UPPSALA
UNIVERSITET

Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology 2104

Practical thinking in programming education

Novices learning hands-on

KRISTINA VON HAUSSWOLFF

ACTA
UNIVERSITATIS
UPSALIENSIS
UPPSALA
2022

ISSN 1651-6214
ISBN 978-91-513-1372-6
URN urn:nbn:se:uu:diva-461455

Dissertation presented at Uppsala University to be publicly examined in Room 101195 (nya hus 10), Ångströmlaboratoriet, Lägerhyddsvägen 2, Uppsala, Thursday, 17 February 2022 at 09:15 for the degree of Doctor of Philosophy. The examination will be conducted in English. Faculty examiner: Dr. Tony Clear (Auckland University of Technology, Computer Science and Software Engineering, Auckland, New Zealand).

Abstract

von Hausswolff, K. 2022. Practical thinking in programming education. Novices learning hands-on. *Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology* 2104. 64 pp. Uppsala: Acta Universitatis Upsaliensis. ISBN 978-91-513-1372-6.

Understanding how novices learn to program is of national and global concern. A substantial number of studies have been conducted within computing education research (CER), and, although some understanding has been gained, research still finds that programming is hard to learn. This thesis presents a detailed investigation of novice students' learning of computer programming. The research project used a mixed-methods approach that ties together a controlled study with upper secondary school students with naturalistic classroom studies at the university level.

Underpinning the research is a pragmatic philosophical view on knowledge and learning. Pragmatism places "doing" at the forefront of knowledge acquisition, and doing is also central to this thesis. Learning that takes place in the computer lab is the primary object of investigation. Throughout the research, hands-on learning (direct control of keyboard and thus interaction with the computer environment) is compared with hands-off learning (other participation in programming activities).

Results from the controlled study demonstrated that hands-on learning decreased stress and was beneficial for learning over time. However, the usefulness of hands-on learning could not be confirmed in terms of improved learning outcomes immediately after students' first encounter with programming. The beneficial effect of hands-on learning over time could be explained by emotional factors as mediators.

Findings from the naturalistic settings confirmed that emotions and the social setting in which the learning takes place were important. Novices experienced frustration when getting stuck but also joy when succeeding — an "emotional roller coaster". All students report that hands-on code writing is necessary when learning to program.

Students in this research appreciated time for individual interaction with the computer environment, and time for working together in pairs to handle the emotional roller coasters. Experiencing negative emotions, or not getting personalized help when stuck, seemed to negatively affect student beliefs about themselves as programmers, and may result in decisions not to continue with computing.

Two new concepts, 'practical thinking' and 'come to agreement', were developed, which tie pragmatic knowledge theory to learning to program. These concepts help to understand and re-describe the purpose of an introductory programming course that in a way emphasizes the "doing". This novel use of pragmatism in CER is one important contribution to the research field. The research presented in this thesis has implications for how programming education can be understood beyond the dichotomy of theory and practice.

Keywords: Programming education, Novice programming, Hands-on learning, Higher education, Pragmatism, Dewey

Kristina von Hausswolff, Department of Information Technology, Division of Scientific Computing, Box 337, Uppsala University, SE-751 05 Uppsala, Sweden.

ISSN 1651-6214
ISBN 978-91-513-1372-6
URN urn:nbn:se:uu:diva-461455 (http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-461455)

To Fredrik,
Elvira, Gottfried,
and Åsa

List of Papers

This thesis is based on the following publications, which are referred to in the text by Roman numerals. Reprints are made with permission from the respective publishers.

I von Hausswolff, K., Eckerdal, A. (2018) Measuring Programming Knowledge in a Research Context. *In 2018 IEEE Frontiers in Education Conference*, (pp. 1-9)

II von Hausswolff, K., Eckerdal, A., Thuné, M (2020) Learning to program hands-on: a controlled study. *In Koli Calling'20: Proceedings of the 20th Koli Calling International Conference on Computing Education Research*, (pp. 1-10)

III von Hausswolff, K., Weurlander, M. (2020) Social dimensions in the lab session when novices learn to program. *In 2020 IEEE Frontiers in Education Conference*, (pp. 1-9)

IV von Hausswolff, K. (2021) Practical thinking while learning to program–novices' experiences and hands-on encounters. *Computer Science Education*, (pp.1-25)

V Weurlander, M., von Hausswolff, K. (2021) Engineering students' strategies to learn programming correlate with motivation and gender. *In 2021 IEEE Frontiers in Education Conference*, (pp. 1-9)

VI von Hausswolff, K. (forthcoming) Practical thinking and learning strategies: novices experiencing learning programming. *Manuscript*

Author contributions

Contributions of the respective authors for each paper are detailed below.

I For Paper I, I did the data collection and developed the structure of the paper. The setting and the aim of the study were developed by the second author together with the project group HOPE. I did statistical analysis and wrote the first draft.

II For Paper II, I did the data collection together with the second author. The setting and the aim of the study were developed by the second and third author together with the project group HOPE. I developed the structure of the paper together with the second and third author. I did statistical analysis and wrote the result part. All the other parts of the paper all the authors contributing equally to the writing.

III For Paper III, I did the data collection and developed the theoretical ideas. The setting and the aim of the study were developed by me. I developed the structure of the paper together with the second author and wrote the first draft.

IV For Paper IV, I am the sole author of this paper.

V For Paper V, the first author did the data collection and developed the structure of the paper. The survey was constructed by both me and the first author and I developed the theoretical ideas together with the first author. I did the statistical analysis and wrote the part about statistics. The first author wrote the first draft.

VI For Paper VI, I am the sole author of this paper.

Supporting work

Jevinger, Å. and von Hausswolff, K., 2016, March. Large programming task vs questions-and-answers examination in Java introductory courses. In *2016 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)* (pp. 154-161). IEEE.

von Hausswolff, K., 2017, August. Hands-on in computer programming education. In *Proceedings of the 2017 ACM Conference on International Computing Education Research* (pp. 279-280).

von Hausswolff, K., 2017, November. Practical thinking in programming education. In *Proceedings of the 17th Koli Calling International Conference on Computing Education Research* (pp. 203-204).

von Hausswolff, K., 2018, August. Practical Thinking while Programming: A Deweyan Approach to Knowledge in Computer Science. In *Proceedings of the 2018 ACM Conference on International Computing Education Research* (pp. 268-269).

Berglund, A., Dorji, T., Lhamo, D., Tshering, P., Von Hausswolff, K., Wangchuk, K., Wangchuk, T. and Wangchuk, Y., 2020, October. The exchange programme between new and different partners, Royal University of Bhutan and Uppsala University. In *2020 IEEE Frontiers in Education Conference* (pp. 1-4). IEEE.

Grande, V. and Von Hausswolff, K., 2020, October. “Mature” to Doubt: Using Ethical Theories for Role Modeling in Computing Education. In *2020 IEEE Frontiers in Education Conference* (pp. 1-9). IEEE.

Contents

Introduction

Computer programming is now considered to be relevant for citizens in many countries. Australia, England, Estonia, Finland, New Zealand, Norway, Poland, South Korea, and the US have all introduced or plan to introduce programming in the elementary school curriculum. In Sweden, a decision to include programming in both primary and secondary school was made in 2017 by the Swedish National Agency of Education. Programming was decided to be integrated into the existing subjects of Mathematics and Technology, and that initiated a curriculum revision in 2018. Explicitly introducing programming in schools was part of an overarching aim of strengthening the digital competencies in elementary education in Sweden. With this as a background, understanding how novices learn to write computer programs is an important national and global concern. There is a growing need for research on how novices learn to program.

Novices learning to program in introductory programming courses in higher education has been studied for decades. In Computing Education Research (CER), a substantial amount of studies has been conducted, particularly in engineering education. Some understanding has been gained but research still echoes the widespread conception that programming is hard to learn. Researchers in CER continue to ask "Why?" (e.g., Bosse and Gerosa (2017) or Guzdial (2010)). Now, more and a broader range of students are learning to program as part of their education which enhances the interest.

Computing education research, similar to research in other areas of science education, like physics, has emphasized the learning of concepts over practical leaning in the labs. This might be related values from a positivistic tradition in mathematics and science, which have been closely related to computer science. This tradition implies an objectivistic view of reality, and knowledge acquisition as forming a correct mental model of that reality. As a consequence of and compatible with that view, a cognitivistic theory would seem suitable. The research presented in this thesis takes a different approach. This thesis is focused on the practice of programming, including a theoretical framing that helps us bridge the gap between "theory" and "practice". Skill acquisition takes place in the computer lab, which is the primary object of investigation in this thesis.

Lab sessions are a central part of nearly all programming education similar to other laboratory subjects. Students' active hands-on learning is perceived

to be very important. Nevertheless, there are significant gaps in understanding how, when, and why practical hands-on learning has positive effects.

Students that participate in the studies are in the age range of 16 to early twenties. Empirical studies have been conducted both in upper secondary school and in higher education, and all students have in common that they encounter programming for the first time. Another shared aspect is that the practical skill of making a computer program run, performing a specific task, is front and center.

This thesis takes a mixed method approach to investigate novice students' learning of computer programming. To this end, a number of factors were investigated including e.g. knowledge gain, stress level and motivation. The thesis uses a pragmatic philosophical view on knowledge and learning. Pragmatism places "doing" at the forefront of knowledge acquisition which is also the center of this thesis.

This thesis grew out a research project called HOPE—after its focus on hands-on learning in programming education. In the HOPE project, hands-on learning refers to learning in practical situations, where students learn through manipulation of physical objects or tactile experiences, which by necessity leads to reflection and observation. Conversely, hands-off learning refers to learning by observation, discussion, or reflection only (Eckerdal, 2015a). The HOPE project and the research settings are described in dedicated sections below. Research presented in this thesis is part of the project, but also includes a broader discussion on the value and role of practice in computer science and especially when learning to program.

Overarching research questions

Novices that encounter programming for the first time are studied both in a controlled setting and in an authentic classroom setting. The theoretical and methodological basis is a pragmatic view of knowledge and learning, resulting in a mixed methodology approach.

Students' own exploratory activities in the computer lab are considered important in learning to program (Höök & Eckerdal, 2015; Lahtinen et al., 2005; Winslow, 1996). It is accepted that practical "doing" is essential. Minogue and Jones (2006) state that research supports the claim that hands-on activities enhance learning but does not clearly address the questions of how and why hands-on activities are important. The computer lab setting includes a student, a computer environment, and a social environment. One part of the social environment can for instance be that the students work together in pairs. In higher education, pair-programming in the computer lab is a common way of working with assignments and is also a part of the research setting here.

As there are many factors that could come into play in this kind of learning situation, a research approach was adopted that seeks to include many factors

at the same time. The natural environment for this learning, sitting by a computer and interacting, also creates a natural way to follow the students' activities during a learning session, through webcam and screen recording.

This thesis has two overarching research questions.

Research question 1: What role do hands-on activities on the keyboard have in a pair-programming learning situation for novices?

To answer RQ1, an investigation was conducted of the effects of hand-on activities on learning outcomes after a three-hours teaching session, measured immediately after the session and after one week. In addition to learning outcomes, the effects on other relevant factors such as motivation and stress were investigated. Background factors that may affect learning outcomes are also in scope. A question following RQ1 is how a researcher should go about constructing a valid way of measuring learning outcomes, from a specific research context.

Research question 2: How do students experience learning to program as novices, focusing on the role of writing code, including emotional and social aspects?

For RQ2, the whole learning situation where the novices encounter programming by writing code is important. Because of this, the theoretical framing becomes central and an investigation of theoretical concepts that may be useful in understanding the learning situation was included.

Background and previous research

This thesis is partly a result of research done in a cross-disciplinary research project that involves several parts and several researchers. This section first briefly presents previous research in the area of novices' learning to program, then describes the research project, HOPE (hands-on learning in programming education), and finally describes the research settings of the studies included in this thesis.

Novices learning programming in CSE

In a review of research on novice programmers and introductory programming Robins (2019, p. 330) writes: "CS1 [a beginning programming course] has typically been regarded as difficult for students, with persistent and widespread reports of high student failure and dropout rates". Failure rates and reasons for poor performance in introductory programming courses, have received considerable attention from the computer science education community (Bennedsen & Caspersen, 2007; Lahtinen et al., 2005). Problems with introductory programming seem to prevail: a recent review of introductory programming found that "student engagement levels in computing are benchmarked as among the lowest of any discipline" (Luxton-Reilly et al., 2018, p. 86). Common problems novice students encountered were understanding the task and various aspects relating to the design and structure of the program (Robins, 2019).

Novices' experiences in introductory programming courses may affect their attitudes towards the subject negatively. Novice students have been found to experience that their skills for learning CS were insufficient and they tended to use "ask for help" problem-solving strategies (Schulte & Knobelsdorf, 2007). Negative emotions may negatively affect their beliefs about their abilities (self-efficacy beliefs) in programming (Kinnunen & Simon, 2012; Rogerson & Scott, 2010). Their beliefs about themselves as programmers may result in decisions not to continue to study computing.

Poor results has been attributed to the students' cognitive abilities, background, motivation etc. (Robins, 2010). Another possible explanation is the content: is programming especially hard to learn? A substantial amount of research has gone into the area of threshold concepts in computer science education (Boustedt et al., 2007; Eckerdal et al., 2006; Rountree et al., 2013;

Sanders & McCartney, 2016; Thomas et al., 2010). The theory of threshold concepts aims at identifying particular concepts that function as thresholds in different subject areas, like keys to the subject and likely problematic for students (Meyer & Land, 2005). Meyer and Land (2005) also introduce the related term "liminal space" for the transitional period between beginning to learn a concept and fully mastering it. The liminal space is troubling to be in, according to Meyer and Land (2005), and they warned that students may experience "difficulty and anxiety" in relation to learning threshold concepts. Eckerdal et al. (2007) found in an interview study on students learning to program that the concept "liminal space" was useful in understanding their experiences, and the results highlighted the emotional reactions of the students. One conclusion was that students emphasized the importance of practice. Another conclusion was that different students take different routes through the liminal space, with the possibility of getting stuck at multiple places.

Sorva (2010) examines the possibility of threshold concepts in introductory programming but is not convinced and discusses the apparent difficulties of finding consensus as to which they are—if they indeed do exist. Robins (2010) claims that the compound and intertwined nature of the programming subject is a reason for students' problems with learning to program. He claims that the concepts involved in programming are unusually tightly integrated, and this suggests that if a student fails to acquire some concept, the learning of further concepts becomes harder.

A common approach in introductory programming education is to let students collaborate in different ways (Sanders et al., 2017), one example is pair-programming. A literature review concludes that pair programming is beneficial for students' learning (Hanks et al., 2011). Pair programming seems to be beneficial for weaker students (O'Donnell et al., 2015) but the collaboration pattern within a pair are important. If one student dominates the work, this inequity could negatively influence the teamwork and the perception of pair programming (Bowman et al., 2019; Cao & Xu, 2005). Programming education often emphasizes lab work where students work hands-on with programming assignments, often in pairs, assisted by the teacher in the computer laboratory. Previous research has for example acknowledged that syntax as well as practical skills can be problematic for novices (Du Boulay, 1986) and also pointed to the beneficial effects of hands-on work (Eckerdal et al., 2007; Höök & Eckerdal, 2015). The important role of hands-on work in computing education, in general, is highlighted in the considerable body of research on the role of practice, both as means to reach learning goals, and as a goal in itself (Gross & Powers, 2005; Sheard et al., 2011). There seems to be a consensus among students as well as teachers that hands-on work is important (Lahtinen et al., 2005). Robins (2010) shows that besides the application of theoretical knowledge to practice, the other way is also important—from practice to conceptual knowledge. In line with Robins, Shneiderman (1977) argues for a spiral approach to teaching programming. Conceptual understanding is built up

by visiting the same concept several times in greater depths. The complex and intertwining dependencies between practice and theory in the computer lab are further examined by Eckerdal (2015b). Research in programming education seems to show that collaboration and hands-on work are beneficial for learning, but we lack explanation of why that is and an underlying theory to anchor that explanation. Novice programming is a topic of ongoing interest within the area of computer science education and this research aims to contribute to the understanding of described difficulties experienced by students when learning to program.

The research project HOPE

HOPE (hands-on learning in programming education) is a research project with several parts started in 2016, funded by the Swedish Research Council, aiming to gain a better understanding of hands-on learning in the context of computer programming education. The HOPE project included both a controlled study and studies in naturalistic settings.

Students' active learning in the form of hands-on encounters is perceived as positive for learning to program by both students and teachers (Lahtinen et al., 2005). In a review article Minogue and Jones (2006) argued that there is a research gap on how, when, and why practical hands-on activities have positive effects on learning.

These gaps become especially important in a subject like programming when, in practice, it is common for students to work in pairs where one student is more hands-on than the other. The importance of hands-on learning to facilitate learning has been demonstrated in previous research in areas like medical education (Preece et al., 2013; Weurlander et al., 2016), science education (Bivall et al., 2011; Jaakkola et al., 2011), and in computing education (Höök & Eckerdal, 2015; Kaila et al., 2010).

The controlled study explored the conditions of learning computer programming hands-on/hands-off involving upper secondary school students and included fMRI scanning of the participants' brains. The naturalistic classroom studies were all conducted at university level in introductory programming courses. The research group consisted of researchers from education, computer science education, and neuroscience (three senior researchers and two Ph.D. students). A description of how the project was organized and how it is connected to this thesis follows here. The next section then gives a more detailed account of the parts in focus for this thesis.

Data has been collected in three different contexts. Context A: Upper secondary school students (age 16-19) work with programming assignments in a three-hour extra-curricular lesson. Context A includes a controlled study and a pre-study. The pre-study (referred to as pre-study I) establish the validity of measuring programming knowledge in this specific context.

Context B: brain imaging data collection of the same students tested on programming questions in an fMRI scanning camera, one or two days after the three-hour lesson. Context C: an introductory programming course at university level, following a group of novices (age 19-20) throughout the course.

Context A and B are part of the same controlled study. Context C is a naturalistic classroom study that includes a second pre-study (referred to as pre-study II) and the main classroom study. The content in this thesis is from Context A and Context C. In the remaining text, the controlled study in Context A is referred to as the controlled study. See Table 1 (p. 25).

In the controlled study, hands-on learning refers to learning in practical situations, where students acquire knowledge while using the keyboard directly interacting with the programming code, which leads to reflection and observation. Conversely, hands-off learning refers to learning by observation, discussion, and reflection only. At the core of the study are comparisons between those two groups of students and drawing conclusions both on the differences and also on the factors that are shown to impact the learning outcome. Context B investigated differences in activities in different brain areas when those two groups answered questions about programming (for results and further description see Lidström (2021)).

Research settings of the studies in the thesis project

As we have seen, this thesis contains two main studies, the controlled study and the main classroom study, with a pre-study for each. An overview of the different parts, the education level involved, and the data collected is presented in Table 1 (below). Thereafter, the settings of the four studies are described in more detail. A mapping of the studies to thesis papers is presented in the Methods and methodology section.

Table 1. Overview of the four studies included in this thesis, with the educational level included as well as the data collected.

STUDY	LEVEL	DATA
Pre-study I	Upper secondary school.	Knowledge test (n=60)
Controlled study	Upper secondary school. After-school activity.	Knowledge tests, video recordings, questionnaires (n=53)
Pre-study II	First year university level. First programming course.	Interviews (n=7), observations, survey (n=77), course material.
Main classroom study	First year university level. First programming course.	Interviews (n=20), observations, survey (n=67), course material.

The controlled study

In the controlled study in upper secondary school, students (n=53, 27 students worked hands-on and 26 hands-off) learned basic Java programming for three hours. This was a voluntary after-school activity. Factors suggested by previous research that could play a part in this learning situation were included when examining possible effects of the hands-on/hands-off condition.

The students work in groups of two, a pair-programming-inspired setting. They alternate between being the "driver", who types the code, and the "navigator" (Salleh et al., 2010), who is responsible for detecting errors and helping to solve the problem. In our setting, however, the driver and navigator never change roles. The reason for this design is that it supports the investigation of hands-on vs. hands-off, in that both students in a pair are involved in the problem solving, but only one is hands-on during the teaching session.

The controlled study was exploratory. After a three-hour teaching session, the students had to answer a knowledge test, questions on a Likert scale on motivation and stress both before and after, in addition to several other tests (for details, see pp. 61-63). After one week, their knowledge was measured again. The results of these measurements were analyzed to examine possible influencing factors given differences between the group typing on the keyboard (hands-on) and the group not typing on the keyboard (hands-off). A measure of social interaction was collected through analysis of films of the students teaching in progress.

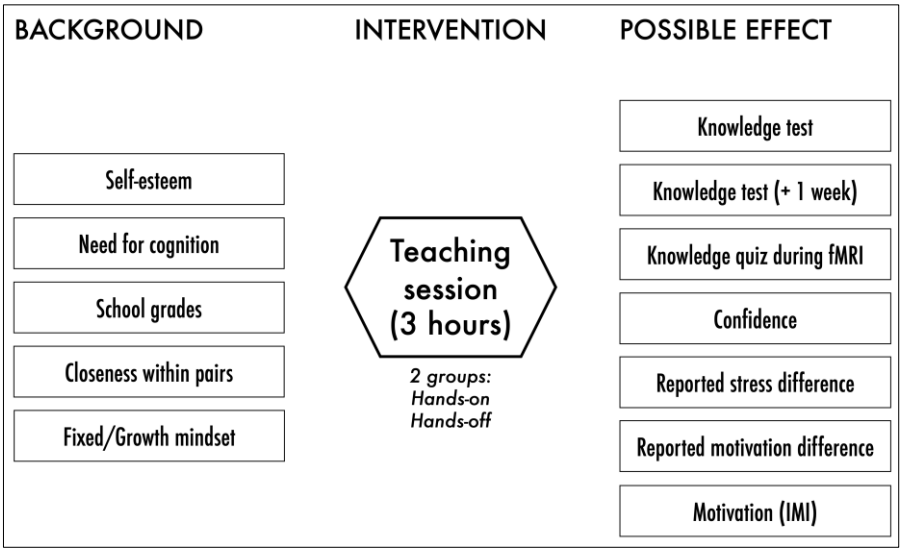

Figure 1. Variables in the controlled study (von Hausswolff et al., 2020, p. 2)

Pre-study I

To validate a test constructed based on the content of the three-hour teaching session, data were collected from 60 upper secondary school students with about half working hands-on and half working hands-off. The data consisted of the answers to programming questions after the session, and background questions (age, gender, and previous experiences of programming). Only students with no previous experience of programming were included in the analysis. Even though the hands-on or hands-off conditions were randomized some students did not comply so they were categorized after how they had actually worked in the session.

Pre-study II

During two following years a university introductory programming course was studied. First year engineering students were introduced to text-based programming using Python. In Sweden, undergraduate education is based on courses, taught and assessed separately. During the first year, when this programming course was given, it can be seen as a minor. Students could choose a specialization during their second year that would make them Computer Science majors. This course corresponded to five weeks full time, and ran part time in parallel to other courses. The structure of the course had been consistent for several years at this particular university, and the content was typical to CS1: basic programming structures such as selections, loops, functions, lists and files. The course consisted of lectures, exercises, assignments and six

mandatory computer labs (duration about two hours each). The class was divided into lab groups of about ten students, each assigned a teaching assistant (TA).

The focus of this study was six labs were pair programming was implemented. Students were instructed to act as the driver every 20 minutes. The pair programming setting was beneficial as it allowed for comparison between hands-on and hands-off experiences.

The students' experiences were studied by gathering backgrounds observation, a survey handed out at the last lab session, and seven student interviews within a month after the last lab session. The survey was mostly used to select which students to recruit for the interviews. The seven interviews were the primary data source used in analysis.

The main classroom study

The course setting remained similar to that of pre-study II, but with a variation on how to work during the lab sessions. Students could now work individually or together and there was no instruction of strict pair programming. The purpose of the entire classroom study was qualitative, but a survey was used to complement the other empirical material. Some of the same surveys/questionnaires as in the controlled study were used, which made it possible to collect quantitative data and analyze it with statistical methods.

The focus was on the five initial lab sessions (the first five weeks). This was studied by gathering background data as structured classroom observation in a lab group (12 students), filming of the lab session for some of the students (faces, audio, and screen capture), interviews with six students on three separate occasions during the course (beginning, middle, end). The survey was handed out in conjunction with an exam and 67 out of about 155 students answered the survey (response rate of 43%). The survey consisted of questions on their motivation and overall tendency to engage in thinking measured by standardized tools, essentially the same as had been used in the controlled study. There were also questions on background and prior programming experience, as well as questions on they work in the computer lab, how they learned programming (strategies), and experiences of and attitudes toward programming. These last questions about programming were constructed from findings in the controlled study and pre-study II and measured as agreement on statements on a Likert scale.

Theoretical framework

In the endeavor to investigate questions in the educational domain, learning theory can be used to frame the results and to motivate the research design. Just as important are theories of knowledge (or the result of learning) that anchor into philosophical assumptions about the world. Closely connected to stance in the theoretical domains stated above are methodological considerations that result in concrete method choices. In my understanding, all these questions are linked and will necessarily be intertwined with each other, which will be apparent in this section and the following.

In this section, I will position my research in a pragmatic framing, starting with a reflection of learning theory, then explaining my theoretical stance using pragmatism. As this is a thesis in computer science (CS) and a pragmatic stance entails a critical gaze at the learning content, a theoretical description of the selective tradition in CS follows. Upon that, I add a reflection of pragmatism as a theory in computer science education and of its comparability with using "mixed methods" methodology.

Learning theories and worldview

As described by Hodkinson et al. (2008), learning is often framed in one of two positions in a dichotomy; either in an individual view or in a social view. The dichotomy can be understood as stemming from two theoretical positions: a cognitive and a socio-cultural (Hodkinson et al., 2008, p. 29). The cognitive position can be described as a tendency to draw upon the root metaphor of acquisition to conceptualize learning, whereas situated learning theorists draw upon the metaphor of participation (Mason, 2007; Sfard, 1998). The cognitive approach places the individual thinker at the center in learning a specific content such as programming. The socio-cultural approach instead focuses on the situation where the learning is to take place, and looks at for example the classroom situation as well as the specific tradition of the learned subject such as computer science. The distinction is not always clear-cut and a cognitivist approach could include social aspects, while socio-cultural approaches could include aspects of individual thinking.

Hodkinson et al. (2008) argue for the incorporation of the individual view in a cultural theory of learning using pragmatism as a part of the theory. They remind us of the fact that John Dewey already questioned a too simplistic view

of cognition: all individual learning is through practice and continuous interaction (transaction) with the environment. Hence all individual learning is also social, there is a social dimension of the interactions taking place (where communication is a type of interaction).

In this thesis, I try to take both the individual and the social view into account when studying novices learning to program and argue for a pragmatic view of knowledge and hence learning. I try to connect these two views in a specific setting; learning programming in the computer lab. The research questions are centred around students' hand-on encounters with programming (in a wide sense). This entails both a narrower focus studying actual learning outcomes after a three-hour session, and a long-reaching aim in understanding students experiences during a course and connecting this with the computer science tradition. At the center of this research are individual students, although as Hodkinson et al. (2008) say, the individual student environment is created in interactions (transactions) with incorporated social dimensions. To gain knowledge, different aspects of a situation are separated (such as emotions and learning outcomes) even though they may very well be best described as being a part of a whole experience.

This thesis is mostly about learning programming in formal settings and is hence a part of a research area called computer science education. At the same time, it is also part of a research area of subject didactics which has its focus on teaching specific content. In Sweden, pragmatism has been used as the theoretical basis for subject didactics for some time (Wickman, 2012), although rarely for computer science. As Wickman discusses the matter of didactic inquiry he connects it with pragmatic theory of knowledge.

> Pragmatist epistemology is about knowing in action and about making sense of the consequences of actions in ways that help us proceed with our undertakings according to the purpose, for example when teaching for the learning of a certain content in the classroom. Knowledge is not ultimately a mental state of correctly understanding how language propositions correspond to the world […] Gaining knowledge thus means learning to successfully transform patterns of action in relation to situated consequences. The use of the term situated indicates that actions are not evaluated and judged by an isolated private mind, but always as part of already established social practices and institutions.(Wickman, 2012, p. 486)

The relationship between reality and the world according to a pragmatic view is not "as a matter of getting reality right, but rather as a matter of acquiring habits of action for coping with reality" (Rorty, 1991, p. 1). Looking at learning this way makes a path for discussing which content is used (and accepted) in a discipline and this is discussed using the concept "selective tradition" (Öhman & Östman, 2019). Subject didactics include questions of "what content?" and "why this content?". A selective tradition is shown in established syllabus and could be investigated through empirical examinations of syllabi.

These investigations could result in one or many traditions existing simultaneously. The term originated from (Williams, 1973), who points out that educational choices are embedded in culture and already established praxis.

Another concept in the tradition of Swedish subject didactics is *companion meaning*, developed by Östman (e.g. Östman (2010)). He means that alongside with the "content learning", students learn other norms and values not explicitly stated, as well as how to regard themselves in relation to all of the above.

Pragmatism

Charles Peirce, William James, and John Dewey are often referred to as the founders of pragmatism, starting in the latter part of the 19th century (Hamlyn, 1987). The philosopher Immanuel Kant named situations where knowledge and action are intimately linked "pragmatic" which inspired the name "pragmatism" (Biesta & Burbules, 2003). Action and knowledge are intertwined for the pragmatists. There is no knowledge without actions (Biesta & Burbules, 2003). Pierce's summary of pragmatism, which has become known as the pragmatic maxim, states the following:

> Consider what effects which might conceivably have practical bearings we conceive the object of our conception to have, then, our conception of these effects is the whole of our conception of the object (Peirce, 1878/1989, p. 88)

Dewey describes the world as a series of *transactions*, not a finite state that can be represented with the help of our language and give us objective knowledge (Biesta & Burbules, 2003). The transaction is "the point of contact" between an environment and an individual, and reality is experienced as a function of the transaction. Reality does exist in Dewey's ontology, but the way to "know" the world is through transactions with the environment. Biesta and Burbules (2003) calls Dewey's ontology transactional realism.

Transactions create experiences, knowledge is one form of experience. Knowledge shows itself in the way the individual has transactions with the environments, to "better" (more intelligently) respond to changes, first in the muscles, and then in the mind (Biesta & Burbules, 2003). Only when we actually act can we "know" if it was a good response, though the consequences. Thus, we can only gain knowledge through action. The meaning of an action and the meaning of its consequences are linked. The individual tries to get balance with the environment and then develops *habits*. Dewey also connects habits with communication in this way:

> The very operation of learning sets limit to itself, and makes subsequent learning more difficult. But this holds only of a habit in isolation, a non-

> communicating habit. Communication not only increases the number and variety of habits, but tends to link them subtly together, and eventually to subject habit-forming in a particular case to the habit of recognizing that new modes of association will exact a new use of it.(Dewey, 1925/1995, p. 280)

Dewey explains that the reason why we experience and describe the world uniformly is intersubjectivity (Biesta & Burbules, 2003). We as humans act together to achieve common goals and, in that process, the individual goals must be adapted so that a coordinated response to the action can be achieved. Through this kind of process, the world of the individual changes, and a common intersubjective world is created. Dewey names this process *communication*.

The world is in constant change and we can never be sure that our pattern of actions, as we have formed before, will suit problems that arise in the future. Biesta and Burbules (2003) denotes Dewey's account of knowledge an "action and communication theory" instead of an epistemology because of the completely different grounding in comparison to classical epistemologies. The action and communication theory includes fallibilism: we can never be completely sure of our knowledge.

When we learn, both the learner and the object of knowledge change through transaction. Dewey denotes this process of acquiring knowledge of the world—in other words, to learn—*inquiry* (Biesta & Burbules, 2003, p. 58). An inquiry is based on a situation that is *indeterminate*, where the predetermined habits do not work. In the inquiry process, the indeterminate situation is reformulated into a problematic situation, which is a conceptualization of the situation. To proceed in the process, the individual reflects and can use resources available in the situation, including as already acquired "knowledge".

Dewey especially directs attention to the fact that the connection between the thought model and the outcome of the action is in reality only a functional connection (Biesta & Burbules, 2003). The formulation of the problem and the consequences of different path of action are connected through the whole inquiry process, during this process both becomes clearer. When a satisfactory answer is reached, the process is completed and as a result, the individual has learned something new.

Pragmatism in educational science

The implications for teaching (and research on teaching) of Dewey's pragmatism are summarized by Biesta and Burbules (2003) in three statements. First, the fundamental difference between theory and practice has been abolished. Second, that knowledge relates to the world through action. Knowledge (models or theories) is only interesting if it helps us to solve practical problems in a rational and responsible way. Third, to strive for intersubjectivity rather than

for objectivity. Intersubjectivity is also Dewey's defense against relativism (Biesta & Burbules, 2003, pp. 105-106).

There is a growing body of research within subject didactics in different subjects that uses a pragmatic approach. Wickman (2013), in agreement with Dewey, argues that learning takes place according to the *principle of continuity*. He also connects this principle to the writings of the later Wittgenstein (Wickman, 2013, p. xii). This principle means that there is a history before the learning event and something coming after the event that is the purpose. The individual builds on previous knowledge and experiences and reconsiders these in the encounter with a new subject in relation to new experiences. There is thus a background history for both the individual and the subject that is a prerequisite in an inquiry. Continuity means that there is a purpose to the learning situation where encounters take place involving both the individual and the subject, resulting in a new development.

Dewey states that *meaningful* learning are those experiences that enable many new experiences in the future. Individuals use habits already acquired often not consciously reflected upon when dealing with events in the world. When these habitual actions are interrupted there is a reason to change direction, to focus attention on the interruption. They try to bridge the gap between what they already know to what they aim to understand by creating relations. This includes doubting, questioning, and reaching out for possible resources. Östman et al. (2019, p. 131) connect this with loops in the inquiry process and use this as the base of the transactional learning theory. Learning can be viewed as resolved gaps in inquiry loops. Minor disturbances result in small loops and minor changes in habits, whereas a major disturbance can result in a reflection, perhaps re-describing the problem or reaching out for resources available.

Furthermore, in a pragmatic framing, the purpose of education is discussed. (Biesta, 2009) describes three functions: qualification, socialization, and subjectification. Qualification is the learning of both *knowledge* and *skills*, for example to be a mathematician, but also including to be a citizen in society. To become a programmer also includes knowledge and skills in the same way. Biesta says that this is the most important function of the three. Socialization is to be a part of an already established order to acquire *norms* and *values* that are a part of being a programmer or a Swedish citizen. Subjectification is defined as the opposite of socialization, the part of the educational development that is unique to the individual: to "become" oneself in the educational setting. Van Poeck and Östman (2019) use Biesta's analytical division with the addition of dividing subjectification into person-formation and identity. This becomes useful when analyzing educational experiences and understanding norms and values as a part of an educational setting. Van Poeck and Östman (2019) argue that values are part of every discipline and that norms and values are acquired alongside the learning of knowledge and skills; called companion meanings or even *companion norms*. Östman et al. (2019) also emphasize the

connection between cognition and bodily feeling "all our knowledge, concepts etc. have the origin in bodily feeling" (Östman et al., 2019, p. 136) referring to both (James, 1912/2003) and (Dewey, 1925/1995). Thus, from a pragmatic stance, it is possible to discuss emotions and values that accompany cognitive learning. Educational researchers have a theory with which to connect actions and feelings to acquired knowledge and accompanied values. Johnson says:

> There is no cognition without emotions, even though we are often unaware of the emotional aspects of our thinking. The idea that meaning and understanding are based solely on propositional structures is problematic because it excludes (or at least hides) most of what goes into the way we make sense of our experience. (Johnson, 2007, p. 9)

Computer science as a discipline and Western epistemology

A foundation in Dewey's theory of knowledge (an action and communication theory) is his critique of Western epistemology in The Quest for Certainty (Dewey, 1930/2013). The basis for Western epistemology can be traced back to Plato's dualistic worldview; one world of ideas and one world of senses (experiences). Plato's two worlds were aiming at explaining both the experience of change in our world while also granting stable unchangeable knowledge and identity (Hamlyn, 1987). Plato's division is the start of several different divisions: objective-subjective, theory-practice, thought-action, mind-body. In this separation, a hierarchy also was established: objectivity, theory, thought, and mind is better than their opposites. It is not Platonism per se that Dewey is hostile to according to the neo-pragmatist Rorty but "a collection of philosophical distinctions (appearance - reality, matter - consciousness, created - discovered, sensual - intellectual and so on)" (Rorty, 1999, p. 7).

The division of theory (objective, eternal, unchangeable truths) and practice (subjective, situated, changeable beliefs) was adequate in the environment of ancient Greece as Dewey trace this division back to Aristotle (Plato's student). In this environment, humans were submitted to unpredictable changes, and understanding the world through thinking was the best way of handling this. In the theory of Aristotle (among others), thinking is the way of obtaining absolute certainty and truth. For many thinkers, this certain knowledge is also connected to God, who guarantees the eternal, unchangeable status of this knowledge.

Another way to handle a changeable world is to act and control the conditions of the environment. This is the way of arts and technology and as it relies on change it will always contain an element of uncertainty. Dewey argues that

the Aristotelian epistemology is an underlying misconception for all philosophical questions:

> They all flow – such is my basic thesis – from the separation (set up in the interest of the quest for absolute certainty) between theory and practice, knowledge and action. (Dewey, 1930/2013, p. 27)

Computer science emerges as a new discipline from 1930 to 1980, as the scientific discussion about certainty, truth, and value was ongoing. Computer science (CS) is the scientific approach to computers and information processes suitable for computer processing (Tedre, 2014). It involves both software and hardware as well as uses and interactions by and with humans, sometimes denoted computing. CS, with its roots both in mathematical logic and in engineering, has an inherent conflict between the two ways of going about knowing in this domain.

> Computer science is not primarily an empirical science. Its most basic parts can be seen as a form of applied mathematics, and otherwise, it is a technical science, in so far as it deals with principles for the construction of a certain class of artifacts. (Nationalencyklopedin, n. d.; Sandewall, 2021) [author's translation from Swedish].

The CS tradition includes both a mathematical part with a quest for unchanged knowledge and the technical part, solving problems in an ongoing changeable world. An additional layer on this is that the computer is human creation stemming from ideas of human thinking with its roots in the positivist paradigm (dominating during 1930-1940) (Goldfarb, 1996).

This quest for certainty was very central to discussions about a foundation of all knowledge (especially scientific) involving both logic and the discipline of mathematics in the 20th century (Tedre, 2014). Practice is valued less in this quest for certainty and this quest is central in the emerging of the new discipline of computer science and debates of valued knowledge in the new discipline (Tedre, 2014).

Developments in mathematical logic and the philosophy of science were strongly connected during the 20th century (Goldfarb, 1996). To ensure a "foundation" for mathematical knowledge and with that, all other scientific knowledge, several "programs" were created during the 20th century (Lakatos, 1980) For example logicism or Hilbert's program that mathematical knowledge could have a secure foundation if it could be shown that the system was without contradictions and was complete. But all programs failed: logicism with Russell's paradox in 1907 and Hilbert's program by Gödel's second theorem (1933) that states that all the truths generated from a set of axioms (if the axioms are sufficiently numerous) can never be captured by a logical system (Lakatos, 1980; Tedre, 2014).

Even though the "quest for certainty" in mathematics failed, mathematical knowledge is presently viewed as among the most certain, and the discipline of mathematics is highly valued in the scientific field. The "quest for certainty" that Dewey question continued to have bearing on how disciplinary knowledge are valued, one example are tensions within the new discipline of computer science.

Tedre (2014) describes the tension between mathematics and computer science as a divorce where computer science wants to be a discipline in its own right, while also keeping the highly valued mathematical knowledge claim (of almost certain knowledge). As such, the engineering part of computer science was devalued as engineering is not about truth but what works (value-ladder). Tedre states "for decades engineers, with its practical aims, was seriously undervalued in academic computing" (Tedre, 2014, p. 9). The engineering approach can easily be sorted under the technology and art category where practical knowledge was available but without any guarantee of safety.

Computer logical thinking and pragmatism

Computational thinking (CT) is a concept that has gained renewed interest in relation to digital competence and programming in schools (Tedre & Denning, 2016). The discussion has revolved around what this thinking is, how important this thinking is, and also its relationship to practical programming. According to Tedre and Denning (2016, p. 12), there is no definition of CT that everyone agrees on despite the popularity of the concept. One possible definition that the authors put forth because of its clarity is that CT is "thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms" (Aho, 2011). The authors also believe that the enthusiasm for CT risks overestimating theory in relation to practice. It is possible to see connections between Dewey's investigation of theory and practice with today's discussion of CT.

Three approaches to the relationship between CT and technology have been described in the debate (NRC, 2010, p. 12): CT is completely separate from technology, CT and the technology is inseparable, and that CT has evolved from the technology but is now to be regarded as separated from it. Dewey has strongly argued against separating action and thought and I believe that from a pragmatic perspective, CT and technology should be treated as inseparable. This also means that what is included in "computer science thinking" can change over time as technology changes and that knowledge is situated and should be viewed from a context (von Hausswolff, 2017).

As we have seen, the dichotomy between theory and practice has its equivalent within the computer science selective tradition, stemming from the mathematical (theoretical) and the technical (practical). There are examples of how epistemological traditions have been linked to theories of teaching in the computer science literature. Stein (1999) points to problems with the computer

science tradition from an "embodied interaction" perspective and believes that teaching needs to change the metaphor from "computation as calculation" to "computation as interaction". Among other things, she criticizes the *sequence* (instructions following each other in sequence and are executed in that order) as a prototype of programming and believes that it expresses a desire to abstract away events in the physical machine. Björkman (2005) explicitly points to connections between epistemological traditions, teaching, and gender issues in computer science. She questions the strong position of mathematics and abstractions in the computer science knowledge tradition and links it to the fact that computer science is male-dominated. Ottemo (2015) has examined the relationship between masculinity and computer science education and points to the glorification of "pure" incorporeal thinking that manifests itself in a type of masculinity that is "disembodied".

Educational science research and pragmatism

Educational science research includes a theoretical framing. It all relies on approaches to the philosophical questions: What exists (ontology)? What is truth? What is knowledge? And how do you learn? The answer to these questions can be said to describe a researcher's worldview or basic assumptions about the area to which the research relates.

The literature describes a paradigm war, from the middle of the 20th century onwards, within the disciplines of social and behavioral science (Polkinghorne, 1983; Tashakkori & Teddlie, 1998) where educational science is included. A positivist paradigm on one side of the struggle with roots in scientific research and on the other side a more interpretive/constructivist/heuristic paradigm on the other, focused on understanding human phenomena. This paradigm war also connects to methods preferred in research. The positivist paradigm is associated with a quantitative approach and the constructivist paradigm with a more qualitative approach.

However, these types of positions of researchers are made in a social and cultural context. The researcher is both part of a tradition and an interdisciplinary discussion, and needs to take a stand on these. A basic view that dominates a research field for a time is usually described as a paradigm (Kuhn, 1962) or the core of a research program (Lakatos, 1970). Which research questions that are considered relevant and important in the field of research are also related to the tradition in the field.

The paradigm war described in the literature is the view that these two conceptions of ontology, truth, knowledge (and learning) cannot be reconciled. Each side says that their starting point is more correct or more fruitful. In recent years, however, other paradigms have emerged. Tashakkori et al. (2021, p. 42) describe pragmatism as third way of approaching research and a methodological philosophy that takes hold of both positivism and constructivism

methods and logic. Therefore, pragmatism is suitable as a theoretical stance for research design with "mixed methods" (Johnson & Onwuegbuzie, 2004; Tashakkori & Teddlie, 1998).

Johnson and Onwuegbuzie (2004) believe that both qualitative research and quantitative research have similarities, e.g. that they examine reality empirically, describe data and construct explanatory arguments with data, and speculate about the result of the investigation. Both include ways to avoid misinterpretations or misconceptions in the interpretation and analysis of data. All research in the social sciences involves claims about people and the environment in which people are. Descriptions of this can involve both holistic phenomena as well as more specific micro-phenomena. Here, there can be room for both a social description of reality as well as a clear material reality, according to Johnson and Onwuegbuzie (2004). Mixed methods "entails a combination of 'qualitative' and 'quantitative' approaches with the ambition to generate a more accurate and adequate understanding of social phenomena than would be possible by using only one of these approaches." (Biesta, 2012, p. 147).

Tashakkori and Teddlie (1998, pp. 24-30) highlight several points where the pragmatic knowledge theory provides the opportunity for combinations of methods and analysis types. They mention that the research cycle from a pragmatic point of view can contain both deductive and inductive logic. This can be seen as an application of Dewey's inquiry: observations and facts create models inductively that provide the opportunity for predictions/hypotheses that can be tested deductively against observations and facts.

The pragmatic "knowledge theory" provides an opportunity to both describe events "objectively" (or intersubjectively) and at the same time provides space for subjective knowledge. During certain parts of a research project, the researcher and the research object may be more interwoven, and during other parts, the researcher may take a more distant, "objective" position. Tashakkori and Teddlie (1998) describe "objective" and subjective knowledge as a question of how involved the researcher is in the knowledge descriptions (the knower's relationship to the known). "Objective knowledge" is thus better referred to as "intersubjective knowledge". The knowledge descriptions concern knowledge that different people can arrive at independently of each other. Intersubjectivity relates to our commonly perceived reality and in some cases includes specialist knowledge in disciplines. As an example, statements such as "programming is a central part of the subject of computer science" can be an intersubjective statement that those involved in computer science agree on. The pragmatists believe that values/norms play a role in the inquiry process and also govern the choice of research questions and possible knowledge claims. The important thing for a researcher is to describe the value-based conditions for the study and that all parts of the study are coherent.

Tashakkori and Teddlie (1998) further argue that the most controversial aspects of pragmatism are its ontology and view of causality. The transcendental realism of pragmatism means that social phenomena exist in an objective reality and that there are stable connections between different parts of reality. Reality thus does not depend on our experience of it. But pragmatists do not seek the "truth" of this reality. The knowledge claims that can be made are never completely independent of beliefs and interests. As a result of the ontology, pragmatists believe that causal connections may exist but that it is not entirely possible to determine what they look like.

Other questions to consider are: Which knowledge claims are meaningful to make? Can different ontological and epistemological views be combined? From this follows problems in describing the purpose of the research. Can the purposes that include causal explanations be combined with more interpretive purposes? Even questions about the practical orientation follow from the epistemological and ontological positions. Can the research be both solution-oriented and oriented towards critical understanding? The mixture can be more complex and the research design should be explicitly argued for with regard to epistemological and ontological issues. It may turn out that certain combinations are difficult to make or even incompatible with each other. Both ontological and epistemological positions affect what can be achieved with the research and what practical consequences the research results can have.

Methodology and methods

Methodology can be described as the way researchers approach the focused research area, more precisely to answer the research questions. Methodology includes a philosophical stance that anchors both the method chosen and the theoretical framing. From this philosophical stance follows the choice of logic, axiom of value, and a view of causal relations which is all a part of the chosen methodology (Tashakkori & Teddlie, 1998, p. 23).

Arguments for a research design with "mixed methods" are often described as putting the research question at the forefront guiding the research design and choice of methods.

Based on a quantitative approach, the choice of method often falls on so-called quantitative methods, such as experiments/ questionnaire studies analyzed with statistics. The purpose is often to generalize and these generalizations can then be expressed in the form of numbers. Based on a qualitative approach, the choice of method often falls on so-called qualitative methods such as interviews and observations, where the purpose is often to in words describe events or phenomena.

"Mixed methods" on the other hand are described by Biesta (2012) as the emergence of the development of triangulation as an approach to scientific inquiries. Triangulation is often used in social science research where one and the same phenomenon is studied with several different methods and where the multi-perspective approach of the phenomenon strengthens the credibility of the result. Triangulation is a possible purpose of using "mixed methods" according to Biesta: "seeking convergence and corroboration of results from different methods and designs studying the same phenomenon" (Biesta, 2012, p. 147).

He further describes four other possible purposes; complementary, initiation, development, and expansion. Complementary means that results from one method can clarify or reinforce results from another method. Initiation means the opportunity to detect any contradictions in the research question and as a result, reformulate it. Development means that results from one method are part of the development of another method. Expanding means that the domain for knowledge claims can be expanded by using different study methods in different parts of the research.

Biesta (2012) also distinguishes seven possible levels where the different approaches can be mixed: data, method, design, epistemology, ontology, purpose of the research, practical orientation. When the term "mixed methods" is

mentioned, it may include one or more of these levels, and this is often unclear. However, mixing different types of data, methods and designs is relatively unproblematic according to Biesta (2012). He believes that data that contains both text and numbers can be combined and that different types of data collection methods and different types of analysis of data can be combined. By design, Biesta means that experimental/intervention design can be combined with naturalistic investigations.

This thesis concerns three levels of mixed methods: data, methods and design.

Research design and methods

A pragmatic approach to educational science puts the research questions at the forefront in combining methods in a mixed-method research design and therefore we are reminded of the research questions:

RQ 1: What role do hands-on activities on the keyboard have in a pair-programming learning situation for novices?

RQ 2: How do students experience learning to program as a novice, focusing on the role of writing code, including emotional and social aspects?

To describe the different methods in a research project that contains several parts, a notation model is usable (Creswell et al., 2003), and in this thesis, the Morse model is used to describe the research design. The research can be simultaneous (+) or in sequence (→). The research paradigm that is dominant in the research design is described with the notation uppercase letters for dominant (KVAN, KVAL) and lowercase letters subordinate (kvan, kval).

In the controlled study, quantitative data was collected by means of questionnaires. Also, qualitative data (films of student pairs working at the computer) was converted into quantitative data.

A measure of social interaction was collected through analysis of films of the students during the three-hour session. Parts of the material needed further interpretation to be converted into quantitative data, for example, the case where students' interaction was captured on film. The knowledge test also involved post-processing of the collected material for quantification. Those two processes were the qualitative parts of the controlled study, described both in Paper I and in Paper II but most of the collected data were quantitative, see Figure 2 (p. 59).

The controlled study generated a lot of numerical data about one and the same teaching situation (three-hour teaching session), which enabled in-depth knowledge of connections. This study should be described as KVAN + kval.

The in-classroom study was mostly qualitative, but parts of the same questionnaires as in the controlled study were used in the classroom study, which provides an opportunity for the collection of quantitative data. Attitude questions on a Likert scale were constructed and analyzed with statistical methods. The purpose of the entire classroom study is qualitative, but a survey is used to complement the other empirical material. This study should be described as KVAL + kvan.

The whole research design can thereby be described as:
KVAN + kval → KVAL + kvan.

Methods in each paper

As we have seen, the four studies resulted in six papers included in this thesis (see Figure 2, p. 59). This section describes how the studies and papers all relate.

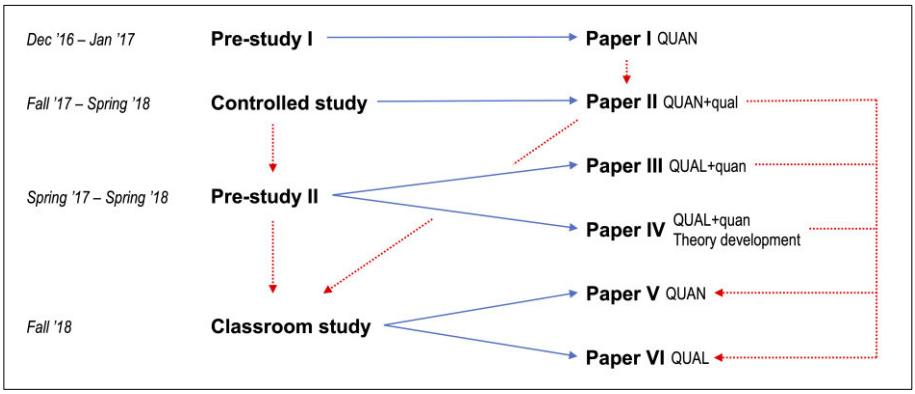

Figure 2. Blue solid arrows show the connections between studies and papers. Red dotted arrows show how prior studies (with results) influence subsequent studies and papers.

The result of the first pre-study was used to validate the measurement of programming knowledge in the controlled study, described in Paper I. The findings from the controlled study informed the design of pre-study II, which was conducted in a naturalistic setting (hand-on/hands-off in a pair-programming setting). The result of the controlled study was used to select survey questions and also influenced the interview questions. Both Paper III and IV describe results that were further built upon in Paper V and VI. In Paper V experiences expressed in interviews were used to develop survey statements. In Paper VI developed theoretical concepts as a result of pre-study II were further investigated and discussed in light of new data.

Paper I: Measuring Programming Knowledge in a Research Context

Paper I corresponds clearly to pre-study I, where data were collected in Dec 2016 and Jan 2017 on three different occasions. Again, the data consisted of the answers to programming questions after a three-hour teaching session, as well as background questions (age, gender, and previous experiences of programming).

The aim of Paper I was to describe a process of establishing validity of a context-specific assessment tool for novice programming and discuss how programming knowledge could be measured after only one three-hour teaching session. Previous research, an empirical sample, and intersubjective knowledge judgments were used to construct and validate the assessment tool. The literature reviews validated the programming content of the three-hour teaching session as well-established content in an introduction to programming course. Empirical data was collected from 60 upper secondary school students with about half working hands-on and half working hands-off. Statistical methods are used to establish the validity of context-specific assessment tool and reliability in terms of interrater reliability (Stemler, 2004).

In addition to that, some group comparisons are done with the independent-samples Mann-Whitney U test (Corder & Foreman, 2014) for differences between students working at the keyboard hands-on and students who did not (hands-off).

Paper II: Learning to program hands-on: a controlled study

The controlled study resulted in Paper II. Participants were upper secondary school students (n=53, 27 students worked hands-on and 26 hands-off) that volunteered to participate in a basic Java programming for three hours outside their ordinary classes. The teaching session consists of the teacher presenting programming concepts and showing examples of problem-solving with programming in pre-prepared filmed teaching session. Students worked in pairs to solve programming problems. In each pair, one student was randomly selected to write the code hands-on, while the other student contributed hands-off. The roles did not switch during the session. In this way the groups hands-on/hands-off were randomly selected. The learning outcome immediately after the session and the long-term learning, after one week, were measured by knowledge tests.

Several questions and questionnaires were handed out to the students at different times during this study. Questions on how well the participants knew the student they were paired with, on the scale “Very well”, “Rather well”, “A little”, or “Not at all”, and if they had chosen him or her beforehand, questions on how stressed they felt both before and after the teaching session, questions on how motivated they were for learning programming both before and after

the teaching session. Both questions about stress and motivation were given on a scale from 1 (least) to 10 (most). In addition, an Intrinsic Motivation Inventory (IMI, McAuley et al. (1989)), was handed out, with questions on interest, competence, effort, pressure, and usefulness related to the teaching session.

A Need For Cognition (NFC) questionnaire (Cacioppo et al., 1984) an assessment instrument that quantitatively measures the tendency for an individual to engage in and enjoy thinking, questionnaire on their self-esteem generally (Rosenberg, 1965), a questionnaire on the attitude to fixed/growth mindset (Grant & Dweck, 2003). All those three questionnaires were designed to measure attitudes relatively stable over time. All the questionnaire was translated to Swedish. To quantify the engagement factor from the films (that captured the students working together) qualitative methods were used. One additional background data, to provide a rough measure of learning ability, were the students' grades from school year nine (the final year before upper secondary school in Sweden). The idea was that students with high grades could be expected to benefit more from the short teaching/learning session than students with more modest grades, so this would be a factor to check for in the statistical analysis.

The tests and questionnaires assess the learning outcome and some other aspects of relevance for learning, see Figure 1 (p. 27). The groups hands-on vs. hands-off could be compared since both students in a pair were involved in the problem solving, but only one was hands-on during a teaching session.

Quantitative methods were used to analyze the data collected. To do statistical analysis the software SPSS was used. The correlation between the measured variables and comparison between the two groups were made, and as our data were not normally distributed we have used non-parametric methods: Spearman's Rho correlation coefficient and the independent-samples Mann-Whitney U test for differences between groups (Corder & Foreman, 2014). However, by appropriate data transformation we managed to use ANCOVA (Hair Jr et al., 2014) to build a model reflecting the results of the analysis (Graziano & Raulin, 1993; Mason et al., 2003).

Paper III: Social dimensions in the lab session when novices learn to program

Paper III stems from the second pre-study, which followed an introductory programming course during two following years. Data were gathered by structured observation in a lab group (10 students), a survey handed out at the last lab session, and seven student interviews within a month after the last lab session and course material. The seven interviews were the main data source used in the analysis, and the remaining data were complementary or used as background information.

The interviews were conducted at the school in a separate room during available daytime breaks. The interviewed students were all novices at programming before taking the course. The interviews were about 30 minutes long and the interviews were recorded and transcribed. All interviews were semi-structured, with questions focused on how students experienced learning to program in the lab setting. The hands-on aspect of learning was in particular focus. The seven interviews were analyzed using a qualitative content analysis. This was done in two steps, the first step was to analyze the stated content (Elo & Kyngäs, 2008). This was to categorize the student's statements into themes looking at what they are actually saying in the interview, only looked for patterns/themes that are explicitly stated in the interview transcripts Themes across the whole dataset were identified, as well as a structure of the content. Quotes from the transcribed interviews were coded in N'Vivo. During the analysis the codes were divided into three groups. Codes that describe experiences of programming that involve the whole setting and the social dimensions, were described in paper III. The grouping is further expanded upon in the Findings section.

In the second step pragmatism as a theoretical framing was used to deepen the understanding of the content. Focusing on one particular group of themes is suggested by (Braun & Clarke, 2006, p. 83), trying to see particular latent themes enlightened by theoretical concepts.

Paper IV: Practical thinking while learning to program – novices' experiences and hands-on encounters

While Paper III focuses on social dimensions, Paper IV present codes that describe an emerging relation between the student and the computer/program environment, evolving during the course.

In Paper IV, the method is the same for Paper III, since the empirical result in the two papers spring from the same data and analyses. In addition to using the qualitative content analysis method, a theoretical anchoring in pragmatism was conducted resulting in the development of new theoretical concepts to help understand the result. Paper IV gives the as of yet most detailed analysis showing how programming education can be understood in the light of pragmatic theory.

Paper V: Engineering students' strategies to learn programming correlate with motivation and gender

Paper V reports on the main classroom study. A survey was handed out to the students in an introductory programming course about halfway through the course. The survey included items from standardized instruments on motivation (IMI, McAuley et al. (1989)), the overall tendency to engage in thinking

(NFC, Cacioppo et al. (1984)) items that was used also in the controlled study. The survey also included items on age, gender, and prior programming experiences, alongside questions about how they work in the computer lab, how they learned programming (strategies), and experiences of and attitudes toward programming. The design of the survey was informed by the results of pre-study II. The survey was handed out in conjunction with an exam and 67 out of about 155 students answered the survey (response rate of 43%). An exploratory factor analysis (Hair Jr et al., 2014) of the items relating to how students learn programming was conducted. Statistical methods were used to investigate possible connections between the data construct we examined. A cluster analysis (Hair Jr et al., 2014) to group student after how the choose to work in the lab setting and comparisons between groups was done using an independent t-tests or a Mann-Whitney test depending on the distribution of the data (Corder & Foreman, 2014). Correlation between the variables were measured using non-parametric methods: Spearman's Rho correlation coefficient or parametric methods: Pearson's r - Pearson product-moment correlation coefficient.

Paper VI: Practical thinking and learning strategies: novices experiencing learning programming

Paper VI also build on the main classroom study. As described previously, a series of interviews were conducted with students in an introductory programming course. To deepen the understanding of the experience of learning to program, the concepts "practical thinking" and "come to agreement" (developed in Paper IV) are used. The main data used in this study were interviews with six students on three different occasions during the course (18 interviews in total). A qualitative research approach was adopted using the inference method that could most accurately be understood as abduction. Abduction was used as the analysis initially was built on predetermined theory, but as the analysis progressed, new directions were incorporated resulting in a deeper understanding of both the predetermined theory and the situation. Alvesson and Sköldberg (2017) describe abduction as a third logic of inference besides deduction and induction. Abduction relies on empirical evidence (as induction) and also uses theory to make conclusions on specifics (as deduction). In that sense abduction has elements of both the two other "classical" inference logics, but should be considered an inference logic in its own right not just as a "mix" of the other two (Alvesson & Sköldberg, 2017, p. 5). Theory is used and included in the patterns emerging from the data during the analysis. Abduction could include both confirmative elements of results or theory described in previous research.

Some data analysis was performed already during the interviews, by noting connections to course material and previous observations. The interview guide

was refined during the process in discussions with a second researcher. Immediately before the second and the third interviews, the audio recordings of previous interviews with that student were reviewed. Analysis of the complete data was both broad, by examining all data in chronological order, and focused, by analyzing data for each individual student. Throughout this process, quotes in N´Vivo were coded with increasing refinement. Three main stories emerged after several iterations (see Paper VI).

Validity and trustworthiness

I have been a part of a project including four senior researchers and one other Ph.D. student, where at least three different disciplines have been represented. During the controlled study, all results were discussed in this group, which arguably accounts for some validation of the reached conclusions. The use of standard statistical methods in the quantitative parts with appropriate measures of validity discussed together with the results adds to the trustworthiness of the obtained results.

One threat to the generalizability of the results is that all the students in the controlled study had grades above average and studied at a math-intensive study program. They also volunteered to be part of the study, meaning a potential problem with self-selection.

In the classroom studies, the research process of analyzing the results were continuously reviewed and discussed with at least one senior researcher in education with experience of doing qualitative research. Sometimes my initial interpretations of data have been reconsidered after discussions and good advice. Validity is strengthened by including context descriptions and referring to additional data sources collected at the time. In my reports, I present quotes richly together with my interpretations, which enables the reader to build an opinion of the data and how I have reasoned during the analysis.

As only one researcher did the data collection and the main analysis work, and the researcher's theoretical lens could bias both interviews and analyses, measures were taken to strengthen the credibility. This included in-depth discussions about the construction of questionnaires and interview guide with a second researcher, and transparency with the questions asked in the reporting of the results. The second researcher took part in discussing the whole analysis process and in particular in validation of the coding. All the interviews were recorded and professionally transcribed, and the transcripts in N´Vivo were part of the two researcher's validation work.

As to argue for dependability (Guba & Lincoln, 1982), it can be stated that in total 15 students were interviewed at three different instances of a programming course. Those 15 students make no claims of statistic representation, although a form of "generalization to a broader theory" (Cohen et al., 2011, p.

294) can be done. That is the transferability Guba and Lincoln (1982) refer to as a part of trustworthiness in qualitative research.

I have collected data from informants that represent the type of students which I address in my research questions and when the participants were recruited I aimed for a variation in age, gender, and experience to avoid a one-sided view of the phenomena in question. But it is important to note that all the participants in the classroom studies are doing their first year in engineering education and this could have some impact on the result.

The mixed methodology enables triangulation between different data sources and data collection methods which further strengthens the results of the research presented in this thesis.

Ethical considerations

Ethical approval was obtained for the controlled study from the regional Ethical Board in Uppsala, Dnr 2017/178. The approval form can be made available upon request.

Students were informed of the study, the purpose of data collection, and how data was stored and processed to protect their integrity. The information was given in writing. It was voluntary to participate in the study, and the participants signed a written consent to confirm their participation. In the controlled study most of the participants were younger than 18 years and requiring also their guardians to sign a written consent.

Information that could lead to identification of the educational program or the university was not included in this text so that the students' anonymity could be ensured. Names presented in the text are pseudonyms. Quotes are chosen to represent a holistic interpretation of the interviews. Throughout this research, I followed the national ethics guidelines for social science research (CODEX, 2020).

Findings

Findings are presented in this section in accordance with the three main empirical studies in this thesis. First the controlled study, including pre-study I. Second, pre-study II, and finally the main classroom study. This means that findings in the papers are presented together, with two papers in each section. Answers to the research questions connected to the papers are presented in connection.

The timing of the results can be good to bear in mind when reading. Results from the controlled study and results from pre-study II were obtained approximately simultaneously. The findings from the main classroom study were obtained at a later stage, influenced by the earlier results and could for that reason be seen as a continuation of those studies. In the last part of this section, the main findings regarding the two overall research questions are summarized.

Findings in papers I and II

The controlled study relies on the validity of measured effects of the condition hands-on/hands-off and the effects that are in focus are the learning outcome of an introductory three-hour teaching session. The aim of paper I was to describe a process of establishing validity of a context-specific test with a connected scoring rubric for novice programming (von Hausswolff & Eckerdal, 2018). This assessment of programming knowledge was used to construct two knowledge tests for the controlled study, one used immediately after the teaching session and the other used after one week. Based on a pragmatic view of knowledge, the validity argumentation relies on three ways for how our assessment of knowledge is considered valid: previous research, an empirical sample, and intersubjective knowledge judgments. See Paper I.

The test questions were of two types: writing code questions and multiple-choice questions. Three content areas were covered: syntax, sequence, and loop—consistent with the learning content and grounded in the tradition of how programming often is taught, established through previous research. The main result of our study is the test with the scoring rubric and our arguments for its validity. We also present some early findings from the empirical data collected on the effects of hands-on in learning programming. Initial statistical

testing on differences between hands-on and hands-off showed indications of students in the hands-on condition performed slightly better.

An important conclusion is that measuring programming knowledge is a relational activity. As has already been pointed out by e.g. (Cronbach, 1951), important aspects of a knowledge test belong both to the test itself and also to the context in which it is used. With an intersubjective view of knowledge (Biesta & Burbules, 2003), we focus our analysis on the communicative practices involved in creating and using the knowledge test and we conclude that such communicative practices benefit from careful considerations of consensus and disagreements. Overly focusing on judging specific questions could be a danger, and the process benefited from complementing the scoring rubric with holistic assessments. This was done by evaluating the total scores in comparison to the knowledge shown in the whole test and adjusting the scoring rubric accordingly.

The controlled study included several background factors and some measurements of possible effects. The setup is shown in Figure 1 (p. 27), in the section describing the research design. The hands-on condition showed no effect on the learning outcome immediately after the teaching session. The background factors that correlated with the learning outcomes directly after the teaching session were high scores on an NFC questionnaire and school grades from elementary school. The Need For Cognition (NFC) questionnaire measures the tendency for an individual to engage in and enjoy thinking.

The same background factors were important when measuring the learning outcomes after a week, with an addition of "closeness in the pair". This was measured by questions on how well the participants knew the student they were paired with, on the scale "Very well", "Rather well", "A little", or "Not at all", and whether they had previously chosen to pair up. When taking the background factors into account, an ANCOVA model showed an effect of the hands-on condition on the learning outcome after one week (see Figure 3) on a $p<0.1$ sig. level (two-tailed).

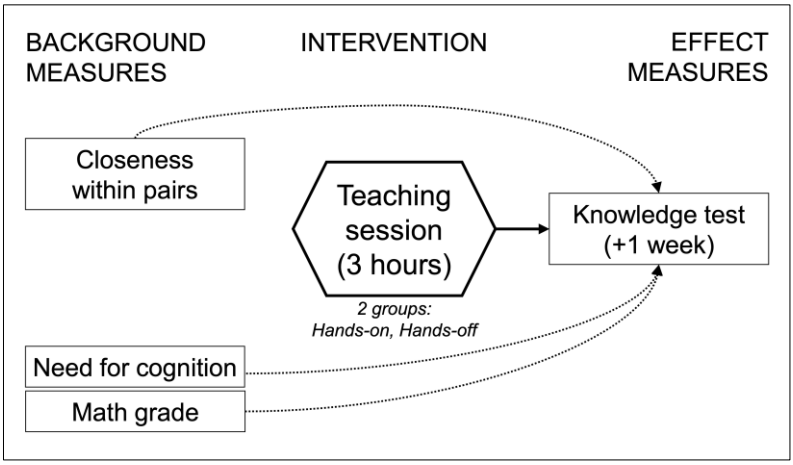

Figure 3. ANCOVA variables (von Hausswolff et al., 2020, p. 7)

Interestingly, the findings from the controlled study showed only a small effect on the long-term learning outcome of learning hands-on, and the ANCOVA model including factors that affected the test one week after the teaching measured a 6.4 % effect of the hands-on factor (von Hausswolff et al., 2020).

As the study included several measurements of the possible impact both on the hands-on condition and on learning outcomes, other associations were possible to observe. The reported decrease in stress during the session was affected by the hands-on conditions. This decrease also correlated to learning outcome differences between the two knowledge tests (immediately vs. after a week), a positive correlation. As the hands-on condition had an effect on stress we took a closer look at factors that correlated with the reported stress. The background factors of being female, scoring low on self-esteem and NFC tests, low motivation, and higher art grade correlated with reported higher stress. Also, the effect factor of reported confidence when answering programming questions had a negative correlation with reported higher stress.

Interestingly, when considering only the decrease in reported stress during the teaching session, the single factor that still remained correlated to the decrease in stress was an increase in reported motivation.

One background factor found beneficial for long-time learning (accounting for 12.7% in our model) was how well the students working together knew each other when the session started. This factor and reported difference in stress only play a role in the long-term learning outcome and not immediately after the learning session.

Conclusions of the study are “that emotions play a non-negligible role when students learn to program. Our results indicate that reduced stress and how well the students knew each other are emotional factors that have an effect on long-term learning. Finally, we conclude that working hands-on has

an important emotional component. The strongest conclusion from our study concerning working hands-on versus working hands-off is that working hands-on clearly affected reported stress reduction. Consequently, the beneficial effect of reduced stress can at least partially explain the effect of hands-on on the learning outcome after one week." (von Hausswolff et al., 2020, p. 10).

Findings in papers III and IV

In pre-study II a naturalistic classroom situation was studied and student interviews were the main data source, complemented with classroom observations, surveys, and course material as background information. This pre-study was conducted in parallel with the controlled study. Here the experiences of the students in the computer lab were in focus. After the initial coding a structure emerged when examining the quotes and the codes several times, thereby dividing the codes into three different groups:

1) codes that describes an emerging relation between the student and the computer/program environment, evolving during the course.

2) codes that describe experiences of programming that involve the whole setting, the social dimensions in the programming experience that includes experiences of belonging and thoughts on continuing to program as a part of the education.

3) "in-between" codes that discuss experiences that emerge from a person's interaction with the computer environment and proceed to establish a relationship to programming that involves a relationship to the whole computer environment. This group makes important connections between the two preceding groups but is not a part of the following two papers.

The findings were sorted and described in two different domains. The first was the domain of the individual student AND the computer environment corresponded with the codes in group 1, described in Paper IV. The second domain is the social dimensions of the learning situation corresponded with the codes in group 2, described in Paper III. The relationship between the two domains is also of interest but is not examined here and thus not part of the results.

The first domain involves a closer look at the individual students emerging relationship with the computer environment and studying that with a focus on the "hands-on" doing. In doing so, it entails an investigation on the philosophical stance to ground the understanding of the situation and descriptions of the relationship. The purpose of an introductory programming course is

commonly described with emphasis on "doing" (actions in the computer environment), the new concepts "come to agreement" and "practical thinking" were introduced (von Hausswolff, 2021). Those findings are described in paper IV.

The second domain is a closer examination of the pair programming condition as well as of values in the discipline of computer science both aesthetic norms ("good-looking" code) and values in connection to different ways of learning. Those findings are described in Paper III. Social interactions within the pairs were crucial for the quality of lab sessions. One finding was that novices had experiences of an emotional roller coaster when encountering programming in the lab. In withstanding this emotional roller coaster, they found comfort in having a partner to work with and not being forced to face the programming assignments alone. Students felt that they learned less during the time their partner had the keyboard, but for social reasons, mentioned above, they still preferred the pair-programming setting. Time at the computer is in opposition to withstanding an emotional roller coaster (with the support of not being alone).

Other results are values connected with discipline of computer science discussed by students, both aesthetic norms ("good-looking" code) and values in connection to learning by yourself ("it is better to figure things out by yourself than together with others"). One way of understanding good-looking code is that less code is better, which is an expression of the computer science norm of "abstraction", i.e. expressing a lot in a compressed form (short) is preferred. The other understanding of "good-looking" code is readability so that the code could be understood by others. One interpretation of using shorter code was that it was a way of expressing that you "controlled" the environment. In this case, the value of short-code was the opposite of readable code, so the concept "good-looking code" includes contradictory values.

The choice not to continue with programming could in part be explained by some students not enjoying working alone and embracing the joy of being social when programming. Time at the computer, withstanding an emotional roller coaster, and self-perception as a newcomer in the discipline, all come together with aesthetic values and norms and impact the decision of continuing with programming (von Hausswolff & Weurlander, 2020, p. 8). These themes are part of the social dimensions of learning to program "hands-on".

In Paper IV two new concepts emerged as important tools when analysing the data from a pragmatic perspective. Pragmatic theory as develop by Dewey and the later Wittgenstein were used in this endeavour and the thinking of Deleuze was also helpful. The developing of new theory, the two new concepts in the domain of programming education were important parts of understanding and re-describing the purpose of an introductory programming course emphasizes the "doing". Taking a closer look at what "hands-on" entailed (actions in the computer environment) was synthesized with the novel concepts "come to agreement" and "practical thinking" (von Hausswolff,

2021). “Students’ learning processes can be understood as ‘come to agreement’ and habitual actions when doing programming as ‘practical thinking’” (von Hausswolff, 2021, p. 5)

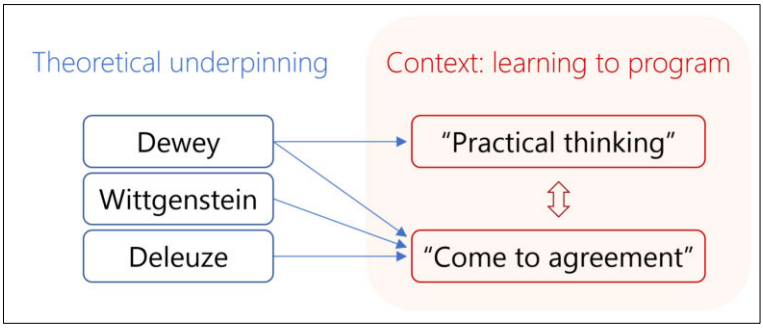

Figure 4. Theoretical underpinning (von Hausswolff, 2021, p. 5)

Practicing “practical thinking” takes courage and time. Programming was perceived both as open (for involving your own creativity in developing context-specific concepts) and at the same time strict (as the conversation with the computer environment could halt with any mistake by the student). “Practical thinking” involved movement in the computer environment testing hunches not expressed in a natural language. Actions in the environment gave responses in small loops that allowed students to follow their own creative thoughts adding on expressions in the created computer program. To “do” something in the computer environment seemed crucial and also that this action was related to one’s own hunches or building on curiosity at the moment (“try, try, try”).

The computer responses were also important (“dig into the error messages”) and this could be seen as a conversation. When the conversation ran smoother and the individual and the computer environment reached “agreement in action” they have ‘come to agreement’. This state could be a description of the desired learning outcome of an introductory programming course. So, in a sense, there are active movements in this environment that involves the student’s creativity (perceived as expressions of the student’s own thinking) and also a novel (for the student) set of boundaries of expressiveness in the formal character of a coding language.

Findings in papers V and VI

The findings highlight the importance of the social learning environment. Social aspects affect emotions as well as learned content and consider the wholeness of the students’ experiences using a pragmatic view of learning.

The main classroom study, conducted last of the empirical studies, were informed from the results of both the controlled study and pre-study II. It entailed further studying of students in a naturalistic classroom setting in an introductory programming course. The setting and the methods are described in previous sections in detail and included interviews with six students on three different occasions during the course, as well as surveys. To deepen the understanding and investigate generalizability of the findings already made, this study included interviews over several occasions and a larger number of participants (as in the survey).

The findings presented in this thesis from the main classroom study are concentrated on the findings from the surveys and the interviews of the six students.

An exploratory factor analysis (n=66) on the part of the survey concerning the students' preferred and stated strategies for learning after about half of the course revealed differences in students' strategies when learning programming. Three distinct learning strategies were found: Individual thinker, Social reader, and Interactive problem-solver (Weurlander & von Hausswolff, 2021). The Social reader and Individual thinker strategies were more common for novices. The third strategy, interactive problem solving, was mostly used by students with more experience in programming. Descriptions of approaches to actively solve problems in close connection to writing code (thinking and writing at the same time), in conjunction with the case that the more experienced uses it more frequently, connected this strategy with practicing "practical thinking" while programming. Our findings suggest that learning programming involves developing practical thinking, a strategy where students interact with the computer, thinking and trying ideas while writing code, a strategy we call Interactive problem-solver.

We also found that female students preferred the Social reader strategy and that this preference was correlated with not planning to continue with programming. "Allowing students to choose how to work, and who to work with, together with fostering a more collaborative, inclusive learning environment may be one way to encourage more women to feel that they belong, and choose to continue with programming." (Weurlander & von Hausswolff, 2021, p. 8).

In paper VI, six stories about learning to program give additional depth to the understanding of "practical thinking" and "come to agreement", and also give directions of how those concepts could be useful when studying the learning process of programming (von Hausswolff, forthcoming). Practical thinking was described at two different stages of the learning process by the students. First at the initial encounter with the programming environment, at the beginning of the course. Second, at the end of the course where they experienced more fluency in moving around the computer environment. Students using practical thinking in the first stage needed personalized help, constantly getting stuck. When students came to agreement with the computer

environment they also came to use the interactive problem-solver learning strategy and hence doing practical thinking but now more freely, seemingly getting unstuck without needing help in person, instead using the internet/google.

Students who were accustomed to and liked working alone could use the individual thinker strategy and this combined with practical thinking/testing/interaction seemed to be helpful. Two of the students, the females in this study, were more accustomed to a social reader strategy. They stated that their path forward was to program in close relation to someone else and that they would prefer to learn more collaboratively. Interestingly, in the end, they seemed to be as independent in their problem-solving compared to two male students that had used the individual thinking strategies. The learning outcome was more similar than the way to reach it.

All students described that the programming course demanded time. They all described struggling alone, frustration, and the experienced of time wasted. Novices experience that the resources in the school were limited or not timely. Other resources in the form of social connections/social abilities or an unemotional personality seemed to come into play. All these circumstances that are different when students get unstuck tend to foster an unequal situation. A student without time on weekends or evenings, or one who does not want to put that much time into programming, tended to conclude that they were less skilled at programming. But at the same time an intrinsic motivation, a feeling of accomplishment, was reported when creating a program of their own, seeing their lines of thought add up. The male students all stated values in finding their own solution, seeing it as both a creative thing and a matter of pride (of their own thinking). Some of them explicitly valued their own way of expressing themselves and it seemed more prevalent in the students applying an individual thinker strategy to learning.

The females that preferred the social reader strategy struggled more to find their way in the course. But all students did struggle, experiencing at least one major hurdle. Emotions during the struggle to learn to program seemed to play a role both in the form of grades and in the decision to continue to study programming in the studied context. The students who did not report feeling any negative emotions when getting stuck received the highest grade. The others felt frustration and anger but two of them still wanted to continue with programming. The remaining two students, both female, did not want to continue with programming despite a positive attitude towards the subject. They concluded that programming was not for them. They also reported low self-esteem, self-blame, self-doubt in connection to programming in contrast to the two male students who received approximately the same grade (D, satisfactory).

Contradictions and anomalies

Students describing their experiences during the course gave several examples of "practical thinking" while programming. Five of the six students described the importance of testing and interacting with the computer, but one seemed not to have had that kind of experience. He maintained a description of thinking before writing throughout the whole learning to program experience and he also did well in the course. This may indicate that even if "practical thinking" is a useful concept to understand the process of learning to program it does not include all students/learners and it is not the only way of approaching this learning.

Another aspect that becomes apparent when studying the student applying more social learning is that the concept "practical thinking" and "come to agreement" focus on the individual and the computer environment, while not really including social dimensions. The two female students´ stories include to a large extent learning together with others. This is not fully captured in the two concepts used in the analysis. This also is shown in that the two female students do not emphasize that their *own* thinking is expressed in the code.

Contributions

This thesis aims to deepen the understanding of how hands-on learning comes into play for novice students learning to program. Why and how is hands-on learning important? Which other factors are important and are there connections between them? As learning is always situated in a context, hands-on learning in an authentic learning situation is also of interest. In this thesis, understanding hands-on learning in programming also entails a philosophical framing novel to the field of computer science education. New concepts have been developed and shown to be useful to interpret students' learning experiences. In addition to this, a rather complex research setting involving different empirical studies and a mix of many methods is shown to be useful and together paint coherent picture of the factors influencing novice students' learning and the learning situation.

Hands-on learning

Previous research has shown that, but not why, hands-on learning is beneficial for learning. For example, Höök and Eckerdal (2015) report positive effects of hands-on learning compared to hands-off, at the end of an introductory programming course. Kaila et al. (2010) found that students experience hands-on learning at the computer to be the most important way to learn to program. The class room study confirmed the latter finding, but the controlled study did not find improved learning out-come in the hands-on group immediately after students' first encounters with programming. One contribution to the field of hands-on learning is that, in this case, the benefits of hands-on learning did not appear immediately. In this study, reduced stress was the main effect of the hands-on condition but this could be linked to increased motivation and also improved learning outcomes over time. Beside background factors such as school grades and enjoyment of thinking (NFC), how well students in a pair knew each other influenced the learning outcome. Hands-on learning was shown to be beneficial for learning over time and here emotional factors could be mediators for connecting positive feelings with programming and therefore increase students' motivation to engage in programming (Paper II). In pragmatic theory, actions and feelings are connected to acquire knowledge, there is no cognition without emotions (Johnson, 2007), and it all stems from the wholeness of an experience.

Emotions and the social dimension in learning programming

In the computer science education literature, emotions are acknowledged as important when learning to program in a formal setting in higher education. However, mostly negative emotions are discussed (cf. Kinnunen and Simon (2012)) compared to results from our controlled study that highlights positive emotions in connection to hands-on learning.

As Kinnunen and Simon (2012) reported, novices experienced frustration when getting stuck. Findings from the naturalistic settings confirmed this, but also positive emotions such as pride and joy were described in an "emotional roller coaster" ride when learning (Paper III). Many students experience negative emotions when they get stuck and frustration if they do not get personalized, "close to the code" help. But not all students in the study had negative emotional reactions when getting stuck, and those students were also more successful in terms of grades received. They achieved the highest final grade. (Paper VI.)

The novices in the study ran into hurdles of such magnitude that they needed time-consuming help from a knowledgeable other at one or several occasions. That help was crucial (Paper III, Paper IV). However, such experiences seem to have negative effects on students' beliefs about themselves as programmers, and may result in decisions not to continue to study computing (Paper III, Paper V, Paper IV). These findings are in line with earlier results from Kinnunen and Simon (2012) about negative emotions having negative effect on students' beliefs about their programming abilities (self-efficacy beliefs). In the findings reported in Paper V and VI those negative effects seem to be especially prominent for women, which is in line with previous research about low representation of women in computer science and cultural factors in the discipline, see for example Rasmussen and Håpnes (1991) and Sax et al. (2018). In the survey study presented in Paper V, having a social reader learning strategy also correlated with identifying as female, experiencing more emotions when programming, and considering not to continue with programming. Those findings were in line with follow-up interviews (Paper V).

All students report that hands-on writing code was the most important aspect when learning programming. Time for the individual interaction with the computer environment is appreciated, as is being two to handle the emotional roller coasters the novices face (Paper III). The social aspect seems more important for some students than for others. A value picked up by some students is that figuring out things for yourself is better than collaborative work (Paper III), which could be a value associated with the selective tradition in computer science.

A pragmatic view on learning

Results from empirical research have suggested that the practice part is the most troublesome for students (Eckerdal et al., 2007), and in this thesis, the "actions" and "doings" in the computer environment are at the forefront. One contribution of this thesis is the use of pragmatic theory to analyze learning to program with a focus on writing code. An important contribution in itself is framing novices learning to program in the two developed concepts "practical thinking" and "come to agreement" (Paper IV) which can be used to understand the programming learning process in a course setting over time. Akin to Wittgenstein's account of learning a first language (Wittgenstein, 1953/2009, 1969/1972), the knowledge of the (computer) world and the meaning of the vocabulary used to communicate are happening simultaneously. As the problem domain becomes large, a need for immediate help from another knowledgeable person to overcome the hurdles seems commonplace for students to be able to continue with their "practical thinking".

The struggle with learning to program

Thinking, creativity, and self-perception as a thinker all are important features of the process of learning to program. Already in 1974, Dijkstra described the problem-solving process in programming as special in the sense that it demanded developing new concepts and putting them to use in coding.

> And this is what a programmer has to do all the time: he has to introduce new concepts -not occurring in the original problem statement - in order to be able to find, to describe, and to understand his own solution to the problem. (Dijkstra, 1974, pp. 610-611)

This feature of programming could be an explanation of why students get stuck: they struggle with concepts of their own creation, materialized in the body of code. It may also explain why it is so important to get individual, specific help to untangle the root of the problem.

Learning to program demands time, sometimes to struggle for hours and not being able to produce code that works, and having to start all over again. However, time spent struggling is beneficial for the learning. The descriptions of computer science students' liminal space by Eckerdal et al. (2007) is echoed in the findings of students' struggles and anxieties when getting stuck (Paper VI). Students describe "coming to agreement" as crossing a threshold (compare: research into threshold concepts in computer science education), sometimes becoming fluent in programming overnight (Paper VI)—benefiting from when fragmented understanding suddenly come together.

Didactical implications

By emphasizing "practical thinking", teachers could communicate that time spent struggling, testing different lines of thought, is an important part of the learning process rather than simply being stuck. The student's own thoughts during a problem-solving task are materialized in the written code and become part of the computer environment. When students get stuck, the conversation with the computer environment grinds to a complete halt. The computer environment (the code structure) is partly a creation of the students' own line of thought, and it becomes hard to get unstuck without qualified help.

The quality of the resources for getting unstuck seems crucial, for both their self-perception as programmers and willingness to continue with programming later on in their education. Too little time set aside for struggling with assignments for the novices, and too little time for the TAs helping the students getting unstuck, tend to create unequal opportunities for different students. When the resources offered by the university are not enough, or not offered at the right time, other resources in the form of social connections or students' unemotional reactions seem to come into play.

When novice students get stuck, they need personalized help close to the code from a knowledgeable other. For that reason, programming teachers are encouraged to meet students as close as possible to their specific thinking, externalized in their code. Instead of standardized feedback, students need individual, concrete feedback on their specific solution. Importantly, new TAs may not be trained in how to personalize their feedback to enhance the connection to each student's own thinking. One possible implication of this is to emphasize training for the TAs, as is discussed by researchers in computer science such as (Riese et al., 2021).

The timing of feedback is also important as it could be impossible for students to do anything without this help, causing frustration and anxiety. In my findings, students often rely on voluntary help from friends, which is both a shaky ground for a learning environment and can also create unequal learning opportunities. A possible way forward is allocating TAs on call online, to answer specific questions related to the student's own code.

As stated in Paper IV, "practical thinking" emphasizes the importance of exploring and connecting your own movements with the new environment, putting the experience of programming at the center (von Hausswolff, 2021). The experiential framing of "practical thinking" can inform course design and be a counterweight to the well-established notion of "computational thinking" (cf. von Hausswolff (2017)).

Future work

The students that used the social learning strategy, to a large extent learning together with others, did not emphasize that their own thinking is expressed in the code. But according to the final grades, they had learned to program just as well as students using an individual strategy. This connection between how the learning is experienced, learning strategies, and emotions could be examined further, possibly also with connections to gender. Research findings in this thesis suggest such connections can be made. The selective tradition and the values of self-expression and aesthetic expressions need closer examination (see Paper III). The interaction between the social dimensions of learning to program and the concepts "practical thinking" and "come to agreement" need to be closer examined especially in the context of deciding to continue with programming.

Finally, one could conclude that the "practical thinking" and "come to agreement", although informed by pragmatic theory, are still somewhat in line with a cognitivistic tradition in computer science education (examining the individual learning the subject). To get a fuller picture, future research needs to connect more explicitly those concepts with social dimensions in both the classroom and in the computer science tradition.

Summary in Swedish

Programmering anses numera vara en viktig färdighet och därför införs programmering i skolundervisningen runt om i världen. I Sverige integreras programmering in i de befintliga ämnena matematik och teknik genom en läroplansrevidering 2018. Detta är en del av en större satsning för att stärka undervisning som syftar till att öka medborgarnas digitala kompetens. Forskning om hur noviser lär sig programmera framstår som allt mer angeläget, såväl nationellt som globalt.

Forskning inom området datavetenskapens didaktik (Computing Education Research, CER) har under flera decennier studerat nybörjarprogrammering med ett rikligt antal publikationer som resultat, ofta med fokus på högre utbildning såsom ingenjörsutbildningar. Inom forskningsområdet råder konsensus kring uppfattningen att det är svårt att lära sig programmera (Robins, 2019). När både ett större antal och ett bredare spektrum av elever/studenter ska lära sig programmera finns ett ökande intresse för kunskap inom programmeringsdidaktik.

Det praktiska handhavandet, att lära sig "hands-on", betonas ofta inom labbintensiva utbildningar som fysik och medicin. I programmeringsundervisning innebär “hands-on”-lärande att studenter sitter vid en dator och löser programmeringsuppgifter under så kallade programmeringslaborationer. Under laborationstillfället, som ofta är på plats i en datorsal, finns labbassistenten eller en lärare att tillgå för praktisk kodnära hjälp. Det är också vanligt att studenterna arbetar två och två i en så kallad parprogrammering under dessa laborationer. Det innebär att en student skriver "hands-on" på tangentbordet medan den andra i paret sitter bredvid, det vill säga "hands-off". Rollerna byts under tiden men båda deltar kontinuerligt i problemlösningen och löser uppgiften tillsammans. Denna typ av samarbete har visat goda läranderesultat.

Såväl studenter som lärare och forskare är överens om att studenternas aktiva lärande "hands-on" är en viktig komponent i att lära sig programmera. Ändå finns ett betydande gap inom forskningen om förståelsen av hur, när och varför "hands-on"-lärandet har positiva effekter. Denna avhandling fokuserar på betydelsen av "hands-on"-lärandet för programmeringsnoviser och använder en "mixed method" metodologi. Den inkluderar en explorativ experimentstudie med gymnasieelever som undersöker vilka faktorer som spelar roll för lärandet av programmering och jämför lärande hands-on med hands-off. Avhandlingen inkluderar också naturalistiska klassrumsstudier med universitetsstudenter. Det teoretiska ramverk som används som grund för kunskapssyn

och lärandeteori är filosofisk pragmatism. Pragmatismen placerar handlingar/görandet i fokus för kunskapsförståelsen, vilket också är centralt i denna avhandling.

Att skriva kod upplevs av studenterna i klassrumsstudierna som det viktigaste när man lär sig programmering. Detta bekräftades dock inte av experimentstudien när kunskapen mättes direkt efter den lektion där eleverna möter programmering för första gången. Eleverna som arbetade vid tangentbordet uppvisade lika bra kunskaper som eleverna som satt bredvid. Elever som arbetade hands-on rapporterade dock en minskning i stress under lektionen och de hade också bättre kunskaper i programmering efter en vecka. Lärande hands-on visade sig vara fördelaktigt för lärande över tid och en hypotes är att emotionella faktorer medierar detta (Artikel II). Den totala upplevelsen under undervisningstillfället kan göra att eleverna som arbetade hands-on associerade positiva känslor med programmering och vid senare tillfällen, när det återaktualiseras, medför detta ökad motivation till att ta sig an programmering eller eventuellt minns man bättre.

Resultaten från klassrumsstudierna i naturalistisk miljö bekräftade att känslor och den sociala miljö där lärandet sker är viktiga komponenter i lärandet och detta stämmer överens med tidigare forskning , (Kinnunen & Simon, 2012). Nybörjare upplevde frustration när de fastnade i en programmeringsuppgift och inte kom vidare utan hjälp, men de kände också glädje när de lyckades. Upplevelserna liknar en "känslomässig berg- och dalbana". Att arbeta tillsammans i par för att hantera de känslomässiga berg- och dalbanor som noviserna möter uppskattas. Ändå betonar studenterna att det är nödvändigt att själv skriva kod för att lära sig programmera, att få tid för den individuella interaktionen med datormiljön (Artikel III).

Studenter som upplevde negativa känslor och inte får individuell kodnära hjälp när de fastnar verkar få en negativ uppfattningen om sig själva som programmerare. Detta kan påverka dem i deras beslut om huruvida de ska fortsätta studera programmering. Dessa negativa uppfattningar om sig själva som programmerare verkar vara vanligare bland kvinnliga studenter. (Artikel III, V och VI)

Två nya begrepp, "practical thinking" och "come to agreement", har tagits fram under arbetet med att kontextualisera den pragmatiska filosofiska utgångspunkten inom domänen programmeringslärande. Användandet av ny teori har visat sig framgångsrikt för att analysera studenternas lärprocesser under en programmeringskurs (Artikel IV). Mot slutet av kursen kände studenterna i klassrumsstudien att "konversationen" med datormiljön flöt på bättre, utan större avbrott eller hinder. Detta beskrivs som att studenterna har "come to agreement" med datormiljön (Artikel VI). Handlingar formas till vanor under tiden man skaffar sig programmeringserfarenhet, dessa handlingar utgör "practical thinking". "Practical thinking" innebär bland annat att testa uttryck som bygger på intuition och outtryckta tankar. Studenten får möjlighet att

tänka/handla och i korta sekvenser av handling och återkoppling interaktivt bygga upp en konversation med datormiljön.

Acknowledgements

First and foremost I would like to thank my supervisors, my main supervisor Anna Eckerdal, and my co-supervisor Maria Weurlander. They have believed in and supported me through this journey and made it possible. It has been a privilege to have such knowledgeable and wise persons by my side.

We were all part of the HOPE research project and conducted much of the research in collaboration. The research project also consisted of three additional members, Michael Thuné, Sara Bengtsson, and Andreas Lidström. I am grateful for the vivid discussions and active research collaboration in the project. I would especially like to thank Michal Thuné for his insightful comments and thorough readings.

Special thanks go to my partner (at home), Fredrik Ohlin, with whom I have discussed all my research ideas, and who is the first person I show my texts. Another contributor that deserves special thanks is Kate Sanders, who took the time to review one of my papers and made it much better as a result.

Throughout my time as a Ph.D. student, I have had the pleasure to be a part of several groups and formations. I would like to acknowledge all of those. I appreciate the opportunity to be a part of Uppsala Computing Education Group (UpCERG), and especially appreciate my colleagues and friends Tina Vrieler and Virginia Grande, not forgetting Anne-Kathrin Peters and Anders Berglund. I also appreciate having been a part of the research school UpRiSE (Uppsala Research School in Subject Education) and meeting Ph.D. students from different disciplines with a common interest in didactics. A highlight has been philosophical discussions about pragmatism with Gita Berg and Mona-Lisa Henriksson. I also would like to mention my friend and Ph.D. student Åsa Olovsson. I cherish her memory.

It has been very valuable for me to be a part of TePlab (Laboratory for teaching practices, formerly SMED). To take part in the SHEAR (Swedish Higher Education Research Network) seminars was also valuable.

I would like to thank all teachers, TAs, and in particular, students, who participated in the studies.

Finally, I would like to thank my family; my partner Fredrik, my daughter Elvira, alongside my father Gottfried, and my friend Åsa, for just being there.

References

Aho, A. V. (2011). Ubiquity symposium: Computation and computational thinking. *Ubiquity*, *2011*.

Alvesson, M., & Sköldberg, K. (2017). *Reflexive methodology: New vistas for qualitative research*. Sage.

Bennedsen, J., & Caspersen, M. E. (2007). Failure rates in introductory programming. *ACM Sigcse Bulletin*, *39*(2), 32-36.

Biesta, G. (2009). Good education in an age of measurement: On the need to reconnect with the question of purpose in education. *Educational Assessment, Evaluation and Accountability (formerly: Journal of Personnel Evaluation in Education)*, *21*(1), 33-46.

Biesta, G. (2012). Mixed methods. In J. Arthur, Waring, M., Coe, R. and Hedges, L. (Ed.), *Research methods and methodologies in education* (pp. 147-152). Sage publications.

Biesta, G., & Burbules, N. (2003). *Pragmatism and educational research*. Rowman & Littlefield.

Bivall, P., Ainsworth, S., & Tibell, L. A. (2011). Do haptic representations help complex molecular learning? *Science Education*, *95*(4), 700-719.

Björkman, C. (2005). *Crossing boundaries, focusing foundations, trying translations: Feminist technoscience strategies in computer science*. Blekinge Institute of Technology.

Bosse, Y., & Gerosa, M. A. (2017). Why is programming so difficult to learn? Patterns of Difficulties Related to Programming Learning Mid-Stage. *ACM SIGSOFT Software Engineering Notes*, *41*(6), 1-6.

Boustedt, J., Eckerdal, A., McCartney, R., Moström, J. E., Ratcliffe, M., Sanders, K., & Zander, C. (2007). Threshold concepts in computer science: do they exist and are they useful? *ACM Sigcse Bulletin*, *39*(1), 504-508.

Bowman, N. A., Jarratt, L., Culver, K., & Segre, A. M. (2019). How Prior Programming Experience Affects Students' Pair Programming Experiences and Outcomes. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 170-175). https://doi.org/10.1145/3304221.3319781

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative research in psychology*, *3*(2), 77-101.

Cacioppo, J. T., Petty, R. E., & Feng Kao, C. (1984). The efficient assessment of need for cognition. *Journal of personality assessment*, *48*(3), 306-307.

Cao, L., & Xu, P. (2005). Activity patterns of pair programming. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences* (pp. 88a-88a). IEEE.

CODEX. (2020, 2020-11-13). *Rules and guidelines for research*. Swedish Research Council. Retrieved 2020-11-25 from http://www.codex.vr.se/en/index.shtml

Cohen, L., Manion, L., & Morrison, K. (2011). Planning educational research. In *Research methods in education. New York: Routledge Editors*.

Corder, G. W., & Foreman, D. I. (2014). *Nonparametric statistics: A step-by-step approach*. John Wiley & Sons.
Creswell, J., Plano Clark, V., Gutmann, M., & Hanson, W. (2003). Advanced mixed methods research design. . In A. T. Tashakkori, C. T (Ed.), *Handbook of mixed methods in social and behavioral research.* (pp. 209-240). Sage.
Cronbach, L. J. (1951). Coefficient alpha and the internal structure of tests. *psychometrika*, *16*(3), 297-334.
Dewey, J. (1925/1995). *Experience and Nature*. Dover.
Dewey, J. (1930/2013). *The quest for certainty*. Isha Books.
Dijkstra, E. W. (1974). Programming as a discipline of mathematical nature. *The American Mathematical Monthly*, *81*(6), 608-612.
Du Boulay, B. (1986). Some difficulties of learning to program. *Journal of Educational Computing Research*, *2*(1), 57-73.
Eckerdal, A. (2015a). *Hands-on in computer programming education: educational effects and brain processes.* [Grant].
Eckerdal, A. (2015b). Relating theory and practice in laboratory work: A variation theoretical study. *Studies in Higher Education*, *40*(5), 867-880.
Eckerdal, A., McCartney, R., Moström, J. E., Ratcliffe, M., Sanders, K., & Zander, C. (2006). Putting threshold concepts into context in computer science education. *ACM Sigcse Bulletin*, *38*(3), 103-107.
Eckerdal, A., McCartney, R., Moström, J. E., Sanders, K., Thomas, L., & Zander, C. (2007). From Limen to Lumen: computing students in liminal spaces. In *Proceedings of the third international workshop on Computing education research* (pp. 123-132).
Elo, S., & Kyngäs, H. (2008). The qualitative content analysis process. *Journal of advanced nursing*, *62*(1), 107-115.
Goldfarb, W. (1996). *The philosophy of mathematics in early positivism*. University of Minnesota Press, Minneapolis.
Grant, H., & Dweck, C. S. (2003). Clarifying achievement goals and their impact. *Journal of personality and social psychology*, *85*(3), 541.
Graziano, A. M., & Raulin, M. L. (1993). *Research methods: A process of inquiry*. HarperCollins College Publishers.
Gross, P., & Powers, K. (2005). Evaluating assessments of novice programming environments. In *Proceedings of the first international workshop on Computing education research* (pp. 99-110).
Guba, E. G., & Lincoln, Y. S. (1982). Epistemological and methodological bases of naturalistic inquiry. *ECTJ*, *30*(4), 233-252.
Guzdial, M. (2010). Why is it so hard to learn to program. *Making Software: What Really Works, and Why We Believe It. O'Reilly Media*, 111-124.
Hair Jr, J. F., Black, W. C., Babin, B. J., & Anderson, R. E. (2014). *Multivariate Data Analysis* (Seventh Edition ed.). Pearson Edition Limited.
Hamlyn, D. W. (1987). *A history of western philosophy*. Viking.
Hanks, B., Fitzgerald, S., McCauley, R., Murphy, L., & Zander, C. (2011). Pair programming in education: A literature review. *Computer Science Education*, *21*(2), 135-173.
Hodkinson, P., Biesta, G., & James, D. (2008). Understanding learning culturally: Overcoming the dualism between social and individual views of learning. *Vocations and learning*, *1*(1), 27-47.

Höök, L. J., & Eckerdal, A. (2015). *On the bimodality in an introductory programming course: An analysis of student performance factors* Proceedings of the 2015 International Conference on Learning and Teaching in Computing and Engineering Taipei, Taiwan. https://doi.org/10.1109/LaTiCE.2015.25

Jaakkola, T., Nurmi, S., & Veermans, K. (2011). A comparison of students' conceptual understanding of electric circuits in simulation only and simulation-laboratory contexts. *Journal of research in science teaching*, *48*(1), 71-93.

James, W. (1912/2003). Essays in radical empiricism In. New York, NY: Dover Publications.

Johnson, M. (2007). *The Meaning of the Body: Aesthetics of Human Understanding*. the University of Chicago Press.

Johnson, R. B., & Onwuegbuzie, A. J. (2004). Mixed methods research: A research paradigm whose time has come. *Educational researcher*, *33*(7), 14-26.

Kaila, E., Rajala, T., Laakso, M.-J., & Salakoski, T. (2010). Effects of course-long use of a program visualization tool. Proceedings of the Twelfth Australasian Conference on Computing Education-Volume 103,

Kinnunen, P., & Simon, B. (2012). My program is ok–am I? Computing freshmen's experiences of doing programming assignments. *Computer Science Education*, *22*(1), 1-28.

Kuhn, T. S. (1962). Historical Structure of Scientific Discovery: To the historian discovery is seldom a unit event attributable to some particular man, time, and place. *Science*, *136*(3518), 760-764.

Lahtinen, E., Ala-Mutka, K., & Järvinen, H.-M. (2005). A study of the difficulties of novice programmers. *ACM Sigcse Bulletin*, *37*(3), 14-18.

Lakatos, I. (1970). Falsification and the methodology of scientific research programmes. In I. L. a. A. Musgrave (Ed.), *Criticism and the Growth of Knowledge*. Cambridge University Press

Lakatos, I. (1980). *Mathematics, Science and Epistemology: Volume 2, Philosophical Papers* (Vol. 2). Cambridge University Press.

Lidström, A. (2021). *Towards a neuroscience of computer programming and education.* University of East Anglia.

Luxton-Reilly, A., Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., Paterson, J., Scott, M. J., Sheard, J., & Szabo, C. (2018). Introductory programming: a systematic literature review. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education* (pp. 55-106).

Mason, L. (2007). Introduction: Bridging the cognitive and sociocultural approaches in research on conceptual change: Is it feasible? *Educational psychologist*, *42*(1), 1-7.

Mason, R. L., Gunst, R. F., & Hess, J. L. (2003). *Statistical design and analysis of experiments: with applications to engineering and science* (Vol. 474). John Wiley & Sons.

McAuley, E., Duncan, T., & Tammen, V. V. (1989). Psychometric properties of the Intrinsic Motivation Inventory in a competitive sport setting: A confirmatory factor analysis. *Research quarterly for exercise and sport*, *60*(1), 48-58.

Meyer, J. H., & Land, R. (2005). Threshold concepts and troublesome knowledge (2): Epistemological considerations and a conceptual framework for teaching and learning. *Higher education*, *49*(3), 373-388.

Minogue, J., & Jones, M. G. (2006). Haptics in education: Exploring an untapped sensory modality. *Review of Educational Research*, *76*(3), 317-348.

Nationalencyklopedin. (n. d.). Bayes sats. . Retrieved 2021-04-05, from http://www-ne-se.ezproxy.its.uu.se/uppslagsverk/encyklopedi/lång/bayes-sats

NRC. (2010). *Report of a workshop on the scope and nature of computational thinking* (0309153727).

O'Donnell, C., Buckley, J., Mahdi, A., Nelson, J., & English, M. (2015). Evaluating pair-programming for non-computer science major students. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (pp. 569-574).

Ottemo, A. (2015). *Kön, kropp, begär och teknik: passion och instrumentalitet på två tekniska högskoleprogram*. Göteborgs universitet.

Peirce, C. S. (1878/1989). How to make our ideas clear. In H. S. Thayer (Ed.), *Pragmatism: The Classic Writings* (pp. 79-100). Hackett Publishing.

Polkinghorne, D. E. (1983). *Methodology for the human sciences: Systems of inquiry*. State University of New York Press.

Preece, D., Williams, S. B., Lam, R., & Weller, R. (2013). "Let's get physical": advantages of a physical model over 3D computer models and textbooks in learning imaging anatomy. *Anatomical sciences education*, *6*(4), 216-224.

Rasmussen, B., & Håpnes, T. (1991). Excluding women from the technologies of the future?: A case study of the culture of computer science. *Futures*, *23*(10), 1107-1119.

Riese, E., Lorås, M., Ukrop, M., & Effenberger, T. (2021). Challenges Faced by Teaching Assistants in Computer Science Education Across Europe. Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1,

Robins, A. (2010). Learning edge momentum: a new account of outcomes in CS1. *Computer Science Education*, *20*(1), 37-71. https://doi.org/10.1080/08993401003612167

Robins, A. (2019). Novice programmers and introductory programming. In S. A. Fincher & A. V. Robins (Eds.), *The Cambridge Handbook of Computing Education Research* (pp. 327-376). Cambridge University Press.

Rogerson, C., & Scott, E. (2010). The fear factor: How it affects students learning to program in a tertiary environment. *Journal of Information Technology Education: Research*, *9*(1), 147-171.

Rorty, R. (1991). *Objectivity, relativism and truth.* Cambridge University Press.

Rorty, R. (1999). *Hope in Place of Knowledge: The Pragmatics Tradition in Philosophy*. Institute of European and American Studies, Academic Sinica.

Rosenberg, M. (1965). Rosenberg self-esteem scale (RSE). *Acceptance and commitment therapy. Measures package*, *61*(52), 18.

Rountree, J., Robins, A., & Rountree, N. (2013). Elaborating on threshold concepts. *Computer Science Education*, *23*(3), 265-289.

Salleh, N., Mendes, E., & Grundy, J. (2010). Empirical studies of pair programming for CS/SE teaching in higher education: A systematic literature review. *IEEE Transactions on Software Engineering*, *37*(4), 509-525.

Sanders, K., Boustedt, J., Eckerdal, A., McCartney, R., & Zander, C. (2017). Folk pedagogy: Nobody doesn't like active learning. In *Proceedings of the 2017 ACM Conference on International Computing Education Research* (pp. 145-154).

Sanders, K., & McCartney, R. (2016). Threshold concepts in computing: past, present, and future. In *Proceedings of the 16th Koli Calling international conference on computing education research* (pp. 91-100).

Sandewall, E. (2021). Nationalencyklopedin, datavetenskap. http://www-ne-se.ezproxy.its.uu.se/uppslagsverk/encyklopedi/lång/datavetenskap

Sax, L. J., Blaney, J. M., Lehman, K. J., Rodriguez, S. L., George, K. L., & Zavala, C. (2018). Sense of belonging in computing: The role of introductory courses for women and underrepresented minority students. *Social Sciences*, *7*(8), 122.

Schulte, C., & Knobelsdorf, M. (2007). Attitudes towards computer science-computing experiences as a starting point and barrier to computer science. In *Proceedings of the third international workshop on Computing education research* (pp. 27-38).

Sfard, A. (1998). On two metaphors for learning and the dangers of choosing just one. *Educational researcher*, *27*(2), 4-13.

Sheard, J., Carbone, A., Chinn, D., Laakso, M.-J., Clear, T., de Raadt, M., D'Souza, D., Harland, J., Lister, R., & Philpott, A. (2011). Exploring programming assessment instruments: a classification scheme for examination questions. In *Proceedings of the seventh international workshop on Computing education research* (pp. 33-38).

Shneiderman, B. (1977). Teaching programming: A spiral approach to syntax and semantics. *Computers & Education*, *1*(4), 193-197.

Sorva, J. (2010). Reflections on threshold concepts in computer programming and beyond. In *Proceedings of the 10th Koli calling international conference on computing education research* (pp. 21-30).

Stein, L. A. (1999). Challenging the computational metaphor: Implications for how we think. *Cybernetics & Systems*, *30*(6), 473-507.

Stemler, S. E. (2004). A comparison of consensus, consistency, and measurement approaches to estimating interrater reliability. *Practical Assessment, Research, and Evaluation*, *9*(1), 4.

Tashakkori, A., Johnson, R. B., & Teddlie, C. (2021). *Foundations of mixed methods research: Integrating quantitative and qualitative approaches in the social and behavioral sciences* (2ed ed.). Sage publications.

Tashakkori, A., & Teddlie, C. (1998). *Mixed methodology: Combining qualitative and quantitative approaches* (Vol. Applied Social Research Methods Series). Sage.

Tedre, M. (2014). *The science of computing: shaping a discipline*. CRC Press.

Tedre, M., & Denning, P. J. (2016). The long quest for computational thinking. Proceedings of the 16th Koli Calling international conference on computing education research,

Thomas, L., Boustedt, J., Eckerdal, A., McCartney, R., Moström, J. E., Sanders, K., & Zander, C. (2010). Threshold concepts in computer science: An ongoing empirical investigation. In J. H. F. Meyer, R. Land, & C. Baillie (Eds.), *Threshold concepts and transformational learning* (pp. 241-257). Brill Sense. https://doi.org/10.1163/9789460912078_016

Van Poeck, K., & Östman, L. (2019). Sustainable development teaching in view of qualification, socialisation and person-formation. In K. Van Poeck, L. Östman, & J. Öhman (Eds.), *Sustainable Development Teaching: Ethical and Political Challenges* (pp. 59-69). Routledge. https://doi.org/10.4324/9781351124348

Weurlander, M., Scheja, M., Hult, H., & Wernerson, A. (2016). The struggle to understand: exploring medical students' experiences of learning and understanding during a basic science course. *Studies in Higher Education*, *41*(3), 462-477.

Weurlander, M., & von Hausswolff, K. (2021). Engineering students' strategies to learn programming correlate with motivation and gender. *2021 IEEE Frontiers in Education Conference (FIE)*, Lincoln, Nebraska.

Wickman, P.-O. (2012). Using pragmatism to develop didactics in Sweden. *Zeitschrift für Erziehungswissenschaft*, *15*(3), 483-501.

Wickman, P.-O. (2013). *Aesthetic experience in science education: Learning and meaning-making as situated talk and action*. Routledge.
Williams, R. (1973). Base and superstructure in Marxist cultural theory. *New left review*(82), 3.
Winslow, L. E. (1996). Programming pedagogy—a psychological overview. *ACM Sigcse Bulletin*, *28*(3), 17-22.
Wittgenstein, L. (1953/2009). *Philosophical investigations*. John Wiley & Sons.
Wittgenstein, L. (1969/1972). *On certainty* (Vol. 174). Blackwell Oxford.
von Hausswolff, K. (2017). Practical thinking in programming education. In *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*,
von Hausswolff, K. (2021). Practical thinking while learning to program – novices' experiences and hands-on encounters. *Computer Science Education*, 1-25. https://doi.org/10.1080/08993408.2021.1953295
von Hausswolff, K. (forthcoming). Practical thinking and learning strategies: novices experiencing learning programming.
von Hausswolff, K., & Eckerdal, A. (2018). Measuring Programming Knowledge in a Research Context. *2018 IEEE Frontiers in Education Conference (FIE)*, San Jose, California
von Hausswolff, K., Eckerdal, A., & Thuné, M. (2020). Learning to program hands-on: a controlled study. *Koli Calling'20: Proceedings of the 20th Koli Calling International Conference on Computing Education Research*
von Hausswolff, K., & Weurlander, M. (2020). Social dimensions in the lab session when novices learn to program. *2020 IEEE Frontiers in Education Conference (FIE)*, Uppsala, Sweden.
Öhman, J., & Östman, L. (2019). Different teaching traditions in environmental and sustainability education. In *Sustainable Development Teaching* (pp. 70-82). Routledge.
Östman, L. (2010). Education for sustainable development and normativity: A transactional analysis of moral meaning-making and companion meanings in classroom communication. *Environmental Education Research*, *16*(1), 75-93.
Östman, L., Van Poeck, K., & Öhman, J. (2019). A transactional theory on sustainability learning. In K. Van Poeck, L. Östman, & J. Öhman (Eds.), *Sustainable Development Teaching: Ethical and Political Challenges* (pp. 127-139). Routledge. https://doi.org/10.4324/9781351124348

Acta Universitatis Upsaliensis

Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology 2104

Editor: The Dean of the Faculty of Science and Technology

A doctoral dissertation from the Faculty of Science and Technology, Uppsala University, is usually a summary of a number of papers. A few copies of the complete dissertation are kept at major Swedish research libraries, while the summary alone is distributed internationally through the series Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology. (Prior to January, 2005, the series was published under the title "Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology".)

Distribution: publications.uu.se
urn:nbn:se:uu:diva-461455

ACTA
UNIVERSITATIS
UPSALIENSIS
UPPSALA
2022