naur, P. (1954). Elektronregnemaskinerne og hjernen. *Perspektiv, 1*(7), 42–46.

Naur, P. (1965). The Place of Programming in a World of Problems, Tools, and People. *Proc. IFIP Congress 65*, 165–199

Naur, P. (1966). Plan for et kursus i datalogi og datamatik. Regnecentralen.

Naur, P. (1968). Demokrati i datamatiseringens tidsalder. Kriterium, 3(5): 31–32. Nyt Nordisk Forlag Arnold Busck.

Naur, P. (1970). Planer og ideer for datalogisk institut ved Københavns Universitet. Studentlitteratur.

Naur, P. (2005). *Computing Versus Human Thinking*. A. M. Turing Award Lecture Video. ACM. https://amturing.acm.org/vp/naur_1024454.cfm

NGSS Lead States. (Ed.) (2013). Next generation science standards: For states, by states. National Academies Press.

Papert, S. (1980, 1993). *Mindstorms. Children, Computers, And Powerful Ideas*. Basic Books

Pea, R., & Kurland, M. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology, 2*(2), 137–168.

Sands, P., Yadav, A., & Good, J. (2018). Computational Thinking in K-12: In-service teacher perceptions of computational thinking. In M. S Khine. (Ed.). *Computational Thinking in the STEM Disciplines* (pp. 151–164). Springer.

Sveinsdottir, E. & Frøkjær, E. (1988). Datalogy – The Copenhagen Tradition of Computer Science. *BIT Numerical Mathematics, 28*(3), 450–472

Undervisningsministeriet. (1972). Betænkning om edb-undervisning i det offentlige uddannelsessystem. In *Undervisningsministeriet*.

Weizenbaum, J. (1976). *Computer power and human reason: From judgment to calculation*. W H Freeman & Co.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35. https://doi.org/10.1145/1118178.1118215.

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.

Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: Pedagogical approaches to embedding a 21st century problem solving in K-12 classrooms. *TechTrends, 60*, 565–568. https://doi.org/10.1007/s11528-016-0087-7.

Yadav, A., Stephenson, C., & Hong, H. (2017). Computational Thinking for Teacher Education. *Communications of the ACM, 60*(4), 55–62. https://doi.org/10.1145/2994591

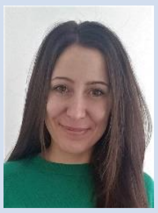
Yadav, A., Krist, C., Good, J., & Caeli, E. (2018). Computational thinking in elementary classrooms: Measuring teacher understanding of computational ideas for teaching science. *Computer Science Education.* https://doi.org/10.1080/08993408.2018.1560550.

Computational Thinking and Technology Comprehension in K-9 schools: A Round Trip

Elisa Nadire Caeli

PhD Student

Danish School of Education (DPU), Aarhus University and Department of Teacher Education, University College Copenhagen



Jeppe Bundsgaard

Professor

Danish School of Education (DPU), Aarhus University

This article is a translated version of:

Caeli, E. N. & Bundsgaard, J. (2019). Datalogisk tænkning og teknologiforståelse i folkeskolen tur-retur. Tidsskriftet Læring Og Medier (LOM), 11(19). *https://doi.org/10.7146/lom.v11i19.110919*

Abstract

In this article, we examine computational thinking and technology comprehension in education in a historical perspective. We present and analyze Danish educational developments in this field since the 1960s and until today, as well as how experiments with the subject known as "datalære" (data education) were conducted throughout the 1970s and 1980s. This subject focused especially on understanding digital technology in a critical perspective and on computational problem solving, and there are many similarities with the new Danish trial subject known as technology comprehension. The purpose of this article is to present the Danish experiences in this field and discuss how these experiences and the lessons learnt at a high cost can inspire us today.

The article presents events and initiatives from this period chronologically. Methodologically, we have selected empirical material that relates to initiatives and discussions about computer science and digital technology in Danish K-9 schools. We identify four periods from 1966 until the present with a view to establishing an overview of trends, central initiatives and discussions.

Introduction

In recent years, computational thinking has received increasing attention in general education all over the world, including in Denmark. Most recently, a number of experiments involving computational thinking have been launched in 46 Danish K-9 schools over a period of three years from 2018 under the name 'Technology comprehension', in which the subject is tested both as an independent subject and as an integrated part of seven existing subjects. Among other things, the aim of the technology comprehension subject is to help students to become critical co-creators of our (digitalized) society, including developing an ability to understand the possibilities and consequences of digital technologies, and to analyze and design digital technologies to solve complex problems (EMU, 2019). For example, a knowledge target for the independent experimental subject is described as having "knowledge of the characteristics of algorithms and their structure, as well as how they are applied in different contexts," and in the subject social science, students must be able to "discuss and consider the significance of digital artefacts or technologies for the development of society." This requires competency in thinking computationally.

In many ways, the subject description is similar to the subject of data education, which was discussed in the 1960s, with attempts at implementation in the 1970s-1980s. In this subject, problem-solving and social-critical perspectives were also in focus. In this context, we can also observe striking similarities between the past and present-day debate in the area, despite many decades of a rapidly accelerating digital society in which new applications are being developed at record speed, and in which we use a wealth of digital tools every single day.

When designing and testing a new subject, it is useful to know about previous generations' insights and experience in the area. This enables us to build an even more solid foundation for a subject than we could without these insights and experiences, and we can better avoid repeating the same errors as in the past. Thus, the purpose of this article is *to explore how the subject has developed, declined and now re-developed over time and up until today in order to derive inspirational experiences from which we can learn today.*

Theoretical and methodological basis

The point of departure for the article is the early period of the subject *Data education*, and the end point is the current period of *Computational thinking and technology comprehension*. The description of the two intervening periods *Operational user competencies and infrastructure,* as well as *Procurement of hardware and development of teaching resources* aims at examining and analyzing what has caused the long-standing struggle for a place on the school timetable.

The historical events in the article are described chronologically. A systematic division into the four periods mentioned above makes it possible to create an overview of trends and thereby compare the key initiatives and debates in the various periods. In order to clarify the chronology of the focus of the different periods and how the periods affected development, we have prepared a timeline of actual events (Annex 1).

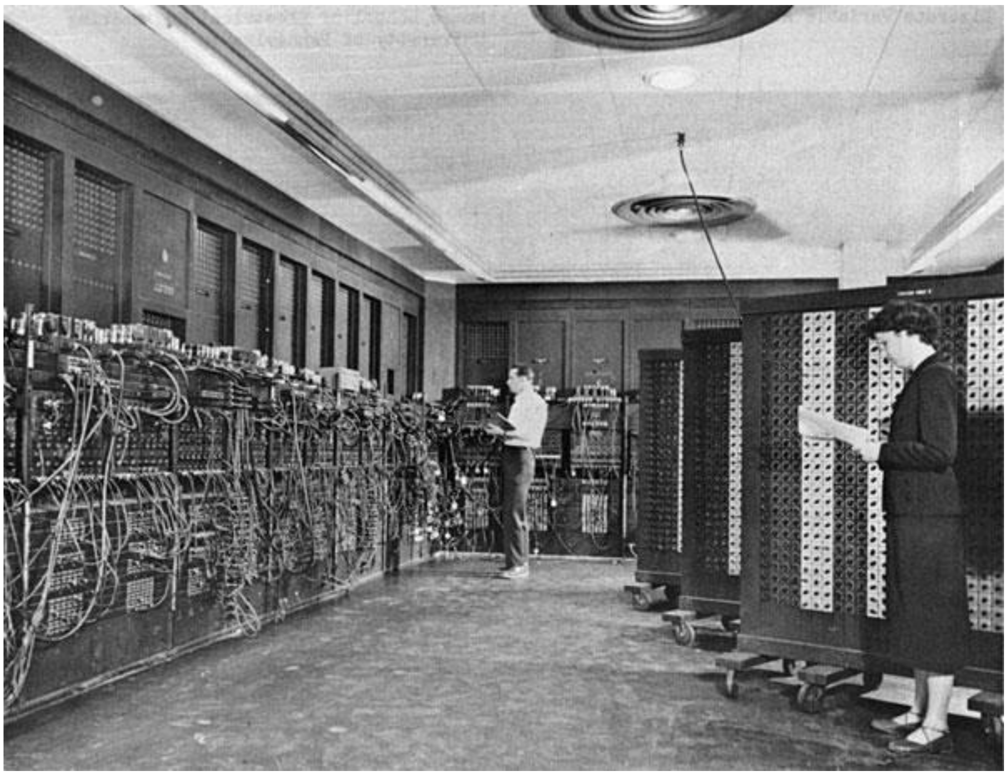
Methodologically, we have selected empirical material, which is limited to initiatives within, and discussions about computer science as well as digital technology in K-9 schools. We present and analyze historical trends and events on the basis of different types of empirical data: statutory provisions in the form of factual presentations of educational policy initiatives in the area, including official subject descriptions and decision-making processes;

academic theory in the form of descriptions and analyses of what researchers within computer science and education have, over time, emphasized as important in a general education school; and events in practice and in the societal debate in the form of descriptions and analyses of how society and schools in practice have acted within the individual periods. Thus, we have conducted an explorative literature search based on key concepts such as "datalogi" (datalogy/computer science) and "datalære" (data education), as well as by searching for key people who have been mentioned in the public debate, participated in legislative work, designed teaching resources, etc., and by talking to stakeholders at that time.

History can help us to establish an understanding of developments and give us a glimpse into a fledgling digital age in which computers were far from commonplace, but when, nevertheless, forward-looking and progressive initiatives in the area from which we can draw inspiration today were taken as early as in the 1960s. However, we will start a little earlier, in the 1940s, when the world's first computer was ready for use.

Everyone has to learn about the principles of computer science

In 1946, the world's first fully functional digital computer was constructed. It took three years to build the ENIAC, which was developed by J. Presper Eckert and John Mauchly at the University of Pennsylvania. The ENIAC was further developed in the following years, so that in 1956 it had more than 100,000 electronic components and a memory of 100 words (approx. 1600 bits). At that time, the "Giant Brain" as it was called, covered 167 square meters, weighed 30 tons, cost DKK 3 million, and consisted of 6,000 cables that technicians had to move around if they had to change the program. In other words, computers in school classrooms were not just around the corner. However, it goes without saying that digital developments have exploded since then. Most of us go around with a computer in our pocket, which, despite its size, processes huge amounts of data and performs tasks for us around the clock. Even though very few would have predicted the extent of the computer's influence today, pioneers in the area were already getting ready for the entry of the machines into society and schools.



The 30-ton ENIAC (Electronic Numerical Integrator and Calculator) was developed by J. Presper Eckert and John Mauchly at the University of Pennsylvania. The picture was taken in the period 1947-1955 at the Ballistic Research Laboratory, and according to Wikipedia, it depicts the programmers Glen Beck and Betty Snyder in the process of programming ENIAC. This first model had no memory, so the program was attached to cable connections. It could therefore take several days to reprogram the machine. (Photo: Wikimedia Commons / Public Domain).

One of the leading pioneers was Denmark's first professor of computer science: Peter Naur. Professor Naur was critical of the way people talked about computers as intelligent – as if there were a link between the ways in which a human brain and a computer functioned. In 1954, he called off the alarm about thinking machines in his article *The Electronic Computer and the Brain* (Naur 1954), making his point that the danger lurked, not with the machines that can perhaps think, but with the people who cannot, and he also pointed out that the machine completely mechanically executes the processes that a human brain has planned for it. The machine lacks both initiative and originality. These points of view formed the basis for the majority of his subsequent work, in which he extensively studied the way the human brain works and thereby refuted that there was an overlap with the way a computer works. Through a large number of initiatives, he advocated that everyone should get to know the basic principles of *datalogy*. He held radio lectures on the importance of computers for society and the individual, he wrote a large number of articles on the position of datalogy in general education, and he designed detailed curricula; work that really took off in the 1960s.

Peter Naur had no doubt that everyone – including children – should develop computational skills in the same way as they developed interdisciplinary skills in languages and mathematics at school. He invented and introduced the Danish concept of *datalogy* (*datalogi*) in connection with the publication of a plan for an educational subject in *datalogy* and *datamatics*, as he considered there was a need for useful designations for these topics in Danish literature and as a protest against the English term *computer science*, which he considered mistakenly indicated the computer as the focal point. Datalogy was about data and data was a question of human understanding (Naur 1966; 2005).

Naur saw a great need for more elementary teaching in datalogy, stretching beyond the use of specific programming languages. He believed that the basic concepts of datalogy were of a more general nature, and that in many ways they could shed new light on everyday tasks (Sveinsdottir & Frøkjær, 1988). In fact, he believed that once one realizes how datalogy combines vital human activities and concepts, and can inspire and create new ideas in all subjects, there can be no doubt that it should have its rightful place in general education. According to Naur, language and mathematics were closest related subjects, as, like datalogy, they entailed the use of tools such as signs and symbols created by humans, and in all three subjects, these tools could be used in a multidisciplinary manner. Thus, he believed that in a time with computers, everyone should learn datalogy as a necessary preparation for life.

> *We all learn writing, reading and arithmetic, no matter whether we end up being artists, doctors, lawyers or anything. ... In my view, there is a strong need for improved interdisciplinary elementary teaching in datalogy and datamatics, and as far as I can see, the only difficulty is that no one has bothered to draw up the requisite course material. (Naur 1966: 7-8, our translation)*

Naur argued for that the subject could be incorporated into a curriculum as either an independent subject or as part of mathematics – the crucial aspect is what is learned, he said, meaning that emphasis should be on data, data representation and data processes as fundamental concepts illustrated by simple experiments. Even though he believed that computers should also be mentioned as part of the subject, he pointed out that this was not the most important element. Instead, he emphasized the importance of datalogy in relation to basic human activities such as learning and problem-solving. While he was passionate about teaching datalogy in general education – because he had a strong belief that it was right and was important for society – he also doubted that it would happen easily or quickly, and he predicted that it would take decades to make the necessary changes due to the academic and organizational inertia of the educational system (Naur 1966).

One of Naur's most important points was that human understanding and formulation of problems is closely linked to the tools available at any time, regardless of whether or not these tools are digital, see figure 1.

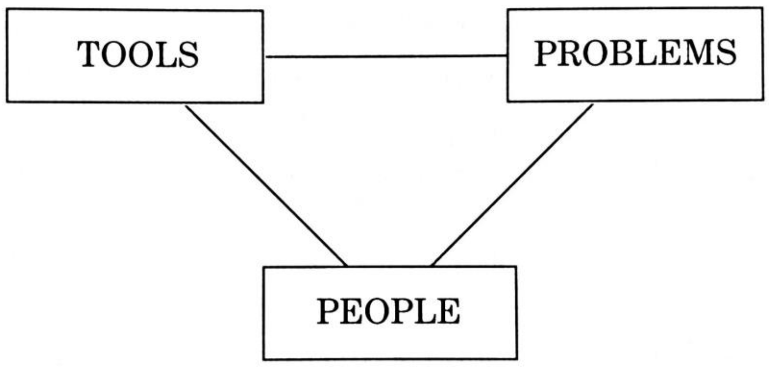


Figure 1. Fundamental components of problem solving according to Naur (Naur 1965).

He pointed out that a problem only exists by virtue of human consciousness, and that a tool only exists as a tool when human beings think of it as something that can solve a problem. As the characteristics of tools for better or worse shape the human mindset and human perception of problems, Naur believed that problem solving should entail an understanding of a tool to make sure that humans are not restricted creatively by its functions (Sveinsdottir & Frøkjær 1988: 463). Thus, he also believed that learning a programming language without a basic understanding was insufficient – a point that was also emphasized in a report on EDP[1] teaching in the public education system some years later, in 1972.

[1] *Electronic Data Processing*

Report on EDP teaching in the public education system

In 1970, the Minister for Education at the time set up a committee to, among other things, propose how EDP teaching could be integrated into general education. This resulted in report no. 666 on EDP teaching in the public education system (Danish Ministry of Education 1972), also known as the 'Johnsen report', in which chapter 4 covered K-9 schools. The committee took its outset in Naurs' concept of datalogy using the variant data education as a broader concept, directed at education. The aim of the subject was to develop better communication and better problem-solving skills, and the Committee described the subject area as the processes in societal life in which data is crucial. The Committee pointed out that the role of the citizen requires specific data, and that communication between people is a form of data processing, as is individual cognition (Danish Ministry of Education 1972: 23). The Committee summarized the methodology of computer science education as:

1. What is the problem, what is the goal?
2. Which model structure or data representation should be applied?
3. What observations or actual data are to be produced?
4. How should data be processed and the method of computation tested (the algorithm)?
5. How should the result be communicated to secure agreement that the problem has been solved, or alternatively that the goals have been achieved?

   (Danish Ministry of Education 1972: 24, our translation).

In its analysis of computer science education at K-9 schools, the Committee coupled the area with the objects of K-9 schools at that time, which among other things emphasized promoting independent attitudes, assessment skills and creativity, and considered teaching in relation to the development of society. The Committee also noted that the aim of K-9 schools was *not* to give students specific vocational skills (Danish Ministry of Education 1972: 39). In the analysis, the Committee emphasized that data education should deal with general, cross-disciplinary concepts and conceptualizations, such as data, problem formulation, model, algorithmization and process, and that the subject would thus play an important role in any problem-solving process. Therefore, they considered it obvious to incorporate the concepts into application-oriented contexts. They pointed out that data education could enhance students' creativity and imagination, since the solution of computational and *datalogical* problems can usually be designed in many different forms, and they warned that students' problem-solving activities could assume specific forms and lack general applicability. "If teaching is oriented towards programming or coding, this risk will be obvious," they pointed out (Danish Ministry of Education 1972: 40, our translation). As *datamatic* tasks are most commonly solved in a collaboration, they noted that the data education subject would encourage group work. Experience in the subject had also shown that it was possible to include elements of data education in all year groups in K-9 schools.

The Committee also presented some subject-specific didactic considerations. They highlighted the interdisciplinary nature of data education, and that the algorithmic way of thinking in data education was most useful. They described the formulation and description of algorithms as fundamental activities in most types of problem-solving, but again they emphasized that the use of existing programming languages as means of description was hampered by the many formal and technical details that they considered as irrelevant to bring into K-9 school teaching, and which could easily make it seem like coding or programming teaching, which was contrary to the actual aim. Therefore, they suggested using a means of description that was not hampered by a strict formal structure, such as a flowchart, but they also realized that the problem with this could be that, even though it was suitable for communication between people, it would be difficult to use in communication with a computer. The value in a student being confronted directly with the consequences of a proposal solution could easily be lost, and students would not gain an understanding of the iterative nature of the algorithm formulation. Although formal programming skills were not the main focus of data education, it was, however, a question of developing solutions that could be *executed by a computer*.

Teacher training for K-9 school teachers had a dedicated chapter in the report. The Committee stated that introducing the subject into teacher training was a necessary consequence of introducing the subject in K-9 schools. This was argued in particular by head of department and associate professor at the Danish School of Education (*Danmarks Lærerhøjskole*), Allan Malmberg, in the 1970s.

Teacher training as a bottleneck

One of Malmberg's points was that, more than any other factor, teacher training could become a bottleneck and could determine where computer science would end in the educational scene. Initially, he advocated educating mathematics teachers by virtue of course activities focusing on problem-solving with relevant thought processes:

> *After careful formulation of the submitted task situation, a plan is drawn up specifying the important phases in a work procedure that is expected to lead to solution of the task. On the basis of this, an algorithm for the solution process is drawn up, i.e. a detailed description in which the individual operations in the workflow are broken down into such an elementary level that, without further division, they can be included as individual instructions in the program that is later to be passed on to the computer. (Malmberg 1970: 274, our translation)*

Malmberg continued by explaining that, prior to the computer processing, it was necessary to translate the algorithm into a programming language, but he pointed out that this phase was the least interesting:

> *The actual – and often highly demanding – task is preparing the algorithm. (Malmberg 1970: 274, our translation)*

For this reason, Malmberg also advocated including activities in teaching to express algorithms in a way that could be passed on to the computer – without necessarily having to learn to program. But it was important to work on developing the mental *datalogical* thinking processes that precede programming.

What Malmberg wanted for the future was to implement and aim a subject within the field at teachers from all fields of study – not just mathematics and physics teachers. In this context, the role of the computer in society and the relationship between humans and computers should be addressed. He thought that the subject should be broader than was possible through just mathematics. It should not be stuck too tightly to mathematics, he said, because "society's use of computers is predominantly outside the mathematical field." In the years to come, therefore, he hoped that general education (K-9) and teacher training would be taken into account in development of a computer capacity within teaching and research as an "investment that would have the greatest impact on all of society" (Malmberg 1970: 276, our translation).

Datalogy as a mental tool with a general purpose

At about the same time, in 1974, the American professor of computer science, Donald E. Knuth, considered Denmark and Peter Naur's introduction of *datalogy* rather than the confusing *computer science*. Knuth pointed out that datalogy cleverly indicates that this science involves more than numerical equations. It has to do with data, i.e. the "stuff" that the algorithms manipulate, he said (Knuth 1974).

Knuth described an algorithmic view as a useful way of organizing knowledge very generally. He considered the question, "What can be automated?" as one of the most inspiring philosophical practical questions in civilization. He described the educational bonus effects of developing computational skills as knowing how to construct, manipulate, understand and analyze algorithms, and he pointed out that this knowledge would prepare a person for much more than writing good computer programs:

> *It is a general-purpose mental tool which will be a definite aid to his understanding of other subjects, whether they be chemistry, linguistics, or music, etc. (...) It has often been said that a person does not really understand something until he teaches it to someone else. Actually a person does not really understand something until he can teach it to a computer, i.e. express it as an algorithm. (Knuth 1974: 326-327)*

He explained that the computer forces very precise thinking, which leads to a much deeper understanding than if we tried to understand things in a traditional way.

Six years later, in 1980, Seymour Papert revolutionized the field with his work *Mindstorms*. In the introduction to the second edition, he wrote that a programming language such as his own LOGO, which was the first child-friendly programming language, could help students develop new ways of thinking and learning (Papert 1993). In most of the educational situations in which children came into contact with computers, the computer programmed the child, but the relationship in the LOGO environment was reversed: The child could now program the computer.

> *In teaching the computer how to think, children embark on an exploration about how they themselves think. (Papert 1980: 19)*

Papert's ideas stemmed from his work with Piaget. Papert was impressed with the way in which Piaget perceived children as active creators of their own intellectual structures, and he believed that the development of programming skills with the associated exploration of own thinking could be transferred to other areas. For example, he pointed out that programming was about being good at isolating and correcting errors – i.e. correcting the parts that prevent a program from working. "The question to ask about the program is not whether it is right or wrong, but if it is fixable," he said (Papert 1980: 23). If this mindset were transferred to how society thought about knowledge, he also believed that we would be less intimidated by the fear of "making mistakes".

Papert described his ideas about new opportunities for learning, thinking and emotional and cognitive development based on computational technology and computational ideas, as dependent on a future in which computers would be a significant part of every child's life (Papert 1980: 17-18). He was the first to apply the specific terminology of computational thinking when he explained in his book how it was not yet possible in 1980 to create computational environments in the form of social "computer clubs". Even though there had been attempts to do so, the environments were too primitive, he said.

> Their visions of how to integrate computational thinking into everyday life were insufficiently developed. But there will be more tries, and more and more. And eventually, somewhere, all the pieces will come together and it will 'catch'. *(Papert 1980: 182)*

Experiments with data education in the 1980s

At the same time, in Denmark, in 1983, the Danish association of headmasters (*Danmarks Skolelederforening*) published a practice-oriented booklet on EDP in schools (Frandsen 1983), which contained a status report on EDP in teaching at K-9 schools, and described four initiatives in which K-9 schools could, at their own initiative, or with funds from the Danish schools trials council (*Folkeskolens Forsøgsråd*) experiment with data education in teaching. Despite the 1972 Johnsen Report, "for inexplicable reasons" there was no data education on the school schedule.

The introduction to the booklet stated that, despite their very short lifespan, computers had already had a very large influence on society and everyday life, and it was predicted that this was probably only the beginning of a transition to an information society. It was stressed that there were two different interfaces with regard to use of the computer and EDP at schools: EDP as a means to solve tasks, and teaching about EDP, computers and consequences for us all, "but unfortunately, these things are often mixed together, with consequential ambiguity" (Frandsen 1983: 3). The first interface was further divided into the computer used as a medium, the computer used as a tool (aid) and the computer used for administrative tasks, while the second interface was mainly about data education as an interdisciplinary subject.

It was up to the individual schools themselves to formulate the objectives of the experimental teaching and what they thought the subject should contain. However, a suggested curriculum in 1985 described it as:

The objective of the teaching is to allow students to acquire insight into electronic data processing and its applications.

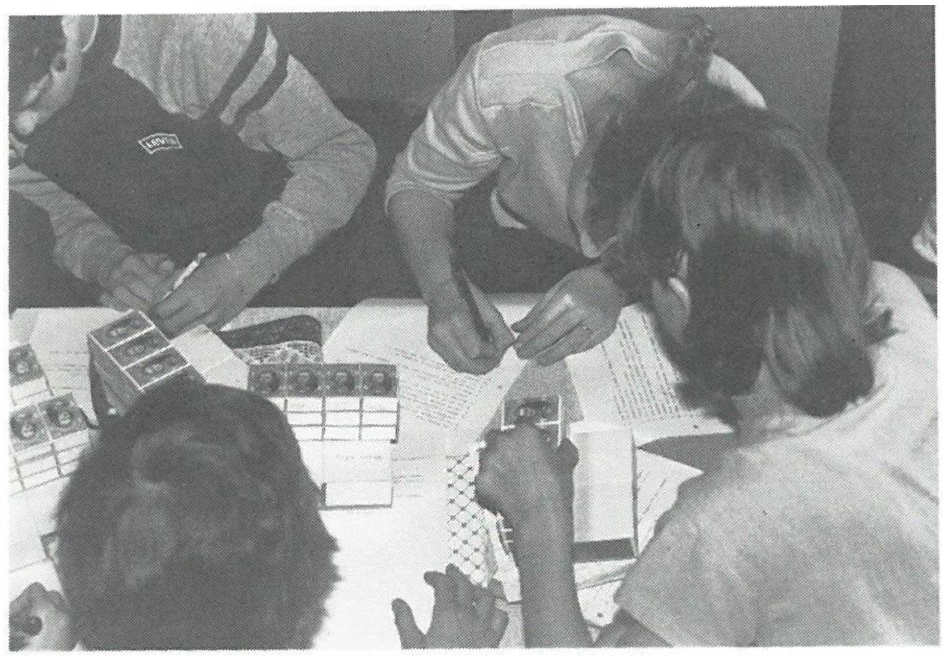
(2) The teaching will give students the opportunity to experience problem-solving through use of the computer.

(3) The teaching will help students acquire a basis to assess and consider the opportunities, influences and consequences that result from the use of computers.

(Skole og edb 1985: 149, our translation)

In 1986, in collaboration with the Danish schools trials council (*Folkeskolens Forsøgsråd*), more than 3,000 pages were written in reports on various topics: data education as a compulsory and independent subject; experiments in which elements of data education had been integrated in other subjects; experiments with data education as a stand-alone subject; and experiments with computer-aided teaching (Hansen & Jensen 1986: 35). There are also reports on experiments with data education in groups of exclusively girls or boys (for example Hjort Jensen 1986).

Common for all these reports was that, at that time, computers were by no means commonplace, and the obvious fact that computers had nowhere near the same intuitive user interfaces as they do today. This may be why teaching focused on the mental and social processes surrounding the understanding and development of computer programs, and why it was based on technologies *unplugged* (without electricity).



To illustrate the structure of a computer, these fifth-grade students (11 year-olds) made their own model of a computer, SKJOLD, out of matchboxes. The model consisted of an input unit, a control unit, a calculation unit, an internal storage unit and an output unit. They then went through various examples of user instructions (programs). In other words, they learnt about the different components of a computer through something concrete and tangible, but without using electricity. (Photo: Danmarks Skolelederforening, Frandsen 1983).

In 1981, a computer science education team held a fictive municipal election. The students examined the rules for elections and initially made the counts and calculations by hand. They then transferred their calculation methods to EDP and assessed the positive and negative consequences of this transfer. In fact, in the same year there were real municipal elections, and the students were given a room so that they could listen when polling results were telephoned in. They entered the results on their computers and, after a minute, they could come up with a distribution of successful candidates. (Photo: Danmarks Skolelederforening, Frandsen 1983).

In the US, researchers in the area were also discussing the nascent introduction of computers in schools. In a 1984 article, Roy D. Pea and D. Midian Kurland wrote about the revolutionary changes that were taking place in education at the time, with widespread access to computers at schools, which were used for learning activities across curricular, for example to design their own software. However, they pointed out:

> *... virtually all educators are as anxious and uncertain about these changes and the directions to take as they are optimistic about their ultimate effects. Now that this admittedly powerful symbolic*

device is in our schools, they ask, 'what should we do with it?' (Pea & Kurland 1984: 137)

Pea and Kurland described the environment at that time as uncritically optimistic about potential cognitive benefits from learning to program. They saw a risk of what they called naive "technoromantic" ideas; that ideas were anchored in the curriculum by affirmation rather than because of empirical verification through research and development.

Among other things, the two researchers compared programming with reading. Like reading, which has historically been considered as equal to decoding, programming was often considered as learning vocabulary and syntax in a programming language. However, the researchers pointed out that skilled programming, like reading, is a complex and context-dependent process that requires comprehension. They believed that there was far too much focus on the grammar and rules of programming, and they also maintained that neither programming languages nor computers should teach students programming. Teachers should.

During the same period, the important role of teachers was also being stressed in Denmark, highlighting that the computer is a tool in the teacher's hand: not a replacement for them.

BK.
Datamaskinen – et redskab i lærerens hånd!

The microcomputer was here to stay, but it should not be used at any price – and certainly not if other means or interaction between teacher and student could process the material in a better way. The computer could not, it was stressed, replace the teacher, and it was only a tool in the teacher's hand. Schools did not want to be taken lying down by the commercial interests of hardware manufacturers and book publishers; they wanted pedagogically designed teaching materials to be developed so that the electronic devices could be used appropriately. (Drawing: Danmarks Skolelederforening, Frandsen 1983).

As mentioned previously, data education was never realized, despite the many recommendations and in-depth descriptions from ambitious educators and researchers in the area. For a short period of time in the 1980s, the subject could be offered as an elective subject, but the Minister for Education, Bertel Haarder, later made it a compulsory section 6 subject in line with sex education and road-safety, which nobody took responsibility for: like the interdisciplinary subject 'IT and media' today. Over the next two decades, focus shifted from problem-solving and democratically oriented data education to IT skills and procuring software and hardware, followed by a phase in which IT was seen as a tool to support changed educational practices and to be integrated as such in all subjects. We briefly present these two phases in the following, before arriving at the situation today, where there is renewed interest in computer science as an approach and as content.

Focus on procurement and IT skills

In 1993, a new Danish School Act (*folkeskolelov*) came into force with a requirement that EDP was to be integrated into all subjects for all year groups. The Act included an additional three elective courses for 8th-10th grades (14-16 year-olds): text processing (as a replacement for typewriting), technology and media (Dalgaard 1994). In 1992, the Ministry of Education had conducted a survey of the number of computers schools had at their disposal for teaching, and this showed that there was a big difference between the schools' ability to comply with these new statutory requirements and how the municipalities prioritized financial resources in the EDP area. Lack of educational programs was also a problem. This triggered a number of initiatives.

Denmark's first internet-based network, *Sektornettet*, was established in 1993-1994 with the aim to connect the entire educational system to the internet. The network was largely in place before 2000. The work was carried out under the auspices of UNI-C (now the Agency for IT and Learning (STIL)).

Operational user competences or IT skills came into focus in 1996, when a number of PC user certificates were introduced, including a certificate for teachers, followed a few years later by a Junior PC user certificate for K-9 school students. The purpose of this certificate was to ensure that the students received a minimum basic knowledge about a PC. They received a certificate of IT skills with which they could document that they had the necessary standard qualifications to participate actively in the information society (Hansbøl & Mathiasen 2003). The focus of the certificate was on operations such as being able to change font to italics or save data.

In parallel with the focus on ensuring accessibility and IT skills, in the mid-1990s the Centre for Technology-supported Teaching (*Center for Teknologistøttet Undervisning*) was granted DKK 100 million to allocate to projects on IT in education, an IT toolbox called Poseidon was established, and the schools' database service (SkoDa) was set up (Bundsgaard, Petterson & Puck 2014).

Focus on pedagogical development and the use of IT in all subjects

This focus on the pedagogical aspects of IT in teaching was strengthened in 2001, when funding of DKK 323 million was implemented through the *IT and Media in K-9 schools* (*IT og Medier i Folkeskolen,* ITMF) project. Among other things, ITMF supported development projects, the production of a media library and continuing education for teachers. Later, in 2004-2008, the *IT in K-9 schools* (*IT i Folkeskolen,* ITIF) followed, with a budget of DKK 750 million, primarily to support computer procurement and development of six subject-specific digital teaching resources (Bundsgaard, Petterson & Puck 2014). In general, these teaching resources had a very short lifetime and did not form the basis for further development.

In 2009, the common goals (*Fælles Mål*) initiative, introduced in 2003 to replace the previous clear goals (*Klare Mål*) initiative, was revised, and in this connection Booklet 48, *IT and media competencies in K-9 schools* (*It- og mediekompetencer i folkeskolen)*, was published (Danish Ministry of Education 2009). The introduction to the booklet focused on digital service and functional mastery of IT as a means of communication, and it described children and young people as front runners because they quickly took to digital technologies and thus helped drive development. Among other things, it was said that "good digital skills are increasingly required to find and utilize recreational activities, for example participation in many outdoor activities requires that you seek information and register via the internet." Digital competences were defined as "the possession of certain skills within IT", but there was also an emerging focus on competencies in "critical information search, data processing and the IT user's ability to interpret diverse representations from digital media." Basically, the booklet consisted of the four interdisciplinary themes: information search and collection; production and dissemination; analysis; and finally communication, knowledge-sharing and collaboration.

In the late 00s, IT-didactic research began to gain momentum, among other things as a result of the requirement for follow-up research for the large ITMF grant, and because a number of PhDs had been educated in the area (Bundsgaard 2017).

Hardware procurement was also still in full swing, particularly as interactive whiteboards at first, and from 2011 as tablets, especially iPads. However, these hardware-focused initiatives also met with criticism. Among other things, procurement of interactive whiteboards in 2012 by the City of Copenhagen for all the municipality's schools hit front pages with headlines like "City of Copenhagen wastes DKK 21 million on interactive whiteboards" (Hansen 2012), and the Municipality of Odder's purchase of iPads for all teachers and students in 2011 for DKK 8 million was also subject to much criticism (Mortensen 2012).

In 2011, the government allocated DKK 500 million and the municipalities a further DKK 500 million for an initiative to increase the use of IT in K-9 schools, initially ending in 2015 and later extended to 2017. The initiative specifically supported municipal procurement of digital learning resources, development of digital learning resources, establishment of a teacher network, impact measurement of digital teaching resources, and the digital infrastructure in the schools. Educational development projects such as demonstration schools experiments were also supported (Danish Ministry of Education 2018b). In the meantime, IT and media were introduced as a cross-disciplinary theme under the new 2014 School Act by Minister for Education, Christine Antorini, and specific academic IT goals were included in all subjects. At the same time, Booklet 48 was withdrawn.

The conclusions of the demonstration school experiments indicated, among other things, that the prototypical teaching practice and integration of IT in K-9 schools was traditional and controlled by conservative logic, but they also showed that IT could promote creativity and innovative solutions (Hansen & Bundsgaard 2016).

Forward to the past: technology comprehension and computational thinking

In 2017, Minister for Education, Merete Riisager introduced an experimental elective subject in technology comprehension in 13 schools. The purpose of the elective subject has similarities with the data education subject in the 1970s as well as the elective subject in the 1980s, although with greater focus on programming skills than at that time.

In 2018, the Danish Ministry of Education (2018a) published an action plan for technology in teaching with a vision that Danish children should be creative with digital technology rather than just use it. The challenges in society are described as both future growth in Denmark and the degree of individual freedom. The background for the action plan is thus to ensure that we can all participate actively in our democratic society, including being critical about algorithms, and that we are prepared for the changes in core services predicted in the labor market in future years, where it will be possible to digitalize many tasks, and where there will be an increasing demand for IT specialists.

In 2018, the minister also established a number of experiments with the subject over a three-year period. Different models for technology comprehension were to be tested as an independent subject and integrated into other subjects, in much the same way as in the experiments in the 1980s, although at that time without the digital technologies that we have today. In December 2018, common goals and subject objectives were published with four areas of competence: digital empowerment, digital design and design processes, computational thinking, as well as technological knowledge and skills (EMU 2019).

In upper secondary schools, in 2015 the IT subject, informatics, was introduced as a permanent field of study and elective subject after a four-year trial period for the subject under the name information technology (Danish Ministry of Education 2017). The identity of the subject is based on abstraction and logical thinking, with an innovative approach to IT product development, which provides the basis for understanding the development and structure of information technology and its interplay with users and society. The experimental subject technology comprehension for K-9 schools has particular focus on the general educational nature of the subject, and similar to the upper secondary subject it entails that students develop computational thinking, among other things.

The concept of computational thinking was re-introduced by Professor Jeannette M. Wing (2006) as an important competence in line with reading, writing and arithmetic. In Wing's understanding, computational thinking generally involves:

... solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. (Wing 2006: 33)

In a report on the development of computational thinking in compulsory education, the authors point out that there is no agreement on the definition of computational thinking, but that it usually includes core concepts and competencies such as abstraction, algorithmic thinking, automation, breaking-down problems, troubleshooting and generalization (Bocconi et.al, 2016), and that programming is regarded as a constituent that can make computational thinking more concrete. For example, algorithmic thinking comes ahead of programming.

Wing's re-introduction has meant that an increasing number of countries at global level have developed and are developing curricula that aim to ensure that students develop computational thinking skills. The aforementioned report concludes that eleven countries in Europe (DK, FR, FI, HR, IT, MT, PL, PT, TR, UK-EN, UK-SCT) have recently undergone reforms that include integration of computational thinking and related concepts. Seven others (CZ, GR, IE, NL, NO, SE, UK-WLS) plan to integrate computational thinking into compulsory education, and yet another seven countries (AT, PT, CY, IL, LT, HU, SK) integrate computational thinking by building on long-term traditions within computer science, primarily in upper secondary schools. Some of these also extend computer science teaching to include K-9 schools (Bocconi et al. 2016). Furthermore, the *International Computer and Information Literacy Study (ICILS)* from the International Association for the Evaluation of Educational Achievement (IEA) was expanded in 2018 to include an evaluation of students' computational thinking competencies (IEA 2018).

Discussion

The purpose of this article is to explore how the subject we today call technology comprehension has developed, declined and now re-developed over time and up until today in order to derive inspirational experience that we can learn from.

Looking back on developments over the past 50 years, we can identify four periods. The first we call 'data education', and it begins in 1966 with Peter Naur's ideas about the need and content of datalogy teaching in K-9 schools. Focus here is on enabling students to think critically about the role of computers in society and to think computationally, which is more than just being able to program. Naur sowed some important seeds for a report on the subject of data education (1972), although the subject was only introduced as an elective subject for a short period in the 1980s, and in practice it took many different forms – from focusing on understanding data and the structure of the computer, to pure programming courses. Naur was forward-looking with his thoughts on data education long before the computer arrived in the classroom. For this reason, this period is also longer than the following, when developments in the area accelerated.

The following period, "operational user competences and infrastructure," began in 1990, when data education was abolished as an elective subject. In this period, focus turned to enabling students to manage what has now been established as machines that can be used in almost all areas of life. During this period, computers were generally thought of as something that needs to be integrated into all subjects and support teaching. Therefore focus was also on getting a sufficiently large number of devices and not least getting schools connected to the burgeoning internet.

The third period starts around 2000, when focus on procuring hardware continued, although now also in the form of interactive whiteboards, tablets and later robots, 3D printers, etc. During this phase, there was also strong focus on developing teaching resources and seeking methods to move away from the many isolated initiatives and projects in some schools towards integration in daily teaching at all schools. Among other things, this was through massive efforts to generate interest and a market for digital learning resources. We call this period "procurement of hardware and development of teaching resources".

In recent years, we can see an emerging new period, which we call "computational thinking and technology comprehension". This period began to develop five to ten years ago with initiatives such as Coding Pirates and FabLab@SCHOOLdk as well as IT-didactic debate in the area. In one sense, Denmark is an extension of a large number of countries that have implemented reforms and initiatives aiming to teach students how to use IT to promote creativity and innovative solutions through the development of computational thinking.

When, in 2006, Professor Jeannette M. Wing reintroduced computational thinking as an important competence in line with reading, writing and arithmetic, she had a different perspective than originally[2]. While Peter

2 Parts of this and the following section are written in an email dialogue with Peter J. Denning in 2018.

Naur encouraged all children to learn to understand and use computers, Wing suggested that they should learn to think like a computer scientist (Wing 2006). Through her work at the US National Science Foundation, she started an international movement to get computational thinking into schools. This movement is very much based on the idea that computational thinking is problem-solving like computational steps and algorithms, i.e. it clearly highlights programming. Any emphasis on programming largely excludes other areas such as artificial intelligence, data analysis, neural networks, quantum mechanics and so on, all of which depend on computational thinking with hardware, computer systems, networks, simulation and design. Computer scientists incorporate all these things, not just programming.

The movement has been under criticism from several educational researchers, especially Professor Peter J. Denning (2017), an American pioneer within computer science, who points out that the latest definitions of computational thinking feign that it all started in the modern computer age, even though many computer science methods for algorithms and machines have been a part of human history for thousands of years. Focus on programming and algorithms has created a narrow understanding of computational thinking, which has caused a number of misunderstandings about algorithms and machines with a risk that students will believe that computers can do more than they actually can, and that we will fail to distinguish between what people can do that computers cannot. Such a misunderstanding is the notion of an algorithm as any step by step instruction. This ignores the essential requirement that the algorithm has to be accurate enough for a machine to execute it without human assessment or interpretation. Another is the idea that a computer is not important in the formulation of algorithms, even though, throughout history, any computational algorithm is designed to control or direct a machine.

Denning argues that computational thinking involves mental habits and methods to find out how to get computers to do a job for us. This entails that, when solving problems, students incorporate those who are to use their programs and technologies. Users – not programmers – decide whether a solution successfully performs a job and is useful. Therefore, students should learn to listen to users and incorporate what users say in their design. Notwithstanding the importance of education in computational steps and algorithms, it is far from the computational thinking that children need.

Another important issue is the idea that the reliability of the programs depends on formal evidence. Evidence is useful when possible, but most large systems depend on many other methods for reliability. In this connection, Denning points out that: “Much of computing is not about programming but design of computations, and much of design draws from engineering rather than mathematics.” This view is consistent with Peter Naur’s philosophy. Naur had a comprehensive and inclusive view of computer science – not a narrow approach only including programming and formal structures. He said that children should learn to understand computers, not just learn programming.

Denning points out that computational thinking is a very valuable skill, and he is positive about efforts to make computer science more accessible, but with the Finnish computer science professor Matti Tedre, he warns that a lack of insight into the long and comprehensive history of the concept may lead to weaker and less ambitious versions of computational thinking and this lack of insight may trigger decline rather than progress. In other words, they say:

> *When researchers do their homework well, they know what previous generations of scientists have tried and done, and where they have succeeded and failed. They avoid ‘reinventing the wheel’ by acknowledging predecessors who built the foundations on which the current generation of researchers is now working. (Tedre & Denning: 2016)*

Perspectives

We should be aware of the foundation on which we build up the subject – and compared with international trends, it seems that Denmark differs by focusing on other aspects of what Denmark has termed “technology comprehension”, and by historically having been at the forefront with a broader general education focus.

The aim of introducing the elective subject of technology comprehension in 2017 was, among other things, to ensure that everyone is able to participate actively in democratic society, including to relate critically to algorithms. In continuation of this, the experimental program for technology comprehension now aims for students to acquire basic knowledge about networks, algorithms, programming, logical and algorithmic thinking, abstraction and pattern recognition, data modelling, and tests and testing; that they develop an understanding of design processes for complex problem-solving; and that they are taught about the importance of technology and automation for society, including developing an understanding of security, ethics and consequences of digital technology (EMU 2019).

As pointed out above, today the area should not only involve understanding traditional sequential algorithms that are performed step by step. Many computer programs are now developed using artificial intelligence, for example in the form of neural networks, built up from parallel algorithms executed simultaneously. Whereas sequential computer programs execute rules by steps, neural networks are constantly learning new behaviors through continuous input (this is referred to as machine learning). This makes it far more difficult – and often impossible – to fully grasp the consequences of neural networks. When we feed computer systems with our data, we, the users, are in fact giving the program instructions. For example, artificial translation programs learn from users how to translate a specific word in the future; location services learn how long a given route takes by car; and search engines learn what is to have highest priority. While this has created far greater opportunities, it has also spawned a large number of moral dilemmas and consequences that we cannot examine in more detail in this article.[3]

However, we can conclude that such issues have only made it even more urgent for children to develop a fundamental understanding of algorithms and data processes. And, in a certain sense, we can see a repetition of Peter Naur's exactly 50-year-old point that:

> *Power over a highly computerized system clearly [will] lie with those who understand how it works,*

and that:

> *... understanding of computer programming must be brought into general education and thus become part of the public domain. ... There's no way around it, we all have to understand computers. (Naur 1968: 32, our translation)*

It will be interesting to see whether these ideas are realized in Denmark after the ongoing experiment – or whether the important ambitions flounder once again in a ministerial reshuffle or a change of government.

References

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., Engelhardt, K. (2016). *Developing computational thinking in compulsory education – Implications for policy and practice*. European Commission, Joint

[3] This major issue is dealt with in Caeli & Bundsgaard, 2020.

Research Centre. http://publications.jrc.ec.europa.eu/repository/bitstream/JRC104188/jrc104188_computhinkreport.pdf

Bundsgaard, J.; Petterson, M. & Puck, M. R. (2014). *Digitale kompetencer. It i danske skoler i et internationalt perspektiv.* Aarhus Universitetsforlag

Bundsgaards, J. (2017). Fagdidaktik og it. *Learning Tech – Tidsskrift for læremidler, didaktik og teknologi*, (2): 6-31.

Caeli, E. N. & Bundsgaard, J. (2020). Teknologikritik i skolen – et demokratisk perspektiv på teknologiforståelse. In: Haas, C. og Matthiesen, C. (Eds.). *Fagdidaktik og demokrati*. Samfundslitteratur.

Dalgaard, L. (1994). Edb i undervisningen i folkeskolen. *Nytt om data i skolan*, nr. 1: 28-31.

Denning, P. J. (2017). Viewpoint. Remaining Trouble Spots with Computational Thinking. *Communications of the ACM*, 60(6): 33:39

EMU (2018). *Teknologiforståelse valgfag (forsøg) – Fælles Mål og læseplan*. https://www.emu.dk/modul/teknologiforst%C3%A5else-valgfag-fors%C3%B8g-%E2%80%93-f%C3%A6lles-m%C3%A5l-og-l%C3%A6seplan

Frandsen, K. (1983) (Ed.) *EDB i skolens undervisning.* Danmarks Skolelederforening.

Hansbøl, M. & Mathiasen, H. (2003). *Junior PC-kørekort. Forskningsrapport ITMF-Projekt 373*. Danmarks Pædagogiske Universitets Forlag.

Hansen, K. F. & Jensen, P. E. (1986). *Informationsteknologi og skole. Status og udviklingslinjer*. Danmarks Pædagogiske Institut. Munksgaard.

Hansen, T. H. (2012). Kritik: Københavns Kommune spilder 21 millioner på interaktive tavler. *Version 2*. https://www.version2.dk/artikel/koebenhavns-kommune-bruger-21-millioner-paa-fiasko-skole-it-46203

Hansen, T. I., & Bundsgaard, J. (2016). *Effektmåling af demonstrationsskoleforsøg: Afrapportering af kvantitative undersøgelser på tværs af de tre demonstrationsskoleprojekter i AUUC-konsortiet.* Læremiddel.dk.

Hjort Jensen, H. (1986). Datalære – også for piger: det kan være en fordel at dele klassen op i rene drenge- og pigegrupper i faget datalære. *Folkeskolen.* 103, 19: 808-809.

IEA (2018). *ICILS. International Computer and Information Literacy Study.* http://www.iea.nl/icils

Jacobsen, J. (2001). Pc-kørekort til skoleelever. *Folkeskolen.dk*, April 5, 2001. https://www.folkeskolen.dk/13030/pc-koerekort-til-skoleelever

Knuth, D. E. (1974). Computer Science and its Relation to Mathematics. *The American Mathematical Monthly*, vol. 81: 323-343

Malmberg, A. C. (1970). *Datalogi i skolen: Læreruddannelsen – en flaskehals.* Undervisningsministeriets tidsskrift, 272-276.

Mortensen, H.N. (2012). *Ingen dokumenteret effekt: Skoler køber iPads i blinde for millioner*. Version 2, September 18, 2012. https://www.version2.dk/artikel/ingen-dokumenteret-effekt-skoler-koeber-ipads-i-blinde-millioner-47792

Naur, P. (1967). Datamaskinerne og samfundet. *Søndagsuniversitetet – Bind 85.* Munksgaard.

Naur, P. (1968). Demokrati i datamatiseringens tidsalder. *Kriterium*, 3, 5, June 1968. Nyt Nordisk Forlag Arnold Busck.

Naur, P. (1954). Elektronregnemaskinerne og hjernen. *Perspektiv* 1 (7): 42-46.

Naur, P. (1966). *Plan for et kursus i datalogi og datamatik.* Regnecentralen.

Naur, P. (1965). *The Place of Programming in a World of Problems, Tools, and People.* Proc. IFIP Congress 65: 165-199.

Papert, S. (1980, 1993). *Mindstorms. Children, Computers, And Powerful Ideas.* Basic Books.

Pea, R. & Kurland, M. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology 2*, 2, 137-168.

Skole og edb (1985). Vejledende forslag til læseplan for valgfaget datalære i folkeskolen. *Skole og edb*, 1985: 149-152

Sveinsdottir, E. & Frøkjær, E. (1988). Datalogy – The Copenhagen Tradition of Computer Science. *BIT Numerical Mathematics,* 28 (3), 450–472.

Tedre, M. & Denning, P. J. (2016). The Long Quest for Computational Thinking. *Proceedings of the 16th Koli Calling Conference on Computing Education Research*, November 24-27, 2016, Koli, Finland: 120-129.

Undervisningsministeriet (1972). *Betænkning om edb-undervisning i det offentlige uddannelsessystem. Betænkning nr. 666.* Undervisningsministeriet.

Undervisningsministeriet (2018a). *Handlingsplan for teknologi i undervisningen.* https://www.stil.dk/-/media/filer/uvm/udd/fgu/180201-nyhandlingsplan-for-teknologi-i-undervisningen-februar-2018.pdf?la=da

Undervisningsministeriet (2018b). *Indsatsen for øget anvendelse af it i folkeskolen.* https://www.uvm.dk/folkeskolen/laering-og-laeringsmiljoe/it-i-undervisningen/oeget-anvendelse-af-it-i-folkeskolen

Undervisningsministeriet (2017). *Informationsteknologi C og B.* https://www.uvm.dk/gymnasiale-uddannelser/fag-og-laereplaner/laereplaner-2013/forsoegsfag-i-de-gymnasiale-uddannelser/permanente-forsoegsfag/informationsteknologi-c-og-b

UNI-C. *På forkant i 40 år. Jubilæum 1965-2005*. https://www.yumpu.com/da/document/view/17696419/historisk-tilbageblik-unioc/5

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM*, 49(3), 33-35. https://doi.org/10.1145/1118178.1118215

Appendix 1. IT in K-9 schools 1966-2018

Year	Event	Focus
1966	Professor Peter Naur introduces “*datalogi”* as an alternative Danish term for the English term *computer science*. He formulates “*datalogi”*/datamatics as a cross-disciplinary tool in line with language teaching and mathematics, and predicts that it will take a similar position in the educational system.	Data education
1968	The Danish School of Education (*Danmarks Lærerhøjskole*) starts computer teaching for mathematics teachers, focusing on problem-solving.	
1970	Allan C. Malmberg, head of the Danish School of Education (*Danmarks Lærerhøjskole*), says that teacher training should be taken into account in the development of a computer capacity in teaching and research in all subjects at a completely different scale than previously in order not to become a bottleneck for the subject.	
1972	In Report No. 666, a committee presents an analysis of data education as an important subject in K-9 schools.	
1973	A committee prepares a draft of teaching instructions for data education.	
1975	New school act, but for inexplicable reasons data education has been dropped, so it is only an elective subject for 10th grade (16 year-olds).	
1976	The Data Education Association (*Datalæreforeningen*) is founded (now known as the Danish Association of IT Supervisors (*Danmarks IT-vejlederforening*)).	
1976-1986	A wide range of experimental and development work in the area: in subjects, as subjects, as courses, girls-boys. Funded by the Danish schools’ trials council (*Folkeskolens Forsøgsråd*) and the schools’ own initiatives. In 1986 more than 3,000 pages of reports were written in collaboration with the Danish schools’ trials council (*Folkeskolens Forsøgsråd*).	
1980	In his opening speech in the Danish Parliament in October, Prime Minister Anker Jørgensen says that “we expect to introduce data education in this parliamentary year.”	
1981	The Educational Council for Danish K-9 schools sets up the Haase Committee to prepare a report on data education in K-9 schools.	
1982	The Haase Committee concludes that data education should be introduced in K-9 schools as an independent subject, but when they are to present their analysis to the Educational Council, Anker Jørgensen loses his position as prime minister and the meeting is cancelled. Bertel Haarder becomes the new Minister for Education and shuts down the Educational Council. The Committee’s report is never published.	
1983	Minister for Education, Bertel Haarder, appoints a new committee to continue work on the plans for data education: description of objectives, teaching supervision and curriculum for an elective subject, as well as suggestions for how it can be included in existing courses.	
1985	Indicative proposals for a curriculum for the elective subject of data education published.	
1984-1990	Data education offered as an elective course in 8th-10th grades (14-16 year-olds).	
1985	The calculation centers at the University of Copenhagen (RECKU) and Aarhus University (RECAU) are merged and become UNI-C (with the double meaning "University center" and "unique"). Since then, the teacher training college IT center	

	(SITC) as well as the county and municipal organization ORFEUS have also become part of UNI-C.	
1986	Bertel Haarder's new committee presents its analysis.	
1990-1994	Data education is withdrawn as an elective subject and is changed to a compulsory section-6 subject in line with road safety and sex education.	Operational user competencies and infrastructure
1993	New Danish act on basic education. Minister for Education, Ole Vig Jensen, approves data education as an elective course for the 8th-9th grade (14-15 year-olds), but data education is lost from the act as an elective subject and as a compulsory subject. The comments on section 7 state that "EDP (...) has been withdrawn as a compulsory subject, as it is assumed that the content will be integrated into the compulsory subjects for the youngest year groups."	
1993-1994	Denmark's first internet-based network, *Sektornettet*, was established with the aim to connect the entire educational system to the Internet.	
1996	PC user certificates are introduced, including a PC user certificate for teachers, and a couple of years later, a junior PC user certificate for K-9 school students.	
The mid-1990s	Several projects are launched. For example, the Centre for technology-supported teaching (*Center for Teknologistøttet Undervisning*) is granted DKK 100 million to allocate to projects on IT, a toolbox for IT in teaching (Poseidon) is established, and the schools' database service (SkoDa) is set up.	
2000	Almost all schools and educational institutions are connected to the Internet through the sector network (Cisco Systems Denmark).	
2001	IT and Media in K-9 Schools (ITMF) is launched with a budget of DKK 323 million to support development projects in a collaboration between schools and researchers, production of a media library, continuing education for teachers and other minor initiatives.	Procurement of hardware and development of teaching resources.
2003	Common goals (*Fælles Mål*) replaces Clear goals (*Klare Mål*).	
2004-2008	The Danish Ministry of Education initiates IT in K-9 Schools (ITIF), with a budget of DKK 750 million, which primarily consists of support to procure computers for 3rd-grade students (9 year-olds) and to develop six subject-specific digital teaching resources.	
2009	Fælles Mål is revised, and Text Booklet 48, IT and media competences in K-9 schools is published.	
2009	Digital national tests introduced.	
2011-2012	City of Copenhagen procures interactive whiteboards for all schools. Municipality of Odder procures iPads for all teachers and students. Many municipalities follow suit over subsequent years with large hardware purchases.	
2014-	Trials with digital examinations in several subjects.	
2011-2017	A new initiative is launched: initiative to increase the use of IT in K-9 schools. DKK 500 million from the government and DKK 500 million from municipalities are earmarked to upgrade broadband at schools and co-finance procurement of digital learning resources. The project consists of support to procure digital learning resources, impact measurement of the use of digital teaching resources in schools,	

	establishment of demonstration school pilot projects, development of digital teaching resources and a teacher network.	
2013	UNI-C transfers the Danish internet institutions DiX'en, DK-CERT and the research network (*Forskningsnettet*) to DTU as part of a reorganization, in which, with its position under the Ministry of Children and Education, UNI-C is to focus on IT tasks associated with this area.	
2014	UNI-C changes name to STIL (Agency for IT and Learning) due to the changed focus on supplying IT solutions for K-9 schools and youth education without a commercial aim. The most important goal for the agency is to support IT in teaching (solutions), while procurement and implementation are a municipal task.	
2014	New school act with simplified common goals under Minister for Education Christine Antorini. *IT and media* introduced as cross-disciplinary themes, and specific academic IT goals are incorporated into all courses. Text Booklet 48 is withdrawn.	
2016	Conclusions from the demonstration school trials suggest that the way IT is integrated in K-9 schools is traditional and directed by conservative logics – but that IT has innovative opportunities.	Computational thinking and technology comprehension
2016	After Professor Jeannette M. Wing reintroduces the concept of computational thinking, a large number of countries undergo reforms or implement initiatives that include the integration of computational thinking, including Denmark.	
2017	Minister for Education, Merete Riisager initiates the elective subject technology comprehension for final-year classes as an experiment. Thirteen participating schools.	
2018	The *International Computer and Information Literacy Study (ICILS)* from the International Association for the Evaluation of Educational Achievement (IEA) is expanded to include an evaluation of students' computational thinking skills (IEA 2018).	
2018	Minister for Education, Merete Riisager initiates trials to incorporate the discipline of technology comprehension in other subjects and as an independent subject at 46 schools over a period of three years.	

Education Tech Research Dev
https://doi.org/10.1007/s11423-019-09694-z

CULTURAL AND REGIONAL PERSPECTIVES

Computational thinking in compulsory education: a survey study on initiatives and conceptions

Elisa Nadire Caeli[1] · **Jeppe Bundsgaard**[1]

Abstract
This article communicates the results of a Danish survey study conducted in 2018 that aimed to examine initiatives relating to computational thinking in primary and lower-secondary schools, as well as the professional development of teachers and the perceptions of school principals in this area. The context is an increasing interest in this field, motivated by a sense that it is important for children to learn computational thinking skills. However, educators struggle with questions regarding what computational thinking in education actually is—and consequently, how they should teach and assess it. In this survey, we wanted to explore existing practices and current situations to find out what school principals regard as important; thus, we designed an electronic questionnaire on this topic. 98 principals started the survey, and 83 completed it. Our analysis suggests that many initiatives connected to computational thinking are currently being implemented, but according to the principals taking part, teachers are not trained to teach this subject. The principals have inclusive views and focus on broad aspects of what computational thinking involves. According to them, computational thinking is not about pushing students into computing careers; rather it is about supporting the well-rounded development of human beings in a free and democratic society. However, the principals do report limited understanding of this subject, which suggests that teachers are not the only ones in need of training—principals also need help to develop a culture and mindset around this subject and implement it efficiently into schools.

Keywords Computational thinking · Technological understanding · Computing curriculum · Compulsory education

Introduction

There is increasing interest in the world today in the idea of embedding computational thinking in primary and secondary education. In Denmark, the Ministry of Education recently expressed their ambition to establish technological understanding (*teknologiforståelse*) as a compulsory subject for K-9 students (Undervisningsministeriet 2018b). The

Elisa Nadire Caeli
elisa@edu.au.dk

[1] Danish School of Education, Aarhus University, Tuborgvej 164, 2400 Copenhagen NV, Denmark

motivation for this move is a sense that all students need to develop computational thinking skills to participate fully in the highly digitalized world of today. For a three-year trial period, the ministry now sponsors various approaches to teaching this subject.

Eleven countries in Europe have recently launched reforms that include the integration of computational thinking into compulsory education, and more are planning to do the same (Bocconi et al. 2016). In the US, leading educational organizations such as the Computer Science Teachers Association (CSTA), and the International Society for Technology in Education (ISTE) have argued for the need to expose students to computational thinking ideas, and computational thinking is included in the Next Generation Science Standards (NGSS 2013). For example, ISTE wants students to "develop and employ strategies for understanding and solving problems in ways that leverage the power of technological methods to develop and test solutions" as part of "enabling them to engage and thrive in a connected, digital world" (ISTE 2016).

In a review published in 2013, Grover and Pea framed the current state of discourse on computational thinking in K-12 and found a number of elements that were "widely accepted as comprising CT and form the basis of curricula that aim to support its learning as well as assess its development" (Grover and Pea 2013). These include abstractions and pattern generalizations (including models and simulations); systematic processing of information; symbol systems and representations; algorithmic notions of flow of control; structured problem decomposition (modularizing); iterative, recursive, and parallel thinking; conditional logic; efficiency and performance constraints; and debugging and systematic error detection.

But technology and automation are changing rapidly, and we need to keep examining shifting views and priorities in K-12. In addition, despite increased attention and a variety of curriculum designs and other initiatives, there is still little common understanding of what computational thinking is, how it should be taught, and how it can be assessed. By emphasizing the importance of programming, the current movement and definitions have produced narrow conceptions that have led to less ambitious versions of computational thinking than in the past, and to unsubstantiated claims of what it is good for (Denning 2017; Tedre and Denning 2016).

Historical perspectives regarding computer science education

Pioneers of computer science have been discussing the power of computational thinking for decades, stressing the value of teaching fundamental computing principles from early primary school. To identify precursors of computational thinking in Danish education, we examined how different types of discussions of computing in schools have been carried out in Denmark since the middle of the 1960s (Caeli and Bundsgaard 2019). It is interesting to note that current discussions on this topic were also present in the past, though with different emphasis. In Denmark, for instance, Professor Naur (1966, 1968, 1992) has discussed the need for students of all ages to be able to think critically about how computers influence life and society. He proposed a subject for compulsory education that involved not only the programming of computers, but also other skills and competences, such as computational designs and understanding algorithms. Additionally, pioneers back in the 1960s and 1970s argued for the importance of training pre-service and in-service teachers to teach such competences. They foresaw a risk that the failure to train teachers in this way would make it harder to implement computing initiatives in schools (Malmberg 1970; Undervisningsministeriet 1972).

For policy-related reasons, *datalogy* (computer science) was never implemented in the curriculum. During the 1990s, the focus changed towards a more user-oriented perspective and lower-level skills. For example students being able to use computer functions such as how to turn on and off the machine, how to save work, and how to use functions in word processors or spreadsheets. Decision makers in education focused on developing computer labs with a powerful infrastructure. During the 2000s, municipalities were occupied with buying hardware, especially interactive whiteboards and tablets. However, teachers were not educated in integrating these devices into teaching practices, and many questions about their relevant, meaningful use were raised and discussed.

Computational thinking and curriculum design in Denmark

This new period of computational thinking and technological understanding has been emerging worldwide, although it is based on a new and rather narrow conception of what computational thinking involves (Denning 2017). The proposed curriculum design in Denmark seems to take into account more inclusive views regarding which computing principles are relevant for children to learn today. Specifically, the Danish experimental subject of technological understanding aims to develop competences within four areas:

- Digital empowerment: the ability to analyze technologies and their purposes, examine their use, and assess their consequences.
- Digital design and design processes: framing, generating ideas, constructing, and assessing digital designs.
- Computational thinking: knowledge of data, algorithms, structuring, and modeling.
- Technological capability: capability with regard to computer systems, networks, programming, and security (EMU 2019).

As such, technological understanding focuses not only on formal programming skills, closed-ended solutions, and concepts related to math, but also on design and engineering skills as well as critical understanding.

This probably reflects the formal aims of the Danish *Folkeskole*, as stated by the Danish government, concerning promoting the well-rounded development of the students. Specifically, and among other things, the aim of the *Folkeskole* is:

> to prepare the students to be able to participate, demonstrate mutual responsibility and understand their rights and duties in a free and democratic society. The daily activities of the school must, therefore, be conducted in a spirit of intellectual freedom, equality and democracy (Undervisningsministeriet 2018a).

There are no formal training courses for *teachers* when it comes to teaching this new subject. Therefore, we hypothesized challenges in teaching computational thinking skills. Recent school reforms have downsized the amount of preparation time allowed to Danish teachers, which might have left teachers with insufficient time to become competent and confident with the content and teaching of this new subject. This lack of preparation time has been discussed a great deal, being regarded as one of several factors that might have caused an increasing number of teachers to experience stress owing to high pressure (see for example Schäfer 2018.).

Purpose of this study

In the following, we present the results of a survey study, conducted and analyzed in 2018. The purpose of the study was:

- to identify the types of technology initiatives involving computational thinking existing in Danish schools today and their frequency.
- to examine Danish principals' perceptions as to whether teachers have the requisite professional skills to teach this subject.
- to analyze the ideas of Danish school principals regarding computational thinking, for example whether they find it relevant in compulsory education, and what challenges they anticipate in terms of embedding it in school practices.

Other surveys have looked into *teachers'* conceptions of computational thinking. For example, a survey study measured the attitudes of pre-service teachers and examined how introducing computational thinking into training courses could influence the understanding of pre-service teachers regarding computational thinking (Yadav et al. 2014). Another study measured the understanding of in-service teachers regarding computational ideas for teaching science and examined how the ideas of computational thinking could be embedded within elementary subjects (Yadav et al. 2018).

Danish primary/lower-secondary school and school administration

Denmark has 10 years of compulsory education, which is called the *grundskole* (kindergarten, primary and lower-secondary school, grades 0–9). Most schools are public (*folkeskoler*, owned by the municipalities), but around a fifth of the students go to private schools (religious, Freinet, Waldorff, etc.). Schools are administered by a principal, a vice-principal, and heads of departments (for example, a head of department for grades 1–3, grades 4–6, and grades 7–9 respectively). The Danish Ministry of Education stresses that the school administration is of great importance for students' learning (Undervisningsministeriet 2015), and the Danish Union of Teachers (DLF) emphasizes that the Folkeskole plays a pivotal role in the development of Danish democracy, communication and development of culture and opinions, and socialization and development of common values (DLF n.d.).

As part of the latest school reform, a number of Danish parties agreed to improve standards in the Folkeskole to "maintain and develop the public school's strengths and academic standards". In order to fulfil their goals, they concluded that "an enhanced professional development of teachers, pedagogical staff and school principals" was necessary (Undervisningsministeriet 2013).

The Danish Evaluation Institute EVA (2015) identified four elements of effective school administration with the purpose of developing learning opportunities for all: facilitate learning processes and school direction; develop a professional culture and build up trust; support employees during changes; and include both experienced and research-based work.

Teachers' understanding and preparedness are essential when introducing and developing a new subject; however, principals also play a central role in ensuring the success of such initiatives. Hence, in our study we decided to measure the ideas of principals in order to examine the way computational thinking is taught in schools more broadly and the way in which principals support such initiatives.

Method

Research design

The study was designed as a survey based on an electronic questionnaire ("Appendix") which was sent to the principals of K-9 schools.

It consisted of three parts with closed-ended single and multiple-choice questions:

- Part 1: Technology initiatives in schools (1–2 questions)
- Part 2: Teachers' professional development (1–2 questions)
- Part 3: Conceptions of computational thinking (6 questions)

Part three of our survey mainly consisted of four-point Likert scale questions (Strongly Agree, Agree, Disagree, and Strongly Disagree) to measure how much principals agreed or disagreed with various statements. We chose a forced opinion scale with no neutral option to make respondents less likely to make fake and faster responses by choosing the undecided alternative.

Research questions

To measure the extent to which Danish schools focus on computational thinking, we asked principals about current and planned learning activities in terms of teaching computing initiatives to students, and the extent to which their teachers had the professional skills needed to teach such initiatives.

In addition, to capture different conceptions of computational thinking definitions and different arguments for embedding it into curriculum, we examined participants' ideas about what computational thinking involves and why it is an important skill for students to develop. The options in this section were designed to cover the most commonly heard arguments. Specifically, policy documents, research articles and debates regarding the idea of teaching computer science to everyone have informed our statements.

In particular, we wanted to study:

- Question 1: To what extent do Danish K-9 schools offer specific technology initiatives or subjects that involve computational thinking?
- Question 2: To what extent are their Danish teachers trained to teach computational thinking from the perspectives of Danish principals?
- Question 3: To what extent are Danish principals familiar with computational thinking; and what are their perspectives on embedding it in compulsory education?

To assess possible problems or obstacles in our survey, we asked four pilot principals to pre-test our questionnaire and give feedback on its clarity and length, as well as sharing all the comments that occurred to them during testing. We adjusted our final version to incorporate their comments.

One principal thought that it would be good to add a question on familiarity to find out whether answers were based on guessing or some degree of knowledge. This made us add question *3. To what extent are you familiar with the term computational thinking?* (see "Appendix") as an important part of the survey.

Table 1 Number of principals responding

Q1	*Q1A*	Q2	*Q2A*	Q3	Q4	Q5	Q6	Q7	Q8
N=98	N=*62*	N=94	N=*76*	N=89	N=87	N=87	N=84	N=84	N=83
Part 1: Initiatives		Part 2: Teacher PD		Part 3: Principals' conceptions					

Italics indicates that it depended on the previous answer whether questions 1A and 2A were given or not

Another principal suggested adding more options to question *1A. What type(s) of specific technology initiatives is your school offering in the 2017/2018 academic year?* For example, at his school, students worked with robots and he needed an option for this initiative. Similarly, he thought it would be good to distinguish between *Technology training integrated in one or more subjects* and *Another optional subject/specific technology course*. We adjusted question 1A according to his comments.

Data collection and participants

The study was a Danish supplement to the International Computer and Information Literacy Study (ICILS). By means of systematic and random stratified cluster sampling, 145 participants were randomly selected to represent Danish schools with grade-eight students.[1]

We specifically asked *principals*, and not for instance teachers, to capture the overall initiatives and preparedness of schools with regard to embedding computational thinking into their practice. In addition, we asked them about their knowledge and perceptions regarding computational thinking, as we consider their views important when it comes to implementing computational thinking in the culture and daily practices of their schools.

The survey was distributed electronically on January 30, and data was collected until February 25, 2018. 145 people received the questionnaire (97 men and 48 women). 98 respondents started the survey (68% participation), and 83 completed it (a dropout rate of 15%) (see Table 1). The participants who completed the survey included principals from all Danish regions with no significant differences in terms of gender, school size, or achievement scores.

We used the information completed on incomplete questionnaires in our data analysis below.

Data analysis and results

In the following, we report data on frequencies and correlations by analyzing each part of the questionnaire individually and summarizing our conclusions.

Technology initiatives in schools

In the first part of the questionnaire, we asked principals whether their school offered any specific technology initiatives, either integrated into existing subjects or in the form of separate courses or subjects, in the 2017/2018 academic year.

[1] The sampling method is specified in Jung and Carstens (2013).

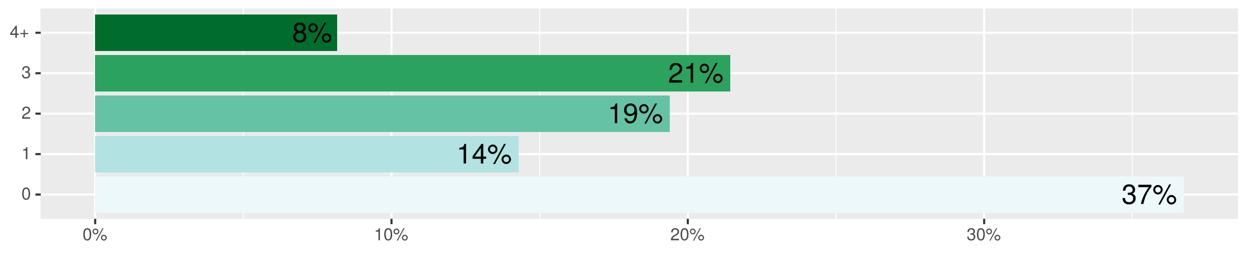


Fig. 1 Technology initiatives in the 2017/2018 academic year (Texts and sentences are shortened in Figs. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. The full text is available in "Appendix")

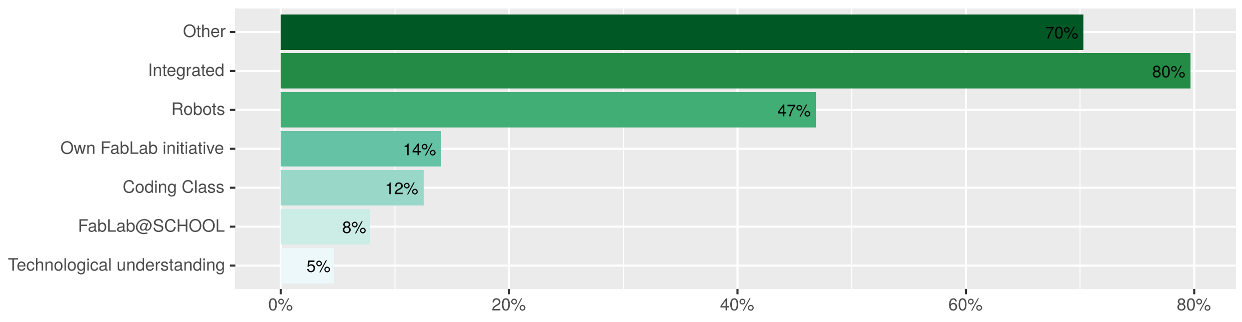


Fig. 2 Types of IT initiatives in the 2017/2018 academic year

98 responded to the question, with 62% offering one or more technology initiative and 37%[2] not offering any (Fig. 1).

We asked principals about technology initiatives, and not explicitly computational thinking, because the term "computational thinking" has not yet been embedded into the school curriculum. For example, as stated in our introduction, Denning (2017) argues that teachers struggle to understand what computational thinking is, and we hypothesized that a number of principals would either have little knowledge of the concept or would not have heard of it yet.

Instead, as a next step we examined the content of these initiatives to identify indicators of possible computational thinking content. We asked principals who said they had one or more specific technology initiative about the type of initiatives their school offered in the 2017/2018 academic year.

The different types of initiatives were selected by examining main technology initiatives in Danish K-9 schools[3] (Technological Understanding, FabLab@SCHOOLdk, and Coding Class). Subsequently, we added options regarding other initiatives that focused on different aspects of technology in education, in particular those which we believed might involve computational thinking (such as robotics, programming, and digital design/fabrication). 62 respondents answered this question (Fig. 2).

Most schools with technology initiatives offered initiatives which we had not mentioned as a specific option (80%), or had technology initiatives integrated into one or more subjects (70%).

We asked participants to specify what these initiatives encompassed. Their answers included Unity Coding, game design, makerspace, digital fabrication and design thinking, e-sports, media literacy, information technology, Appwriter (for dyslexics), and

[2] The figures have been rounded off, so the total is not always 100%.

[3] Danish compulsory education.

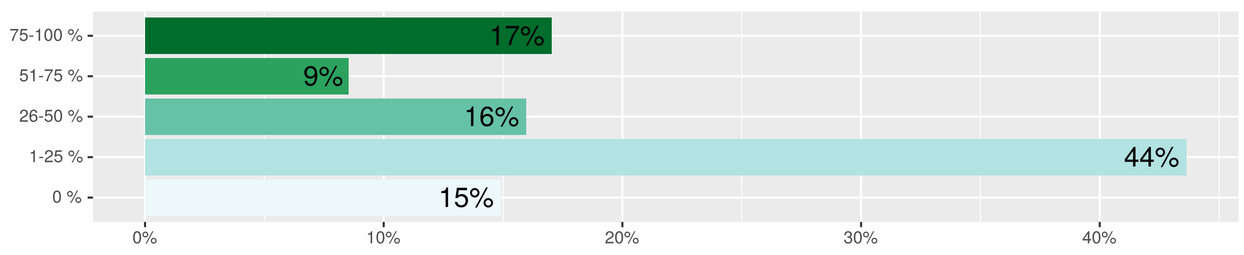


Fig. 3 Teachers' PD in teaching technology initiatives

digital music production. Integrated technology initiatives included Lego Wedo, Lego Mindstorms, Lego Education, math coding, coding, Raspberry Pi, common tools (Google docs, GeoGebra etc.), gaming, Kubo, drones, Beebots, and various aspects of technological understanding.

These results indicate that some of the principals had rather diverse understandings of a "specific technology initiative". In our understanding, common computerized tools such as Google Docs or GeoGebra, as well as specific technologies to support students with dyslexia (for instance), do not constitute specific initiatives unless the technology in itself was the primary focus.

Analysis and conclusions regarding technology initiatives in schools

Our conclusion regarding this first part of the study is that a large number of schools have integrated technology into their teaching (62%). A closer look at specific initiatives suggests that several focus on programming, and some on design.

A comparison of our data to the formal aims of the upcoming subject known as "technological understanding" (see introduction) indicates that according to the principals, current initiatives seemed to prioritize some competence areas over others, specifically those relating to digital designs and technological capability. What is not clear, however, is whether digital empowerment and computational thinking principles are embedded in these initiatives, even though they may not be the primary focus.

Teachers' professional development (PD)

In the second part of the questionnaire, we initially examined the number of teachers who according to the principals had undergone teacher training courses to teach technology initiatives in the classroom (Fig. 3). 94 responded to the question.

The principals reported that 15% of the schools did not have any teachers with professional development in this area. In many of the remaining 86% of the schools, the principals reported that only 1–25% of their teachers had been engaged in professional development (44%); whereas in 42% of the schools, the principals said 26% or more had done so.

Furthermore, we wanted to explore what kind of skills the teachers actually possessed, so we asked the principals who said that at least one of their teachers had been engaged in professional development. We identified nine different types of courses in the field of

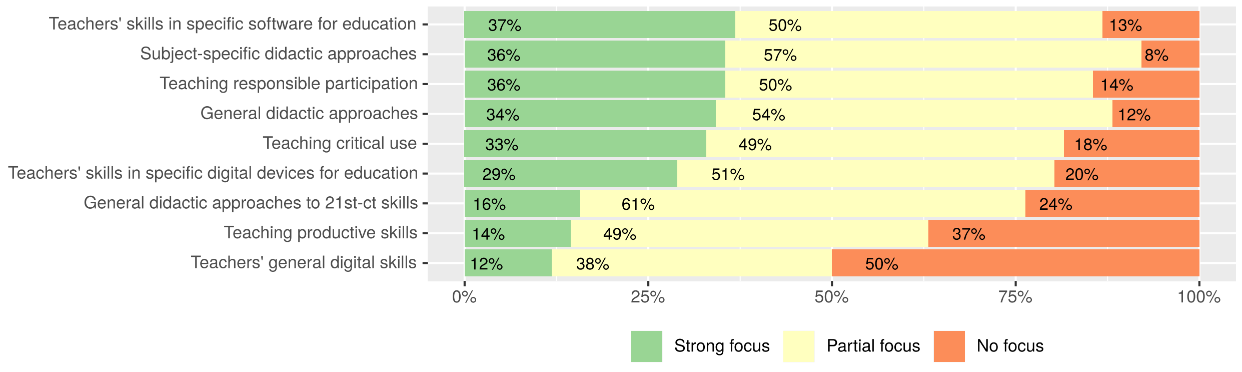


Fig. 4 Teachers' PD focus

computer science education and didactics[4] (Fig. 4, see full version in "Appendix"), and presented them to the principals in a randomized order.

The principals reported that teachers were mostly trained to use specific software for education (37% courses with a strong focus) and in didactic approaches to the use of technology (36% with a strong focus on subject-specific didactic approaches and 34% with a strong focus on general didactic approaches). In addition, they had been trained to teach critical and social online competences (responsible participation—36% and critical use—33%). There seemed to be less focus on teachers' digital skills in using specific digital devices for education (29%) and on common digital skills (12%). 16% of the courses focused strongly on general didactic approaches to teaching twenty-first century skills, and only 14% were trained to teach students productive skills.

Analysis and conclusions regarding teachers' skills

According to the principals, 62% of the schools in our survey offered one or more technology initiative; therefore, the number of teachers who engaged in professional development to teach these initiatives in schools is relatively small.

We do not know whether the teachers who teach these initiatives are the ones who have the requisite training, and there is a risk that the principals did not know whether individual teachers had recently attended training courses, for example online training on their own. This analysis focuses on how many teachers were engaged in professional development, and there is no guarantee that the professional development of teachers leads to the acquisition of skills.

The fact that the principals report that many teachers were trained to use specific technologies might reflect an aspiration that teachers should use the hardware in which Denmark has invested so much money. We regard the focus on didactics—and not on technology alone—as positive. This might follow from heavy debates on the importance of pedagogical aspects as drivers of education and not technology. Furthermore, the focus on the importance of knowing how to teach critical and social online competences might be due to worldwide debates on children's participation in social media and a focus on the sweeping presence of *fake news*. The fact that teachers (according to the principals)

[4] In this article, the term *didactics* is based on Nordic educational traditions, defined as knowledge and skills relating to teaching processes, e.g. professional factors concerning learning intentions, students' learning process, settings, conditions, content, and assessment. We distinguish between *general* and *subject-specific* didactics.

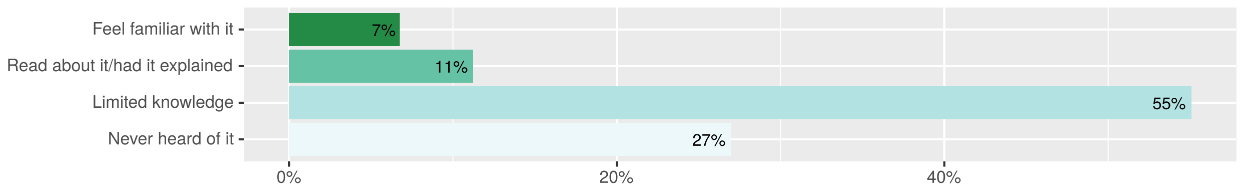


Fig. 5 Principals' familiarity with CT

were not trained to use specific digital devices or to develop common digital skills may be due to the fact that most people today already possess such low-level functional skills. The absence of focus on training teachers to teach students productive skills conflicts with the fact that many of the technological initiatives that have been launched focus on production in terms of programming and design.

Conceptions of computational thinking

While the first and second parts of the questionnaire addressed initiatives related to technology initiatives in education more broadly, this last part concentrated on computational thinking. Specifically, it addressed the ideas of school principals regarding computational thinking within the six categories: *Familiarity*, *Definition*, *Importance*, *Relevance*, *Challenges*, and *Advantages*. In the following, we present data from these six categories individually and sum up with an analysis and conclusions.

Familiarity

To assess principals' knowledge of computational thinking, we asked to what extent they were familiar with the term (Fig. 5).

27% had never heard of computational thinking, whereas 73% had various degrees of knowledge of the term. Specifically, 55% reported that their knowledge was limited, 11% felt they had some degree of knowledge, and 7% thought they had a good sense of what computational thinking involved.

The following results regarding *Definition*, *Relevance*, *Challenges*, and *Advantages*, were derived from the principals who thought that they had at least some degree of familiarity with the term. We reasoned that if the principals were not familiar with the term "computational thinking", their responses to subsequent questions could not be valid.

Definition

We asked principals whether they agreed or disagreed with eight different statements about what computational thinking involved (Fig. 6).[5] The results are presented below, with the highest degree of agreement on the left and the highest degree of disagreement on the right. While blue and green colors represent agreement with the statement, orange and red

[5] Questions related to Figs. 6, 8, 9 and 10 were all given in a randomized order of the items on a four-point Likert scale: Strongly Agree, Agree, Disagree, and Strongly Disagree.

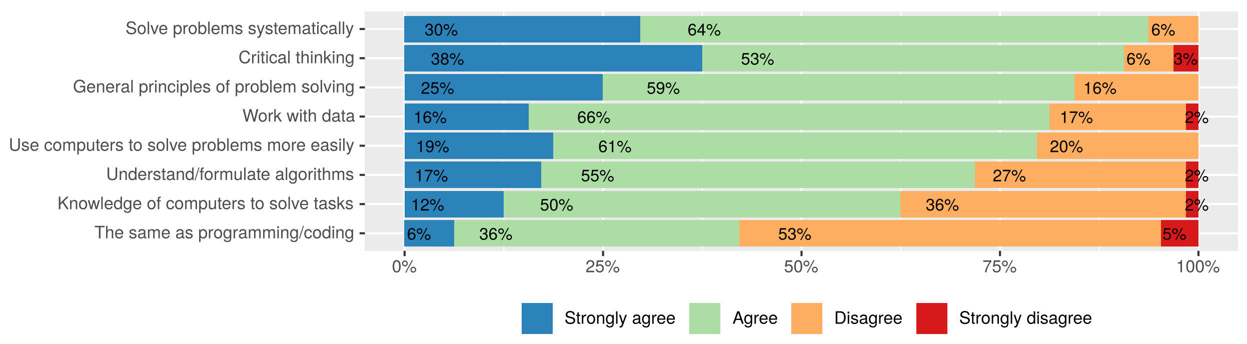


Fig. 6 Principals' definition of CT [In Figs. 6, 8, 9 and 10, the answers are sorted in an increasing order ranging from the bottom (disagree and strongly disagree) to the top (agree and strongly agree).]

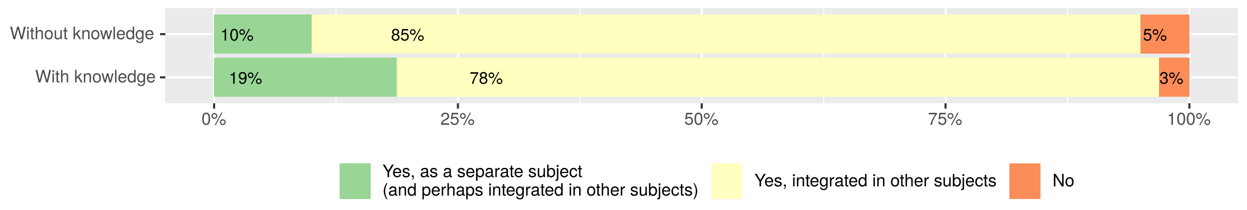


Fig. 7 Principals' views on the importance of teaching CT as a subject or not

represent disagreement. Thus, we combine the Strongly Agree and Agree totals, categorizing them as "agreed"; and we combine the Strongly Disagree and Disagree totals, categorizing them as "disagreed".

In general, the answers were characterized by a high degree of agreement. The point on which there was most agreement was that computational thinking involves solving problems systematically (94%), while most of the principals (58%) disagreed that computational thinking is the same as programming and coding.

Three categories are evident: the principals agreed the least on computational thinking as being equal to basic technical skills and knowledge (*the same as programming/coding* and *knowledge of computers*). Moreover, they agreed less on subject-specific definitions (*understand/develop algorithms*, *using computers to solve problems more easily* and *work with data*) than on definitions including more demanding mental skills such as critical thinking and problem-solving approaches (*general principles of problem solving*, *critical considerations* and *solve problems systematically*).

Importance

Secondly, we wanted to find out whether principals think that students in compulsory education should be taught computational thinking as a separate subject, taught as part of existing subjects, or not taught at all. In our analysis, we differentiated between principals with knowledge of computational thinking and principals who have never heard of computational thinking (Fig. 7).

The general picture is the same whatever degree of knowledge principals have: most of the principals thought that computational thinking should only be integrated into existing subjects and not offered as a stand-alone subject. However, principals who had knowledge of

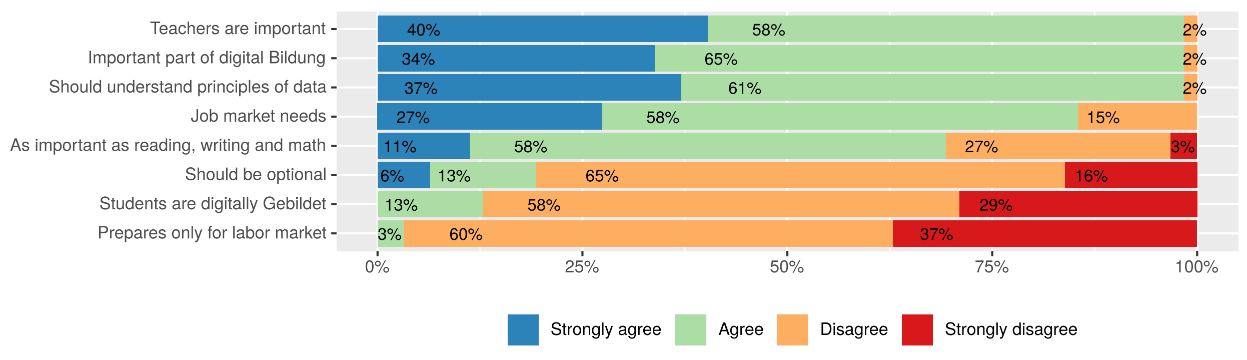


Fig. 8 Principals' attitudes towards the relevance of teaching CT

computational thinking tended to have a more positive attitude towards a subject: 19% with knowledge were positive compared to only 10% without knowledge.

Relevance

To elaborate on their thinking, we asked principals to respond to eight different statements about the relevance of teaching computational thinking in compulsory education, asking whether they agreed or disagreed with these statements (Fig. 8).

Most of the principals disagreed that computational thinking only prepares students for future work (97%) as well as disagreeing that students are already digitally *Gebildet*[6] when they start school thanks to their experience of digital devices in pre-school (87%). Furthermore, principals regarded computational thinking as more than just being able to handle and use a digital device—students should, for example, also understand the principles of data (98%). Most of the principals disagreed that computational thinking should only be an elective for those interested in the field (81%).

The last five statements all generated a considerable degree of agreement, suggesting that the principals had broad views on what computational thinking is good for. They regarded computational thinking as being as important as reading, writing and math skills (69%). They agreed that the future job market has a need for more digitally skilled employers (85%). They thought compulsory education should introduce the principles of data in order to understand and act in an increasingly digitalized everyday life (98%). They regarded computational thinking as an important part of the students' digital *Bildung* (98%). And they regarded teachers as important in helping children to understand the threats and potentials of technology (98%).

Challenges

Earlier research, debates and history within the field strongly demonstrate that teaching computational thinking and computer science in school settings is challenging. As a result,

[6] Verb used to describe a person with *Bildung*. *Bildung* (a German word, in Danish: *dannelse*) is a holistic and humanistic approach to public education and its content which has long-standing traditions in Danish and other Nordic countries. In our view, it encompasses the individual, social and cultural development of the whole child, which is reflected in the legislation relating to what the *Folkeskole* should be concerned with (e.g. students developing into active citizens with social competences and the ability to understand and take part in the democratic processes as well as their individual overall development as human beings).

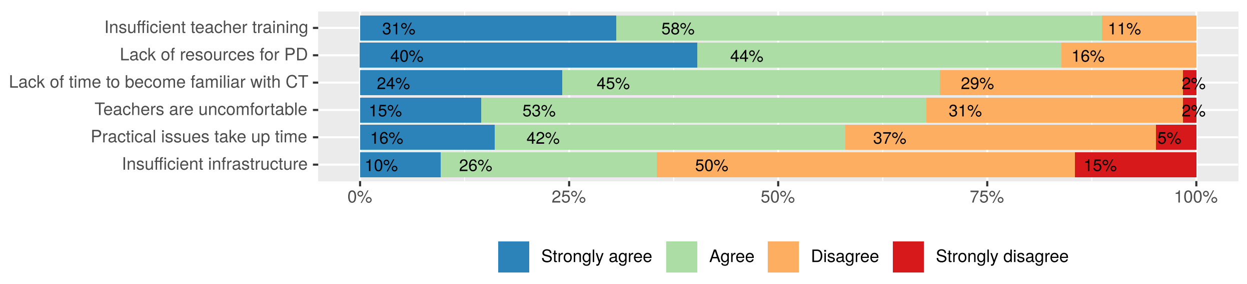


Fig. 9 Principals' view on potential challenges of teaching CT

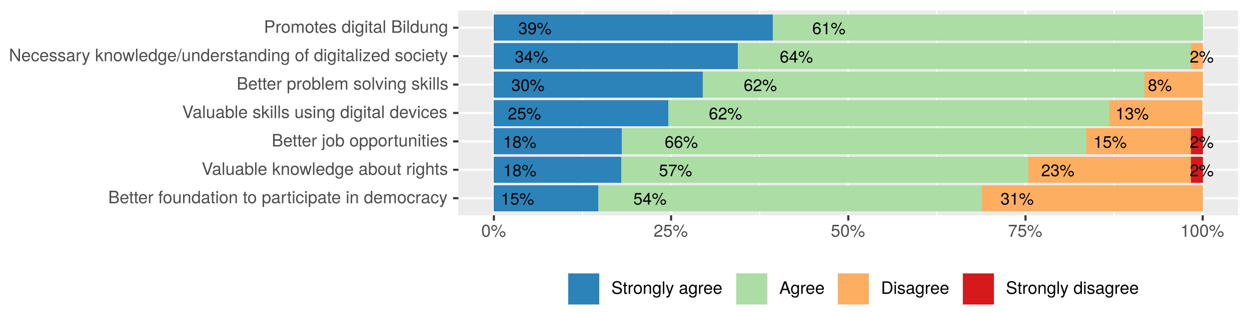


Fig. 10 Principals' view on potential advantages of teaching CT

we wanted principals to specify *what* they found challenging in implementing computational thinking based on six different statements (Fig. 9).

It is interesting, though from a Danish perspective not surprising, that most schools according to the principals have a sufficient infrastructure (65%). More than half of the principals reported the following: practical problems take up too much time (58%), teachers are uncomfortable teaching technology initiatives (68%), teachers do not have the time to learn about computational thinking themselves (69%), there are not enough resources for teacher professional development (84%), and teacher training courses do not prepare teachers to teach technology initiatives (89%).

Advantages

We also we wanted to examine principals' views on the possible advantages of implementing computational thinking in compulsory education. So we asked the principals to respond to seven different statements (Fig. 10).

Our results indicate that the principals held a comprehensive view on the potential advantages of teaching computational thinking. A majority of the respondents agreed on all seven arguments: 1. Computational thinking gives students a better foundation for acting and participating in a democratic society (69%); 2. Computational thinking provides students valuable knowledge about their rights in an increasingly digitalized society (75%); 3. Computational thinking allows students to get better jobs (84%); 4. Computational thinking offers students valuable skills in using digital devices (87%); 5. Computational thinking gives students the chance to address problem-solving in more appropriate ways (92%); 6. Computational thinking presents students with the necessary knowledge and understanding of an increasingly digitalized society (98%); and 7. Computational thinking promotes students' digital *Bildung* (100%).

Analysis and conclusions regarding conceptions of computational thinking

Our results suggest that the principals taking part regarded computational thinking as a broad and deep skill. It means more than the use of computers and the formal act of writing code. These findings are interesting in comparison with the worldwide focus on programming environments.

The fact that the principals were not in favor of teaching computational thinking as a separate subject is both interesting and controversial, given that the Ministry of Education intends to teach all students computational thinking as part of technological understanding, and in the light of the fact that automated information processes are so pervasive in the world in which we need to prepare children to live. This might reflect principals' self-assessed limited knowledge on computational thinking as well as their broad views on computational thinking as a subject with cross-curricula benefits.

However, the attitudes of the principals taking part with regard to the relevance of computational thinking suggest an inclusive view on what computational thinking is and what it is good for. For example, they did not think of children as digitally *Gebildet* before school, and deemed it to be important that all children should develop computational thinking skills. Many debates in Danish media have centered on this topic, and on why children are not digitally *Gebildet* even though they are almost born with a tablet in their hands and a smartphone in their pockets. Our findings indicate that this focus has resulted in a common understanding that digital *Bildung* requires education.

Most of the principals taking part disagreed with the idea that computational thinking should be an elective for only those interested in the field. This is not surprising, given that most principals thought that computational thinking should either be taught as a separate subject or be embedded in existing subjects.

With regard to the relevance of teaching computational thinking in compulsory education (Fig. 8), there was general agreement in two areas: functionalistic advantages for education/job market (*important as reading, writing and math* and *digitally skilled employers*), and general advantages for *Bildung* (*understand principles of data in a digitalized everyday life* and *digital Bildung*). It is interesting that there was a tendency towards greater agreement on the potential of *Bildung* (98 and 99%) than on job and education possibilities (69 and 85%).

With regard to advantages, the same trends appear again: general advantages for *Bildung* (*understanding of our society* and *digital Bildung*) as well as functionalistic advantages for education/job market (*better job opportunities* and *valuable skills*). As with the arguments relating to relevance, there was a tendency towards greater agreement about the potential of *Bildung* (98 and 100%) than on job and education possibilities (84 and 87%).

These findings might reflect traditions and cultural values illustrated in the formal aims of Danish compulsory education as presented in our introduction, for example a well-rounded development. However, there was an imbalance in terms of how principals assessed the importance of *Bildung* when compared with the importance of education and the job market, with a tendency to regard *Bildung* as more important than future education and work.

In our data regarding relevance, there was considerable agreement regarding the importance of teachers supporting children's learning process. However, the principals taking part also agreed that this is the most challenging area. The resources allocated to professional development are insufficient, and teachers are not adequately trained in this field. The principals also regarded technology initiatives as too time-consuming, both in terms of practical problems and in terms of their own familiarity with what computational thinking involves. This possibly reflects the fact that teachers are not trained to teach this subject, as well as recent reforms that downsized the amount of teachers' preparation time. Consequently, with not enough time or energy to do tasks you are already familiar with, it becomes even more difficult to develop the pedagogical and subject-specific competences needed to teach a new domain.

In our historical review of computational thinking in public education, briefly summarized in the introduction of this article, we found that municipalities spent hundreds of millions Danish kroner on hardware and infrastructure during the 1990s and 2000s (Caeli and Bundsgaard 2019). So it is not surprising that principals do not regard satisfactory infrastructures as a challenge. This foundation is advantageous to now allocate our resources on didactics and pedagogy.

Discussion

The aim of this study was to examine initiatives regarding computational thinking in compulsory education (primary and lower-secondary school) as perceived by principals. Specifically, we wanted to identify the types of technology initiatives involving computational thinking existing in Danish schools today and their frequency; examine principals' perceptions of whether teachers have the requisite professional skills to teach this subject; and analyze the ideas of school principals regarding computational thinking. (For example whether principals found computational thinking relevant in compulsory education, and what challenges they anticipated in terms of embedding it in school practices).

The context of the study is an increasing interest in computational thinking in school settings, motivated by a sense that it is important that children learn more in this area. However, educators struggle with questions about what computational thinking in education really is—and consequently, how we should teach and assess it.

This survey was the first part of a larger-scale research study that aims to suggest answers to these questions. In this part, we wanted to look at existing practices and what school principals think about this field—both with a view to setting the scene by exploring the current situation, and with a view to bringing practitioners' ideas to the fore: What do principals regard as important?

In general, the principals taking part had inclusive views on what computational thinking involves. The idea that computational thinking is the same as programming attracted the least level of agreement, which is interesting given the high focus worldwide on programming. This indicates that a Danish view might differ from international trends because it focuses on broader aspects of computer science with the overall term "technological understanding", which also encompasses competences with regard to designing systems, modeling and analyzing among other things.

Our results reflect long-standing Nordic traditions about what compulsory education is about. Computational thinking is not about pushing students into computing careers; rather it is about supporting the well-rounded development of human beings in a free and democratic society. Students are not digitally *Gebildet* when they enter school, and education is important.

Consequently, the professional development of teachers is necessary. This is not only revealed in our survey, but also seems to be regarded as a challenge worldwide across multiple computer science approaches and curricula. For example, Yadav et al. (2017) emphasized the challenge that few teacher-training institutions offer programs specifically for computer science teachers, and that there is a limited understanding of how to engage pre-service teachers from other subject areas in computer science and computational thinking.

The participants in our study were principals, and most of them reported that they had limited knowledge of what computational thinking really is. So we cannot develop definitions on computational thinking from their answers alone, nor was this the purpose of the study. Their limited understanding suggests that teachers are not the only ones in need of training—principals also need help to develop a culture and mindset around this subject and implement it efficiently into schools.

One limitation of this study is that the only thing we examined was the views of principals. For example, they might not know whether teachers had attended training courses, or they might lack understanding of the specific content of the professional development of their teachers. In addition, we cannot know whether professional development leads to the acquisition of competences.

To examine conceptions of computational thinking and its relevance more deeply more research is needed. As such, while in this study we surveyed principals with a view to gain an idea of the culture of computer science education in schools more broadly, in-service teachers as well as pre-service teachers could be surveyed next to examine their conceptions of this subject and how well they feel they are prepared to teach it.

Another good place to learn and look for answers is in the traditions and history of both computer science education and pedagogy, as well as in society today. Future parts of our studies aim to look into this.

Moreover, it would be interesting to measure the consistency of our analysis of Danish views on computer science education in terms of any differences compared with international perspectives, for example by surveying principals in other countries on this topic. Different mindsets and traditions could open up discussions on what computer science for all in a connected world should really be about.

Compliance with ethical standards

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

Ethical approval All procedures performed in studies involving human participants were in accordance with the ethical standards of the institutional and/or national research committee and with the 1964 Helsinki declaration and its later amendments or comparable ethical standards.

Informed consent Informed consent was obtained from all individual participants included in the study.

Appendix: Questionnaire[7]

Introduction

Study of the focus of Danish schools on computational thinking

We would like to ask for your help in answering 8–10 questions regarding your school's focus on computational thinking. The questionnaire is the first step of a research project linked to ICILS 2018, which aims to gain a deeper understanding of the ICILS results.

Thank you very much for your time and help!

[7] This is a translation of the original Danish version of the questionnaire.

Part I: Technology initiatives in your school

1. Is your school offering specific technology initiatives in the 2017/18 academic year?

❑ Yes
❑ No

[if your answer is yes]

1A. What type(s) of specific technology initiatives is your school offering in the 2017/18 academic year?

	Offered in grade 8	**Offered in other grades**	**Not offered**
Technological understanding (experimental elective subject)	❑	❑	❑
FabLab subjects in collaboration with FabLab@SCHOOL	❑	❑	❑
	Offered in grade 8	**Offered in other grades**	**Not offered**
Subjects as parts of the coding class initiative	❑	❑	❑
Your own FabLab initiative	❑	❑	❑
Elective that focuses on robots (e.g. Lego Mindstorms/ Lego First League)	❑	❑	❑
Technology training integrated in one or more subjects (e.g. programming, robots or digital fabrication)	❑	❑	❑
Another optional subject/specific technology course	❑	❑	❑

Please state the name of the school's FabLab initiative:

____

Please state the name of the school's IT subject that focuses on robots:

____

Please state what kind of technology training is integrated in what subject(s):

____

Please state the name of other technology subjects/courses:

____

Part II: Teachers' professional development

2. How many of your teachers have been engaged in professional development to teach technology initiatives?

- ❑ 0%
- ❑ 1-25%
- ❑ 26-50%
- ❑ 51-75%
- ❑ 76-100%

[if your answer ranges between 1 and 100%]

2A. To what extent did teachers' professional development focus on the following?

	Strong focus	**Partial focus**	**No focus**
Teaching students critical use of technology	❑	❑	❑
Teachers' skills in using specific technologies in their teaching (software, e.g. apps or technologies for creating homepages)	❑	❑	❑
General didactic approaches to 21st-century skills in education	❑	❑	❑
Teaching students productive skills with digital technologies (e.g. FabLabs or coding class)	❑	❑	❑
Teachers' skills in specific digital devices for education (hardware, e.g. iPads/tablets, 3D printers, robots, interactive whiteboards)	❑	❑	❑
Teachers' general digital skills (e.g. how the OS, user interface or printer works)	❑	❑	❑
Teaching students responsible participation in online life, incl. web ethics	❑	❑	❑
General didactic approaches to using digital technology in education	❑	❑	❑
Subject-specific didactic approaches to using digital technologies in education	❑	❑	❑

Part III: Your conceptions of computational thinking

There is no common definition or understanding of what computational thinking involves.

In the following questions, we are interested in the way you understand the concept and your perspective on the importance of teaching K-9 students CT.

3. To what extent are you familiar with the term computational thinking?

- ❑ I have never heard of it
- ❑ I have heard of it but have limited knowledge
- ❑ I have read/watched videos about it and/or had it explained by colleagues/at conferences
- ❑ I feel familiar with the concept and national/international discussions on it

4. How strongly do you agree or disagree with the following statements about what computational thinking involves?

	Strongly agree	**Agree**	**Disagree**	**Strongly disagree**
CT involves understanding and formulating algorithms	❑	❑	❑	❑
CT involves working with data (collecting, processing, communicating)	❑	❑	❑	❑
CT involves knowledge of computers to solve tasks using a computer	❑	❑	❑	❑
CT involves critical thinking regarding the role of digital technologies in our life and society	❑	❑	❑	❑
CT is more or less the same as programming and coding	❑	❑	❑	❑
CT involves solving problems in systematic and logical ways	❑	❑	❑	❑
CT involves using computers to solve problems and perform tasks in easier ways	❑	❑	❑	❑
CT involves a number of general principles of problem-solving with or without the use of computers	❑	❑	❑	❑

5. Do you think K-9 school students should be taught computational thinking?

(1) ❑ Yes, as a separate subject (and perhaps also integrated in other subjects)
(3) ❑ Yes, integrated in other subjects
(2) ❑ No

6. How strongly do you agree or disagree with the following statements about the relevance of teaching computational thinking in K-9?

	Strongly agree	**Agree**	**Disagree**	**Strongly disagree**
CT only prepares students for the labor market, and K-9 should not focus on developing this skill	❑	❑	❑	❑
K-9 students should learn to understand principles of data in order to understand and act in an increasingly digitalized everyday life	❑	❑	❑	❑
Developing CT is as important as developing reading, writing and math skills	❑	❑	❑	❑
CT should be an optional subject for those interested in the field, and everyone should not be forced to learn CT principles	❑	❑	❑	❑
The future job market has a need for more digitally skilled employees	❑	❑	❑	❑
CT is an important part of students' digital *Bildung*	❑	❑	❑	❑
From their experience with digital devices in pre-school, students are already digitally *Gebildet* when they start school	❑	❑	❑	❑
Teachers are important when facilitating technology training (e.g. discussing danger/potentials of technology or setting challenging tasks)	❑	❑	❑	❑

7. How strongly do you agree or disagree with the following statements about the potential challenges of implementing computational thinking in K-9?

Teacher training does not train teachers sufficiently to teach technology initiatives	❑	❑	❑	❑
Teachers do not have time to learn about new fields such as CT	❑	❑	❑	❑
There are insufficient resources to train teachers in this field	❑	❑	❑	❑

7. How strongly do you agree or disagree with the following statements about the potential challenges of implementing computational thinking in K-9?

	Strongly agree	**Agree**	**Disagree**	**Strongly disagree**
The infrastructure of the school and/or hardware is insufficient (e.g. unstable connections, lack of devices or insufficient Wi-Fi coverage)	❑	❑	❑	❑
Practical issues around technology take up too much of the time for teaching (e.g. launch of devices, updates, or outdated software)	❑	❑	❑	❑
Teachers are uncomfortable when teaching technology initiatives (e.g. different roles, non-transparent processes or students knowing a program better than the teacher)	❑	❑	❑	❑

8. How strongly do you agree or disagree with the following statements about the potential advantages of implementing computational thinking in K-9?

	Strongly agree	**Agree**	**Disagree**	**Strongly disagree**
CT helps students to get better jobs	❑	❑	❑	❑
CT gives students a better foundation for participating in a democratic society	❑	❑	❑	❑
CT gives students the necessary knowledge and understanding of an increasingly digitalized society	❑	❑	❑	❑
CT gives students valuable knowledge about their rights in an increasingly digitalized society	❑	❑	❑	❑
CT gives students the chance to solve problems in more appropriate ways	❑	❑	❑	❑
CT gives students valuable skills in using digital devices	❑	❑	❑	❑
CT promotes students' digital *Bildung*	❑	❑	❑	❑

If you have any comments, please add them here:

____

____

Thank you for your help!

References

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). *Developing Computational Thinking in Compulsory Education - Implications for policy and practice.* Joint Research Center, European Commission. https://doi.org/10.2791/792158.

Caeli, E. N., & Bundsgaard, J. (2019). Datalogisk tænkning og teknologiforståelse i folkeskolen tur-retur. *Tidsskriftet Læring Og Medier (LOM)*. https://doi.org/10.7146/lom.v11i19.110919.

Denning, P. J. (2017). Viewpoint. Remaining trouble spots with computational thinking. *Communications of the ACM, 60*(6), 33–39. https://doi.org/10.1145/2998438.

DLF (n.d.). *The school of the community.* http://eng.uvm.dk/primary-and-lower-secondary-education/the-folkeskole/about-the-folkeskole.

EMU (2019). *Teknologiforståelse*. https://www.emu.dk/grundskole/teknologiforstaelse.

EVA (2015). *4 elementer i god skoleledelse.* https://www.eva.dk/grundskole/4-elementer-skoleledelse.

Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher, 42*(1), 38–43. https://doi.org/10.3102/0013189X12463051.

ISTE (2016). *ISTE standards for students.* https://id.iste.org/docs/Standards-Resources/iste-standards_students-2016_one-sheet_final.pdf?sfvrsn=0.23432948779836327.

Jung, M., & Carstens, R. (eds.) (2013). International Computer and Information Literacy Study. ICILS 2013 *User Guide for the International Database.* IEA Secretariat.

Malmberg, A. C. (1970). Datalogi i skolen: Læreruddannelsen—en flaskehals. *Uddannelse: Undervisningsministeriets tidsskrift*. *Årg., 3,* 72–76.

Naur, P. (1966). Datalogi og datamatik og deres placering i uddannelsen. *Magisterbladet*, 15. maj 1966.

Naur, P. (1968). Demokrati i datamatiseringens tidsalder. *Kriterium*, 3(5). Nyt Nordisk Forlag Arnold Busck.

Naur, P. (1992). *Computing: A Human Activity*. New York: ACM Press.

NGSS States. (2013). *Next generation science standards: For states, by states*. Washington, DC: National Academies Press.

Schäfer, M. V. (2018). *Ny undersøgelse: Hver fjerde lærer føler sig stresset.* Folkeskolen.dk. https://www.folkeskolen.dk/624472/ny-undersoegelse-hver-fjerde-laerer-foeler-sig-stresset.

Tedre, M., & Denning, P. J. (2016). *The Long Quest for Computational Thinking*. Proceedings of the 16th Koli Calling Conference on Computing Education Research, November 24–27, 2016, Koli, pp. 120–129. https://doi.org/10.1145/2999541.2999542.

Undervisningsministeriet. (1972). *Betænkning om EDB-undervisning* (p. 666). Betænkning nr: Det offentlige uddannelsessystem.

Undervisningsministeriet (2013). *Agreement between the Danish Government (the Social Democrats, the Social-Liberal Party and the Socialist People's Party), the Liberal Party of Denmark and the Danish People's Party on an improvement of standards in the Danish public school (primary and lower secondary education).* http://eng.uvm.dk/-/media/filer/enguvm/enguvm/pdf/13/131007-folkeskolereformaftale-eng-red–2-.pdf?la=en.

Undervisningsministeriet (2015). *Ledelse af den nye folkeskole. Syv ledelsesfelter til skoleledelser og forvaltninger.* https://arkiv.emu.dk/sites/default/files/UVM%20Skoleledelse_WEB.pdf.

Undervisningsministeriet (2018a). *The Aims of the Folkeskole.* http://eng.uvm.dk/primary-and-lower-secondary-education/the-folkeskole/the-aims-of-the-folkeskole.

Undervisningsministeriet (2018b). *Undervisningsministeren vil gøre teknologiforståelse obligatorisk i folkeskolen.* https://uvm.dk/aktuelt/nyheder/uvm/2018/jan/180126-undervisningsministeren-vil-goere-teknologiforstaaelse-obligatorisk-i-folkeskolen.

Yadav, A., Krist, C., Good, J., & Caeli, E. N. (2018). Computational thinking in elementary classrooms: measuring teacher understanding of computational ideas for teaching science. *Computer Science Education, 28*(4), 371–400. https://doi.org/10.1080/08993408.2018.1560550.

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*. https://doi.org/10.1145/2576872.

Yadav, A., Stephenson, C., & Hong, H. (2017). Computational thinking for teacher education. *Communications of the ACM, 60*(4), 55–62. https://doi.org/10.1145/2994591.

Elisa Nadire Caeli is a PhD student at the Danish School of Education, Aarhus University, and the Teacher Education, University College Copenhagen. Her research focuses on children's development of computational thinking skills in primary and secondary education to prepare them to live and work in a digitalized society.

Jeppe Bundsgaard is a professor at the Danish School of Education, Aarhus University. His research focuses on curriculum studies, innovative teaching and learning, and the use of computers in education. His recent research has been on scenario-based standardized assessments of twenty-first-century skills, including computer and information literacy.

Technology Criticism in Schools – a Democratic Perspective on Technology Comprehension

Elisa Nadire Caeli and Jeppe Bundsgaard

This article is a translated version of:

Caeli, E. N. & Bundsgaard, J. (2020). Teknologikritik i skolen – et demokratisk perspektiv på teknologiforståelse. I Haas, C. & Matthiesen, C. (Eds.): Fagdidaktik og demokrati. Samfundslitteratur.

It is no secret that we spend a large part of our day looking at a screen – both inside and outside the school setting. We are slowly realizing that the services we use so indiscriminately collect and process our digital behavior. Even when we are not actively using them. For example, some self-tracking enthusiasts use a device while they are sleeping that tracks their pulse throughout the night and can tell them how they have slept and how they can improve their sleep. We can use an app to scan the food we eat, which then tells us how many calories we are allowed for the rest of the day, or how far we have to run to earn a piece of strawberry shortcake. Geo-tracking apps collect data about our location every second of the day, so we can be guided to places we want to go to, or be matched with people we want to meet. Streaming services tell us which series we want to watch and what music we want to listen to; we do not even need to click on the next film or create a playlist – this is automatically done for us. The world's most popular search engine, Google, provides us with answers to our questions – often without us having to leave the search engine page – and Facebook shows us content from those of our friends that we are most interested in, and shows us ads from products that best match our needs. Based on our nonstop online behavior, a very precise data image can be drawn using information about our individual lives: our name, age, address, family, friends, work, interests, attitudes, tastes, etc. Today, our data are not only collected and isolated in individual programs, they are also collected across programs, and you need to really know your way around data privacy settings to avoid contributing to the data pool.

But what does it actually cost to be able to save music in our own time capsules, and what are the consequences of being exposed to election slogans that are tailored to trigger us emotionally by presenting us with solutions to issues we have discussed with friends and family in our 'private' Messenger threads? New dilemmas arise when online programs surveil our behavior in the physical world we know and have learnt to understand, and in which we have earned rights. We might be tempted to think that being shown targeted content is useful, and that it is fair that machines process data about us all in the same way. We might even think that this ensures objectivity, because, unlike people, a computer is not biased by its own subjective values and opinions. However, computer programs are created by people. Any order that a computer executes is in essence conceived by a human being and is thus built on human opinions and values formalized in code. A computer can only do what it has been told to do. And it can be more or less impossible to work out which algorithms a digital service consists of, to what extent the service manipulates you to perform a particular behavior or even dictates that you do so, to what extent it gives us a warped image of the world, and to what extent it contributes to injustice.

Over the past years, we have become increasingly aware of the scope this issue and its consequences for people. For example, in her book *The Age of Surveillance Capitalism,* Shoshana Zuboff, professor *emerita*, describes a phenomenon she calls surveillance capitalism and that refers to the worrying way in which technology giants design digital products and services that monitor, collect, sell and buy data about our behavior. She explains, among other things, how they do this at the expense of democracy and freedom (Zuboff 2019). And in the books *Weapons of Math Destruction* (O'Neil 2016) and *Automating Inequality* (Eubanks 2018), mathematician Cathy O'Neil and Associate Professor Virginia Eubanks also describe how the design of algorithms amplifies discrimination and contributes to inequality.

As early as in the 1960s, the Danish professor of computer science Peter Naur pointed out that "the understanding of computer programming must be included in general education and thus become common knowledge" (Naur 1968: 32, our translation). If we failed to do this, computer programmers would have so much power that it could lead to the death of democracy, because "the power over a highly computerized system [will] evidently lie with those who understand how it works" (ibid., our translation). This fear is becoming a reality today. As a democratic society founded on the rule of law, we need to take seriously the fact that digital technologies lack a sense of justice and that algorithms lack transparency. When advertisements move from the side of public buses to our individualized newsfeeds, and when people are replaced by anonymous automatic data processing, we need to view data processes and digital technologies in a completely new way. Children should develop this kind of critical understanding of technology during their years of basic schooling, the purpose of which is, among other things, that children learn how to be critical consumers.

A subject-specific didactic contribution to technology criticism

Worldwide, no one seems to disagree with the tenet that computer science must be included in the school curriculum. Countless initiatives within the field of computational thinking have been launched (Bocconi et al. 2016). However, these initiatives tend to focus on programming, and their objective is often expressed as educating students to meet the needs of the labor market. A larger educational, social perspective is not the primary focus. In Denmark, we are currently testing the subject area, technology comprehension, and the subject description includes the development of basic critical competences. The objective of the subject is to "shape and educate students to be able to participate as active, critical and democratic citizens in a society characterized by increasing digitalization" (Undervisningsministeriet 2020, our translation). For example, students in the fourth to sixth grade must develop a digital awareness that enables them to "assess the intentionality and applications of digital artefacts in order to be able to act prudently in specific situations" (ibid., our translation). However, as of yet, no subject-specific didactics have been developed for the subject; instead the subject description refers to a general theme about pedagogy and didactics, practical experience as well as a pilot study and an mid-term evaluation of the test subject. Seen in this light, the objective of this article is to offer subject-specific didactic perspectives on a rapidly developing field.

The primary focus of the present article is the development of an understanding of *technology criticism* among K-9 students. The article consists of two parts that each address one of the basic questions of subject-specific didactics. Part 1 focuses on *what* technology criticism is in order to underline the significance and importance of including this area in the academic content of the school curriculum. We will present examples of how a society risks evolving in undemocratic directions if its citizens fail to acquire an understanding of data and digital technologies. In Part 2, we will discuss *why* teaching this type of academic content in general K-9 schools is important, and what we must consider when implementing it – both when it is taught as an independent subject and when it is integrated in another subject – and we include examples of *how* this kind of awareness can be boosted. Even though a subject-specific didactic foundation is still lacking for critical technology comprehension, pioneers of the contemporary field of computer science have shared numerous considerations and experiences that can put the subject into perspective and inspire our thinking today. We will therefore include historical discussions and relate these to the situation today throughout the article. Let us begin with a short history of the computer.

The power of the machine in society – historical highlights

Ever since the advent of the computer, we have discussed its influence and power in society – and for just as long, we have feared the robot that would render humans redundant due to its far superior processing capacity compared with the human brain. This sentiment is reflected in the name given to the first fully electronic computer built in 1945: the 'Giant Brain', and similar names like 'electronic brain' were used to describe the frightening fact that a machine could replace humans in many tasks that had previously been considered to rely on brain power (Bednerik 1967).

But for just as long as we have feared this scenario, experts have reassured us that our fears are unfounded. In line with this, Peter Naur (1954) wrote that he did not fear computers that could think. The danger was rather humans who could not think. This is because a machine can only execute a process that a human brain has planned. And this is one of the cardinal points of this article: that a democratic society needs humans to develop their understanding of computer science as a human activity.

The idea that machines can think is still with us today. Concepts such as artificial intelligence insinuate that a computer can be equipped with intelligence and can thus make intelligent choices. But perhaps we should refrain from understanding the term quite so literally. The concept of artificial intelligence grew out of the 1956 vision that "every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it" (Denning & Martell 2015: 26). However, this notion was subject to criticism from several parties. For example, some experts pointed out that rule-based machines might appear to be able to have a conversation in Chinese, but this did not mean that the machine actually understood any Chinese. In the mid-1980s, researchers abandoned the idea of simulating the way a human brain works (referred to as strong AI) and instead shifted their focus to designing systems that could take over cognitive work performed by humans (referred to as weak AI). Weak AI systems do not work in the same way as the human brain; instead they are designed to perform specific tasks (ibid.: 27-28). Google's self-driving car is an example of this. The 'intelligence' of the car's computer system is based on the choices that the designer of the system has embedded in the code. That is, the car only does what the designer has asked it to do, and by collecting data and analyzing statistics, it "learns" and predicts – quite mechanically and without human awareness – what it should do.

The risk of (mis)use of personal data has also been predicted and discussed previously. In the 1970s, the lack of clear rules about what kind of information about individuals could be collected and stored was discussed, raising in particular concerns about who such information could be shared with, and the risk of unauthorized individuals gaining access to personal data. In connection with this, a number of moral dilemmas related to collecting personal data were discussed. Computers made it possible to manage unprecedented volumes of data and thus made it possible to obtain a detailed picture of an individual's behavior. This entailed a risk that obsolete information stored in data registers could be used for unfair purposes. (Fischer, Frøkjær & Gedsø 1972).

The advent of the computer made it possible to use unprecedentedly large volumes of data, and new automated data processing processes created dilemmas that we are still struggling with today. To better understand these dilemmas, we will focus on the basic principles of data and data processes in the following.

About data and data processes

To understand how computer systems work, we need to understand what data and data processes are. In the words of Denmark's first professor of computer science, Peter Naur, data are first and foremost "auxiliary tools, good tools to do things in this world" (Naur 1966, our translation). Human beings process data all the time: every single day. For example, at a pedestrian crossing with traffic lights, the two actions 'stop' and 'walk' are represented by different types of data: the colors red and green and the image of a man standing still and a man walking. Before we cross the road, we process these data and ascribe meaning to them. Thus old data become new data – they become information. In our social context, we have agreed that the color red represents the information 'stop' and that the color green represents the information 'walk'. Thus data represents information, but it is not until we process the data that they become meaning-bearing.

When we process data, we can transform existing realities into desired realities, and thereby we can change things in the world (see Figure 1). In the example above, the current reality is that we are standing on one side of the road, and the desired reality is that we are standing on the other side.

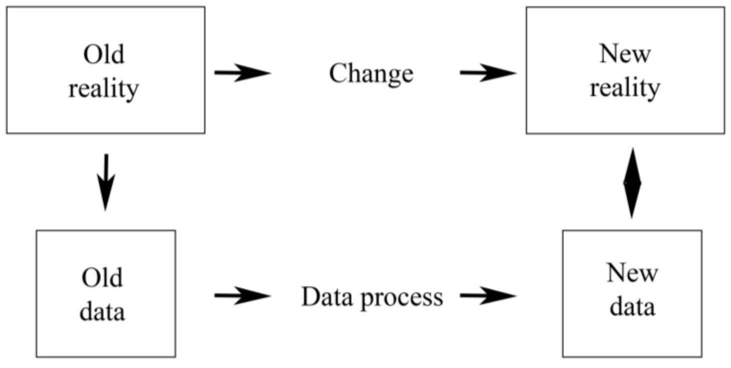


Figure 1. Naur's model for simulated data processes (reconstructed based on Naur 1966).

Another example – of a more complex reality – is when we are physically at home and our desired reality is that we are lying by the pool in southern Spain. To realize our desire, the smart move is to conduct a data process to identify the easiest and cheapest way to get to a pool in southern Spain instead of just getting in our car and driving. That is, we allow data to represent our current and our desired reality. In the given example, we would use data about our current reality (our address, which represents where we live) and data about the desired reality (the address in southern Spain that represents where we want to go). We can execute this data process by preparing step-by-step driving instructions (an algorithm) for ourselves using non-digital technologies such as a pencil, a piece of paper and a map of Europe. However, it can easily become a time-consuming and cumbersome process to follow roads (data in the form of lines and colors), choose and not choose exits (more data) and compare possible routes (data on distances, estimated times, types of roads, beautiful scenery, etc.). And is it does not stop here. We have to ask ourselves whether it is cheaper to go by car or by plane. And if we are concerned about the climate, we might ask ourselves what is actually the most environmentally efficient way to travel. And finally, we have to decide which hotel matches our budget and our wishes.

Today, many digital services have created an abundance of streamlined data processes. Often we do not even have to provide a location-based service with the data that represent where we live – the service already has this data (in many cases it has the data, even though we thought we had disabled the 'share location' function). When we have bought our plane ticket, we are (un)pleasantly presented with recommendations for nice hotels (with a pool) for the exact same dates for which we have bought tickets, and the hotel automatically sends us a link that tells us if we book a private driver "online now", we will get a 30 percent discount. When we arrive at our destination, we will get a complimentary drink – that is, after we have checked in on social media... And it goes on and on.

This is just a small – and perhaps at first glance innocent – example of how today the individuals who develop application services use our data to help create desired realities. The question is, do they (only) do this for our benefit, or do they (also) benefit from helping us? When do we as consumers strike the better bargain? We want to gain something and they want to gain something. We achieve our goals through a data process to which we might not give much thought, and it can seem overwhelming to understand how the individuals who have created a given digital service use our data to create the algorithms that take us, with just three clicks on our computer, from our home in Denmark to a pool in Málaga for less than DKK 3,000, and what their motives are. Nevertheless, we all need to be able to understand these processes.

Today, technology comprehension requires more than simply understanding traditional sequential algorithms, executed step by step. This is because many computer programs are developed as neural networks built up of parallel algorithms that are executed simultaneously. Whereas sequential computer programs execute rules in steps, neural networks are constantly learning new behaviors through continuous input (this is referred to as machine learning). This makes it far more difficult – and often impossible – to fully grasp the consequences of neural networks. When we feed computer systems with our data, we, the user, are in fact giving the program instructions. For example, artificial translation programs learn from users how to translate a specific word in the future; location services, learn how long a given route takes by car; and search engines learn how to prioritize searches. While this has created far greater opportunities, it has also spawned a number of moral dilemmas and consequences that we examine in more detail in the following.

Data and (amoral) algorithms in use

Denmark has been in first mover with regard to e-government since the 1960s (Jæger & Löfgren 2010), and has developed databases with information about its citizens. Since 1968 all these data have been coupled through a personal identification number called the CPR number (Nielsen 1991). As a result of this, Denmark's public sector stands to benefit from developing algorithms that can improve public services as well as prevent and monitor certain behaviors.

Access to large data sets helps government institutions prevent and investigate crime. For example, the Danish police use software from the company Palantir to analyze data from police reports to find patterns that can be used in decisions about where to patrol and who to keep an extra watchful eye on. This is called predictive policing (Fribo 2018). The state also uses data to identify citizens who commit welfare fraud or who have mistakenly been paid too much in welfare benefits (Hovgaard 2018). This is done, for example, through pooling registers from educational institutions, hospitals, electricity companies, doctors, psychologists, unemployment insurance funds, banks, employers, etc. This enables caseworkers to detect benefits fraud. For example, fraud with cash benefits or child allowance, by identifying dwellings that house a large number of people in a very small space. Such dwellings are typically used as a front address and several of the individuals registered as living there actually live somewhere else with their family.

Preventative effects and biased side effects

On the face of it, it would seem as if society stands to gain from preventing and stopping crime, and those of us who never attract the attention of the police or public authorities may take no issue with our data being included in these analyses – because we have nothing to hide. But as a democratic society based on the rule of law, we must carefully weigh the disadvantages against the advantages. Historical data are crucial in situations where the police use data about previous patrols and criminal activities that have already been committed and discovered to predict *who* is more likely to commit a crime and *where* a crime can be expected to happen. As a consequence of this, if the police have previously had blind spots or have been biased in their investigations, these prejudices will be reproduced in the proposals that the algorithm comes up with.

In the US, where predictive policing has been very successful, discussions have centered on whether, for example, in some areas young black men are subjected to unreasonable suspicion, and whether it is reasonable that they are stopped by the police again and again, just because they are young and black. The same bias may apply in the Nørrebro borough of Copenhagen or in the so-called ghetto areas in some of the bigger Danish towns. If the police exclusively or predominantly patrol these areas, they will exclusively or predominantly arrest individuals in these areas, and this in turn will intensify the algorithm. Biased algorithms are not only problematic with regard to identifying potential criminals, they are also problematic when used to decide who is a good candidate for a job, how teachers perform, who should be treated with a specific medication, etc. (see e.g. O'Neil 2017; Mann & O'Neil 2016; Davidson 2014). Precisely because they are made by humans, algorithms are just as prejudiced as the individuals who develop them – and as the individuals from whom they learn what to do.

Control through social credit

In China, the government has taken things a step further with regard to the use of data in their aim to create 'a good citizen, who is an asset to society'. In 2014, the Chinese State Council presented the initiative "Planning Outline for the Construction of a Social Credit System" (Prosa 2017, our translation). The objective of the system is to register and update the so-called social credit of enterprises and individuals, so that "professional credit assessments are extended to guide the development of professional, ethical and behavioral norms". The system identifies, for example, 'insincere behavior'– i.e. "behavior that seriously undermines the normal social order... seriously undermines transmissions in cyberspace" and "gatherings that disrupt social order and threaten national defense interests" (Prosa 2017, our translations).

Mitchell & Diamond (2018) describe how, for example, making political statements without gaining prior permission to do so or sharing news that the Chinese government does not like can reduce your ranking in these systems. A high 'citizen score' may give an individual access to, for example, a visa to travel abroad, higher internet speed or even the right to travel by train or plane within the country's borders, and conversely, a low score penalizes the individual by preventing them from gaining access to these benefits. Technically, the government can also monitor the behavior of an individual's friends and family, whose conduct may then also impact the individual's score. In this way, the government can educate the population to behave and think in a specific way by penalizing or rewarding them for the 'right' and the 'wrong' behavior. "China's experiments with digital surveillance pose a grave new threat to freedom of expression on the internet and other human rights in China. Increasingly, citizens will refrain from any kind of independent or critical expression for fear that their data will be read or their movements recorded – and penalized – by the government", Mitchell & Diamond (2018) warn. Several pilot projects have been run to test the social credit score system. One of these was the app Honest Shanghai, which was introduced in 2016. When the user activates the app with the help of a national ID number and facial recognition, the app collects and analyses up to 3,000 different types of information about the user from 100 public-sector institutions.

The Chinese equivalent to Amazon and Facebook, Alibaba, that offers e-commerce, social media platforms, an app store, digital payment methods and much more, launched another pilot project called Sesame Credit in 2015. The Sesame Credit system analyses, among other things, users' activity on social media. The algorithm is secret, but when the algorithm's developers are asked to give examples, they mention that gaming 10 hours a day is considered negative, whereas regularly buying diapers is considered positive because it signals that you are a responsible parent. The score is used, among other things, on the dating site Baihe, where individuals with a high score are considered more attractive partners than individuals with a low score.

China, unlike Denmark, is a one-party system with a tradition of not only extensive social control and surveillance of its citizens, but also very limited freedom, and the data analyses are therefore also used for purposes that we in Denmark would consider highly problematic. It is estimated that by the end of 2021 more than 560 million cameras will be installed in China (Ricker 2019), which, in combination with facial recognition technology, will make it possible to monitor the movements of the Chinese population, thereby entailing an obvious risk of violating the individual's right to privacy. It is not unlikely that similar disturbing systems for measuring the 'creditworthiness' of individuals will spread to other countries, and to some extent it is already happening. In Denmark, we have already seen how insurance companies add to rising social inequality through their use of algorithms to determine how much a customer should pay for their policy.

'Customized' prices based on data collection

Insurance companies in Denmark have access to an abundance of information about their customers and their potential customers (PFA 2018; LB 2018). When assessing whether they are willing to insure a potential customer and at what price, insurance companies can draw on "information from the Danish Civil Registration System about name and address, information from the Central Register of Buildings and Dwellings about the customer's address and dwelling in connection with the purchase of household insurance, and information from the department of motor vehicles in connection with insurance for motor vehicles" (LB 2018, our translation). In many cases, the information they access not only relates to the customer applying for insurance, but also to customers who live nearby, have the same type of car, the same type of job, the same family structure, etc. And insurance companies can also retrieve "information from publicly available sources such as search engines, virk.dk, the yellow pages, Krak, social media, OIS.dk and boligejer.dk." By using data from search engines, in principle an insurance company has access to all the information about the customer that the customer or others have shared online. This information can be used to find out who the new customer is 'friends' with, what kind of social and political circles the customer moves in, as well their religious associations, and this data can then be used to create profiles of individuals who resemble the customer, that is, customer profiles created based on real data. Insurance companies continuously analyze data about their customers in order to identify patterns in claims, fraud, switches to a new provider ('loyalty'), etc., giving them a 'risk profile' for each customer as well as potential customers (PFA 2018).

All these data are entered into a secret algorithm that calculates how much a customer should pay for their insurance as well as the terms and conditions of the policy. So when two potential customers who live in the same town enquire about the price of insurance on the same day, one customer may be offered an insurance policy in the most expensive group and be subject to a number of special conditions that the other customer, who lives in another neighborhood, is not a member of the same political party, is a different gender or has another family structure, or whose friends belong to different social circles than the first customer, is not.

Danish insurance companies are also looking into the possibility of collecting data about "how many kilometers motorists drive, how quickly they accelerate, and how hard they brake," and some companies are considering how they can use access to customer data via mobile phones, for example, by collecting data about the customer's movements throughout the course of a day. "For individuals who work out, it's great that they can get a discount if they use a pedometer, but what about the people who don't run a half marathon, or whose GPS tracking shows that they often frequent fast-food restaurants? Is it fair that they have to pay a higher premium?" asks a legal adviser from the Danish Consumer Council Tænk (Mørch 2017, our translation). It is impossible to know what determines which customers pay the most for their insurance, just as the ethical aspects and consequences hereof – quite conveniently for the insurance company – are hidden in the algorithms used by the insurance company to calculate their insurance premiums. However, it is safe to say that decisions about how much a customer is to pay for their insurance premium are not objective, despite the fact that a machine calculates the premium objectively. Even though data about two different customers are not processed using two different algorithms, the embedded subjective values of the algorithm may give rise to, and even amplify, inequality. And this inequality may in fact be based on what other customers who resemble the potential customer have done. You might be completely innocent and yet still be prevented from taking out insurance simply because you resemble the wrong people or live in the wrong place.

Whereas in China the state is behind most initiatives aimed at promoting desired citizen behavior, in capitalist societies, where governments are subject to democratic control, companies are more involved in developing algorithms that use data to promote the desired response patterns (for them). Nevertheless, such practice still raises a number of questions. Firstly, who decides what is desirable, and how far are we as a society willing to go (and let companies go) in terms of determining what constitutes desirable behavior? Secondly, is it reasonable that an individual's possibilities and challenges are determined in whole or in part by who they resemble, where they live and who is in their social circle? And finally, what data are available and therefore included in assessments of whether an individual's behavior is appropriate?

Democratic participation in a network society

The growth of the bourgeois and democratic public sphere in the 18th and 19th centuries was closely linked to the literary sphere, to meeting places in coffee houses, salons, etc., and to the rise of newspapers as a medium for the fledgling public debate (Habermas 1962). In the early years of the Internet, a number of pioneers saw in it the potential for further democratization through new modes of activism and new ways of accessing knowledge and news that were not being printed in traditional news media (Rheingold 1993). In many ways, reality has far exceeded their expectations. All over the world, people are joining discussions about sexual violence and harassment of women, global campaign organizations such as Avaz.org have emerged and have set the agenda and changed decisions in countless areas, and activists have used mobile technologies to organize their actions and overthrow regimes. These technologies have enabled activists from different parts of the world to learn from one another, to coordinate their actions and support each other. However, terrorist organizations have also benefitted in the same way from these technologies.

Democratic bodies at all levels are trying to involve citizens, including young people, in their decisions, and they are trying to engage young people in these bodies. However, we do not always make use of the options available to us. Young people in Denmark are very well informed about the world around them, and they have a sound understanding of democracy. But they show little interest in becoming involved in democratic bodies (Bruun, Lieberkind & Schunck 2017) or in other organizations in which they could make a difference.

Today, the Internet provides us with access to information, insight and inspiration from all over the world – both via the 'old' media's online channels and from all the new media, blogs and campaign bodies that have emerged. We can read about the fight against an oil pipeline on the border between Canada and the United States in our newspaper and then visit the websites hosted by activist forces to learn more about the situation. We can read about any given topic that interests us on specialist media from all over the world, and we can follow discussions between the world's leading minds in any field. Again, this is made possible by algorithms. We especially see this with regard to what news items we are presented with in our feed on Facebook, YouTube, etc. The algorithms at play here most likely use knowledge about what we have clicked on before, what our friends have seen and shared, etc. However, we cannot know for sure how the algorithms work as this knowledge is confidential (Mosseri 2018).

The next step in automated access to news might be that algorithms take over a substantial share of the news production itself. The British company Urbs Media has developed an algorithm that retrieves data from a data source – for example, about the current birth rate – and then enters these data into an article template that then can be used to produce hundreds of the same article targeted at different local newspapers (Rogers 2018). Several other companies are developing a system that can extract articles from different news sources and then combine them into unique, new articles. One of the major challenges in the democratic debate of our time is what has been termed fake news.

What is fake news?

Fake news is news – or rather propaganda – that is produced more or less with the intent of fooling the reader or viewer to believe in a false representation of the truth. Fake news is included in the category of journalism known as yellow journalism – that is sensationalized news. Fake news uses a one-sided representation of facts through a deliberate distortion of facts – or may even present fabricated lies as facts.

Artificial intelligence provides the producers of the latter with a very effective tool. For example, today it is possible to create extremely realistic photographic reproductions of people who have never existed, as was shown in a research project by the chip manufacturer NVIDIA (2017). When combined with similarly extremely realistic videos of physical surroundings, it is possible to create a completely realistic, but entirely artificial video of massacres of civilians in a war zone, for example. Videos like this can then be shared on social media or via news channels. Stories that contain human suffering, evil, condemnable actions, etc., are often shared hundreds of times and thus tend to 'go viral' and are thereby widely spread all over the Internet.

Fake news may have played a crucial role in the American presidential election of 2016 and in the Brexit referendum the same year. Triggered by these and similar events, researchers and tech companies are now working to develop technologies that can identify fake news and prevent its spread (see e.g. Cherry-Michigan 2018; FNC 2018; Hern 2017; Wingfield, Isaac & Benner 2016). But such systems come with their own disadvantages. An example is Facebook's outsourcing of its seemingly impartial fact checking system to the conservative American news site The Weekly Standard, which has given the news site the power to block any articles they deem to be false. As a consequence, in September 2018, based on a headline, The Weekly Standard marked an authentic news article from the progressive news site ThinkProgress as being fake, and as a result, numerous Facebook readers were prevented from seeing the article (Ingram 2018). So, how does it affect the democratic debate when companies are tasked with curating, evaluating and excluding opinions, news, and worldviews? Are socialist perspectives on the world fake news? Fascist perspectives?

And what about sexually provocative news? How does it affect the Danish democratic debate when American companies who have their own perception of what is right and wrong decide which information is presented to people in Denmark?

The examples mentioned above are just a few of the many issues that we as citizens in a democratic society must consider. Other issues include the thousands of videos based on an algorithm and promoted through the mass production of keywords; products that are individualized through algorithms; workflows and actions that are automated (from self-driving cars to health platforms and digital learning platforms) and many, many more. It can be quite overwhelming to grasp all of this. So how should we approach this important area in schools? What do students need to learn? And how? We will discuss this in the following.

What do students need to learn – and how?

Globally, the question of whether computer science – or elements hereof – should be included in curricula for compulsory education is gaining more attention. Discussions are based on the perspective that computer science skills are just as important as reading, writing and arithmetic in a modern society that is increasingly permeated by digital data processes that have a huge impact on society as a whole and on the individual. This development brings new opportunities for democracy, but it also poses a threat to democracy as we have shown in several examples above. Just how important a role computer science plays seen from a social and democratic perspective was formulated as early as in 1968 by Peter Naur who emphasized that:

> ....the understanding of computer programming must be included in general education and thus become common knowledge. [...] If this this broad *understanding* of programming is not effectuated, expert programmers will gain a power position that can lead to the end of democracy. (Naur 1968, our translation and italics)

We can all seem to agree at both national and international levels that students need to learn 'something about computer science', but we cannot seem to agree on what this 'something about" should actually include. What do students need to learn? What skills do teachers need? And who will teach the teachers these skills?

We can, however, agree that no consensus exists. A questionnaire survey from 2018 among representatively selected heads of Danish K-9 schools indicates that only just under seven percent of headmasters feel they really understand what computational thinking actually entails (Caeli & Bundsgaard 2019a). The remaining 93 percent understand to some degree that computational thinking entails more than just programming skills, and they consider it a significant skill with regard to educational development and Bildung – with the main emphasis being on Bildung. For example, a total of 91 percent either *strongly agree* or *agree* that computational thinking entails critical thinking. According to the surveyed headmasters, one of the biggest challenges of implementing the subject in practice is that teachers have no training in the subject. If we look at initiatives being taken internationally, we see that many of these initiatives are based on programming environments. Thus, focus is more on formal coding skills than on understanding what lies behind the coding. But perhaps we do not need to look abroad; some of the answers can be found here in our Danish traditions.

From data education to technology comprehension – IT development in Danish schools

In 1972, a working group presented a number of well-founded subject-specific didactic considerations about what computer science should comprise in K-9 schools and in the teacher education degree program – both as an independent subject and when integrated in other subjects. The working group based its considerations on the draft revision of the objects clause for K-9 schools, which was implemented three years later in 1975 and that laid the foundation of the objects clause in place today. The objects clause emphasized that "skills taught at school should be seen in relation to developments in society, because these developments will create new situations that the individual must learn to master, and they will create new problems in the relationships between people" (Undervisningsministeriet 1972: 39, our translation). The working group proposed that computer science should be included as an independent subject (data education) and be integrated into other subjects.

In light of the focal point of this article, the working group's perspectives about an interdisciplinary approach are particularly relevant, especially their suggestion to integrate computer science into social studies, history, geography and biology. The working group wrote in its report: "In these subjects, and when dealing with issues relating to social studies in general, computer science, including teaching its applications, can help students achieve a relatively in-depth understanding of the area" (our translation). And general, interdisciplinary concepts and conceptualizations such as data, problem formulation, model, algoritmization and process should be included in elementary data education (ibid.: 40). The working group also noted that their ideas could not be realized without also educating teachers – which in turn required education of teacher trainers. They therefore recommended as an initial step to launch an intensive program for upgrading teachers' skills. In the long term, they recommended recruiting university graduates with a degree in computer science to teach computer science in teacher training programs. They also emphasized that it would be advisable to establish a Master's of Education program in computer science, whose graduates could teach computer science as a main subject in teacher training programs. Pedagogics and didactics were important components.

We are faced with similar considerations today. In the 1970s, the working group's visions crumbled when it was decided to introduce photography as a new subject instead of data education. Then followed a period in which several pioneering schools unilaterally introduced data education, and later it was introduced as an elective subject for a few years. However, it lost its momentum, and in the early 1990s it was eventually replaced by the interdisciplinary subject EDP that no one really took responsibility for. For many years following this, focus was on operational user competencies and infrastructure, such as a PC user certificate and Internet connections, as well on purchasing PCs for schools. From around 2000 and onwards, focus shifted to the procurement of interactive whiteboards and tablets – and later robots, 3D printers, etc. Didactics and reflections on what was actually needed were not the primary focus (Caeli & Bundsgaard 2019b).

Three aspects of technology comprehension as a subject in K-9 schools

Today is the first time since the 1970s and 1980s that we have moved closer to actually establishing a subject that resembles the abandoned subject of data education. The new subject is called technology comprehension and was originally introduced by the Danish Ministry of Education as an experiment that had an overall vision that all children learn to *comprehend* technology – as indicated by the name (Undervisningsministeriet 2018a). According to the subject description, this entails not only basic knowledge about computer science, iterative design processes, complex problem-solving and programming, but also – and this is particularly important to underline in this context – "the skills to assess the applicability, intentionality and consequences of digital artefacts for the individual, the community and society as a whole" (Danish Ministry of Education 2020, our translation).

Figure 2 visualizes three main aspects of technology comprehension as a subject in K-9 schools: computer science, design and technology criticism.

Technology comprehension as a subject in K-9 schools		
Datalogy	Design	Technology criticism
↕	↕	↕
Subject-specific didactics	Subject-specific didactics	Subject-specific didactics

Figure 2. Three aspects of technology comprehension as a subject in K-9 schools: What should schools teach, why and how?

In general, *computer science* can be said to refer to a basic understanding of what data are, their characteristics and use; *design* refers to the implementation of design processes and development of digital technologies that solve relevant issues; and *technology criticism* refers to an understanding of the consequences (risks and possibilities) of digital technologies. How can we include all these aspects of technology comprehension in K-9 schools?

Three perspectives of technology criticism

We have ascertained that design and programming initiatives – regardless of how their importance – are not enough. Students must acquire knowledge about what data are and how they are collected and processed today, so they themselves can determine the possibilities and risks posed by automation for us as individuals, as communities and as societies, see Figure 3.

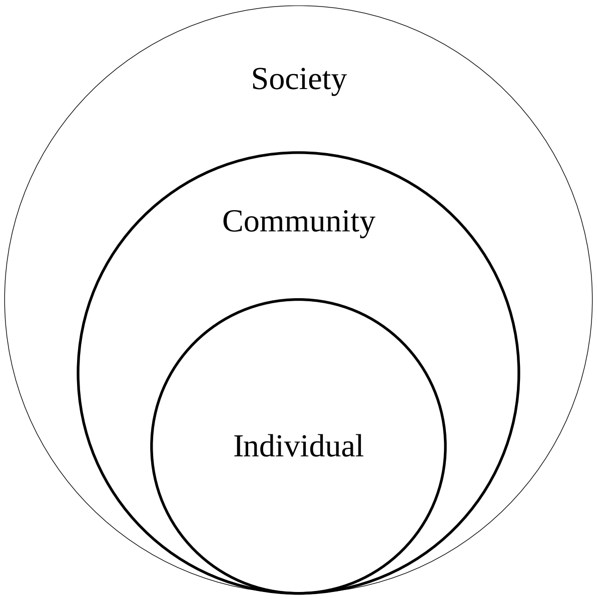


Figure 3. Three perspectives of technology criticism.

Students must develop an understanding of how systems work, and they must be able to discuss in a broader perspective the moral dilemmas and consequences of the choices a designer makes when designing an algorithm.

Students are more likely to acquire basic knowledge and skills if these are taught as an independent subject. Consequently, teachers with specialist knowledge and training to teach the fundamental principles of computer science are needed. Native language teachers or social science teachers can hardly be expected to take on this responsibility. Their academic focus is – and should be – on other areas. Having said that, perhaps these two subjects can take on a central role in the discussion about how technology and automation impact society and the ethical issues this raises? For example, teaching materials could include relevant contemporary dilemmas for the students to discuss and consider. A classic example is the moral dilemmas that arise when programming self-driving cars: Who should the computer program select to die if the choice stands between two passengers in the car and five pedestrians? What if the two passengers are from the same family? What if four of the five pedestrians are children?

Seen in the context of today, we cannot ignore the importance of technology comprehension in any of the subjects taught in K-9 schools. In each individual subject, we need to consider how the interdisciplinary nature of computer science affects the objectives of each subject. For example, in Denmark, in the subject of visual arts, students should be able to “analyze the function of an image in a given context” (Undervisningsministeriet 2019b, our translation). And in history, students should be able to “relate changes in everyday life and living conditions over time to their own lives” (Undervisningsministeriet 2019c, our translation).

Even though a visual arts teacher or history teacher is not responsible for teaching students the principles of computer science – just as they are not responsible for teaching students how to read or write texts – they must incorporate the elements of computer science that are relevant in their academic context in same way that they must teach students how to read academic texts from their field. That is one side of the story: the relevance of the subjects and areas of responsibility.

Another side of the story is the training of teachers. As the working group pointed out in 1972, in the long term it would make good sense to educate ‘didactic computer scientists’, in much the same way as Master’s students in mathematics should acquire pedagogical and didactic competencies when they are responsible for educating teachers of mathematics. Just as it is not enough to merely focus on the didactics of a subject, it is not enough to merely focus on the academic content of the subject computer science. Danish K-9 schools are not based on the objective to educate specialists. Instead, students should acquire both ”knowledge and skills that will prepare them for further education” as well as skills that prepare them “to be able to participate, demonstrate mutual responsibility and understand their rights and duties in a free and democratic society” (Ministry of Children and Education 2018).

Conclusion

The purpose of this article was two-fold. First, based on several examples of how data and algorithms are used today, we have emphasized how important it is for children in a democratic society to develop the ability to think critically about the way in which digital technologies use data. By using automated data processes, humans have achieved enormous progress for individuals and society as a whole, but we have also seen an abundance of examples of how secret algorithms increasingly mislead us rather than guide us. As a democratic society founded on the rule of law, we are responsible for ensuring that, as part of their basic education, children learn how to be critical toward data and algorithms. In this respect, we want this article to serve as a wakeup call.

Second, we have discussed how this kind of thinking can be promoted in Danish K-9 schools, bearing in mind how quickly the area has developed and how far behind the Danish school system is, despite several warnings from experts over the years. Computer science is not a mandatory subject in Danish K-9 schools, neither is it integrated in other subjects – yet. However, several progressive initiatives in this area exist. A new subject – technology comprehension – is being tested, the objective of which is to “shape and educate students to participate as active, critical and democratic citizens in a society characterized by increasing digitalization” (Undervisningsministeriet 2020, our translation).

If these progressive thoughts are to be implemented in practice, we need to educate teachers to teach the subject. Technology comprehension should be an independent subject in line with other subjects offered as part of the teacher training program – and should not only include computer science as a discipline in itself but also how to teach computer science. In addition, it is important to consider how the academic content of subjects that are already taught is related to the issues we have outlined in this article. Which aspects of technology comprehension are relevant to address in each individual subject?

“There is an increasing tendency to give up and leave it up to the ‘experts’. But this is absolutely untenable for an area that affects key decisions," said Peter Naur in 1968 (our translation). He realized that those who understand how the system works are those who hold the power.

> This fact is the reason why many of us who work with computers and think about what consequences computers have for society feel compelled to continue emphasizing that the understanding of computer programming must be included in general education and thus become common knowledge. (Naur 1968, our translation)

The warning signals have been flashing for 50 years – and today the signals are more powerful than ever. It is time for us to take our responsibility seriously and ensure that our children are ready for their future lives in the democratic society to which they belong.

References

Bednerik, K. (1967). *Programmørerne: Automationens elite*. Albertslund: Det Danske Forlag.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A. & Engelhardt, K. (2016). *Computational Thinking in Compulsory Education*. Brussels: Joint Research Center. European Commission.

Bruun, J., Lieberkind, J. & Schunck, H.B. (2017). *ICCS 2016: internationale hovedresultater*. Copenhagen: Danish School of Copenhagen, Aarhus University. https://pure.au.dk/portal/da/publications/iccs-2016(2ac8e389-3c6e-4d1e-ad96-ba33c5bd8e51).html

Caeli, E.N. & Bundsgaard, J. (2019a). Computational thinking in compulsory education: A survey study on initiatives and conceptions. *Educational Technology Research and Development*, 68(1): 551-573.

Caeli, E.N. & Bundsgaard, J. (2019b). Datalogisk tænkning og teknologiforståelse i folkeskolen tur-retur. *Tidsskriftet Læring Og Medier (LOM)*, 11(19): 30. https://doi.org/10.7146/lom.v11i19.110919

Cherry-Michigan, G. (2018). Algorithm beats humans for sniffing out fake news. *Futurity*, August 22. https://www.futurity.org/fake-news-detecting-algorithm-1844942/ (accessed 19.06.20).

Davidson, J. (2014). The 7 social media mistakes most likely to cost you a job. *Money*, October 16. http://www.time.com/money/3510967/jobvite-social-media-profiles-job-applicants/ (accessed 19.06.20).

Denning, P.J. & Martell, C.H. (2015). *Great Principles of Computing*. Cambridge: The MIT Press.

Eubanks, V. (2018). *Automating Inequality: How High-Tech Tools Profile, Police, and Punish the Poor*. New York: St. Martin's Press.

Fischer, C., Frøkjær, E. & Gedsø, L. (1972). *Datalære i skolen. Om data og EDB i samfundet.* Copenhagen: Gads Forlag.

FNC (2018). Exploring how artificial intelligence technologies could be leveraged to combat fake news. *Fake News Challenge*. http://www.fakenewschallenge.org/ (accessed 19.06.20).

Fribo, A. (2018). Politiet: Predictive policing er interessant – men datagrundlaget er for småt. *ING/Version 2*, August 28. https://www.version2.dk/artikel/politiet-predictive-policing-interessant-datagrundlaget-smaat-1086050/ (accessed 19.06.20).

Habermas, J. (1962). *Strukturwandel der Öffentlichkeit: Untersuchungen zu einer Kategorie der bürgerlichen Gesellschaft*. Frankfurt: Suhrkamp.

Hern, A. (2017). Google acts against fake news on search engine. *The Guardian*, April 25. https://www.theguardian.com/technology/2017/apr/25/google-launches-major-offensive-against-fake-news (accessed 19.06.20).

Hovgaard, L. (2018). 12.000 borgere udtages årligt af Udbetaling Danmark til datatjek for fejl og snyd. *Version 2*, September 17. https://www.version2.dk/artikel/12000-borgere-udtages-aarligt-udbetaling-danmark-datatjek-fejl-snyd-1086211/ (accessed 19.06.20).

Ingram, M. (2018). *The Weekly Standard* and the flaws in Facebook's fact-checking program. *Columbia Journalism Review*, September 18. https://www.cjr.org/the_new_gatekeepers/the-weekly-standard-facebook.php (accessed 19.06.20).

Jæger, B. & Löfgren, K. (2010). The history of the future: Changes in Danish e-government strategies 1994-2010. *Information Polity*, 4: 253-269. https://doi.org/10.3233/IP-2010-0217

LB (2018). Forsikringstilbud og kontaktoplysninger. *Lærerstandens Brandforsikring*. https://www.lb.dk/personoplysninger/10-forsikringstilbud-og-kontaktoplysninger (accessed 19.06.20).

Mann, G. & O'Neil, C. (2016). Hiring algorithms are not neutral. *Harvard Business Review*, December 9. https://hbr.org/2016/12/hiring-algorithms-are-not-neutral (accessed 19.06.20).

Ministry of Children and Education (2018). *The Aims of the Folkeskole*. https://eng.uvm.dk/primary-and-lower-secondary-education/the-folkeskole/the-aims-of-the-folkeskole (accessed 24.08.21).

Mitchell, A. & Diamond, L. (2018). China’s surveillance state should scare everyone. *The Atlantic,* February 2.

Mosseri, A. (2018). Bringing people closer together. *Newsroom*, January 11. https://newsroom.fb.com/news/2018/01/news-feed-fyi-bringing-people-closer-together/ (accessed 19.06.20).

Mørch, T. (2017). Forbrugerrådet Tænk bekymret over forsikringsbranchens nye tiltag. *Finanswatch*, February 26. https://finanswatch.dk/Finansnyt/Forsikring/article9393163.ece (accessed 19.06.20).

Naur, P. (1954). Elektronregnemaskinerne og hjernen. *Perspektiv*, 1(7): 42-46.

Naur, P. (1966). Datalogi og datamatik og deres placering i uddannelsen. *Magisterbladet*, May 15.

Naur, P. (1968). Demokrati i datamatiseringens tidsalder. *Kriterium*, 3(5): 31-32.

Nielsen, H. (1991). CPR – Danmarks Folkeregister. *CPR-kontoret*. https://www.cpr.dk/media/17546/cpr-danmarks-folkeregister.pdf (accessed 19.06.20).

NVIDIA (2017). Generating photorealistic images of fake celebrities with artificial intelligence. *NVIDIA Developer*, October 30. https://news.developer.nvidia.com/generating-photorealistic-fake-celebrities-with-artificial-intelligence/ (accessed 19.06.20).

O’Neil, C. (2016). *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*. New York: Crown.

O’Neil, C. (2017). Don’t grade teachers with a bad algorithm. *Bloomberg Opinion*, 15. maj. https://www.bloomberg.com/view/articles/2017-05-15/don-t-grade-teachers-with-a-bad-algorithm (accessed 19.06.20).

PFA (2018). Behandling af personoplysninger. *PFA*, July 2. https://pfa.dk/andet/behandling-af-personoplysninger/ (accessed 19.06.20).

Prosa (2017). Kinas system til social kontrol. *Prosa*, November 2. https://www.prosa.dk/artikel/kinas-system-til-social-kontrol/ (accessed 19.06.20).

Rheingold, H. (1993). *The Virtual Community: Homesteading on The Electronic Frontier*. Cambridge: The MIT Press.

Ricker, T. (2019). The US, like China, has about one surveillance camera for every four people, says report. https://www.theverge.com/2019/12/9/21002515/surveillance-cameras-globally-us-china-amount-citizens (accessed 30.06.20).

Rogers, G. (2018). Reporters and data and robots: Why 2018 will be the year of automation in news. *Urbs Media*, December 31. https://medium.com/@urbsmedia/reporters-and-data-and-robots-ee352220c5f1 (accessed 19.06.20).

Undervisningsministeriet (1972). *Betænkning om edb-undervisning i det offentlige uddannelsessystem. Betænkning nr. 666*. Undervisningsministeriet.

Undervisningsministeriet (2020). *Teknologiforståelse*. Copenhagen: Undervisningsministeriet. https://www.emu.dk/grundskole/teknologiforstaelse (accessed 30.06.20).

Undervisningsministeriet (2018a). *Undervisningsministeren vil gøre teknologiforståelse obligatorisk i folkeskolen*. Copenhagen: Undervisningsministeriet. https://uvm.dk/aktuelt/nyheder/uvm/2018/jan/180126-undervisningsministeren-vil-goere-teknologiforstaaelse-obligatorisk-i-folkeskolen (accessed 19.06.20).

Undervisningsministeriet (2019b). *Billedkunst. Faghæfte 2019*. Copenhagen: Undervisningsministeriet. https://emu.dk/sites/default/files/2020-06/GSK_Billedkunst_Fagh%C3%A6fte.pdf (accessed 29.06.20).

Undervisningsministeriet (2019c). *Historie. Faghæfte 2019*. Copenhagen: Undervisningsministeriet. https://emu.dk/sites/default/files/2019-08/GSK.%20Fagh%C3%A6fte.%20Historie.pdf (accessed 29.06.20).

Wingfield, N., Isaac, M. & Benner, K. (2016). Google and Facebook take aim at fake news sites. *The New York Times*, November 14. https://www.nytimes.com/2016/11/15/technology/google-will-ban-websites-that-host-fake-news-from-using-its-ad-service.html (accessed 19.06.20).

Zuboff, S. (2019). *Overvågningskapitalismens tidsalder: Kampen for en menneskelig fremtid ved magtens nye frontlinje.* Copenhagen: Informations Forlag.

Technology Comprehension in Schools

Computational Design for Solving Authentic Problems

Elisa Nadire Caeli, Danish School of Education (DPU), Aarhus University and the Department of Teacher Education, Copenhagen University College

Martin Dybdal, Department of Computer Science, University of Copenhagen

This article is a translated version of:

Caeli, E. N. & Dybdal, M. (2020). Teknologiforståelse i skolens praksis. Datalogisk design til autentisk problemløsning. Tidsskriftet Læring Og Medier (LOM), 12(22). https://doi.org/10.7146/lom.v12i22.115613

Abstract

This article is a didactic contribution to the subject area known as "technology comprehension" in Danish K-9 schools. In this context, technology comprehension is regarded as a discipline that involves developing competencies in the fields of computational thinking, design thinking, and critical thinking regarding the use of computer science in society.
The article presents the results of an experiment conducted in a Danish eighth-grade class. The purpose was to examine a way of teaching computational thinking and design thinking by developing a computational design to solve an authentic problem, thereby identifying didactic opportunities and challenges.
The experiment was based on two primary hypotheses: We expected the students would dislike the idea of changing their designs after user feedback, and we thought they would have a narrow view of computer science, regarding it as the equivalent of programming. These hypotheses were partially confirmed. However, other opportunities and challenges were also identified, including changed conceptions of computer science and new opportunities for success within the field, success with creative breaks and varied ways of working, the positive importance of authenticity with regard to the students' understanding, and the wish for a greater degree of freedom than they were granted in this project.

Introduction

This article presents the results of a Danish research project conducted in a Danish eighth-grade (14-15 year-olds) class in March 2019. We designed a three-day course aimed at teaching the students to develop a digital solution to a real problem. The objective was to examine a way of teaching in which the students developed comprehension of and competencies in computational thinking, including design and programming, using data as a means to develop a solution to an authentic problem for humans, rather than the acquisition of these skills being a target in itself, detached from real practices. Specifically, we wanted to examine the subject-specific didactic opportunities and challenges in developing and implementing this kind of course.

The project was conducted at a time when significant international attention is being directed at computer science as a discipline in basic education. All over the world, attempts are being made to introduce computational thinking, either as an independent subject or as part of other disciplines in basic education. In Denmark, this has led to, among other things, the Danish Ministry of Education launching experiments on *technology comprehension* as a stand-alone subject and as a discipline included in other subjects. Since March 2019, technology comprehension has been taught at 46 schools with the overall objective of "developing academic competencies and acquiring skills and knowledge, so that they [students] can constructively and critically participate in the development of digital artefacts and understand what such artefacts mean" (EMU, 2019, our translation).

Methodologically, this project was designed as a design experiment in the subject area referred to in the Danish K-9-school context as technology comprehension, and it is based on a problem within science as a subject. We planned the experiment together, but divided the roles between us: one of us took on the role of teacher and the other took on the role of observer. Our experiment was planned and conducted independently of the Ministry's experiment. This means that our teaching design was not based on the curriculum for the experimental subject area technology comprehension; instead we developed our own goals and didactic considerations based on research in the area and theoretical analyses. We consider technology comprehension as a discipline that involves developing competencies in the fields of computational thinking, design thinking and critical thinking with regard to how computer science is used in society (Caeli & Bundsgaard, 2020). In this context, we specifically examine competencies within computational thinking and design thinking through a computational design developed by eighth-grade students.

Structure and research contribution of the article

In this article, we first present the societal and educational context in which the experiment is based, as well as its theoretical and methodological basis. Then we introduce our course design and describe and analyze what we did based on specific examples. Next, we discuss our results, including what worked and what did not work as seen from our perspective and from the students' perspectives, and we address the opportunities and challenges we see in connection with implementing this kind of technology comprehension in practice. Finally, we round off with our concluding remarks.

Our experiment is intended as a specific and authentic example of how teachers can teach the aspects of technology comprehension that we refer to as computational thinking and design thinking, and therefore a detailed description of practice is included. We also identify other potential opportunities and challenges that we saw in our experiment, and we discuss cultural changes that we believe are necessary.

Computer science and project-specific focus

Our experiment is based on the tenets of former professor of computer science Peter Naur. He believed that computer science is an interplay between people and computers, and that the subject must always include focus on how it is used in practice.

Naur pointed out that the starting point of all projects is an incomplete description, for which many very different solutions can be fully acceptable (Naur, 1970). He also emphasized that programming is merely a tool in this kind of process, and he illustrated this by introducing a model that described the relationship between people, problems and tools (Figure 1).

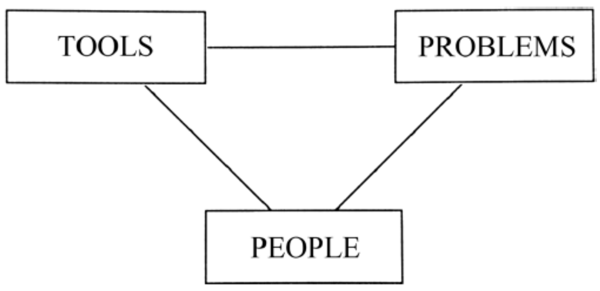


Figure 1. The relationship between problems, tools and people (Naur, 1965)

Naur explained that we need tools to understand problems, and that tools that are designed to solve problems that nobody understands are meaningless. He also said that problems only exist in the minds of people and only relative to understood tools, and that tools only exists as tools if people think of them as appropriate things with which to solve a problem (Naur, 1965).

One of the authors of this article has interpreted Naur's model so it can be used for subject-specific didactic purposes in a contemporary context (Caeli, 2021). In this article, we exemplify how teachers can use the model when planning, implementing and assessing their teaching to be aware of all three perspectives and how they are related to one another by asking themselves and the students the following questions:

> What is the relationship between **problems** and **people**? Do the students consciously solve a problem that has relevance for people? The questions allow the teacher to see whether the students are working with authentic problems, or whether they risk working with digitalization simply "because we can".
>
> What is the relationship between **people** and **tools**? Do the students understand and think about the tools as appropriate things for solving problems? These questions will help the teacher become aware of whether the students have what it takes to solve problems, and whether they understand the tools used for open problem solving, or whether there is a risk that students' limited functional user skills will limit their use of tools.
>
> What is the relationship between **tools** and **problems**? Are the tools used to solve problems that the students actually regard as problems? Here, the teacher can become aware of whether the tools are used to solve problems, or whether there is a tendency for using the tool to become the goal in itself. (Caeli, 2021)

According to Naur, problem-solving should unfold as a process in which there is a conscious relationship between the three factors. We believe this is important in a teaching context so as to avoid, for example, uncritically focusing on learning how to use a specific tool without having a real problem or without understanding what the tool can be used for in real life. In our experiment, the students were asked to work creatively with problem-solving through design thinking and computational thinking.

Design thinking, computational thinking and computational design

In recent years, the concept of design thinking has been increasingly understood in a broad sense to refer to a method of problem solving; however, the term has actually existed for many years. Professor John Edward Arnold is considered to be one of the first to use the term when in 1959, in his book *Creative Engineering*, he described it as a specific approach to creative problem-solving. The design researcher Nigel Cross explains that design thinking involves processes such as analyzing context, generating and delineating problems, generating ideas, thinking creatively, making drafts, sketches, models and prototypes, testing a design and evaluating (Cross, 2011).

In our project, we couple computer science and design. Professor Peter J. Denning discusses this kind of coupling in an article from 2017, in which he introduces the concept of *computational design*. We find this concept to be well-suited to our experiment. *Computational design* is the intersection between computational thinking and what is referred to as computational doing. According to Denning, computational thinking refers to a process in which a computational solution to a problem or a concern is found[1], and computational action refers to the actual use of computer science and computational tools to solve problems. In contrast, computational design refers to the creation of new computational tools and methods that solve problems (or concerns) for people, and – not least – that people actually use. About this relationship, he says:

> Clearly, designers are a subset of thinkers because you need to be a computational thinker to design computational tools; and not every thinker is a designer. Also, designers are tool users, but not all tool users are designers or thinkers." (Denning, 2017)

The model below (Figure 2) illustrates this relationship.

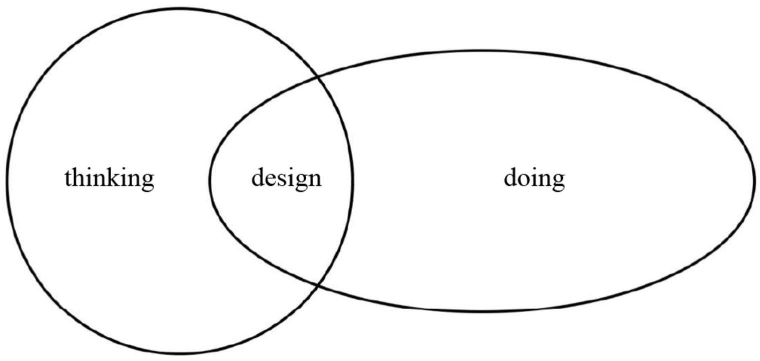


Figure 2. The relationship between computational thinking, computational doing and computational design (Denning, 2017).

This coupling was a mainstay of our experiment, and this will be demonstrated below in the section about the experiment in practice.

Methodological approach and didactic reflections

Research question and hypotheses

The purpose of the experiment was to examine a way of teaching computational thinking and design thinking in Danish K-9 schools by developing a computational design to solve an authentic problem, thereby identifying the subject-specific didactic opportunities and challenges of this kind of teaching.

[1] Denning uses the term *concerns* instead of *problems* because we often use computer science to solve concerns that are not actually problems. An example of this is when we develop a game; this can hardly be seen as a response to solving a problem.

We wanted the students to design and program a product that could solve an authentic problem in the world. The intention was, and is, that by presenting our experiment and our analyses of practices, we can contribute to a subject-specific didactics of technology comprehension that others can build on or be inspired by. To achieve this aim, we have been guided by the following overall research question:

What are the subject-specific didactic and cultural opportunities as well as challenges in connection with teaching computational thinking and design thinking in a K-9-school context?

We had a hypothesis that an iterative approach to computer science and design in a school context could be problematic, and that students were more accustomed to school work following a linear process from A to B. This assumption was based on, among other things, previous research in the area that has shown that teachers sometimes find it challenging to teach flexible processes (Smith et al., 2016). More specifically, we hypothesized that it could prove challenging to get the students to change their design to accommodate user feedback if they already believed that their design was finished. However, we agreed that it was important the students acquired an understanding of how a computational designer really works. This meant that they also needed to learn how iterations can be used.

Furthermore, we hypothesized that the students' perception of computer science would be limited to thinking of it in terms of programming as in "writing in code". We therefore focused on ensuring that the students discovered that a computer scientist does a lot more than merely write code. This hypothesis was based on, among other things, the way in which society in general uses the terms programming and coding. We see a tendency to equate programming with coding; however, in our understanding, programming involves all the stages of system development, including the design of the system, and not just the specific action of entering codes. When students do not fully understand what programming entails, they risk neglecting the design aspect and instead only focus on learning how to code correctly.

Methodological approach

The project was designed as a design experiment, inspired by design-based research. The purpose of design-based research is, among other things, to explore complex teaching practices in a way that is valuable for others. That is, in a way where the construction of the experiment can be transferred from one specific context to another. It is a formative, process-oriented assessment method, through which a phenomenon is investigated to understand both its intended and unintended consequences. Thus, design-based research is often conducted as iterations in the form of repeated, improved interventions that lead to the generation of new theories (Barab & Squire, 2004). In our experiment, we tested the theories presented above, and analyzed and discussed improvements of our design based on the experiences we realized. This enabled us to contribute to the generation of subject-specific didactic theory in this new area. Even though the experiment was conducted without iterations, our descriptions and discussions can pave the way for others to complete and further develop our course.

Didactic standpoint and Bildung[2] perspective

Our experiment is based on the German Bildung theorist Wolfgang Klafki's critical-constructive understanding of how the individual's Bildung is also a societal issue. This means, among other things,

[2] A holistic and humanistic approach to public education and its content that encompasses the individual, social and cultural development of the whole child,, e.g. students developing into active citizens with social

that, “Bildung theory and Bildung practice have the opportunity to, and are tasked with, not only responding to conditions and developments in society, but also with assessing and helping shape these based on an educational responsibility for current and future living and development opportunities...” (Klafki, 2001, our translation). Thus he breaks with the classic approach of having a basic core consisting of a fixed set of cultural elements. Instead, Klafki presents a new core based on a critical historical-societal-political and, also, educational awareness, in which teaching focuses on “key problems of contemporary life and the presumed future that are typical of the period”.

In our experiment, we used the environment as an example of a dominant problem characteristic of our time. We especially focused on the part Klafki describes as:

> developing the insights required to develop resource-saving and energy-saving technology as well as environmentally friendly products and forms of production, just as we must limit our consumption and make it environmentally friendly.

Klafki also stresses that general didactics cannot disregard the dimensions of area and subject-specific didactic concretisation:

> Only by drawing on area- and subject-specific didactic discoveries is it possible to find answers to general didactic questions. (Klafki, 2001, our translation)

In our experiment, we primarily focused on technology comprehension as a subject, and this is also the focus of this article, which is why science as a subject is not dealt with as such here. We drew on the specific expertise of teacher specialists and content specialists within computer science, and this enabled us to combine subject-specific knowledge with general didactic knowledge to create a subject-specific didactic approach[3].

Design and implementation of the experiment

The experiment was run over three full school days in an 8th grade (14-15 year-olds) in March 2019. The school was located in the western part of the Greater Copenhagen Area. There were 20 students in the class, 10 boys and 10 girls, of whom 95 per cent had a different ethnic background than Danish (19 students), and 80 per cent were multilingual (16 students).

Prior to the lessons, we prepared an overall lesson plan (see Appendix 1), as well as slides for the lessons and worksheets[4], and we made sure we had all the materials and resources we needed for the actual lessons[5]. The lesson plan presented in this article has been modified to allow the reader to gain insights

competences and the ability to understand and take part in the democratic processes as well as their individual overall development as human beings (Caeli & Bundsgaard 2019).

[3] Elisa Nadire Caeli is originally educated as a teacher, has a master’s degree in learning and innovative change, and is now a PhD student doing research on the development of computational thinking and technology comprehension in K-9 schools. Martin Dybdal has a master’s degree in science and a PhD in computer science, and is now a special consultant. He conducts research into, and works with, computer science didactics and communication from basic education level to university level.

[4] Slides and worksheets (in Danish) can be viewed and downloaded at https://silo1.sciencedata.dk/shared/designeksperiment.

[5] See the section “The experiment in practice” for more information about materials and resources.

into what actually took place in the classroom and use these insights in their own teaching. Time intervals are indicative.

We designed and conducted the experiment together, but we took on different roles when conducting the experiment. Martin Dybdal was responsible for teaching, while Elisa Nadire Caeli observed the teaching, wrote field notes and had conversations with the students. Even though Elisa Nadire Caeli did no teaching, throughout the experiment she acted as an observer and discussed didactic and pedagogical initiatives and changes with Martin Dybdal. In addition to Elisa Nadire Caeli and Martin Dybdal, Maja Hvidtfeldt Håkansson, who is a computer science student and a trained architect, participated as an assistant teacher, as we considered it necessary with two teachers for this kind of project-oriented teaching in a new subject area.

Data collection and analysis

The experiment was designed by the two authors as a collaborative effort: one author served as a content specialist and the other as a teacher specialist. The experiment was based on the academic and subject-specific didactic theories described above. This resulted in a lesson plan in which the students were to develop a computational design based on a central contemporary problem within science. Data were collected by testing the experiment in practice: we observed and video-recorded the teaching and wrote field notes. We extracted a number of quotations from the teachers and students and we have analyzed these quotations in the final part of this article. We specifically focused on our assessment with the students. This assessment was partially unstructured, as we wanted the conversation to flow freely; we wanted to hear the students' immediate feedback. The empirical material on which this article is based is thus qualitative in that we analyze and discuss the process on the basis of quotations from the students and our own assessment.

We will present and discuss the experiment and the assessment in more detail in the following. We share our lesson plans and slides, and describe the experiment in detail so that others can test and further develop the course or draw on it for inspiration.

The experiment in practice

The experiment was in its form partially closed. The purpose was to model problem-solving through computational design as a *first step* in the development towards becoming more independent computational designers. As mentioned above, we hypothesized that the students were not used to working in this way. This also meant that we had already selected the problem the students were to work with as well as parts of their solution: They were to develop a prototype (a physical and programmed design) that used LED strips. The purpose of the prototype was to help humans reduce their CO_2 emissions by reducing their electricity consumption. The students were to retrieve data about Danish real-time CO_2 emissions from the data provider electricitymap.org, and then program the LEDs to "report" (i.e. *light up* following a certain pattern) when CO_2 emissions in Denmark are low and high, respectively. Thus their design would show when electricity consumption is emitting the least CO_2 (for example because wind energy is being used), and therefore when electricity consumption is least harmful to the environment. The LEDs were to be integrated into a physical prototype. The students were to come up with their own ideas for their physical design, including where to place it, and whether it should be integrated into an existing product or a new product. They were also to come up with ideas for how the LEDs were to show whether it was a good idea to use electricity or not and then program this. If, for example, consumers can see that CO_2 emissions are high at a given time of day, they may be encouraged to delay turning on their dishwasher or charging their mobile phone until emissions are lower, because this would entail that the energy being used is greener.

Our general didactic approach was based on the environmental issue as a typical contemporary key problem, with a view to providing students with an insight into the need to develop resource-efficient and energy-saving technologies, as well as environmentally friendly products. We used the subject areas computer science and design as cross-disciplinary auxiliary subjects in our subject-specific didactic approach. To create an overview and clarify what could be a chaotic and unpredictable design process, we used a design model (Figure 3) to illustrate and talk to the students about where we were going, where in the process we were, and what the next step would be[6].

In the following, we describe what we did on each of the three days by drawing on specific examples. We have chosen to focus primarily on the subject-specific content and the subject-specific didactic choices of the teaching. In other words, we do not delve deeper into contextual and cultural factors, for example, students who walked in and out of the classroom, or students who were noisy, used mobile phones or ate snacks during the lessons. Even though these factors are important and should be considered when a teacher is teaching a group of students that they know well, thereby allowing the teacher to address the culture in the class, they are not the main focus of this article – both because they fall outside the scope, and because they are insignificant in relation to the general statements of this article. Having said that, we will briefly discuss some of these challenges in our analysis of the experiment at the end of the article.

Day 1: Understanding the problem, understanding data and defining the project

As can be seen in our teaching slides[7], we started the experiment by introducing Naur's model of the relationship between people, problems and tools (Figure 1). Using this model, we explained to the students that they were to design solutions for *people* for a current *problem* in the world, and that they needed some *tools* to solve the problem. This led us to the design model (Figure 3), which we introduced as the next step to help the students grasp what was going to happen over the course of the next three days, including what our starting point was and where we planned to end. The model was inspired by "Værktøjskassen" ("The Toolbox") (Katapult/TEACH, 2013) and a design model from the School of Design, Stanford University[8], but it was adapted to the specific climate issue the students were to work with.

[6] This approach is inspired by John Hattie and Helen Timperley's model for feedback, which, among other things, aims to involve the students in their learning by making the process visible (Hattie & Timperley, 2013).
[7] See the teaching slides for more about the specific activities, questions, models: https://silo1.sciencedata.dk/shared/designeksperiment.
[8] https://dschool.stanford.edu/

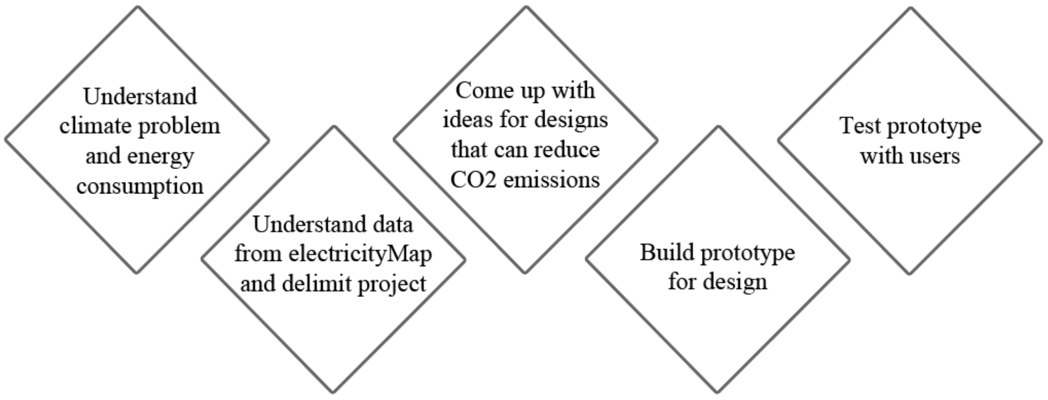

Understand climate problem and energy consumption
Understand data from electricityMap and delimit project
Come up with ideas for designs that can reduce CO2 emissions
Build prototype for design
Test prototype with users

Figure 3. Our design model with the experiment's specific content area: "the climate problem and energy consumption".

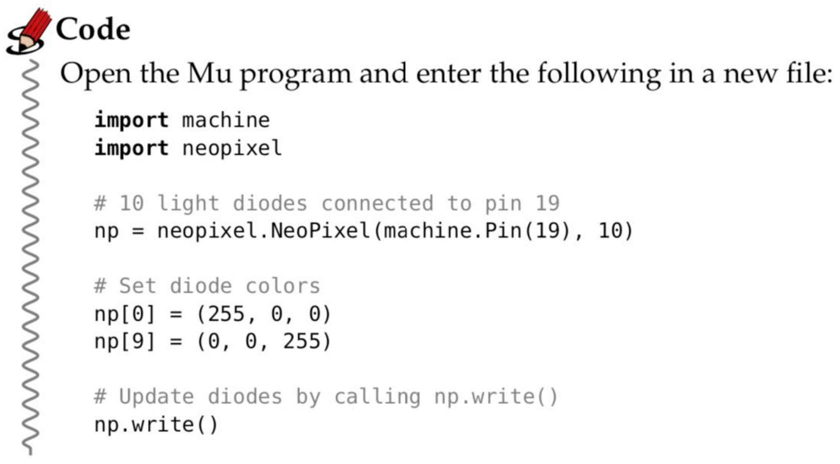
The first phase of the model was about understanding the climate issue and energy consumption. Because we did not know the students before we started, we did not know how much the students already knew about the subject. However, the day before the experiment started, we learned that the students had just begun working on a science project about sustainable energy supply; naturally this was helpful to the experiment.

We started day 1 with an open dialogue about climate change, after which we showed the students a short video about the greenhouse effect and the causes and consequences of climate change. Then we discussed what we can do to counteract climate change; we talked about power consumption and power production; and we introduced the students to electricitymap.org. For example, we talked about why France was much "greener" than Estonia, and why Bornholm was greener than, for example, another Danish island: Zealand. We linked this conversation to the scope of our problem-solving project: reducing CO_2 emissions through the use of data from electricitymap.org. The students were to use electricitymap.org in their own computational design that was to include programming LEDs.

Again, we used Naur's model (Figure 1) to illustrate why a number of well-known *tools* that were designed to solve a *problem* for *people* are designed as they are. For example, why is a doorbell designed to ring and not flash a light when someone is at the door? Why is a traffic light red, yellow and green? And so on.

Then, we introduced the students to how the LEDs work and how to program them. We used our own computers as they were equipped with the software we wanted to use. Before the lessons, we checked that everything worked. We also brought our own ESP32 microcontrollers, LED strips and a 4G Wi-Fi hotspot to minimize the risk of technical or infrastructural problems from systems in place at the school that we were unfamiliar with.

Throughout the experiment, the students worked in pairs and shared a computer. As mentioned above, the experiment was partially closed in the sense that we modelled large parts of the course for the students in a way that provided them with an understanding of how *computational designers* work. That is, it was not our intention for the students to become programmers over the course of the three days – that would also have been quite unrealistic. The intention was for them to experience how programming can be used as a tool to solve a real problem in the world. Therefore we provided them with worksheets with a substantial amount of the code they needed. For example, for worksheet #1, the students were to start by opening the program Mu[9] and entering the code as described below (Figure 4).


Code

Open the Mu program and enter the following in a new file:

```
import machine
import neopixel

# 10 light diodes connected to pin 19
np = neopixel.NeoPixel(machine.Pin(19), 10)

# Set diode colors
np[0] = (255, 0, 0)
np[9] = (0, 0, 255)

# Update diodes by calling np.write()
np.write()
```

Figure 4. Example of code from worksheet.

Then they were to test whether the code worked as intended: The first diode [0] should light up in red, and the tenth [9] should light up in blue.

Next, the students were to add more features to the program. The goal was for the diodes to alternate between red and blue. Finally they experimented with the colors using color codes from the worksheet. This quickly led to a number of questions, such as why only two lamps lit up, and how they could make them all light up. In other words, the students were motivated by the fact that they could influence how their LEDs lit up.

After this first trial run, we wanted the students to learn how to work together when programming by using the method of pair programming[10] as research has shown that working together in pairs, results in much better programs. We introduced the following rules for pair programming:

[9] A simple Python code editor for beginner programmers: https://codewith.mu/
[10] See Tabel et al., 2017 for more about pair programming in a teaching context.

- One person is the driver, the other person is the navigator.
- The driver keeps their hands on the steering wheel (keyboard) and eyes on the road (screen).
- The navigator holds the “map” and focuses on the destination and how to get there.
- The navigator must not touch the keyboard or the mouse, and the driver must not ignore the navigator.
- No one is allowed to issue orders.
- Both parties must talk nicely to one another.
- Both parties must try to keep a conversation going and constantly verbalize what they are doing, so as to verbalize what they are learning.

We told the students that one of the objectives was that they acquire a vocabulary for programming and thereby become better at talking about the programs.

After the students had experimented with colors and more attractive codes, they used the next worksheet (#2) to learn how to connect their microcontroller to the Wi-Fi and retrieve CO_2 data from electricitymap.org. They used the data in their design to make the LEDs visualize CO_2 emissions in Denmark in real time. In the beginning they were motivated by the LEDs lighting up, but they gradually lost their focus and motivation to program. We explained to them that day 2 would be a bit more creative in that they were to draw and build a prototype using cardboard. However, for the LEDs to show the CO_2 emissions, they first had to learn how to use the data they had retrieved from electricitymap.org. We worked on this for the rest of the day.

Day 2: Ideas for designing and constructing a prototype

Day 2 started with a recap of day 1, which served as a formative assessment for us in relation to the students’ level of understanding, and as a reminder to the students about where in the process we were. Recurring statuses and conversations about what was happening enabled us to monitor their understanding, which we considered important in a partially open problem-solving process such as our experiment. There were no right or wrong answers.

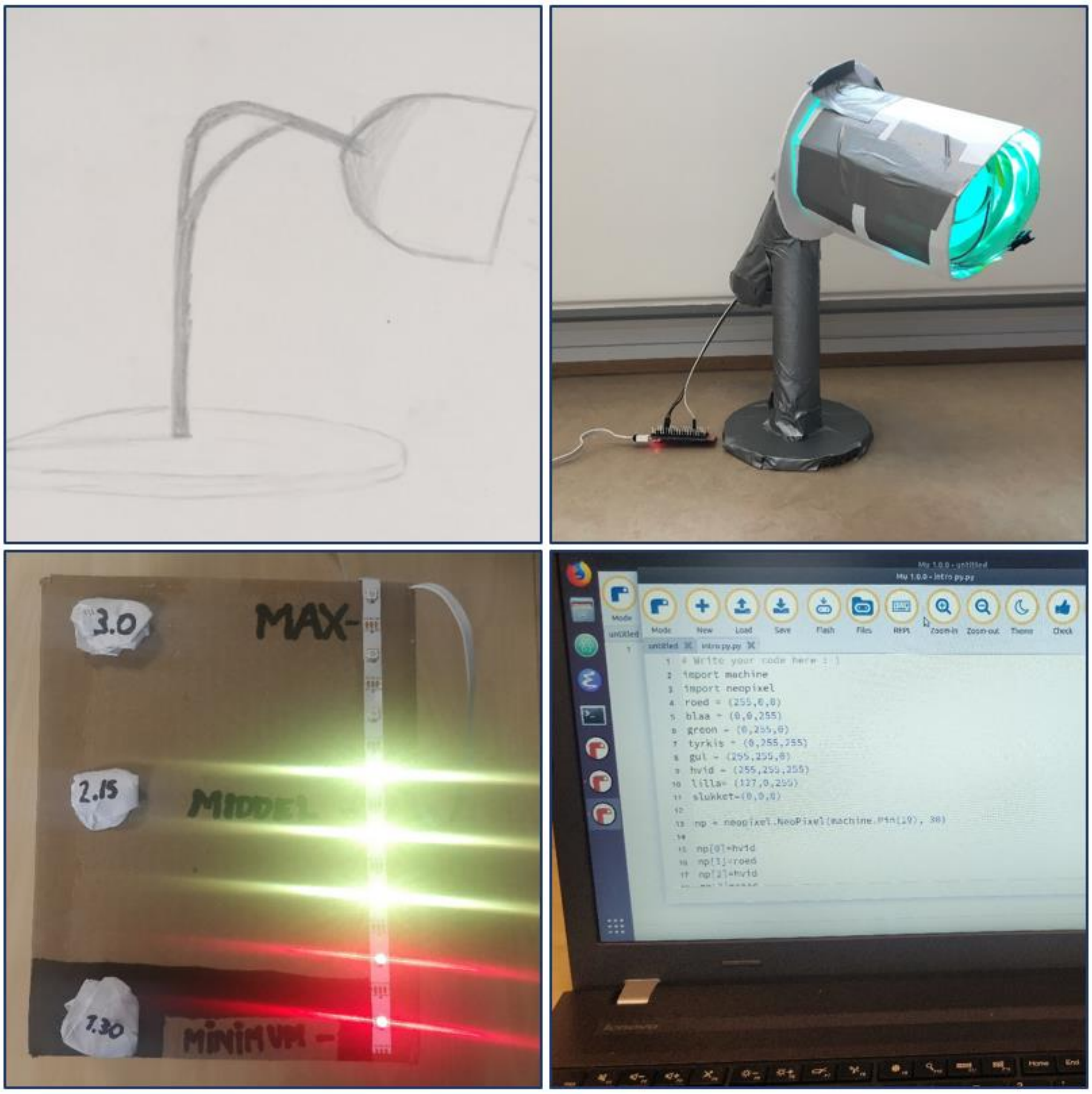
After the recap, we moved on to the next phase: design ideas. Whereas focus on day 1 – in addition to understanding the science problem – was on *computational competencies*, focus on day 2 centered more on *design competencies* and on linking the two competencies.

We started with a brainstorm about what constitutes good and bad design, including what to consider when interacting with users, and what the students themselves thought was important in a design. All their ideas were written on the board. The students listed the following characteristics of good design: appropriate size that fits the body, matches our habits, durable, new and creative, simple, helps people, looks good, feels nice, works, solves the problem, good quality and easy to learn to use. And they described the following characteristics of bad design: poor quality, difficult to understand, breaks easily, too small or too big, difficult to use, does not solve the problem and is annoying. Then we watched a short video in which a number of famous Danes talked about what they think good design is. We had intentionally planned for the students to activate their pre-understanding themselves by sharing their ideas before they were presented with arguments and ideas from other sources.

Next, we asked the students to rank their ideas about what characterizes good design according to their opinions as to what was the most important. For example, we asked them, what is most important: that a design looks good, or that it is a good fit to our body and habits? “That it looks good,” said one student. “It's more important that it’s a good fit,” said another student, to which the first student responded, "Yes, I agree. I've changed my mind." We had planned to have discussions like these, because we found them to be useful for developing the students' collaborative competencies and insight into the perspectives of other users. In our view, and with reference to our design model, such competencies are required of a competent computational designer.

Then we showed the students some prototypes, both to help make their pending task more concrete and to serve as a starting point for a conversation about whether these prototypes were examples of good designs based on the criteria the students had just discussed. This conversation led to discussions about what might be less important in a prototype. For example, we discussed that maybe the students' criteria "good quality" and "looks good" might not be as important in a prototype.

After this, the students got started on planning their design. First, they discussed what their design needed to be able to do, and where it was to be placed, and then they sketched on paper what it should look like. The next phase was Build prototype for design. Prior to the experiment we had asked the students to collect and bring some material they could use to build their prototypes, for example cardboard and polystyrene and we brought some materials too. We also made sure scissors, hobby knives, pens and glue guns were available. Because it was such a central part of the design, the electronics had to be incorporated into this process. This resulted in the students alternating between building with cardboard, putting the LED strip in place and programming the "actions" of the LEDs. Figure 5 provides a sense of the process with examples of sketches on paper, prototypes and code.



```
# Write your code here :-)
import machine
import neopixel
roed = (255,0,0)
blaa = (0,0,255)
groen = (0,255,0)
tyrkis = (0,255,255)
gul = (255,255,0)
hvid = (255,255,255)
lilla= (127,0,255)
slukket=(0,0,0)

np = neopixel.NeoPixel(machine.Pin(19), 30)

np[0]=hvid
np[1]=roed
np[2]=hvid
```

Figure 5. Examples of the groups' paper sketches, codes and prototypes.

Day 3: Testing, user feedback, new prototype and presentation

After a recap from day 2, we presented the next phase: user testing. This was planned as peer feedback. Two groups demonstrated to each other how they had built their design, explained what a finished version would look like and shared their challenges and what they would like some help with. As feedback criteria, we used the students' own ideas from day 2 of what constitutes good design. We did this to emphasize the fact that the users of a design – people – ultimately decide whether your design is successful.

In our experiment, we wanted the students to get a taste of what it means to interact with users. Our intention was that the students experience that, even though they themselves thought that their design was finished, there might still be things that could be improved after they had received feedback from others. Therefore, we told them that, based on the feedback they received, they were to change at least one thing in either their code or physical design. Next they were to prepare a presentation for later that day. The presentation could be no longer than six minutes and had to be based on certain criteria (see lesson slides).

Before the students gave their presentation, we returned to the programming part. This was a slight deviation from the original plan, in which we had set aside more time for the physical design, user feedback and preparation of the presentation. As it turned out, we had more time than anticipated, and we assessed that the students would benefit from a more detailed conversation about what each part of their program did. We asked the groups to take turns at presenting their code to one another while we asked questions. We also discussed in plenary session how to improve the codes, for example, by using more efficient methods.

In the final part of the experiment, the students were to present their design to another class from the same year and two of their teachers. In their presentations they talked about the parts they had struggled with, but that they had found a solution to through testing and feedback. For example, one group described how they had learned that yellow and white light were difficult to distinguish from one another, so they chose to use a different color instead. The same group explained that at first they had struggled with their physical prototype because it was very unstable, so they changed the design based on the feedback they had received. The students also commented on the materials they had used: "It should really be made of metal and not cardboard and duct tape and plastic cups." The students' presentations were the final part of the experiment, and after the final presentation we assessed the entire process with the students.

Results: opportunities and challenges

In this section, we discuss the subject-specific didactic opportunities and challenges seen from our perspective – but also from the perspectives of the students. We had prepared to number of questions[11] with which to assess the experiment thoroughly with the students[12], as we considered it a significant part of a qualitative design experiment like this – and of teaching in general – to discuss developments with the people involved.

We analyzed our results based on our two primary hypotheses: that we would encounter some resistance from the students with regard to changing their designs to accommodate user feedback, and that they would equate computer science with programming. However, we also identified other opportunities and challenges, including changed perceptions of and new opportunities for good experiences with computer science, good experiences with creative breaks and varied ways of working, the positive impact of authenticity with regard to the students' understanding, and the wish for a greater degree of freedom than they were granted in this project.

Resistance to change

After the first iteration of the design process, in which the groups had given each other feedback, we gave the students the extra challenge to make at least one change to their design. The groups responded differently to this challenge. Whereas some students immediately set about stabilizing their design or changed their programming, other groups believed, as expected, that their product was finished and therefore they did not think it was necessary to incorporate the feedback they had received. This became evident when some of the students just left the classroom because they believed they were finished.

[11] See lesson slides, slide no. 63.
[12] The wording of some of the quotes has been slightly edited for the sake of readability, but without changing the meaning of the students' comments.

This resistance to making certain changes was also seen later; for example, when giving their presentation, a group showed signs of not having understood what their LED strip actually showed. The same group had been given extra help with the design and programming of their LEDs, but they had been more interested in getting the LEDs to light up in a specific way, that they thought looked nice, rather than making the LEDs represent information about CO_2 emissions. For example, the group asked for help to make the LEDs light up in green several times, even though we told them that the lamps would only light up in green when CO_2 emissions in Denmark were low. They simply wanted the LED lamps to *always* light up following a specific color pattern, regardless of the changes in the CO_2 emission data.

That is, they demonstrated both resistance to making changes and a desire to place priority on the attractiveness of their product rather than on it actually responding correctly to the data.

What students think about computer science

Our oral assessment with the students showed that, after the course, they still had a narrow understanding of what a *programmer* does when we asked them about this specific term. For example, one student replied that a programmer is a person who: "writes a whole lot of code and tests it and sees and something like that … and doesn't really do much else at work than write code," and another student replied that a programmer is: "someone who knows a lot about different mechanisms for a computer, because I think they work a lot at a computer." This understanding was broader when we discussed the term further and asked what a *computer scientist* and a *designer* do.

One student explained that a designer: "asks himself questions: 'Is this good enough? Does it have all these requirements?' (..) And if it doesn't, he'll start all over again. (…) You keep trying." Another student asked: "Weren't there three, he builds… he designs a prototype (…) and then he moves on to design, prototype, design, prototype, test… and then he goes back?" When commenting on the work a computer scientist does, the students said that a computer scientist is: "a very clever and patient person." Another student added: "a very precise person, because you have to be very precise when you write those codes", to which a third student replied: "Yes, precise and patient (…) and someone who can cope with many things." A fourth student pointed out that a computer scientist is: "someone who can think very broadly (…)", and a fifth pointed out that it is: "someone who has a good overview. Because he must always be able to see if there's an error somewhere." About errors, another student said: "I accidently typed a letter in small caps when it should have been large caps. And that's something you don't notice so easily."

Based on these comments, we conclude that a few of the students have experienced that a computer scientist must be able to think broadly, whereas some of the other students still have the very narrow understanding of what a computer scientist does that we were concerned about. The students' answers may reflect the way in which we asked the question – as two separate areas of work – but we still believe that it is possible to conclude that a course as short as ours is not enough to give the students an understanding of the fact that computer scientists do not merely write code. That they also sometimes work with design, for example.

Changed perceptions and new opportunities for success

Even though the students still did not fully understand the complexity of computer science at the end of the experiment, we did observe other changes in their perceptions. For example, some of the students had expected the assignment to be much more difficult: "When you first explained what we were to do, I thought: 'Okay, I don't think this will go so well,' but then when I got started and actually sat quietly and worked, and it was explained properly, then I thought it was fun." Another student said: "I think it was much easier than I had expected. I've sat there and seen the big companies that just write 100 codes in ten minutes and thought: 'If that's what we have to do… I give up'."

And another student, who said he is planning a career in IT, commented: "I thought this really was a lot of fun. Of course, there was also some new stuff, and it's something I'm interested in, so (…) and I'm going to use it later when I work in IT too. I was also quite surprised, because I'd expected it to be really, really difficult and take a really, really long time, but it was quite fast, and it's much easier than I'd thought it would be."

Several of the students used male pronouns when referring to both a designer and a computer scientist. When we were talking about what a computer scientist does, we asked about this and discussed whether it was always a "he". One of the students replied: "Or she. It's just a person." We asked if they thought that being a computer scientist is a profession for men, to which the student answered yes. Another student replied: "I think it's something that everyone, both men and women, can do, because my mother has also worked with it," and a third replied: "Everybody can do it. Equality." A fourth student said: "We've all done it in here", while a fifth explained how his perspective had changed, because in the beginning he had thought that only men could be computer scientists. But now: "For example, we've met Maja, (…) and she also works with stuff like that. So I guess that means there are just as many women as there are men." These statements emphasize how important it is that we talk about and make visible the fact that computer science is not a gender-specific discipline.

One of the students also expressed surprise at what they were capable of: "I'm surprised at how good Kasper and I are at designing. (…) I've always been bad at building things, and now I've built this. I'm happy." The student's experience does not just reflect a changed perception, it also shows that the design element can provide new opportunities for success. That the link between developing competencies in computational thinking and design thinking can potentially give students new opportunities for experiencing success.

Creative breaks and varied ways of working

By coupling computational thinking and design thinking, we can also provide students with more varied teaching activities that alternate between different domains; this helps prevent lessons from becoming monotonous. Among other things, this was seen in the way some students became invigorated when they were presented with a creative task. One of the students said: "I thought it was a bit boring (…), and I didn't respond very well to it, but I really liked it when we were creative and when we had to build that thing. That was kind of fun."

Another student felt differently: "I think it's been really fun (…) I've learned a lot, and I didn't think any of it was boring, but I think the most fun part was when we built stuff and had to share our ideas."

Working together in pairs when programming also motivated the students because they got to alternate their roles. One of the students said: "The most fun bit was the navigator and driver thing. (…) I thought that was a lot of fun, because we had to try and rephrase things so the other person understood it."

We had designed worksheet #1, which showed how to program LED strips, so that the students began by working with LED strips without using loops. Loops had the advantage of shortening the code so the students would not have to write the same code over and over again. We wanted to ensure that the students did not have to learn too many concepts at once, and to ensure that loops were introduced at a point when the students could see that using loops would make their task easier. We did this based on the expectation that if the students had learned about loops at the beginning – without having experienced what it was like not to use loops – they would not realize how useful they are.

When we asked the students about this part of the lesson, they had different responses. One student said: "I thought it was boring, but some of it was fun too. For example, I didn't know you could do this programming (…) with LED lights."' We asked what had been boring, and the student explained: "That you had to sit and write it [lines of code that were to be repeated], even though there was a faster way."

We talked about how we had designed the task to be boring in the beginning with repetition of lines of code, so the students would discover later on that there was a smarter way.

This was a first experiment. Based on the students' assessment and our own assessment, we find that the lessons were successful; however, some changes in culture are needed, as is more experience with the methods used if the students are to become independent, reflective and innovative computational designers.

Conclusion and perspectives

The purpose of this experiment was to examine a way of teaching computational thinking and design thinking in Danish K-9 schools by developing a computational design for solving an authentic problem, and through this to identify the subject-specific didactic possibilities and challenges of this kind of teaching. We designed the teaching as a design experiment based on a problem within science for which the students were to develop a digital design that could help people reduce CO_2 emissions caused by their electricity consumption.

In the experiment, we used two models in particular that we applied didactically both before and during the experiment. We used Naur's model (Figure 1) of the relationship between problems, tools and people to clarify the goal of developing a solution to a *problem* in the world for *people* by using *tools*. In this respect, it is important to emphasize that, in the real world, being able to program a perfect LED is worthless if no one realizes the applicability of your design. For example, as one group discovered, if human users cannot distinguish between the different colors of the LEDs, it makes no difference that the LEDs were actually programmed correctly. There must be a conscious relationship between the problem, people and tools, and we must bear all three elements in mind when working with the area at schools. We used the design-thinking model (Figure 3) when planning the different stages of the experiment and as a guiding principle throughout to help us maintain focus on where we were, what we had done and what the next step would be. Thus, we combined computational thinking and design thinking with the concept Denning has termed *computational design* (Figure 2).

Our analysis was based on our two primary hypotheses: that we would encounter some resistance from the students with regard to changing their designs to accommodate user feedback, and that they would equate computer science with programming. These hypotheses were partially confirmed. However, with regard to the students' resistance to making changes, the fact that we had chosen a controlled teaching format meant that students who resisted were 'forced' to change at least one thing in their design. In the long term, the intention is for the students to be able to see the possibilities that lie in user feedback and in making adjustments. With regard to what the students thought computer science entails, at the end of our course, a few students understood that computer scientists must be able to think broadly, whereas others had a narrow understanding of a computer scientist as someone who (only) writes codes.

In our analysis of the experiment, we also identified other opportunities and challenges, including changed perceptions of and new opportunities for positive experiences with computer science; positive experiences with creative breaks and varied ways of working; the positive impact of authenticity with regard to the students' understanding; and the desire for a greater degree of freedom. The latter arose from the fact that *we* had chosen the problem the students were to work with to model how a computational designer works. Focused efforts on changing cultural aspects and strengthening project-oriented working methods can over time lead to students being given a greater degree of freedom. Giving students freer rein in their work would, however, require a high degree of independence and changes in culture, and we would need more time to work with these aspects in the classroom.

In this regard, the context and culture of a school affect the ways in which the students are used to working. It was a challenge that we did not know the students in advance, just as we were not familiar with the conventions, cultural codes and ways of working usually at play in the class. Major cultural changes do not happen overnight – they require time and repetition – and it would be unrealistic for us to constantly 'discipline' the students or expect that they change their habits over the course of a three-day experiment. Therefore, we made a conscious choice not to focus on changing the culture in the classroom, but instead focus on the subject-specific content. This meant that, to the extent possible, we followed the conventions of the class and let their regular teachers, who were present for most of our lessons, keep track of, for example, the attendance register, any internal social problems, and any students who left the classroom in the middle of a lesson. For example, some students were more present both physically and mentally in class than others, and we experienced a number of social and academic challenges that we were not able to work with in the short period of time that we were there.

In addition to contributing to the didactics of technology comprehension, the aim of this experiment is also to inspire further subject-specific didactic research within the field of teaching computer science (in Denmark, known as technology comprehension[13]) in K-9 schools. By making our lesson plans, slides and worksheets freely available, we hope that others want to test or further develop our theory and practice.

References

Barab, S. & Squire, K. (2004). Design-Based Research: Putting a Stake in the Ground. *The Journal of the Learning Sciences*, 13(1): 1-14. https://doi.org/10.1207/s15327809jls1301_1

Caeli, E. N. (2021). *Computational Thinking in Compulsory Education: What, Why, and How? A Societal and Democratic Perspective*. PhD Dissertation. ARTS, Aarhus University.

Caeli, E. N. & Bundsgaard, J. (2020). Teknologikritik i skolen – et demokratisk perspektiv på teknologiforståelse. In: Haas, C. og Matthiesen, C. (Eds.). *Fagdidaktik og demokrati*. Samfundslitteratur.

Cross, N. (2011). *Design Thinking: Understanding how designers think and work*. Bloomsbury.

Denning, P. J. (2017). Computational Design. *ACM Ubiquity*. Volume 2017, August: 1-9. https://dl.acm.org/doi/10.1145/3132087

EMU (2019). *Formålet for forsøgsfaget teknologiforståelse*. https://www.emu.dk/grundskole/forsogsfag-teknologiforstaelse/formal. Undervisningsministeriet.

Hattie, J. & Timperley, H. (2007). The Power of Feedback. *Review of Educational Research*, 77(1). https://doi.org/10.3102/003465430298487

Naur, P. (1965). *The Place of Programming in a World of Problems, Tools, and People*. Proc. IFIP Congress 65: 165-199.

Naur, P. (1970). *Planer og ideer for datalogisk institut ved Københavns Universitet*. Studentlitteratur.

Katapult/TEACH (2013). *Værktøjskassen: Model for designtænkning*. The project Next Generation. University of Copenhagen. https://innovation.sites.ku.dk/model/design-thinking/

Klafki, W. (2001). *Dannelsesteori og didaktik – nye studier*. Klim.

Smith, R. C.; Iversen, O. S.; & Veerasawmy, R. (2016). Impediments for Digital Fabrication in Education: A study of teachers' role in digital fabrication. *International Journal of Digital Literacy and Digital Competence*, Vol. 7(4). https://doi.org/10.4018/IJDLDC.2016010103

Tabel, O.; Jensen, J.; Dybdal, M.; & Bjørn, P. (2017). Coding as a social and tangible activity. *Interactions*, 24(6): 70-73. https://doi.org/10.1145/3137099

[13] Even though our teaching design is not based on the curriculum for the experimental subject technology comprehension, there is considerable overlap between our design and several of the areas of competence included in the curriculum, and it fits in with the broad Bildung perspective that we see in the curriculum, as well as with the traditions and overall objective of Danish schools.

Appendix 1. Lesson plan

This appendix provides an overview of our lesson plan. The slides and worksheets (in Danish) are freely available and can be downloaded at: https://silo1.sciencedata.dk/shared/designeksperiment

Day 1

Objective	Introduce to the course and academic content Develop an understanding of the problem	Develop an understanding of data – specifically from electricitymap.org Practical preparations	Develop programming competencies
Contents	**8.00-9.50**	**10.10-12.00**	**12.30-14.00 / 14.15-15.00**
	20 min.: Introduce ourselves, the experiment and the overall plan – show design model **Understand the problem** **20 min.:** Multivoiced dialogue: Introduce the problem/activate pre-understanding of the climate issue Video **20 min.:** Discuss what we can do about climate change **Understanding data/delimitation** **20 min.:** Energy consumption versus production in Denmark **15 min.:** Set up computer, log on, connect to Wi-Fi **15 min.:** Explore electricitymap.org **10 min.:** Sum up and set the stage for the solution we will be working with	**Ideas** **15 min.:** Design adapted to the problem Why wouldn't an app work? Compare with other data indicators found in the home **Prototype** **10 min.:** Introduction to programming in pairs Advantages/disadvantages? **10 min.:** Introduction to microcontrollers, LED strips and the Mu editor **90 minutes:** Worksheet #1 Programming in pairs in practice – time it so they swap roles Mu editor, connecting the microcontroller, connect the LED strip, change colors, make a pattern Introduction to variables Change their code to use variables	**Prototype** **2 hours and 15 min.:** Worksheet #2 Introduction to logging on to the Wi-Fi and retrieving data from ElectricityMap Add code to retrieve CO_2 contamination level from ElectricityMap Introduction to *if* sentences Change the code to change colors based on CO_2 level
Materials	Computer, access to electricitymap.org, 4G Wi-Fi-hotspot	Computer, microcontroller, LED strip, 4G Wi-Fi-hotspot, worksheet	Computer, microcontroller, LED strip, 4G Wi-Fi-hotspot, worksheet

Day 2

Objective	Develop design competencies	Develop programming competencies	Develop design competencies
Contents	**8.00-9.50**	**10.10-12.00**	**12.30-14.00**
	10 min.: Return to the problem: Where are we in the design process? **5 min.:** Introduce to part about design **Ideas** **20 min.:** Multi-voiced dialogue about design: What is a good/bad design? What should you think about when interacting with users? What is important in a design? Write points made on the board **25 min.:** Inspiration from users. What do other people think is good design? Video **20 min.:** Introduce prototype: Sketch on paper before they build it. Build using cardboard, glue, etc. Consider the final version (colors, materials, etc.) **30 min.:** Generate ideas and start building a prototype	**Ideas and Prototype** Own pace Continued idea development and construction of prototype Ask them to proceed with worksheets #3, #4 and #5 Individual supervision, depending on level and speed	**Prototype** **2 hours:** Completion of prototype **15 min.:** Testing own prototype
Materials	Computer, microcontroller, LED strip, 4G Wi-Fi-hotspot	Building materials for prototypes (cardboard, glue guns, flamingo, markers, tape, staples, scissors), computer, microcontroller, LED strip, 4G Wi-Fi-hotspot	Building materials for prototypes (cardboard, glue guns, flamingo, markers, tape, staples, scissors), computer, microcontroller, LED strip, 4G Wi-Fi-hotspot

Day 3

Objective	Develop design competencies Develop feedback competencies	Develop design competencies Develop feedback competencies	Develop communication skills Assessment
Contents	**8.00-9.50**	**10.10-12.00**	**12.30-14.00**
	10 min.: Sum up days 1 and 2 using the design model **Testing** **15 min.:** Introduction to feedback: work in pairs. Explain the design, functions and challenges to each other **20 min.:** Test each other's prototype + give feedback based on the criteria for a good design from day 2 **Prototype** **30 min.:** Revision of design/prototype before presentation: change at least one thing in either the code or design	**Testing** **90 minutes:** The groups present and explain their codes in plenary session **10 min.:** Introduction to presentation, including scope of content and time per group **40 min.:** Prepare and practice presentation	**Testing** **90 minutes:** Presentation of prototypes for each other, another class from the same grade and teachers **45 min.:** Multi-voiced assessment of the project, e.g.: What have they learned that they didn't know before? What worked well? What didn't work so well? What have they learned about data? What have they learned about design? What have they learned about programming? What surprised them?
Materials	Computer, microcontroller, LED strip, 4G Wi-Fi-hotspot	Computer, microcontroller, LED strip, 4G Wi-Fi-hotspot	Computer, microcontroller, LED strip, 4G Wi-Fi-hotspot

Authors

Elisa Nadire Caeli

PhD student

Danish School of Education (DPU), Aarhus University and Department of Teacher Education, University College Copenhagen

Conducts research into students' development of computational thinking and technology comprehension in K-9 education

Martin Dybdal

PhD, special consultant

Department of Computer Science, University of Copenhagen

Conducts research into and works with computer science didactics and communication from K-9 education to university level