Design and Technology Education: An International Journal

29.2

Teaching programming in Technology teacher education: Revealing student teachers' perceptions

Anna Perez, Linnaeus University, Sweden
Maria Svensson, University of Gothenburg, Sweden
Jonas Hallström, Linköping University, Sweden

Abstract

This study explores the changing landscape of technology teacher education, in relation to the increasing integration of digital content, especially programming, in teacher education for grades 4–6 (pupils 10-12 years old) and how student teachers in Sweden perceive this content. Limited research exists on student teachers in technology, particularly focusing on programming. This study therefore investigates student teachers' perceptions of teaching programming in technology education, after completing their technology course in teacher education. We answer the following research questions: What are the student teachers' perceptions of teaching programming in technology education? and How is potential subject didactics knowledge for teaching programming manifested in student teachers' perceptions of technology teaching? Using a phenomenographic approach, 25 student teachers' perceptions of programming in technology education were investigated through semi-structured individual and group interviews. Different perceptions were revealed and presented in four categories: (1) following instructions in a logical order, (2) learning a programming language, (3) solving technological problems, and (4) understanding and describing a technological environment. The results show that student teachers' perceptions of the subject of technology predominantly focuses on following instructions and the learning of a programming language. The identified potential subject didactics knowledge is constituted of an awareness of three critical aspects: understanding programming language, understanding programming as a way of solving problems, and the relationships of technological problems to everyday life and society. This study offers valuable insight into the development of competencies required to teach programming in technology, informing educational strategies and future research in this emerging field.

Keywords

Student teachers, Technology education, Programming, Phenomenography

Introduction

Over the last two decades, our everyday lives have changed and become increasingly digitalised; for example, in the form of robot lawn mowers, vacuum cleaners, and AI-supported banking transactions. The increasing digitalisation of society has contributed to changes in school curriculum documents in Sweden (Skolverket, 2017) and other countries. In Sweden, for example, digital technology and programming have been included as educational content in the technology syllabus since 2018. However, many teachers approach this new educational content with uncertainty (Sentance & Csizmadia, 2017; Vinnervik, 2023; Webb et al., 2017), because programming was not part of their own teacher education and the curriculum does not say how programming should be taught or how any difficulties learners encounter should

be addressed (Passey, 2017). Therefore, there is a need for more knowledge about what is involved in teaching programming as part of teacher education and what competencies technology student teachers need to develop to be prepared for their future teaching career.

In line with this, there is a need to understand the perceptions of student teachers in order to inform teacher education (Koster et al., 2005; Schneider et al., 2013). In a study by Perez and Svensson (2024), the experiences of student teachers with programmed technological artefacts, including elevators, tumble dryers, traffic lights, and keyboards, were investigated. The result shows that student teachers' initial understanding of those programmed technological artefacts can be described as ranging from experiencing only the physical interface to components as and within a system, but these were still only a limited set of all the possible aspects of programmed technological artefacts (Perez & Svensson, 2024). In addition, student teachers' inadequate subject knowledge is a problem reported more frequently by primary school teachers than by their secondary counterparts (Selby & Woollard, 2014). This imbalance may be accounted for by the fact that primary school teachers are responsible for teaching a range of subjects, whereas secondary school teachers concentrate on fewer areas with more comprehensive training.

An important mission for primary teacher education should therefore be to instil student teachers with the ability to plan, implement, and evaluate the content of different subjects and understand the characteristics of each subject. In technology education, there is still a lack of research, specifically on student teachers' knowledge of programming. Therefore, it is imperative to investigate how student teachers perceive their upcoming teaching regarding programming in the subject of technology, after completing the technology teacher education course included in their training.

Aim and research questions

This study investigates student teachers' perceptions of teaching programming in technology education, after completing their technology course in teacher education. The following research questions are posed:

- What are the student teachers' perceptions of teaching programming in technology education?
- How is potential subject didactics knowledge for teaching programming manifested in student teachers' perceptions of technology teaching?

Programming as part of technology education and subject didactics

The study of teaching and learning of a subject content is often referred to as subject didactics, which can be seen as a bridge between subject knowledge and pedagogy (see for example Sjøberg (2001)). Subject didactics in, for example, the Nordic, German and French context thus refers to the subject-specific aspects of teaching and learning (Osbeck et al., 2018; Rothgangel & Vollmer, 2020; Schoenfeld, 1998). It addresses the three key questions: What (the relevant content), Why (the goals), and How (the appropriate methods) of teaching and learning within a certain subject (Rothgangel & Vollmer, 2020). Teacher education develops student teachers' knowledge in subject didactics (Osbeck et al., 2018; Vollmer, 2022). In technology education for grades 4-6 (pupils 10-12 years old) this involves both subject knowledge and its subject didactics. For instance, the 2018 revision to the Swedish compulsory school curriculum aims to

help pupils understand the impact of digitalisation on individuals and society. Programming is included in several subjects, primarily mathematics and technology. The revised technology curriculum further states that pupils should acquire skills to control their constructions or other objects through programming, and reflect on opportunities, risks and safety when using technology in everyday life (Skolverket, 2017). Consequently, teacher education in technology must equip student teachers with the necessary skills to teach programming effectively.

There is a lack of research on both technology teachers and technology student teachers' understanding and teaching of programming. However, the small amount of research that does exist shows that technology teachers feel uncertain about how to teach programming, probably because it has been a marginal part of technology teacher education (Sentance & Csizmadia, 2017; Vinnervik, 2022; Webb et al., 2017). Those who taught programming before it became a compulsory part of Swedish technology education in 2018 were mostly computer enthusiasts who had learned to program themselves (cf. Nouri et al. (2020)).

Furthermore, there is a lack of research specifically on student teachers' understanding of the role of programming in relation to the school subject of technology. We know very little about what these prospective technology teachers learn about programming during their teacher education. Moreover, computational thinking (CT) in teacher education, student teachers' perceptions of programming and what constitutes the nature of programming in technology teacher education, are underdeveloped areas of research. However, Tsai et al. (2021) demonstrate that a game-design project helped improve the programming skills and computational thinking of student teachers in Taiwanese pre-service primary teacher education. Other studies, such as Rowston et al. (2022) are more inconclusive and show that technology integration in teacher education, including programming, can be more haphazard. In conclusion, more research is needed that could potentially shed light on what knowledge components need to be in focus to improve programming teaching in technology teacher education.

A framework for technological knowledge and computational thinking

Technology is created by humans to solve problems or fulfil needs and desires (Kline, 2003; Lindqvist, 1987). Technology is not only about artefacts (objects, products) but also about processes, methods, systems, and activities—and *knowledge* about these—either for innovation and production or for use (Bijker et al., 2012; Hallström & Williams , 2022; Mitcham, 1994; Van der Vleuten et al., 2017). Technology is also something fundamentally material, as can be seen in the entire human-built world that surrounds us (Hughes, 2004; Ihde, 1993; Schatzberg, 2018). In line with this, even digital technology that is made up of abstract machine code in ones and zeros—basically Boolean mathematical expressions—must be considered as technology because it requires electrical signals in physical computers for it to work (Denning & Tedre, 2019; Hallström, 2024). Technology can therefore be referred to as the "designed world", in correspondence with the "natural world" as a term for the environment (Blomkvist & Kaijser, 1998).

Technological knowledge is, in a sense, practical and concerned with designing, crafting, and making (Mitcham, 1994). However, technological knowledge is not only practical and hands-on, nor is it merely an application of scientific or other knowledge for practical use, but it is its own area and tradition of knowledge that is related to the designed world and human material

culture, in all their variety (Schatzberg, 2018). This means that technological knowledge includes skills and know-how to manage the designed world (procedural knowledge), cognitive and other mental conceptions and theories that make sense of the same (conceptual knowledge), as well as an understanding of the relationship of technology to society and the environment (contextual knowledge) (cf. Nordlöf et al. (2022); Williams (2017)). Technological knowledge therefore concerns the material as well as the abstract, the analogue as well as the digital aspects of the designed world (Hallström, 2024).

Computational thinking (CT) widely applied in computer science, is closely associated with programming skills. In line with the above reasoning, it could also be defined as a kind of technological knowledge. Denning and Tedre (2019) claim that CT encompasses the skills and practices essential for creating computations to perform specific tasks through artefacts, as well as interpreting the world as a series of informational processes. As described by Denning and Tedre (2019), CT also has two further dimensions. One focuses on the mechanics of computer operations, code expression in programming languages, and software assembly into systems. The other dimension focuses on anticipating design needs and considering the user's context. Both dimensions contribute to understanding the purposeful design of technological solutions and artefacts to address challenges (Denning & Tedre, 2019), and both require applying a systems perspective to technological and computational solutions; that is, systems thinking: “a set of skills for understanding, analysing, and working with systems consisting of multiple interconnected elements and exhibiting emergent properties” (Ho, 2019, p. 2764).

Methodology

This article is based on the preliminary findings presented at the PATT40 conference (Perez et al., 2023). The analysis has since been completed, rendering the results more reliable through rigorous categorization validation, including at the aforementioned conference.

To answer the research questions, we used a qualitative method using semi-structured individual and group interviews with student teachers, and a phenomenographic approach (Marton & Booth, 1997) was used to analyse the transcripts and find variation in student teachers' perceptions regarding teaching programming in technology.

Phenomenography

Phenomenography as a research tradition is broadly situated within an interpretive epistemological orientation and focuses on the variation in how a phenomenon is experienced by a group of individuals (Collier-Reed, Ingerman & Berglund, 2009; Marton & Booth, 1997). Phenomenography is underpinned by, among other things, a focus on the relational nature of human experience, a non-dualistic ontological perspective, an explicit focus on the experience of phenomena, and the adoption of a second-order perspective. The result of the research is a set of categories which describe the qualitatively different ways of experiencing this phenomenon and which are logically related in structure and meaning. The categories do not describe how individuals perceive the phenomenon - rather they describe the phenomenon at a collective level (Marton, 1981; Marton & Booth, 1997; Runesson, 2006).

This study investigates the different ways in which student teachers experienced, perceived, or understood *programming in technology education*, here labelled as their ‘perceptions’ of this phenomenon. Even though a phenomenographic study investigates the individual experience,

the area of interest here is these experiences taken together; that is, the collective perceptions of a phenomenon (Booth & Ingerman, 2002; Marton, 1981; Marton & Booth, 1997; Trigwell, 2006). Data collected from interviews is interpreted and described by researchers to reshape the individual voice into a collective statement. The perceptions shared by the collective of student teachers is of interest, and therefore all perceptions are collected into a dataset for categorisation (Trigwell, 2006).

The inductive process of creating categories, from these descriptions, involved determining when descriptions about the phenomenon were similar enough to be grouped, and when they were different enough to require separate groupings. This 'set' of descriptive categories forms what is called an outcome space (or space of variation), which contains different groupings of aspects of a phenomenon. Central to this outcome space is that the categories are logically related, typically hierarchically, with each successive category representing a more complex way of experiencing the phenomenon under investigation (Marton & Booth, 1997)

After forming an outcome space, categories were arranged hierarchically depending on both the number of aspects but also the complexity of the aspects, as outlined by Marton (1981). This hierarchical arrangement created a spectrum—a variety—and constituted an intriguing range of understanding in the group of student teachers. It is worth noting that the extent of the range of the established categories is interesting because it provides insights into how great the difference is in how the group as a whole views programming in technology education.

Data collection

Student participating in this study were from three higher education institutions in southern Sweden, enrolled in a four-year program to become teachers for grades 4-6 (pupils 10-12 years old). During one of the eight semesters, they can choose to specialize in either the social sciences or the natural sciences and technology. This part of their professional education aims to enhance their subject knowledge and subject didactics knowledge. The semester includes a five-week course in technology, with the remaining 15 weeks divided among physics, biology, and chemistry. At the time of data collection, the student teachers had completed their technology course, and therefore, it was of interest to investigate what they did and did not discern about the phenomenon of *programming in technology education* and whether and how signs of subject didactics knowledge for teaching programming in technology were manifested. The five-week technology course was taught full-time at these higher education institutions (i.e., 7.5 credits). The course deals with relevant subject theory, together with subject didactics. Among other things, the course's content covers the history of technology and views of technological knowledge, but also construction, mechanics, and technological systems. To ensure that participants had been exposed to similar teaching content, the schedules of the higher education institutions were compared. This established that there were few differences in the teaching content. The proportion of the teaching that involved elements linked to programming corresponds to two full days of the five weeks that the student teachers take the technology course. These elements include the construction of an object which can be controlled by programming. Two data collection sets were used to form a pool of meaning, where the first set is individual interviews and the second is group interviews. The distribution of the participants between individual and group interviews is outlined in Table 1 below.

Table 1. Distribution of participants

Interviews	Participants					Total
Individual	8, including 2 pilot					8
Group	4	2	4	3	4	17
						25

An interview guide was used for both sets of data collection, and they were almost identical, with only a few adjustments made to the second guide. An adaptation was made, given that the data collection on one occasion was conducted individually and the other involved groups, and additional follow-up questions were added to the interview guide for the groups. By using two almost identical guides in two types of interviews, we ensure that the results can be treated equally. The student teachers in the individual interviews came from two different universities and the student in the group interviews came from one of them. Interviews were conducted using Zoom for individual interviews and in person for group interviews. Audio recordings, video recordings, and notes were taken to assist in the analysis process by the first author of this article. Group interviews were planned after the individual interviews. Group interviews allow for in-depth conversations within the group of participants without too much meddling from the researcher. This resulted in allowing more freedom for student teachers to talk, and the researcher played a smaller role than in the group interviews than they did in the individual interviews. At the same time, it can be difficult to capture individuals' in-depth understanding when the researcher does not ask so many follow-up questions. However, the student teachers themselves contributed to a certain degree, posing follow-up questions to each other during the group interviews due to the discussion-like atmosphere (Robson & McCartan, 2016). In both data collection sets, the interview guide included pictures that were used to initiate the conversation. The pictures represented four everyday artefacts: a tumble dryer, traffic lights, a keyboard, and an elevator. These artefacts, familiar to student teachers, were chosen because they can be controlled by programming and they are connected to technological systems, which is an important part of the technology subject in primary schools. The researcher's task during the interview was first to keep participants focused on the phenomenon throughout the interview. The researcher also attempted to gain more depth from the interviewee' answers by repeating the participants' answers and asking whether they would like to elaborate on their answers or add any additional comments or details. Examples of questions that were asked include: "What competences do you think you need to be prepared to teach programming in technology?" "What do you think learners need to know about programming in technology?", and "What is important that we teach them?" Each individual interview lasted approximately 45–50 minutes, and the group interviews lasted slightly longer. Each participating student teacher was informed about the aim and design of the study and consented to participate. The participants were informed that the study follows ethical guidelines from the Swedish Research Council (Vetenskapsrådet, 2017). The responses were anonymised, and data was managed following the General Data Protection Regulation (GDPR).

Method of analysis

The analysis in this study followed a phenomenographic approach, aiming to achieve a comprehensive and nuanced understanding of student teachers' perceptions of the phenomenon in focus—in this case, *programming in technology education*. The analysis of the

transcripts from the individual interviews began with repeated read-throughs, where a researcher, here the first author, reviewed the material to find expressions of the variation between different ways of experiencing the phenomenon. In this way, the researcher first adopts an open attitude to the data which gradually becomes more focused; the researcher then forms a "pool of meaning" pertaining to the entire dataset (Wood, 2000). Within this "pool of meaning", the researcher identified similarities and differences in perceptions, leading to an initial grouping aimed at discerning variations among the participants' perceptions of the phenomenon. While the first groupings were being made, all three authors discussed and debated the groupings together. Once the researchers reviewed similarities and differences, they also made descriptions of what constitutes the variation found between the groups as a help when deciding whether the groups should be merged or new groups should be created. The goal of the analysis process was to consistently identify the qualitative variation in student teachers' perceptions when they describe teaching programming in technology.

From this initial analysis phase of the individual interviews, three categories emerged. These categories describe student teachers' perceptions of teaching programming as: 1) learning a programming language, 2) solving technological problems, and 3) understanding and describing a technological environment. As the analysis extended to include group interviews, the original categories remained and were strengthened, while one additional category also emerged. In this category teaching programming is described as following instructions in a logical order. The new category had fewer and less complex aspects describing the phenomenon than the previously identified categories and is therefore placed lowest in the hierarchy. A change in the numbering of the categories can therefore be seen below, where categories are described with examples of excerpts that are characteristic of each category.

Validity of the study

Multiple efforts were made to strengthen the validity of the data analysis. Based on the questions posed in the study, the method is appropriate and transparent as there are included extracts from the collected excerpts that show answers obtained in the semi-structured individual and group interviews. The overall questions in the interview guide have been mentioned in the text, but the follow-up questions varied depending on the participants' answers. To ensure that collected data have been analysed correctly, the categorisation has been questioned and validated several times during the analysis process by other researchers, both in informal discussions and during conference presentations (Collier-Reed et al., 2009).

Since the process of conducting the group interviews confirmed our created categories but also broadened the variety of perceptions, we could assume that we reached saturation in our data in this context and therefore chose not to continue conducting more interviews.

Results of the study

The analyses resulted in four categories describing student teachers' perceptions of teaching programming in technology education and indicating their potential subject didactics knowledge for teaching programming in technology. The four categories of teaching programming are:

1. following instructions in a logical order,

2. learning a programming language,
3. solving technological problems,
4. understanding and describing a technological environment.

A selection of excerpts is presented below to describe what characterises each category.

Category 1: Following instructions in a logical order

In this category, student teachers describe teaching programming in technology as a series of practical exercises where individuals follow oral or written instructions. What is emphasised when describing the instructions is that it is important to organise them in a logical order and that instructions are used as an input to get a specific output. However, no programming concepts are used, which makes it unclear whether or not they understand the instructions to be synonymous with a programming language. This indicates that the student teachers' focus here is on a structural level—or the order in which instructions are given and how they are followed—rather than on a referential level, where the focus is on the relationship between the instruction and the context in which it is used. The focus on logical order in instructions indicates that some aspects of CT are present.

In the following excerpts, student teachers emphasise the importance of following instructions using a person (analogue programming) to illustrate the logical order needed to make something happen:

> *Cecilia: Yes, but if they're going to work together in pairs, one of them might have to tell the other how to walk, and for them to get there, they have to say all the steps, which is a simple way of showing what programming is.*

> *Frida: It's a lot of following instructions and doing it from the top down and this way, […] for example, you're going to guide your friend and give instructions, or you're going to write down an instruction and then the other person will try to follow it, for example, draw something after the instruction.*

> *Wera: I think it's good to start with analogue programming. Maybe giving instructions to each other [...] and like this, OK, now you're going to programme someone to brush their teeth. Write step one, step two, step three and then it could go wrong in any way.*

The student teachers in this category focus on what and how to teach programming as 'separate' content, without connections to technology knowledge and to the intentions of programming, so the 'why' question is absent in their descriptions. We interpret that as a gap in their potential subject didactics knowledge regarding teaching programming in technology.

Category 2: Learning a programming language

In this category, student teachers describe programming as a language in the form of instructions using specific concepts, such as loops and expressions. In their descriptions, persons are no longer used as tools to follow instructions and there is a stronger link to computers as receivers of the instructions than to individuals as receivers. The instructions are no longer oral; instead, they use a language with commands (code)—a programming language.

Programming in this category is not seen as a solution to a technological problem, instead, student teachers describe that something happens in the language where you get an output for a certain command (input).

Similarly, as in the previous category, there are traces of CT in the form of instructions following a logical order. Instructions are foregrounded with a focus on the output of the programming language. Still, there is a lack of connection to technological knowledge except in terms of rudimentary systems thinking. The student teachers in this category describe how you can connect components; for example, the computer operates on commands, through a programming language. The following excerpts illustrate this and also describe the need for understanding the language. What follows is an excerpt specifically addressing block programming language:

> *Carin: It's these blocks, it's called block programming and so on. That's what I think because otherwise, it's a too advanced a level.*

> *Chris: It [block programming] contains almost the same thing as programming with code, except that it is blocks, so it contains things like loops and expressions and such things. So, they get to learn it in a simpler way with blocks.*

In the excerpt below we see how the student teacher refers to the computer executing an instruction in the form of commands, and it is about learning to use these commands. A basic level of systems thinking is therefore demonstrated:

> *Daniella: [...] so how to start it on the computer, how to use these commands, how to twist and turn so you get comfortable using it.*

Student teachers' potential subject didactics knowledge is still of a lower order in this category since they mainly focus on teaching a specific language with a specific order on the commands, rather than explaining the use of the programmed artefact outside the computer program, which connects to the intentions of programming in technology.

Category 3: Solving technological problems

In this category, student teachers describe teaching as something that includes the use of a programming language when solving technological problems. In contrast to the two previous categories, here the instructions (the programming language) are no longer in the foreground, but rather represent what can be achieved with the help of the instructions. The output is defined in this category as something linked to a technological solution. Therefore, the category also describes that the student teachers relate programming to technology knowledge.

The student teachers in this category also show a greater understanding of systems thinking, as they describe several components and their close connections. CT is present as descriptions where student teachers make links between programming and problem-solving, which did not occur in the previous category.

The following excerpt shows a continued interest in practical engagement, while also emphasising the necessity of problem-solving:

Clara: […] and technology education is largely about, well, how should I put it, identifying needs, and perhaps finding solutions to those needs, and it's quite challenging nowadays to find solutions to needs if you don't have programming skills.

Daniella: […] But I think that when you program something, it's because it's meant to be some kind of tool, like you want to see something, you want to cook something, you want to dry something. It has a purpose, and that purpose is what belongs to technology. It's not just the fact that it's programmed that makes it technology, but it's what comes after, in a sense.

Hanna: […] programming in technology is more like we program, for example, [...] carousels, making carousels that make them spin and stuff. It's about making things work, you know. So, in technology, it's like, it's a bit more computer-oriented, in a way.

In this category, student teachers emphasise what can be achieved with programming as well as the use of a language. We interpret that as a more developed subject didactics knowledge. The 'why', 'what', and 'how' questions are all present to some extent.

Category 4: Understanding and describing a technological environment

In the final category, the teaching of programming is contextualised as part of society and therefore other aspects are in the foreground compared to the previous categories. Here the instructions are clearly in the background and instead components that can be linked to each other are in the foreground. This indicates a more visible systems perspective, which was only hinted at in the previous categories. Programming is described as part of a larger whole, a human-built apparatus, technological environment, or system. The student teachers show CT in this category by suggesting that teaching should include identifying problems, but also that problems can be divided into smaller sub-problems (decomposition), and the effect overall should be made visible. Björn and Daniella describe this by highlighting several components where they describe teaching that deals with consequences for decisions and actions:

Björn: To understand that something is happening behind the scenes. There's a reason why the lights turn off in the school corridor when no one has been there. It happens automatically, and it's programmed to do so. [...] Many things can be done to maybe save electricity or save water, and it can also contribute to sustainability thinking. Because I think many pupils are very concerned about that nowadays. And through technology and programming, there are great opportunities to address those concerns.

Daniella: No, but what I mean is that everything you learn, it's something that gives you power over your life and how you relate to society. And given that we have many more technological gadgets, we also need to have more knowledge about them and how they work so that we can engage with society and its structures. […] So that they can see that programming exists all around, it's in the traffic light when I go to school, what would happen if something went wrong and what would be the effects in a larger context?

Student teachers' subject didactics knowledge, in this category, is related to the environment where programming is present in today's society. This indicates that student teachers perceive that teaching programming means not only conveying it as a language or as a method for solving problems but also as a way of becoming literate in a technologically intensive society.

Summary of the results

It is apparent in the hierarchical categorisation that there is a wide range of perceptions of teaching programming between the highest and most developed perception (Cat 4), and the lowest and least developed perception category (Cat 1). The lowest category contains fewer and less complex aspects of *programming in technology education*, and then the number and complexity of the aspects of teaching increase with each category.

There are therefore two dimensions that vary between categories:

- the way programming instructions or language are understood, and
- the way technological problems are understood.

The understanding of the instructions and the programming language changes significantly between categories 1 and 2 but is maintained and in the background of their understanding in categories 3 and 4. Thus, while the problem-solving dimension does not appear in the first two categories, it is then visible in category 3 and central in the fourth category.

In summary, the dimensions of variation are related to the identified aspects in the categories. Some aspects appear to be critical in transiting one's understanding from a lower category to a higher one. These critical aspects are essential when it comes to choosing to which category the description should be assigned.

The critical aspect that categorises a description into the second category, instead of the first, is that in addition to emphasising programming as following instructions the student teachers also express *an understanding of a programming language by using specific words and terminology.* For a description to be categorised into the third category, the student teachers also need to show *an understanding of programming as a way of solving technological problems*. The critical aspect that needs to be identified for a description to be placed in the fourth category is that the student teachers show *an understanding that the technological problem impacts on, and is impacted by, our everyday life and society in a broader sense*.

Discussion

This study investigates student teachers' perceptions of teaching programming, after completing their technology course in teacher education.

It appears that the studied student teachers' awareness of the connection to technology education is relatively weak and that there is a predominant focus on instructions and the programming language. This is in line with earlier studies of novice students learning to program (Eckerdal et al., 2005), although it also contradicts findings by Vinnervik (2023), who found that even if technology teachers teach basic coding they focus more on broader technological competencies. In any case, student teachers should not be taught only to program but also to be able to teach technology in a broader sense, where programming is one way of using technology in solving societal problems. Therefore, it is imperative that student teachers reach a more developed understanding of programming, including the problem-solving dimension. This information on the studied student teachers' understanding is important for teacher education to be able to develop student teachers' subject didactics knowledge in technology during their teacher education, and the variation in perceptions

indicates what may be important to emphasise. It is important for technology student teachers to acquire a broad competence that is relevant to and covers knowledge components taught in schools (Doyle et al., 2019; Norström, 2014), but they also need programming knowledge that is specific to teacher education; exactly what this is in a technology education context is not clear due to a lack of research, but the results of this study indicate some such knowledge components, for example, systems thinking.

From the results presented above, it is clear that most of the descriptions collected in the study are categorised in the lowest two categories (Cat 1 and 2) and only a few descriptions are categorised in the highest category (Cat 4). Given that a phenomenographic categorisation is hierarchical, this is interesting because it gives us information about how well the understanding of the chosen phenomenon—*programming in technology education*—is developed among student teachers.

The results show that the majority of student teachers have difficulties demonstrating an understanding of how programming is related to technology, in that they do not see programming as part of the problem-solving dimension, nor do they clearly express an understanding of how technological solutions have a function or purpose in society (Mitcham, 1994; Schatzberg, 2018). However, in categories three and four, some student teachers have more fully developed CT which considers, e.g., design and user needs (Denning & Tedre, 2019), and only in category four do a few student teachers mention conceptions of programming that can be linked to more developed technological systems perspective and systems thinking (Ho, 2019; Slangen et al., 2011). Such thinking is important for developing subject didactics knowledge in technology education and goes beyond merely focussing on what is taught in schools.

The study underlines the importance not only of competence in technology but also subject didactics knowledge for developing programming instruction in the subject of technology. The identified knowledge is an awareness of the three critical aspects—*understanding programming language, understanding programming as a way of solving problems* and *the relation of technological problems to everyday life and society*—as well as two variations: *the programming language,* and *the problem-solving dimension*. Together these aspects imply critical components of subject didactics knowledge that is crucial for student teachers' preparation for their coming profession.

The study thus advocates for a teacher education curriculum that not only imparts practical, procedural skills but also promotes conceptual knowledge and contextual understanding related to technology (Björklund & Nordlöf, 2024). This holistic approach ensures that future technology teachers are well-equipped to address the challenges of teaching programming effectively and adapting to the changing demands of the technological landscape in educational settings and in society at large.

References

Bijker, W. E., Hughes, T. P., & Pinch, T. J. (2012). *The Social Construction of Technological Systems, anniversary edition: New Directions in the Sociology and History of Technology*. MIT press.

Björklund, L., & Nordlöf, C. (2024). Product or Process Criteria?: What Teachers Value When Assessing Programming. In *Programming and Computational Thinking in Technology Education* (pp. 325-341). Brill.

Blomkvist, P., & Kaijser, A. (1998). *Den konstruerade världen: Tekniska system i historiskt perspektiv*. B. Östlings bokförl [*The Constructed World: Technical Systems in Historical Perspective*]. Symposion.

Booth, S., & Ingerman, Å. (2002). Making sense of Physics in the first year of study. *Learning and Instruction*, *12*(5), 493-507.

Collier-Reed, B. I., Ingerman, Å., & Berglund, A. (2009). Reflections on trustworthiness in phenomenographic research: Recognising purpose, context and change in the process of research. *Education as change*, *13*(2), 339-355.

Denning, P. J., & Tedre, M. (2019). *Computational thinking*. MIT Press.

Doyle, A., Seery, N., Canty, D., & Buckley, J. (2019). Agendas, influences, and capability: Perspectives on practice in design and technology education. *International Journal of Technology and Design Education*, *29*(1), 143-159.

Eckerdal, A., Thuné, M., & Berglund, A. (2005). What does it take to learn'programming thinking'? Proceedings of the first international workshop on Computing education research, USA, 135 – 142. https://doi.org/10.1145/1089786.1089799.

Hallström, J. (2024). Design and Make—and Code? In J. Hallström & M. de Vries (Eds.), *Programming and Computational Thinking in Technology Education: Swedish and International Perspectives*. Brill.

Hallström, J., & Williams, P. J. (2022). *Teaching and Learning about Technological Systems: Philosophical, Curriculum and Classroom Perspectives* (J. Hallström & P. J. Williams, Eds.). Springer.

Ho, F. M. (2019). Turning challenges into opportunities for promoting systems thinking through chemistry education. *Journal of Chemical Education*, *96*(12), 2764-2776.

Hughes, T. P. (2004). *Human-built world: How to think about technology and culture*. University of Chicago Press.

Ihde, D. (1993). Philosophy of technology: an introduction. Paragon House, New York. In.

Kline, S. J. (2003). What is technology. . In *Philosophy of technology: the technological condition: an anthology* (pp. 210-212). Blackwell.

Koster, B., Brekelmans, M., Korthagen, F., & Wubbels, T. (2005). Quality requirements for teacher educators. *Teaching and teacher education*, *21*(2), 157-176.

Lindqvist, S. (1987). Vad är teknik? In B. Sundin (Ed.), *I teknikens backspegel: antologi i teknikhistoria*. Carlsson.

Marton, F. (1981). Phenomenography—describing conceptions of the world around us. *Instructional science*, *10*(2), 177-200.

Marton, F., & Booth, S. A. (1997). *Learning and awareness*. Routledge.

Mitcham, C. (1994). *Thinking through technology: The path between engineering and philosophy*. University of Chicago Press.

Nordlöf, C., Norström, P., Höst, G., & Hallström, J. (2022). Towards a three-part heuristic framework for technology education. *International Journal of Technology and Design Education*, *32*(3), 1583-1604.

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

Nouri, J., Zhang, L., Mannila, L., & Norén, E. (2020). Development of computational thinking, digital competence and 21st century skills when learning programming in K-9. *Education Inquiry*, *11*(1), 1-17. https://doi.org/10.1080/20004508.2019.1627844

Osbeck, C., Ingerman, Å., & Claesson, S. (2018). An introduction to didactic classroom studies. *Didactic Classroom Studies a Potential Research Direction*, 9-22.

Passey, D. (2017). Computer science (CS) in the compulsory education curriculum: Implications for future research. *Education and Information Technologies*, *22*, 421-443.

Perez, A., & Svensson, M. (2024). Student Teachers´ Experiences of Programmed Technological Artefacts: Range of Understanding and Ideas for Development. In J. Hallström & M. J. d. Vries (Eds.), *Programming and Computational Thinking in Technology Education: Swedish and International Perspectives*. Brill.

Perez, A., Svensson, M., & Hallström, J. (2023). Student teachers' preconceptions of programming as a content in the subject technology. The 40th International Pupils' Attitudes Towards Technology Conference Proceedings 2023.

Robson, C., & McCartan, K. (2016). Real World Research, 4th Edn. Hokoben. In: New Jersey: Wiley.

Rothgangel, M., & Vollmer, H. J. (2020). Towards a theory of subject-matter didactics. *Research in Subject-Matter Teaching and Learning (RISTAL)*, *3*(1), 126-145.

Rowston, K., Bower, M., & Woodcock, S. (2022). The impact of prior occupations and initial teacher education on post-graduate pre-service teachers' conceptualization and realization of technology integration. *International Journal of Technology and Design Education*, *32*(5), 2631-2669.

Runesson, U. (2006). What is it possible to learn? On variation as a necessary condition for learning. *Scandinavian journal of educational research*, *50*(4), 397-410.

Schatzberg, E. (2018). *Technology: critical history of a concept*. University of Chicago Press.

Schneider, C., Pakzad, U., & Schlüter, K. (2013). The Influence of Personal School Experience in Biology Classes on the Beliefs of Students in University Teacher Education. *Journal of Education and Training Studies*, *1*(2), 197-210.

Schoenfeld, A. H. (1998). Toward a theory of teaching-in-context. *Issues in education*, *4*(1), 1-94.

Selby, C., & Woollard, J. (2014). Refining an understanding of computational thinking.

Sentance, S., & Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Education and Information Technologies*, *22*(2), 469-495.

Sjøberg, S. (2001). Innledning: Skole, kunnskap og fag. *Svein Sjøberg, red., Fagdebatikk, Fagdidaktisk innføring i sentrale skolefag, Oslo: Gyldendal*.

Skolverket. (2017). Läroplan för grundskolan, förskoleklassen och fritidshemmet 2011: reviderad 2017. In: Skolverket Stockholm.

Slangen, L., van Keulen, H., & Gravemeijer, K. (2011). What pupils can learn from working with robotic direct manipulation environments. *International Journal of Technology and Design Education*, *21*(4), 449-469.

Trigwell, K. (2006). Phenomenography: An approach to research into geography education. *Journal of geography in higher education*, *30*(2), 367-372.

Tsai, F.-H., Hsiao, H.-S., Yu, K.-C., & Lin, K.-Y. (2021). Development and effectiveness evaluation of a STEM-based game-design project for preservice primary teacher education. *International Journal of Technology and Design Education*, 1-22.

Van der Vleuten, E., Oldenziel, R., & Davids, M. (2017). *Engineering the Future, Understanding the Past: A Social History of Technology*. Amsterdam University Press.

Vetenskapsrådet, S. (2017). Good research practice. *Stockholm: Swedish Research Council*.

Vinnervik, P. (2022). Implementing programming in school mathematics and technology: teachers' intrinsic and extrinsic challenges. *International Journal of Technology and Design Education*, *32*(1), 213-242.

Vinnervik, P. (2023). Programming in school technology education: the shaping of a new subject content. *International Journal of Technology and Design Education*, *33*(4), 1449-1470.

Vollmer, H. J. (2022). International Transfer of Knowledge: Translating Didaktik, Fachdidaktik, Allgemeine Fachdidaktik. *Research in Subject-matter Teaching and Learning (RISTAL)*, *5*(1), 39-55.

Webb, M., Davis, N., Bell, T., Katz, Y. J., Reynolds, N., Chambers, D. P., & Sysło, M. M. (2017). Computer science in K-12 school curricula of the 2lst century: Why, what and when? *Education and Information Technologies*, *22*(2), 445-468.

Williams, P. J. (2017). Critique as a disposition. In P. J. Williams & K. Stables (Eds.), *Critique in design and technology education* (pp. 135-152). Springer.

Wood, K. (2000). The experience of learning to teach: Changing student teachers' ways of understanding teaching. *Journal of curriculum studies*, *32*(1), 75-93.