From programming to computational perspectives in higher educations for humanities students

Møller, Anders Kalsgaard; Kaup, Camilla Finsterbach; Brooks, Eva; Gnaur, Dorina; Schürer, Maja Højslet; Lyngbye, Marie Charlotte

Published in:
Acta Didactica Norden

DOI (link to publication from Publisher):
10.5617/adno.9183

Creative Commons License
CC BY-NC-ND 4.0

Publication date:
2022

Document Version
Publisher's PDF, also known as Version of record

Link to publication from Aalborg University

Citation for published version (APA):
Møller, A. K., Kaup, C. F., Brooks, E., Gnaur, D., Schürer, M. H., & Lyngbye, M. C. (2022). From programming to computational perspectives in higher educations for humanities students. *Acta Didactica Norden*, *16*(4), [9183]. https://doi.org/10.5617/adno.9183

Anders Kalsgaard Møller
Aalborg University
Camilla Finsterbach Kaup
Aalborg University
Eva Brooks
Aalborg University
Dorina Gnaur
Aalborg University
Maja Højslet Schürer
Aalborg University
Marie Charlotte Lyngbye
Aalborg University

DOI: https://doi.org/10.5617/adno.9183

From programming to computational perspectives in higher educations for humanities students

Abstract
Due to the increasing need for IT competencies, university humanities programs have started introducing courses to strengthen students' understanding of informatics. This paper studies how students in a master's program in IT at a humanities faculty developed skills in programming and computational thinking. All students had a compulsory course in Programming and Prototyping, and some of the students had electives in Computational Thinking. Data consisted of observations from the courses, assessment of students' assignments, and four focus group interviews (two groups – one with Computational Thinking and the other with Programming and Prototyping only). We held interviews before and after the courses for both groups to uncover how the students' views changed. Both groups of students saw themselves in coordinating roles where they would collaborate with programmers and other software developers. The students who took the electives in computational thinking showed a richer vocabulary when describing computational concepts, practices, and perspectives. The ability to reflect on the practical tasks, including concepts, practices, and perspectives, seems essential for students' future careers, as humanity students working with technology. The results show how students can develop their understanding of computational thinking through scaffolding for computational empowerment. In the process, we saw how students achieved a computational understanding through working with concepts and practices and where perspectives emerged from combining the computational understanding with information and practices from other disciplines.

Keywords: computational thinking, higher education, humanities, programming

Fra programmering til perspektiver for computationel tankegang for humanistiske, videregående uddannelser

Sammendrag

Med udgangspunkt i det stigende behov for IT-kompetencer er de humanistiske universitetsuddannelser begyndt at indføre kurser, der skal styrke de studerendes forståelse for informatik. Denne artikel undersøger, hvordan studerende på en masteruddannelse i IT på et humanistisk fakultet udvikler kompetencer i programmering og computationel tankegang. De studerende havde alle et obligatorisk kursus i Programmering og Prototyping, og nogle af dem havde valgfag i Computational Thinking. Data består af observationer fra kurserne, vurdering af de studerendes opgaver og fire fokusgruppe-interviews. (De studerende var delt i to grupper – en med Computational Thinking og den anden med kun Programmering og Prototyping.) Vi afholdt interviews før og efter kurserne for begge grupper for at afdække, hvordan elevernes synspunkter havde ændret sig. Begge grupper af studerende ser sig selv i koordinerende roller, hvor de samarbejder med programmører og andre softwareudviklere. De studerende der valgte valgfaget i Computational Thinking, viste et rigere ordforråd, når de beskrev computationel tankegangs begreber, praksisser og perspektiver. Evnen til at reflektere over de praktiske opgaver, herunder begreber, praksisser og perspektiver, synes essentiel for de studerendes fremtidige karriere, som humanister, der arbejder med teknologi. Resultaterne viser, hvordan studerende kan udvikle deres forståelse af computationel tankegang med stilladsering mod computational empowerment. I processen så vi, hvordan studerende opnår en forståelse af computationel tankegang gennem arbejde med koncepter og praksisser, og hvor der opstod perspektiver ved at kombinere computationel tankegang med information og praksis fra andre discipliner.

Nøgleord: computationel tankegang, videregående uddannelse, humaniora, programmering

Introduction: Computational thinking in higher education

Informatics has become an important subject in the Danish educational system and can be seen as a discipline for understanding technology more profoundly and the knowledge behind computer programs (Caspersen et al., 2018). To be prepared for the jobs of the 21st century, students at higher educations (HE) in humanities must not only be digitally literate but also understand key concepts related to informatics.

Computational thinking (CT) has been on the agenda when preparing students for a digital reality in both K–12 and in HE. CT can be seen as a thought process, used in formulating a problem where the solution(s) is expressed in such a way that a computer, human or machine can solve it (Wing, 2006). Brennan and Resnick (2012) have further suggested that CT could be divided into three dimensions: concepts (the concepts designers engage with as they program), practices (the practices designers develop as they engage with the concepts), and perspectives (the perspectives designers form about the world around them and about themselves). In research on CT in education, most attention has been given to K–12, especially Science, Technology, Engineering, and Mathematics (STEM disciplines) (Barr & Stephenson, 2011). According to Wing (2020), scientific environments in general have put a lot of emphasis on CT, whereas the field of humanities, however, have not to the same extent been empirically substantiated. CT should, according to Wing (2006), be a formative skill such as reading, writing and arithmetic. In this perspective, CT should help to process information and tasks systematically and efficiently (Lu & Fletcher, 2009).

According to Lu and Fletcher (2009), pedagogical challenges exist, such as how to teach CT or if programming is required to learn CT: "… teaching CT should focus on establishing vocabularies and symbols that can be used to annotate and describe computation and abstraction, suggest information and execution, and provide notation around which mental models of processes can be built" (Lu & Fletcher, 2009, p. 1). However, students must also understand CT as a thinking process throughout their education to better prepare for learning programming, career opportunities, and intellectual content. In line with that, Jonasen and Gram-Hansen (2019) have examined the potential of problem-based learning (PBL) as a pedagogical framework for developing CT skills in HE. The study included 20 students in a pre- and post-design, where CT was introduced at the conceptual level. The CT skills should provide the students with reflections and a vocabulary to explain their skills. The study aimed to look at differences in the students' CT skills after completing a PBL process. In the first benchmark, a few students could reflect and relate CT skills to their bachelor project. The second benchmark showed that the students had developed a deeper understanding of their problem-solving process. The students' competencies revealed rich means of expression in terms of CT skills. Students related, e.g., specific methods to the process of decomposition. The study found that the PBL approach had increased the students' understanding of the different CT skills to the extent that they could relate them to their practices. This increase was interesting because CT had not been a distinct theme, but a perspective brought for the students' reflection.

Humanities and CT

There is not much research regarding how and why to teach humanity students about CT. A study that does address the topic shows how humanities students (hybrid Humanities/ICT) at HE enrolled in a programming-related course are most interested in learning to program and use it as a tool rather than an ability (Chongtay, 2019). Chongtay has conducted three years of surveys applied to eighty-three humanities students; almost a third expressed interest in being directly involved in future programming jobs. According to the survey, almost half of the students expected to learn programming at a basic level. In contrast, a little less than a half expected to learn up to an intermediate level, and just a few wished to learn to a high proficiency level. As stated by Chongtay, students' motives for choosing a HE program that combines humanities and ICT fall into four categories:

- Improving existing skills in both humanities and ICT
- Finding the combination exciting and relevant for businesses
- Expanding job possibilities
- Learning ICT skills (Chongtay, 2019, p. 166)

Those categories are relevant to be aware of when designing learning activities at HE programs that combine their interest in CT and programming (Chongtay, 2019).

Our scope

In this paper, we enlighten how students at a master's program in IT at a faculty of humanities develop programming and CT skills through a PBL approach to learning. Based on empirical findings from the study of courses in programming and CT, we explore the ways in which students develop their understanding of CT, and how this affects their own professional digital competencies. We apply Brennan and Resnick's (2012) framework to examine how students develop their computational concepts, practices, and perspectives. While we find the division between concepts, practices, and perspectives useful, we believe that the overall framework is too narrowly focused on understanding the concept of computing, programming, and software development. Thus, we further broaden the framework to also include computational empowerment (CE), defined as:

> the process in which [people], as individuals and groups, develop the skills, insights and reflexivity needed to understand digital technology and its effect on their lives and society at large, and their capacity to engage critically, curiously, and constructively with the construction and deconstruction of technology. (Iversen et al., 2018, p. 1)

With the addition of CE, the perspectives are no longer limited to programming skills but also include other perspectives that are better suited to refer to the future multivarious job roles that humanities students partake in. Some of these roles or knowledge areas are made explicit in a course in "technology comprehension" developed by the Danish Ministry of Education (Ministry of Education, 2018) for teaching at elementary schools in Denmark. However, we argue that graduate students need similar knowledge or skills, though, on a different level. The course focuses on four different topics. For transparency, we use the same translation as Dindler et al. (2020, p. 75):

- *Digital empowerment*: Focus on how digital artifacts shape our lives and society and promote an analytical and critical approach to digital artifacts and transformation
- *Digital design and design processes*: Focus on the process and the choices the designer makes in the process of designing digital artifacts

- *Computational thinking*: The ability to analyze, model and structure situations with the purpose of understanding or designing digital artifacts
- *Technological ability*: The ability to express computational ideas in digital artifacts. The ability to make informed choices about the use of technologies

These topics do not represent an exhaustive list of perspectives but show some of the overlapping areas that at least partly cover the perspectives. Students may need to mix knowledge and competencies in other ways than the ones mentioned above. Furthermore, Dindler et al. (2020) argue that the above topics are still too narrowly focused on the development of technology and to a lesser extent raise questions about how technology affects us. This is an important part of CE and lead to questions such as how digital technology challenges our democratic rights, how it alters our personal relations and practices, and how we interpret intentions embedded in everyday technology (Iversen et al., 2018).

As our focus is on humanities in HE and how we, as an education, can prepare students for a professional life after graduation, we believe that a broader definition of especially the perspectives better reflects the different roles that humanities students engage in after they graduate. In this article, we seek to answer the following research questions:

1. How do humanities students in higher education develop their programming and computational skills?
2. What should humanities students learn from programming and computational thinking to meet the requirements for their future careers?
3. How do we best support the humanities students' learning and development of these competencies?

In the following section, we present previous studies that address CT and learning. That section also introduces different forms of scaffolding used in the analysis of the empirical data.

CT and learning activities

Czerkawski and Lyman (2015) conducted a literature review to discuss whether CT is relevant in HE outside of STEM fields. The authors found that experts conceptualize CT within the field of computer science (CS) and that engaging humanities students in CT projects requires a field-appropriate introduction to CT principles. However, spreading CT to other academic areas requires adapting the existing CT theory to match the needs of novices and other non-specialists outside the CS field. They further identified that the use of CT in the humanities field can expand the range of human creativity by including CT. CT requires other, and more specific instructional design strategies. However, teaching CT is often included as an element of other thinking skills, similar to teaching general critical thinking or problem-solving skills. Czerkawski and Lyman (2015) recommend further studies to establish a better connection between CT and HE outside the CS field. On the contrary, Romero et al. (2017) define CT as a skill transferable to other fields than just CS. The interactive nature of computer programs allows students outside CS to represent and test models of a phenomenon, supported by a prototype-oriented approach. According to that, programming should be a part of a pedagogical strategy and not only a technical tool to learn coding techniques.

Romero et al. (2017) investigated how a total of 120 undergraduate students were engaged in a story2code creative challenge. Story2code consists of short text-based stories that engage learners in analyzing, modelling, and creating a Scratch project to represent a story. Scratch is a block-based programming language (https://scratch.mit.edu/). The result indicated that the learning design should include the students' Zone of Proximal Development (ZPD) (Vygotsky, 1978) and refers to and offers the students an appropriate degree of newness, a certain ambiguity, or ill-definition problems to engage the students in a creative learning process. The learning design should not be too guided or structured; otherwise, it loses creativity, and the students' ill-definition problems become easy to solve.

One of the learning components to support the students through the ZPD is scaffolding. Scaffolding has been emphasized as crucial for learning programming (Zhang et al., 2021). Zhang et al. introduced a step-by-step scaffolding approach to improving HE students' CT and academic outcome in a programming course. They used feedback and flowcharts to scaffold students' programming and CT skills. In a quasi-experiment by Zang et al. (2021), students used flowcharts to help decompose programming tasks and support their problem-solving process. The experimental group that used flowcharts showed a higher level of CT skills. According to Lye and Koh (2014), scaffolding and reflection activities can help to foster computational practices and perspectives.

Scaffolding

Scaffolding is typically linked to Vygotsky's social theory of learning (Vygotsky, 1978). Wood et al. (1976) utilized a scaffolding metaphor to explain how a more expert individual assists in joint problem-solving activities with students. The metaphor refers to the direction and assistance given to students to make them able to perform at a level that is higher than students ordinarily could sustain by themselves. Scaffolding supports the creation of higher-order actions so that students can perform the required acts of internalization, appropriation, or reconstruction that eventually bring development changes (Wood et al., 1976; Mascolo, 2005). Mascolo (2005) emphasizes that scaffolding as a metaphor describes the development as a social process. However, the concept of scaffolding provided by Wood et al. (1976) focuses primarily on construction activities carried out by an expert. Fundamentally, the scaffolding metaphor fails to consider how the students contribute to the scaffolding process (Mascolo, 2005). In this regard, Mascolo suggests that scaffolding should imply a coactive dimension. Coactive scaffolding involves "any process outside of an individual's direct control that functions to direct individual action toward novel or higher-order forms" (Mascolo, 2005, p. 187). There are three broad categories of coactive scaffolding proposed and illustrated: ecological, social, and self-scaffolding. In this paper, we draw on the concepts of self-scaffolding concerning the students' development of CT skills.

The traditional understanding of scaffolding is defined by the effects an expert has on a student's action. By involving the student's development in the scaffolding process, the understanding of the student's activity increases, and the student becomes coactive in the process by self-scaffolding. Self-scaffolding relates to students' previous actions creating preconditions and helps guide and support new forms of actions and meaning in other contexts. Mascolo (2005) highlights three forms of self-scaffolding:

- *Cognitive self-scaffolding* occurs when a student performs actions that directly or indirectly change behaviors that create new opinions or changes in the cognitive structure.
- *Bridging* refers to how the student uses partial knowledge to create a structure that helps to bridge old and new knowledge. This bridging is due to an ability to function at two levels of development towards an acquired competency.
- *Analogical mappings* are closely related to bridging. Using analogical scaffolding, the students draw on past action experiences from similar problems and transfer them into new structures to help solve the new problem. The use of analogies in connection with self-scaffolding works best when the student is aware of the connections between existing strategies and new problems.

By involving the student's development in a scaffolding process, the understanding of the student's activity increases. The student becomes coactive in the learning process through the learning design, which guides the student to do self-scaffolding.

Research design, data, and methods

In this section, we first describe the context of our study – the education and courses from which we collected our data. This is followed by the data collection methods and finally our data analysis approach.

Research context

This study's empirical data comes from observations, interviews, and assignments from students in the second semester of a humanities master's program. In that semester, the students have a mandatory course in Programming and Prototyping; in addition, they can choose to follow an elective in Computational Thinking. The students in this program follow a PBL approach to teaching and learning. The education includes different IT-oriented modules as well as modules focusing on learning and organizational change. Due to the COVID-19 pandemic, all teaching was conducted online on Zoom.

Programming and Prototyping (PP)

72 students participated in this course. The students learned the basic concepts of software programming, software development and built prototypes using a micro:bit (https://microbit.org/) and different electronic components and design artifacts. We provided each student with a micro:bit kit including a board and electronic components which they could use during the course. Before each lecture, we made a preparation video that introduced the students to different concepts along with different training exercises with micro:bit using block-code in the micro:bit Makecode environment. This way the students could learn the basic concepts at their own pace. During the actual lectures, we introduced new concepts, answered questions, and did different exercises such as pair programming. We also introduced a programming project, inspired by a design-based learning approach (Puente et al., 2013) and similar to the suggestions in Romero et al. (2017) concerning a more open learning following a design process. The students were tasked with creating a design vision for a programming project of their choosing. The students would work on the project throughout the course. After each lecture they had to report on the process of the project, e.g., what problems that they had, if their design vision had changed etc.

Computational Thinking (CT)

31 students participated in this course. The teaching related to the CT course consisted, among other things, of traditional lectures over Zoom, flipped classroom lectures, and peer discussions. Three times during the elective course, the students were given an assignment that consisted of analyzing a technology or a software program, related to concepts of CT and the literature from the lectures. This could for instance be Beebots or Scratch. The teaching consisted of five teaching sessions of 4 hours duration each. The teaching was structured so that it started with a presentation from the teachers, then the students had to discuss the presentation and the literature for the course in smaller breakout rooms in Zoom. After discussions, the students worked on their assignments with assistance from the teachers of the course. The students could thus get help and assistance if needed during their assignments.

Data collection
Before and after the two courses, we held focus group interviews through Zoom to uncover how the students' views on the two topics had changed. The participants in the focus groups consisted of four students from PP and three students who selected the CT elective. Brennan and Resnick's (2012) framework of concepts, practices, and perspectives informed the interview guide. We observed all students (n=72) during the classes and analyzed their assignments and course activities. The students who participated in the focus group interviews were recruited from the pool of the 72 students. We asked all students if they wanted to participate, and those who accepted participated. All students who participated in the study gave informed consent.

Data analysis
We have used a multiple data approach to analyze different components using both deductive and inductive coding following a thematic approach to data analysis inspired by Braun and Clarke (2006). The first step in the data analysis was to transcribe the interviews and start initial coding (Ravitch & Carl, 2021) using an open coding approach. To get a broader picture of the students' development of CT, we used Brennan and Resnick's (2012) framework to create the themes of concept, practices, and perspectives; this made it possible to analyze the different components of the students' learning from PP and CT. We compared the two different groups of students regarding their development of CT to look for alternative explanations and differences between the groups. During the coding, new themes would also emerge from the data. To validate the coded data, two of the authors have coded the same data to look for shared understanding, and they developed, refined, and codified the data in an iterative process to create a more complex sense of it (Ravitch & Carl, 2021; Creswell & Cresswell, 2018). In the end, it was decided to include the themes of scaffolding and employment competencies in the analysis along with the deductive coding themes. The selected quotes for the data analysis were translated from Danish to English and reproduced to capture their meaning rather than a literal translation. The analysis in the paper shares a rich example of quotes to contextualize the context of humanities students' development in the PP and CT courses.

Results and analysis

In this section we present the results and analysis of the empirical data. First, we analyze how the students build their programming and computational thinking skills using different scaffolding techniques (research question 1). In this process they first develop a computational understanding and then use the computational understanding to learn new perspectives. We then proceed to analyze the knowledge and competencies needed for future careers (research question 2).

Finally, we present a model of how the students develop these competencies and how the learning process can be supported (research question 3).

Concepts and practices

The PP course design conceived programming as a tool, where students learned the basic theoretical and practical principles through a mini-project. The PP course worked as a scaffolding where students developed concepts and practices when working with the micro:bit. The low-level programming language based on block programming enabled the teachers to scaffold the students' learning by working in small steps. "I like to learn the basics first, like what a loop is. What are these things? So, the teaching has worked well for me." (Student A, PP)

Our observations and the students' comments during the interview indicated that the students learned the concepts and practices in three steps. First, they worked with understanding the different components. By components, the students referred to both the concepts and the electronic components such as sensors, LED, motors etc. Second, the students started to learn how to put the different components together, e.g., how to make different input and output devices work, using different concepts. Last, they applied this knowledge to explore different ideas and to code their programs.

> … It was first to get a basic understanding of what all these different components can do and how you put the blocks together, what happens then? After that, you put them together and try something, an idea or program. (Student D, PP)

In the process, the students expressed that they used a trial-and-error inquiry approach or a learning-by-doing approach (Dewey, 1923).

> … To try a lot of things and find out what the different components can and get an idea of what happens when I do this? What happens when I do this? (Student A, PP)

> For me it helped a lot to get it hands-on and test it out and try and fail a bit and try and redo it. (Student A, PP)

According to Wood et al. (1976), students need to be supported in their learning process, and both groups of students felt supported by the learning design. Especially, the recorded videos worked as a scaffold that led the students to work at their own speed and to develop their computational understanding.

> I think it has been nice to have been able to sit with it myself. And by that been able to nerd a little and look at it at all times of the day. So, if I get an idea, I try it out. (Student G, CT)

The learning design enabled students to develop concepts by themselves related to programming and do it asynchronously. The scaffold broke up the learning into smaller pieces and provided a tool and a structure that guided the students' learning. Through the work with programming, the students worked with practical actions and hands-on activities, which gave them proficiency in the work with micro:bit. Peer discussions were also a part of the learning design, and students used them as a part of their scaffolding to discuss essential vocabulary and get help if they needed that in an activity: "... discuss with each other or something? When we discuss it in the groups, or when we have these forums where we can talk and get wiser." (Student F, CT)

Both groups of students used peer discussions to become wiser and develop their understanding. Some of the students used peer discussions to learn from each other.

> I think it is an advantage to be at different levels. Both because I can learn something from someone who knows a little more than me, and the person that knows a little less can learn from me. And it just means a lot more if you have to tell someone how to do it. As I see it, it makes good sense to be on different levels. (Student G, CT)

The students used a scaffolding strategy where they primarily made use of cognitive self-scaffolding to learn the different components and concepts through working with the content of the course with a learning-by-doing process. Later the scaffolding strategy changed towards a more analogical mapping-based process as they applied their previous experience with the concepts in a new way when creating programs from their ideas as part of the mini-project. Working with concepts and practices, the students gained what we refer to here as their computational understanding. Brennan and Resnick (2012) refer to this as "The intersection of computational thinking concepts and computational thinking practices" and give examples such as "Is the learner able to analyze and critique their own and others' code?" and "Is the learner able to meaningfully put the concept to use in design?" While we did not see any difference between the students who took the elective in CT and the students who did not, in terms of their computational understanding, we did see a difference in how they articulated it. For example, the students from PP used examples from the programming classes, while students from CT would more often refer to the actual practices, e.g., talking about decomposition, debugging etc. in a broader perspective. Thus, they were showing a deeper understanding of what they were doing on a meta-reflective level.

From computational understanding to perspectives

The PP course gave the students a vocabulary to explain concepts and practices to develop their understanding of CT. Nevertheless, the students also used CT to reflect on their computational understanding in PP.

> I think programming has given you something, so for example, you may have learned some expressions like this. At least I used it for the task in IFDTT (CT course). Like booleans and conditions, it comes from literature in PP; there was nothing like this in the CT literature. I think they (the two courses) have been able to spar a little with each other. (Student F, CT)

The CT students used partial knowledge to grasp the computational understanding and made it into perspectives. The students used concepts from the computational understanding to explain their knowledge because the idea of CT was too hard to grasp: "... it is difficult to concretize. I think that's why I use programming to explain it." (Student F, CT)

The computational understanding also clearly opened new perspectives for the students. In the quote below, the student provided an example of what Brennan and Resnick (2012) refer to as questioning:

> If I encounter some small problem in everyday life or such, then I will automatically think about how I could solve it by programming myself, and I think, it is knowledge you can apply almost no matter what situation you are in, so to think the technology a little more into everyday problems, how it can be solved by something, i.e., by simple programming. I think that is at least the knowledge I will take with me in the future. (Student D, PP)

Concepts and practices became tools that students could use to bridge between earlier learned knowledge and perspectives in utilizing this knowledge in another context. When addressing perspectives, the students combined the computational understanding with their previous experiences to understand different perspectives.

> Well, if you just start with kindergartens and then these curriculum themes you have to work with, then there is this science aspect, it is something the authorities have decided, and you can always be skeptical about implementing technology in kindergarten. (Student A, PP)

> It is also about the understanding of technology. What exactly can technology do? Many believe that iPads are not something that children should have in their hands. It is better that they come out and get some mud in their face, and I agree with that to a large extent. However, if you start to think a little about what exactly technology can do that is otherwise not achievable for children, then technology becomes important. (Student A, PP)

The above examples indicate digital empowerment where the student can critically reflect on the use of technology. The second example also includes aspects of technological ability. Both groups of students scaffold knowledge from their computational understanding and their knowledge from other domains through bridging to create perspectives. Depending on the different knowledge they combine with their computational understanding, different perspectives emerge. The students that only have PP were able to see some perspectives but were not able to make distinctions between computational understanding and computational perspectives. The missing link between understanding and perspectives indicated that students with the elective in CT have a deeper understanding of the processes they had been engaged in and a richer vocabulary to describe CT processes, hence, the CT students again showed the ability to meta-reflect on their own processes and learning.

Knowledge and competencies for future careers

In the previous sections, we have seen how the students develop their understanding of concepts, practices, and perspectives. In this section, we analyze how the students see this knowledge and these competencies being used in their future careers.

The students recognized that technology today is ubiquitous and that they need to be able to understand and be able to work with technology. They further argue that CT competencies would make them more attractive on the job market.

Rather than being hired as programmers, the students see themselves as collaborating with programmers and in other ways working with technology in situations where they can apply their knowledge from CT.

> I am constantly thinking about future work. It's not going to be programming. There is always going to be someone who's better than me at programming. (Student E, CT)

> The PP course is necessary, so we get a vocabulary to talk to someone [a programmer] who knows how to program … We can also do a little… We understand when they say words like functions and stuff like that. And if they want to change something, then I have a fair overview of the CT. (Student F, CT)

They see themselves as "a link between, e.g., traditional humanities students and programmers" (Student F, CT). In this role, they need a computational understanding that enables them to make decisions and realistically consider options and time frames for specific tasks such as software development or implementation. To do so, it is important for them to be able to communicate with programmers and "be able to speak different languages" (Student G, CT) when talking with different people with various competencies and job roles. The following example shows how the understanding of concepts helps the students to communicate.

> … so, I discussed something with my husband who has an IT background and he said something like that, well you just have to have some Booleans and some loops and something like that, where I actually understood the words, he said. It was an eye opener for me. I did not expect that. (Student G, CT)

There is also a need to understand the different practices when planning IT projects such as being able to decompose the problems into smaller tasks to estimate the time frame and understand how programmers work in terms of iterations and increments, testing and debugging etc. In the interview, some of the CT students explained how they also used practices in situations not related to technology such as decomposition, breaking down a problem into smaller problems, when working with "wicked problems" (Rittel, 1984). Thus, the practices are not only usable to learn for students working with technology but can also be applied to other domains.

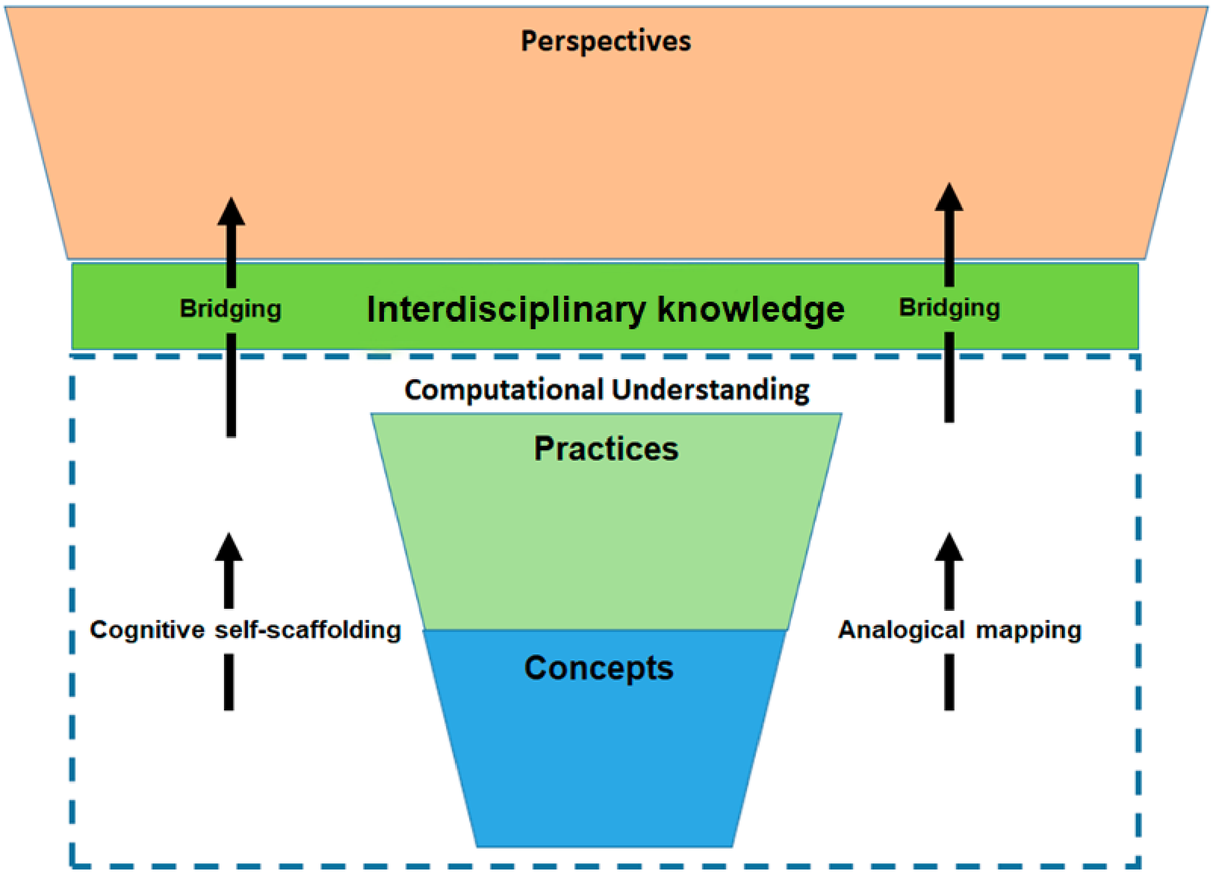
With the current ubiquity of technology and the graduate students' diverse roles as a link in technology development, implementation, and teaching etc., we argue that CT should not only be taught with the aim of learning to program but calls for a wider range of perspectives when working with technology. Thus, it is important for the students to develop their computational empowerment to be able to both analyze, discuss, and reflect on the use of technology in diverse settings and domains while we consider the perspectives the most important aspect to learn for humanities students in HE.

Development of knowledge and required competencies overview

Figure 1 shows an overview of how the students developed their CT skills through different scaffolding processes. The bottom of the model shows that students build a computational understanding by working on concepts and practices in relation to Brennan and Resnick's (2012) conceptual framework using cognitive scaffolding and analogical mapping.

When the computational understanding is combined with information and practices from other disciplines (interdisciplinary knowledge), through bridging, different perspectives emerge. Depending on the interdisciplinary knowledge combined with the computational understanding, we see different perspectives arise. The sizes of the concept box, practice box, and perspective box in Figure 1 indicate the relative importance for future job roles where the perspectives play the most important role followed by practices and concepts.

Figure 1. Overview of the development of knowledge and the required competencies for humanities students


Perspectives
Bridging
Interdisciplinary knowledge
Bridging
Computational Understanding
Practices
Cognitive self-scaffolding
Analogical mapping
Concepts

Discussion

In this study, we examined how humanities students developed their understanding of CT. The students gained a computational understanding through a PBL-oriented learning-by-doing process when working with the concepts and practices where they applied cognitive self-scaffolding and analogical mapping (Mascolo, 2005). The perspectives emerge from bridging the CT with the students' interdisciplinary knowledge. We argue that for humanities students the perspectives are of more importance than practices and concepts. However, both concepts and practices are needed and used to scaffold the learning of perspectives. We are aware that students enter very diverse careers, where the importance of CT competencies will vary from student to student. Obviously, there will be cases where concepts and practices will play a greater role; however, when we argue that perspectives play a larger role, it is from a general viewpoint where the development of CT competencies should not only be used for programming and development, consistent with Lu and Fletcher (2009), Dindler et al. (2020), and Chongtay (2019). According to the students, CT must be related to the field where they can find employment after graduation. In other words, it is crucial that CT teaching is aligned with students' field of study so that they can acquire an appropriate CT understanding (Czerkawski & Lyman, 2015). The way the students build their CT competencies will largely reflect the way they are taught, why other ways of learning may arise from different approaches to teaching (Lu & Fletcher, 2009). This raises the question of how to teach programming and CT in HE is moving forward and what the content should be. Should it be a course for meta-reflection, or should it support the construction of different perspectives? According to Lu and Fletcher (2009), CT should be something that all students are taught from lower level to HE, and it will thus become a skill they can use in their professional career and everyday life. Lu and Fletcher (2009) raised a question: whether programming must be part of CT to make the students' CT proficient; for them, CT can be taught through practices and repeated meetings in the school's regular subjects without using programming.

In our process of teaching CT, the students start with programming and thereby build an understanding of concepts and practices followed by the course in CT where they build on this knowledge and create the meta-reflection; we see the students achieve and become more aware of the processes. In this way, we experience that they gain a stronger computational understanding. We argue in our study that the students not only gain perspectives through the CT course, but through the interdisciplinary knowledge they gain from, among other things, other courses. However, we see, in line with Czerkawski and Lyman (2015), that CT can support other competencies such as critical thinking or problem-solving skills, which are part of their computational empowerment (Dindler et al., 2020). This can be compared to the systemic PBL pedagogy at our university, which prepares students for adopting various knowledge and problem modes, different approaches to problem-solving while drawing on interdisciplinary curriculum, and a plethora of PBL and CT skills (Kolmos et al., 2021; Jonasen & Gram-Hansen, 2019). In this way, students develop meta-competencies in choosing among various perspectives on emergent problems and select concepts and tools accordingly. Similarly, the students in our study reach for the necessary understanding and tools to substantiate their sense of perspective.

Concluding remarks

In this study, we have presented our findings regarding how humanities students in HE develop the CT competencies that students require for their further careers. It implies that students in HE develop their computational understanding through programming and prototyping. The study, however, also indicates that students need CT competencies to enroll in their future career. We provided an example of how lower-level programming with micro:bit provided a scaffolding teaching approach. The teachers used the scaffold to teach students concepts and practices. The students used this computational understanding to grasp and bridge the abstraction of CT and develop computational skills. To develop perspectives, students were drawing from their interdisciplinary knowledge, and through self-scaffolding improved their understanding of CT. It should be noticed that a scaffolding approach facilitates students' development of CT understanding and thereby helps students to provide and create self-scaffolding and let students be coactive in their learning process. This indicates that students of HE need computational understanding to meta-reflect on CT.

The presented study contributes to the literature in HE of humanities and enriches the empirical studies of CT. This research introduced a model of the development of knowledge and the required competencies for humanities students. The model can be used as a tool to increase the general understanding of how students in HE develop CT. However, other studies should investigate the model further to grasp the potential of humanities development of CT in HE.

About the authors

Anders Kalsgaard Møller is an Associate Professor in IT-based Design, Learning and Innovation at Aalborg University. His research focuses on participatory design processes, virtual reality, learning and digital technologies.
Institutional affiliation: Aalborg University, Faculty of Social Studies and Humanities, Department of Culture and learning, Kroghstræde 3, 9220 Aalborg, Denmark.
Email: Ankm@learning.aau.dk

Camilla Finsterbach Kaup is a Ph.D. student in IT-based Design, Learning and Innovation at Aalborg University. Her research focuses on students' development of computational thinking regarding mathematics at the primary level.
Institutional affiliation: Aalborg University, Faculty of Social Studies and Humanities, Department of Culture and learning, Kroghstræde 3, 9220 Aalborg, Denmark.
Email: cfk@learning.aau.dk

Eva Brooks is a Professor in IT-based Design, Learning and Innovation at Aalborg University. Her research focuses on the use of digital technology and its implications for teaching and learning.
Institutional affiliation: Aalborg University, Faculty of Social Studies and Humanities, Department of Culture and learning, Kroghstræde 3, 9220 Aalborg, Denmark.
Email: eb@learning.aau.dk

Dorina Gnaur is an Associate Professor of learning with technologies at Aalborg University. She received her Ph.D. in e-supported workplace learning from Aarhus University in 2010. Her main research areas are technology-enhanced learning in context and innovative approaches to education.
Institutional affiliation: Aalborg University, Faculty of Social Studies and Humanities, Department of Culture and learning, Kroghstræde 3, 9220 Aalborg, Denmark.
Email: dgn@learning.aau.dk

Maja Højslet Schürer is a Ph.D. fellow in IT-based Design, Learning and Innovation and a research assistant at Aalborg University. Her research interest is in digital play, participation, children's perspective, and transition.
Institutional affiliation: Aalborg University, Faculty of Social Studies and Humanities, Department of Culture and learning, Kroghstræde 3, 9220 Aalborg, Denmark.
Email: mhsc@learning.aau.dk

Marie Charlotte Lyngbye is a research assistant at Aalborg University. Her main research areas are computational thinking, virtual reality and health.
Institutional affiliation: Aalborg University, Faculty of Social Studies and Humanities, Department of Culture and learning, Kroghstræde 3, 9220 Aalborg, Denmark.
Email: marielyngbye@gmail.com

References

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. https://doi.org/10.1145/1929887.1929905

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, *3*(2), 77–101. https://doi.org/10.1191/1478088706qp063oa

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 (Vol. 1, p. 1–25).

Caspersen, M. E., Gal-Ezer, J., McGettrick, A., & Nardelli, E. (2018). *Informatics for all: The strategy*.Technical Report. ACM Europe & Informatics Europe. https://doi.org/10.1145/3185594

Chongtay, R. (2019). Computational Literacy skill set – an incremental approach. In N. B. Dohn (Ed.), *Designing for Learning in a Networked World* (pp. 156–174). Routledge. https://doi.org/10.4324/9781351232357-9

Creswell, J. W., & Creswell, J. D. (2018). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. Sage, Los Angeles.

Czerkawski, B. C., & Lyman, E. W. (2015). Exploring issues about computational thinking in higher education. *TechTrends*, *59*(2), 57–65. https://doi.org/10.1007/s11528-015-0840-3

Dewey, J. (1923). *Democracy and education: An introduction to the philosophy of education*. Macmillan.

Dindler, C., Smith, R., & Iversen, O. S. (2020). Computational empowerment: participatory design in education. *CoDesign*, *16*(1), 66–80. https://doi.org/10.1080/15710882.2020.1722173

Iversen, O. S., Smith, R. C., & Dindler, C. (2018). From computational thinking to computational empowerment: A 21st century PD agenda. In *Proceedings of the 15th Participatory Design Conference: Full Papers* (Volume 1, pp. 1–11). ACM, New York. https://doi.org/10.1145/3210586.3210592

Jonasen, T. S., & Gram-Hansen, S. B. (2019). Problem Based Learning: A facilitator of Computational Thinking. In R. Ørngreen, B. Meyer, & M. Buhl (Eds.), *ECEL 2019: 18th European Conference on e-Learning* (pp. 260–267). Academic Conferences and publishing limited.

Kolmos, A., Holgaard, J. E., & Clausen, N. R. (2021). Progression of student self-assessed learning outcomes in systemic PBL. *European Journal of Engineering Education*, *46*(1), 67–89. https://doi.org/10.1080/03043797.2020.1789070

Lu, J. J., & Fletcher, G. H. L. (2009). Thinking about computational thinking. In *Proceedings of the 40th ACM technical symposium on Computer science education* (pp. 260-264). https://doi.org/10.1145/1508865.1508959

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, *41*, 51–61. https://doi.org/10.1016/j.chb.2014.09.012

Mascolo, M. F. (2005). Change processes in development: The concept of coactive scaffolding. *New Ideas in Psychology*, *23*(3), 185–196.

Ministry of Education (2018). *Læseplan for forsøgsfaget teknologiforståelse* [Curriculum for the experiment about Technology Comprehension]. Copenhagen, Denmark. https://www.uvm.dk/-/media/filer/uvm/aktuelt/pdf18/181221-laeseplan-teknologiforstaaelse.pdf

Puente, S. M. G., van Eijck, M., & Jochems, W. (2013). A sampled literature review of design-based learning approaches: a search for key characteristics. *International Journal of Technology and Design Education*, *23*(3), 717–732. https://doi.org/10.1007/s10798-012-9212-x

Ravitch, S. M., & Carl, N. M. (2021). *Qualitative research: Bridging the conceptual, theoretical, and methodological*. Thousand Oaks, CA: Sage Publications.

Rittel, H. W., & Webber, M. M. (1984). Planning problems are wicked. Developments in design methodology (pp. 135–144). John Wiley & Sons, New York.

Romero, M., Lepage, A., & Lille, B. (2017). Computational thinking development through creative programming in higher education. *International Journal of Educational Technology in Higher Education*, *14*, Art. 42. https://doi.org/10.1186/s41239-017-0080-z

Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes*. (M. Cole, V. Jolm-Steiner, S. Scribner, & E. Souberman, Eds.) Harvard University Press. https://doi.org/10.2307/j.ctvjf9vz4

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

Wing. J. M. (2020). *A conversation about computational thinking*. NSW Government website https://education.nsw.gov.au/teaching-and-learning/education-for-a-changing-world/resource-library/a-conversation-about-computational-thinking

Wood, D., Bruner, J. S., & Ross, G. (1976). The role of tutoring in problem solving. *Journal of child psychology and psychiatry*, *17*(2), 89–100. https://doi.org/10.1111/j.1469-7610.1976.tb00381.x

Zhang, J.-H., Meng, B., Zou, L.-C., Zhu, Y., & Hwang, G.-J. (2021). Progressive flowchart development scaffolding to improve university students'computational thinking and programming self-efficacy. *Interactive Learning Environments*. https://doi.org/10.1080/10494820.2021.1943687