DSV Report Series No. 20-014

Stockholm University

Integrating Computational Thinking into Swedish Compulsory Education with Block-Based Programming

A case study from the perspective of teachers

Lechen Zhang

Doctoral Thesis in Information Society at Stockholm University, Sweden 2020

Integrating Computational Thinking into Swedish Compulsory Education with Block-Based Programming

A case study from the perspective of teachers

Lechen Zhang

Academic dissertation for the Degree of Doctor of Philosophy in Information Society at Stockholm University to be publicly defended on Wednesday 9 December 2020 at 09.00 in Aula NOD, NOD-huset, Borgarfjordsgatan 12.

Abstract

Computational thinking (CT) is increasingly widespread in compulsory education worldwide, and programming is considered to be an appropriate and standard way of delivering CT. In particular, block-based programming languages (BBPLs) such as Scratch have successfully attracted an enormous amount of young users and have begun to predominate in classrooms over the past decade. Much of this success can be attributed to the features of BBPLs, such as their media-rich environment and straightforward syntax. As part of the recent computing education reforms taking place globally, Sweden has revised its compulsory curriculum to include CT and programming. This new Swedish curriculum mandates that CT and programming should be incorporated primarily in the areas of mathematics and technology, effective from July 2018. However, dilemmas and challenges arise from this educational reform. Compulsory school teachers may not be equipped with adequate CT competence to implement the new curriculum, and CT and programming have not historically formed part of teacher training, and have not been the focus of professional development. This raises questions over the extent to which teachers with limited competence in a subject can integrate it into lessons. This dissertation is therefore dedicated to investigating the integration process of CT and programming into Swedish compulsory education from the perspective of teachers. More specifically, it scrutinizes two essential aspects of integration: the CT skills that are taught and assessed by the teachers using a BBPL, and the teachers' CT competence.

A case study was conducted to probe the integration process. This was carried out as part of a national research and development project aiming to enhance compulsory school teachers' CT competence and pedagogical skills in programming. Multiple sources of data were collected, triangulated, and analyzed both qualitatively and quantitatively. The key findings can be summarized as follows. Firstly, the CT framework was examined and extended to describe the CT skills that can be acquired through BBPLs. Secondly, the CT skills included by teachers in their instruction and assessment were identified, as well as the learning and teaching difficulties that are encountered. A learning progression for CT skills for young learners was also proposed. Thirdly, the CT skills listed in the examined framework were compared with those taught and assessed by the teachers. The differences yielded by this comparison implied that the teachers had limited CT and programming proficiency. Finally, a CT test was constructed and the teachers' CT competence was systematically surveyed. In addition, the relationship between a teacher's background and CT competence was investigated.

The significance of this study is manifold and unique. The primary contribution is an examination and evaluation of the most important aspects of the integration of CT and programming into Swedish compulsory education. To the best of the author's knowledge, no previous research has provided insights into this matter since the new curriculum went into effect. Hence, this study is the first to update the stakeholders (namely teachers, students, parents, school management, and policy makers) with valuable and specific details of this integration. Furthermore, this work contributes to the establishment of a body of knowledge regarding in-service teachers in CT education research, since studies of this topic have been scarce. By focusing on what teachers choose to teach and their competence in CT, their perspective is brought to the fore. Due to the nature of case studies, the findings in this study can be directly applied in practice, such as when planning instructions and designing assessments, pinpointing areas for improvement, and directing future professional development.

Keywords: *Computational thinking, Compulsory education, Programming.*

Stockholm 2020
http://urn.kb.se/resolve?urn=urn:nbn:se:su:diva-185922

ISBN 978-91-7911-330-8
ISBN 978-91-7911-331-5
ISSN 1101-8526

Stockholm University

Department of Computer and Systems Sciences

Stockholm University, 164 07 Kista

INTEGRATING COMPUTATIONAL THINKING INTO SWEDISH COMPULSORY EDUCATION WITH BLOCK-BASED PROGRAMMING

Lechen Zhang

Stockholm University

Integrating Computational Thinking into Swedish Compulsory Education with Block-Based Programming

A case study from the perspective of teachers

Lechen Zhang

ISBN print 978-91-7911-330-8
ISBN PDF 978-91-7911-331-5
ISSN 1101-8526

Printed in Sweden by Universitetsservice US-AB, Stockholm 2020

Included Articles

Zhang, L.C., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers & Education*, *141*, 103607.

Nouri, J., Zhang, L.C., Mannila, L., & Norén, E. (2019). Development of computational thinking, digital competence, and 21st-century skills when learning to program in K-9. *Education Inquiry*, *11(1)*, 1–17.

Zhang, L.C., Nouri, J., & Rolandsson, L. (2020). Progression of computational thinking skills in Swedish compulsory schools with block-based programming. In *Proceedings of the 22nd Australasian Computing Education Conference* (pp. 66–75). February 3–7, 2020. Melbourne, Australia. Australian Computer Society, Inc. ACM.

Zhang, L.C., & Nouri, J. (2019). Assessing K-9 teachers' computational thinking skills through a computational thinking test. In J. Keengwe & P. Wachira (Eds), *Handbook of research on integrating computer science and computational thinking in K-12 education*. Hershey, Pennsylvania: IGI Global.

Contents

Sammanfattning

Datalogiskt tänkande (DT) tar en allt större plats i grundskoleutbildningen över hela världen. Programmering anses vara ett lämpligt och standardiserat sätt att lära ut DT. Blockbaserade programmeringsspråk, som Scratch, har framgångsrikt attraherat många unga användare i klassrummen under det senaste decenniet. Mycket av framgången kan härledas till de fördelar som blockbaserade programmeringsspråk har, till exempel att de är mediarika och har enkla syntax. Under de senaste globala reformerna av DT har Sverige reviderat läroplanen för grundskolan så att den inkluderar DT och programmering. Den nya svenska läroplanen kräver att DT och programmering huvudsakligen ska ingå i ämnet matematik och teknik från och med juli 2018. Det finns dock stora utmaningar med denna utbildningsreform. Historiskt sett har DT och programmering inte varit en del av lärarutbildningen och är inte heller fokus för lärarnas professionella utveckling och vidareutbildning. Syftet med denna avhandling var att undersöka om lärare kan integrera DT och programmering i ämnet matematik och teknik trots begränsad kompetens på området. Vidare undersöker denna avhandling integrationsprocessen för DT och programmering i svensk utbildning i ett lärarperspektiv. Specifikt så granskar den två viktiga aspekter av integrationen: DT färdigheterna som lärts ut av lärarna i blockbaserade programmeringsspråk och lärarnas kompetens inom DT och programmering.

För att undersöka integrationsprocessen genomfördes en fallstudie inom ramen av ett nationellt forsknings- och utvecklingsprojekt som syftar till att förbättra grundskolelärarnas kompetens i DT och pedagogiska färdigheter i programmering. Flera datakällor samlades in, triangulerades och analyserades kvalitativt och kvantitativt. Undersökningen testade och utvidgade ett ramverk för DT som beskriver de färdigheter som kan förvärvas genom blockbaserade programmeringsspråk. Studien visar också de färdigheter i DT som lärare har inkluderat i sin undervisning och bedömning, liksom de inlärnings- och undervisningssvårigheter som har uppstått. Studien visade vidare en inlärningsprogression av färdigheterna för DT hos unga elever. Studien jämförde de färdigheter som anges i ramverket med de färdigheter som lärts ut och bedömts. Skillnaderna i jämförelsen visar att lärarna har begränsade färdigheter i DT och programmering. Slutligen konstruerades ett test som systematiskt kartlägger lärarnas kompetens i DT. Studien undersökte även förhållandet mellan lärarnas bakgrund och deras kompetens i DT.

Sammanfattningsvis undersöker denna studie viktiga aspekter av integrationen av datalogiskt tänkande och programmering i svensk grundskoleutbildning. Så vitt författaren vet saknas studier på detta område sedan den nya läroplanen trädde i kraft. Denna studie bidrar därför med värdefulla detaljer för integrationen. Dessutom bidrar detta arbete med kunskap om lärares kompetens inom DT, eftersom relevanta studier på området är sällsynta. Genom att fokusera på vad lärarna valde att undervisa i och deras kompetens inom DT, så lyfter denna studie lärarnas perspektiv. På grund av fallstudiens design kan dessutom resultaten tillämpas direkt i praktiken, såsom planeringsinstruktioner och utformning av bedömningar, identifiering av områden för förbättring och styrning av den framtida professionella utvecklingen.

Acknowledgements

This dissertation is a synthesis of guidance and cooperation, most of which was generously provided by my research supervisor, Jalal Nouri, PhD. Associate Professor. Thank you, Jalal, for this doctoral candidacy, for your expertise and efforts in shaping me as a researcher, and for your direction with vision, motivation, ambition, kindness and patience.

I would also like to express my gratitude to my co-authors, Eva Norén, Lennart Rolandsson, Linda Mannila, and to the IFOUS project for their contribution to my research and learning.

Completing my doctorate could have been a lonely journey if it were not for my friends. I sincerely appreciate my friends for their support and caring during my study. My dearest Linus Liljegren, you make me live for the future and long for the past. Henrik Nordström and Martin Hellström, thank you for your witty humor and for experiencing the world with me. Le Anh Ma and Lotte van Doselaar, I wish you the best for your doctoral studies. Hannes Gerhardsson, Nicholas Johansson, and Sebastian Bramefjord, I appreciate the positive influence you have had on my life. James Ford and Li Yuejin, no matter how long it has been since we last talked, we can always pick up right where we left off.

I am extremely grateful to my beloved parents and my late grandfather. I am eternally indebted to you for your sacrifices and your dedication for raising me and educating me. Special thanks goes to my uncle Long for his life wisdom and advice.

Last but not least, I would like to thank my students at NR for their inspiration, my colleague Helén Thorstenson for proofreading the abstract in Swedish, as well as the principal Fredrik Skog for his support.

致,

我的

母亲 和 父亲。

2020-10-10

List of Additional Publications

Ahmed, G., Nouri, J., Zhang, L.C., & Norén, E. (2020). Didactic methods of integrating Programming in mathematics in primary school: Findings from a national project in Sweden. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (pp. 261–267), March 11–14, 2020. Portland, USA. ACM.

Ahmed, G., Nouri, J., Norén, E., & Zhang, L.C. (2019). Students' perceptions of programming in primary school. In *Proceedings of the 14th Workshop in Primary and Secondary Computing Education* (pp. 15–19). October 22–25, 2019. Glasgow, UK. ACM.

Chibas, Å., Nouri, J., Norén, E., Zhang, L.C., & Sjöberg, C. (2018). Didactical strategies and challenges when teaching programming in pre-school. In *Proceedings of the 10th International Conference on Education and New Learning Technologies* (pp. 3345–3350), July 2–4, 2018. Palma, Spain.

Sjöberg C., Risberg T., Nouri J., Norén, E., & Zhang L.C. (2020) Programming as a tool for across-subjects learning in primary school. *RUDN Journal of Informatization in Education*, 17 (3). pp.179-189.

Sjöberg, C., Risberg, T., Nouri, J., Norén, E., & Zhang, L.C. (2019). A lesson study on programming as an instrument to learn mathematics and social science in primary school. In *Proceedings of the 13th Annual International Technology, Education and Development Conference* (pp. 2230–2235), March 11–13, 2019. Valencia, Spain.

Sjöberg, C., Nouri, J., Sjöberg, R., Norén, E., & Zhang L.C. (2018). Teaching and learning mathematics in primary school through Scratch. In *Proceedings of the 10th International Conference on Education and New Learning Technologies* (pp. 5625–5632), July 2–4, 2018. Palma, Spain.

List of Figures

List of Tables

Introduction

Computational thinking (CT) and programming now have an increasing presence in compulsory education worldwide. In 2012, the Australian Digital Technologies Syllabus made it mandatory for all children from Foundation to Year 8 to learn CT (ACARA, 2012). Since September 2014, England has been implementing a new computing curriculum into education for ages 5 to16, and with a strong focus on CT (Department for Education, 2013). The new Finnish curriculum introduced programming as a cross-curricular activity, mainly in Mathematics and Crafts, for Years 1–9 in 2016. This new curriculum included learning objectives that relate to aspects of CT (Heintz et al., 2016). Sweden has also integrated programming and CT into its compulsory schooling, primarily in Mathematics and Technology, since July 2018 (Heintz & Manilla, 2018). Similar changes can be observed in other parts of the globe, including Estonia, New Zealand, Norway, Poland and South Korea (Heintz et al., 2016).

What is CT? The term was coined in 1980 in Seymour Papert's book "Mindstorms: Children, Computers, and Powerful Ideas," with his creation of the programming language LOGO. Papert referred to CT as a mental skill children develop from programming (Papert, 1980). Revitalized by Wing (2006) a decade ago, CT has been gaining considerable attention from researchers, educators, and policymakers (Grover & Pea, 2013). Wing defined CT as "solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science" (ibid, p. 33). Later, she was inspired by computer scientist Alfred Aho, and together with Cuny and Synder provided a new definition: "[C]omputational thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" (Wing, 2011, p.1). Since Wing brought CT back to public attention, there have been many attempts to unify the definition of CT in the research community (Barr & Stephenson, 2011; Grover & Pea, 2013). Despite the dispute over what CT entails, it is common for many definitions and frameworks for CT (e.g. CSTA & ISTE, 2011) to propose and specify a set of computing-related concepts and skills, such as sequences, control flow, abstraction, modularization and debugging.

Why teach CT? Wing highlighted the necessity of CT in K–12 education by stating, “to reading, writing, and arithmetic, we should add computational thinking to every child’s analytical ability” (Wing, 2006, p. 33). Wing argued that “everyone, not just those who major in computer science, can benefit from thinking like a computer scientist”, and that CT could be extended to other disciplines. Although Wing’s vision of “CT for all” has received criticism (e.g., Denning, 2017), there does seem to be a consensus that students at all levels can benefit from exposure to and increased practice with computing-related concepts and skills (Barr, 2014, p. 2–4). Many scholars deem CT an essential 21st century skill (e.g., Nouri et al., 2020; Angeli & Jaipal-Jamani, 2018; Dasgupta et al., 2016; Yadav, Hong et al., 2016; Lye & Koh, 2014). The increasing use of computational methods and the pervasiveness of computational artifacts in society and our daily lives require that we educate students to apply computational approaches to solve problems (Jacob et al., 2018; Barr, 2014; Bundy, 2007). It is therefore important for them to begin working with CT and relevant tools early and often as part of their individual development (Barr & Stephenson, 2011, Lu & Fletcher, 2009). Simply put, the growing demand for a workforce with computational skills requires CT education (Wilson & Moffat, 2010). Another reason for teaching CT in compulsory education is to democratize the digital society and empower future citizens (Iversen et al., 2018). With the rise of artificial intelligence and automation, CT can prepare children to understand and make informed choices about technology and to participate in technological development (Wolz et al., 2011). In this way, CT skills can foster civic engagement by students. Moreover, CT has the potential to cultivate other essential skills. Wing (2011, p. 2) maintained that CT can engage other kinds of thought processes, such as compositional reasoning, pattern matching, procedural thinking, and recursive thinking. Moreover, Mishra and Yadav (2013) posited that CT could move students from being consumers of technology to creating new forms of expression through the building of computational artifacts and cultivating of creativity. Subsequent to Wing’s call for “CT for all”, many efforts were undertaken to incorporate CT into the landscape of K-12 education (Barr, 2014, p. 2–4).

How can CT skills be acquired? There are various ways to obtain this 21st century skill, but CT is often operationalized through computer programming (Bocconi et al., 2016; Barr & Stephenson, 2011). Of the different types of programming languages, block-based programming languages (BBPLs) such as Alice, AppInventor, Blockly, and Scratch are believed to be most suitable for delivering CT to the young learners. This type of programming language uses visual, drag-and-drop blocks to construct programs. The exemption from knowing the language-specific syntax, as required in text-based programming languages (TBPLs), reduces the cognitive load on the students (Kelleher & Pausch, 2005). Furthermore, these approaches are media-rich and can stimulate students' interest and creativity (Kobsiripat, 2015; Resnick et al., 2009).

Research Problem and Aims of the Study

Since these reforms to curricula have taken place only in recent years, the teaching of CT and programming in compulsory education is still in its initial stages, and teachers are inevitably facing challenges, the most urgent of which is insufficient CT competence. As the concept of competence involves a vast area, the present study focuses primarily on teachers' cognitive competence in terms of their content knowledge (CK). CK is defined as teachers' knowledge of the subject matter to be learned or taught (Shulman, 1987). A lack of CK is due to the fact that primary school teachers, and especially at K-6 level, are usually generalists, and typically did not have a strong computer science background during their teacher training (Gadanidis et al., 2017). In addition, neither CT nor programming generally forms part of (pre-service) teacher education, and teachers' professional development (in-service) in terms of incorporating CT lags behind (Yadav, Gretter et al., 2016). What a teacher knows, does and believes has a major influence on how students learn, and hence preparing teachers to teach a new curriculum is equally as critical as drawing up this curriculum (Gal-Ezer & Stephenson, 2010). Teachers therefore need to upgrade their competence in order to integrate CT into these newly revised curricula.

Besides these problems in teaching practice, the research community is also rife with challenges. Firstly, the lack of consensus on a definition of CT (Weintrop et al., 2016) has repercussions for CT education. The inherent tension between attempting to define the "core" of CT versus the "peripheral" aspects (Voogt et al., 2015) can cause inconsistency and confusion for compulsory school teachers when it comes to teaching and assessing CT. Furthermore, due to the novelty of integrating CT into compulsory education, relevant research is still scarce. Historically, programming has been available to only upper secondary school students (grades 10–12) (Heintz et al., 2016; Mannila et al., 2014) and college students. While scientific efforts to explore CT are gaining pace, there is a paucity of literature relating to how teachers implement CT within the context of school-wide CT learning at the elementary level (Israel et al., 2015). Moreover, previous research seldom focuses on the teachers' perspective (Portelance & Bers, 2015), for example how teachers determine which CT skills to embed in their own classrooms (Yadav et al., 2018). Studies on CT have primarily focused on the students' perspective, definitional issues, and tools that can foster CT (Grover & Pea, 2013). Due to the strong influence of teachers on students' learning, more research should be conducted to explore the teachers' point of view.

In summary, this situation gives rise to several problems. Many of these new curricula have already taken effect, obliging teachers to integrate a subject in which most of them have never received any training, and the extant research provides little assistance from their perspective. Hence, this study is dedicated to investigating how teachers integrate CT with programming, an area in which most of them have barely received any training. This is the research problem, in a broadest sense, that is addressed by this dissertation. In other words, how has CT been integrated in classrooms thus far, given teachers' limited competence in CT? This investigation took place in the context of Swedish compulsory education and with BBPLs, in view of their advantages and suitability for K-9 students.

This research problem generated two research aims. The first aim of this study is to map the CT skills that are taught and assessed by teachers using BBPLs. An in-depth investigation was expected to provide an overview of the extent of CT integration in school for all stakeholders (students, teachers, policymakers and researchers). The results can be seen as an evaluation of the integration of CT thus far, allowing us to answer several questions, such as what CT skills are involved, and have all these skills been practiced equally in the classroom? If so, how did they progress over time? If not, why have some attracted more focus than others? In this case, what are the challenges and counter-measures?

The second aim of this study is to close the research gap relating to the teachers' perspective on this issue. An examination of the aspects teachers engage with in the classroom can reflect their knowledge of CT. As stated in the research problem, the CT competence of teachers is limited, and it is therefore necessary to survey their comprehension of CT directly, to bring their CT competence to the fore. This is one, if not the most, crucial factor affecting the way that teachers orchestrate learning in the classroom. This scrutiny of their CT competence is important in two ways. Firstly, it can provide an explanation of the current status of the integration of CT, such as the selection of CT skills engaged in the classroom. Secondly, it can direct the attention of teachers to the underdeveloped or overlooked aspects of their CT competence, and to adjust and improve their teaching accordingly. In other words, it can help them to identify the areas of CT that require professional development.

Research Questions

Two research questions were formulated for this dissertation:

1. *Based on the current status of teachers' competence in CT, what CT skills have been taught and assessed by teachers using BBPLs in compulsory education, and how have these skills progressed?*

2. *What is the teachers' level of CT competence, and how does it influence the integration of CT?*

These two questions were operationalized through four sub-studies. The relationships among these four studies and their contributions to answering the two research questions are illustrated in Figure 1 and described below.

Sub-study 1. To address Research Question 1, and to find out which CT skills are taught and assessed by the teachers using BBPLs first requires us to identify which CT skills can be obtained via BBPLs. An inventory of the CT skills that could be obtained with BBPLs from the existing literature gave an outline of the capacity of BBPLs to deliver CT to young learners.

Sub-study 2. After having identified the CT skills that could be delivered by BBPLs in Sub-study 1, an exploration was carried out as to whether the teachers, with their current CT competence, could actually capture any of these CT skills in the students' learning. As argued above, more research is needed to shed light on the teachers' perspective, and teachers were therefore interviewed in this study to identify which skills they perceived pupils to have developed, or whether they noticed the development of any CT skills at all. The aim of this was to create some tentative insight into the integration of CT education.

Sub-study 3. Based on evidence from Sub-study 2 that CT skills were engaged by teachers in practice, further investigations were carried out to systematically examine which CT skills were being taught and assessed by the teachers in compulsory education, and how these skills were progressing. The results of this investigation provided details of the integration of CT in compulsory schools, and offered more rounded answers to Research Question 1. A comparison between the results of Sub-study 1 and 3 revealed differences between the CT skills that could be obtained theoretically and those were actually engaged by the teachers.

This comparison laid the groundwork for possible improvements, and also reflected the teachers' limited CT competence. Sub-study 3 was a natural extension of Sub-study 2.

Sub-study 4. The last study surveyed the teachers' CT competence. The logical link with the research question is threefold. Firstly, it provided answers to Research Question 2. The level of teachers' CT competence is a major challenge faced by educational institutions, and strongly determines how well CT is integrated in schools. A clear and direct picture of teachers' content knowledge of CT can indicate areas for professional development. Secondly, a scrutiny of teachers' competence may unveil some possible explanations of the results of Sub-study 3, since what a teacher can (or chooses to) teach may reflect the teacher's competence in that subject. Thirdly, and most importantly, the findings produced new knowledge about teachers' perspectives on CT research, as similar research is scarce.

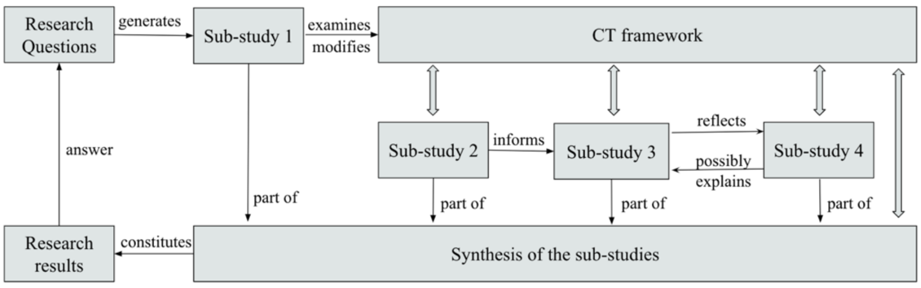

Figure 1. Structure of this dissertation

The four studies included in this dissertation were published in several different outlets.

Sub-study 1: Zhang, L.C., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers & Education, 141*, 103607.

Sub-study 2: Nouri, J., Zhang, L.C., Mannila, L., & Norén, E. (2019). Development of computational thinking, digital competence, and 21st-century skills when learning to program in K-9. *Education Inquiry, 11*(1), 1–17.

Sub-study 3: Zhang, L.C., Nouri, J., & Rolandsson, L. (2020). Progression of computational thinking skills in Swedish compulsory schools with block-based programming. In *Proceedings of the 22nd Australasian Computing Education Conference* (pp. 66–75). February 3–7, 2020. Melbourne, Australia. Australian Computer Society, Inc. ACM.

Sub-study 4: Zhang, L.C., & Nouri, J. (2019). Assessing K-9 teachers' computational thinking skills through a computational thinking test. In J. Keengwe & P. Wachira (Eds), *Handbook of research on integrating computer science and computational thinking in K-12 education*. Hershey, Pennsylvania: IGI Global.

Background

Definition of Computational Thinking

Throughout the evolution of CT, an agreed definition has been absent (Denning, 2017). Despite the increasing recognition of CT, “there is little agreement on what CT encompasses” (Brennan & Resnick, 2012, p. 2), and the studies in the literature have not reached a consensus on a formal definition of CT (Kalelioğlu & Gülbahar, 2014; Lye & Koh, 2014). This lack of a unanimous definition (Weintrop et al., 2016) has led to diverse efforts to define CT. Thus far, three categories of definitions have been put forward in the literature (Román-González et al., 2017). Firstly, generic definitions focus on the thought process used in problem-solving that leads to a solution involving computational steps or an algorithm, as in the work by Wing (2011) and Aho (2012). Secondly, operational definitions are used; for instance, in collaboration with higher education, K-12 education and industry, the Computer Science Teachers Association and International Society for Technology in Education offered a concise operational definition of CT as a problem-solving process characterized by formulating, organizing, analyzing, automating, presenting, implementing, and transferring (CSTA & ISTE, 2011). The third category relates to educational definitions. Table 1 (Zhang & Nouri, 2019) summarizes the definitions and frameworks that are relevant to the scope of this study, i.e., K-9 education.

The common factor in these definitions and frameworks is that they define CT in terms of a list of components that are fundamental to programming. These computational-related components are referred to as CT skills in this dissertation. It can be observed from Table 1 that despite disagreements between researchers, certain CT skills are commonly recognized in these different definitions and frameworks, namely abstraction, algorithms, decomposition, parallelization, debugging, and control flow. In other words, at the heart of these definitions and frameworks, CT is seen as a thought process that utilizes these skills for problem-solving. This dissertation therefore defines CT as a thought process of solving problems by using skills that are fundamental in programming (CT skills), regardless of the discipline of application.

Table 1. Summary of CT frameworks

	Barr & Stephen-son (2011)	Brennan & Resnick (2012)	Selby (2012)	Grover & Pea (2013)	Seiter & Foreman (2013)	Angeli, et al. (2016)	Kalelioglu et al. (2016)	Repen-ning et al. (2016)
Abstraction	✓	✓	✓	✓	✓	✓	✓	✓
Algorithms	✓	✓	✓	✓	✓	✓	✓	
Data	✓	✓		✓	✓		✓	
Decomposition	✓	✓	✓	✓	✓	✓	✓	
Parallelization	✓	✓		✓	✓		✓	
Testing and debugging	✓	✓		✓			✓	✓
Control structure	✓	✓		✓			✓	
Automating							✓	✓
Generalizing			✓			✓	✓	
Simulating	✓		✓				✓	
Event		✓						
Being incremental and iterative		✓						
Expressing, connecting and questioning		✓						
Reusing and remixing		✓						
Efficiency and performance constraints				✓				
Systematic processing				✓				
Conceptualizing							✓	

From the educational definitions and frameworks listed above, this study adopts the framework proposed by Brennan and Resnick (2012) as the basis for defining CT skills. This framework, which was intended for the study and assessment of the development of CT, is considered to provide "a wide coverage of CT" (Kong, 2016, p. 379). Many curricula emphasize the learning of basic CT through the mastery of what Brennan and Resnick (2012) term "CT concepts" (Falloon, 2016). Their framework has also been widely discussed and laid out as a basis for further empirical and theoretical studies (Zhong et al., 2016; Lye & Koh, 2014;). More importantly, this framework has been proposed as a suitable framework for conceptualizing the CT skills that can be developed through BBPLs, such as Scratch (Chen et al., 2017; Sáez-López et al., 2016; Brennan & Resnick, 2012), which is the programming tool investigated in this work. Brennan and Resnick developed this framework with a focus on Scratch (Lye & Koh, 2014), thereby enriching this tool with a theoretical foothold. The choice of Scratch for the present work will be explained in a later section. Their framework (Table 2) defines CT using three dimensions: the concepts, practices and perspectives of CT.

Table 2. Framework proposed by Brennan and Resnick

Dimension	**Description**
CT concepts	These are the concepts with which designers engage as they program, i.e., sequences, loops, parallelism, events, conditionals, operators and data
CT practices	These are the practices developed by designers as they engage with the concepts, i.e., being incremental and iterative, testing and debugging, reusing and remixing, abstracting and modularizing
CT perspectives	These are the perspectives formed by designers about the world around them and about themselves, i.e., by expressing, connecting, and questioning

To ensure the validity of this framework, Sub-study 1 performed a systematic examination of previous empirical evidence, which led to the development of an extended version of the framework (Zhang & Nouri, 2019). This extended version was not applied directly in all the sub-studies in this dissertation; the intention was to give it time to mature in the research community by exposing it to more discussion and critique before applying it in future work. It was therefore adopted only in Sub-study 3, and Sub-studies 2 and 4 were based on the original framework. The extended version is presented in Sub-study 1.

Computational Thinking through Programming

CT and programming are often intertwined. This section presents some necessary definitions and clarifies their relationships to avoid confusion. Within the scope of this study, the intention of programming is assumed to be understanding and solving a problem (Bell et al., 2017, p. 5) and formulating the process that is needed to come to the solution, with enough precision that it is clear how it should be done (ibid, p.11). Even more specifically, in the context of a school, programming typically involves a student solving problems by writing small computer programs so that the program artifacts can perform the desired operations. Although a wide range of activities involve programming without a computer, for instance Computer science (CS) unplugged (Bell et al., 2010), programming typically takes place on a computer.

Programming is not a synonym for CT, but is one possible context for the practice of CT. CS may be the field and practice from which CT skills arose, but it is not the only discipline in which these skills can be found or applied (Voogt et al., 2015, p.717). Training in CT takes place naturally through programming. During the construction of a computer program, CT is engaged automatically, as it utilizes the skills that are fundamental to programming. For example, all programs require the construction of sequences, and almost all of them need to be tested and debugged, which are skills that embody CT. The programming artifacts are the evidence that CT has taken place (Sellby & Wollard, 2010). As Grover and Pea (2017, p. 31) put it, "few would challenge the proclamation that programming is an important – and engaging – vehicle for learning and applying (and hence, teaching and assessing) CT". Furthermore, as CT emphasizes the underlying thought process used in programming, it lowers the barriers to entry to computer programming. Since it is perceived as a nonthreatening term that does not focus on computer programming syntax (Román-González et al., 2019; Heintz et al., 2017), it promotes "programming to learn" rather than "learning to program (ibid, p.80). Particularly with the prevalence of BBPLs such as Scratch, programming is an effective means of developing CT (Berland & Wilensky, 2015; Brennan & Resnick, 2012; Wilson & Moffat, 2010).

Programming Tools in Compulsory Education

In the setting of compulsory education, there are four common types of programming tool for training CT: 'CS unplugged', educational robotics, block-based programming (BBP), and text-based programming (TBP). This section describes these programming tools and illustrates them with examples.

'CS unplugged' offers activities that do not require an automated agent such as a personal computer for training in CT. An example of a learning sequence might be to ask pupils to draw the order of their activities between waking up and arriving at school. These off-computer activities, depending on their complexity, can be introduced as early as kindergarten. Moreover, they are suitable for contexts with a lack of access to computers, or where teachers are not qualified to teach computer programming. While providing valuable introductory activities that can expose children to the nature of CS, unplugged activities may keep learners from the crucial computational experiences involved in the common practice of CT (Grover & Pea, 2013, p. 40). In previous studies, teachers have reported that computer programming allowed students to try things out, make mistakes, explore and create things, which were much more engaging than unplugged activities (Stager & Martinez, 2017, p.42). Older students have also reported dissatisfaction with off-computer or unplugged activities, compared with computer programming (ibid, p.41).

Educational robotics is a powerful, flexible, teaching and learning tool that can encourage students to construct and control robots using specific programming languages (Alimisis, 2014). It provides a valuable path allowing K-9 students to obtain CT skills (Atmatzidou & Demetriadis, 2016). There are two main forms of educational robotics, physical and virtual. Physical robotics such as Bee-bot require students to control the movements of a robot, such as turning left or right and moving forward etc. The manipulation of the robot is achieved by programming in a specific language. Different forms of robotics require different types of programming languages, and some can be programmed in several languages; for instance, Lego Mindstrom works with both BBPLs and TBPLs. Virtual robotics such Light-bot functions similarly. A student develops a program in a programming environment, and when the program is executed, the environment will show a simulation of the robot's movement on the screen. However, although educational robotics is useful, it cannot reach its full capacity for delivering CT without a programming language.

BBP has its origins in visual programming, and due to the popularity of emerging tools such as Scratch and Alice has attracted much attention in both practice and research. This new generation of tools allows the user to compose programs by dragging blocks into a scripting area and snapping them together to form scripts. These blocks usually have different shapes, and to prevent syntax errors, the user can only form scripts by snapping together blocks of shapes that can be joined together. Figure 2 demonstrates an example in Scratch in which the purple and orange blocks can be snapped together, while the green one does not fit. This provides cues that illustrate how and where a given block can be used or combined. Furthermore, the blocks are "easily-browsed and logically organized drawers" (Weintrop & Wilensky, 2019, p.2), and are color-coded to further reduce the barrier to programming. The BBP paradigm provides an intermediate level of abstraction between high-level building blocks and low-level text-based language commands, thus alleviating the burden imposed by syntax on novice programmers (Repenning, 2017; 1993).

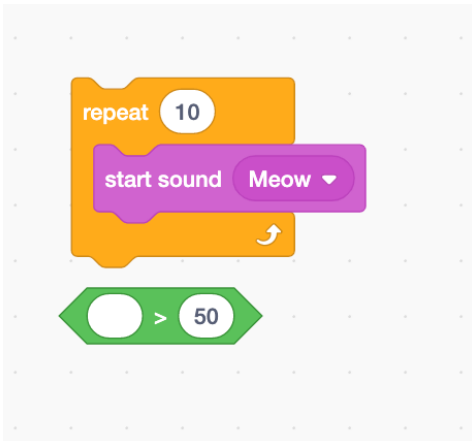

Figure 2. Block commands in Scratch

Of the plethora of BBPLs, this study focuses exclusively on Scratch, for the following reasons. Firstly, Scratch is used more frequently than any other similar tool (Price & Barnes, 2015; Wong et al., 2015). Although it could be argued that both Scratch and Alice are of the widest use in contemporary formal education involving block-based environments (Weintrop & Wilensky, 2017), they have different goals and target groups (ibid). Alice "is best suited for those students who are already coming across problems in other STEM… and have sufficient background to be able to tackle a more rigorous approach toward problem solving" (Utting et al., 2010, p.3). In contrast, Scratch was designed from its inception to help children to get started in this area, from the age of four upwards (Maloney et al., 2010). Furthermore, the large number of users has led to the development of a vast online support system, and these rich online resources mean that Scratch is a compelling way of introducing young learners to programming. More importantly, as mentioned previously, Scratch has its theoretical origins in Brennan and Resnick's framework. Their framework was used to guide the design of the next generation of Scratch, Scratch 2.0 (Brennan & Resnick, 2013). No other similar language has evolved in relation to a CT framework in this way. Last but not least, focusing on one BBPL helps to delineate the scope of this study and make the study manageable.

Scratch was created in the MIT Media Laboratory by the Lifelong Kindergarten Group, and is a media-rich software development environment for novices (Resnick et al., 2009). The Scratch interface has three main sections (Figure 3): a stage area, a palette for blocks, and a coding area to place and arrange the blocks into scripts that can be run. The scripts are composed by dragging and dropping blocks from the block palette into the scripting area. The blocks contain components such as conditionals, data, events, loops, motions etc. When the scripts are executed, animations of graphical characters, for example the Scratch Cat, are displayed in the stage area. The latest version, Scratch 3.0, supports direct use via a web browser.

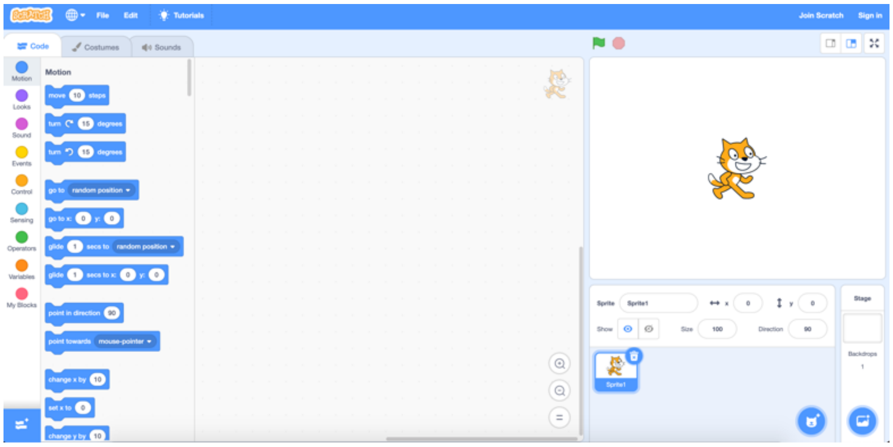

Figure 3. The Scratch environment in a web browser

TBP can be done using the conventional professional programming languages e.g., C#, Java, Python, by typing various characters to create syntax or a list of codes that is readable in a particular language. Although these powerful and sophisticated TBPLs can deliver CT, these are less appropriate for young learners, since few seven-year old students can comprehend syntax such as “public void” or “return true”. BBPLs, on the other hand, were invented for the young, and are age-appropriate for K-9 CT education. Previous research has shown that BBPLs are viewed as easier to use than TBPLs due to “the ease of the drag-and-drop composition, the lack of needing to memorize commands, and fact that in the tools we used, blocks were closer to natural language than text-based programs, making them easier to read” (Weintrop, 2015, p. 302). In contrast, young students associated TBPLs with intimidation, frustration and a need for substantial help from experts (Good & Howland, 2016); however, this does not necessarily exclude TBPLs from compulsory education.

History of Programming Tools for Children

Education in computer programming has been an active area since CS was established as a discipline in higher education institutions in the 1960s (Passey, 2017). However, while programming flourished at these institutions over the subsequent six decades, the penetration of programming into young children's learning took place much later. The personal computer reached a widespread presence only in 1983 (Parker & Davey, 2014). Alongside the adoption of computers in schools during the 1980s, programming started to become available to schoolchildren; in fact, programming was "one of the primary activities and, indeed, one of the main rationales for buying the computers in the first place" (Resnick & Silverman, 2005). At that time, programming activities for children were often carried out using Logo (ibid), an educational programming language designed in 1967 by Seymour Papert and his colleagues to underpin Piaget's constructivism.

This section describes the modern evolution of programming tools for children based on Logo, due to the kinship between Logo and Scratch, which provides a context that illustrates the strengths of Scratch. According to Gunion et al. (2009), several programming tools based on Logo were developed before Scratch, such as Microworlds EX and MultiLogo. Logo, one of the first educational programming languages designed for children, featured a triangular turtle that roamed across the monochrome screen used at the time. Children could steer the turtle using simple syntax such as forward, backward, pen up, pen down etc. The turtle was able to draw using pens of varying colors, and could create different patterns, for example. Logo supported looping, branching, functions and recursion (Gunion et al., 2009). Later, Microworlds EX emerged as a derivative of Logo, with similar commands and syntax; however, it differed from Logo in that it allowed multiple turtles to work on the screen at the same, enabling parallelism and allowing users to manage the control flow of these parallel tasks. It also provided 'kid friendly' error messages. Microworlds led to the development of other end-user programming tools such as AgentSheets, "a massively parallel, visual, end-user programmable computation idea" (Reppening, 2017, p.69). In view of the complexity of parallel processing in programming, especially for novice programmers, Mitch Resnick developed a concurrent extension for Logo in the late 1980s, naming it MultiLogo. This program was intended to "give people (particularly non-expert programmers) a simple yet powerful model for thinking about and programming parallel processes (Resnick et al., 1990, p. 60). As an extension of Logo, MultiLogo used the familiar turtles, graphics and commands, and could also be connected to and used to control the hardware of Lego robots, thus becoming the prototype for today's Lego Mindstorms. Lego Mindstorms was a product of a cooperation between MIT and the Danish company Lego. The main idea was that Lego bricks could be manufactured with sensors and motors, and that all of these components could be programmed. Lego hardware can currently be programmed with many different languages, both text-based and block-based.

However, it has been shown that the initiatives mentioned above did not live up fully to Papert's standard of programming tools for children (Resnick et al., 2009), and were not introduced into mandated curriculum documents (Gadanidis, 2015, p. 309). According to Resnick et al. (2009), Papert envisioned that programming languages should have a "low floor" (i.e. should be easy to start with), a "high ceiling" (i.e. should offer opportunities to create increasingly complex projects over time), and "wide walls" (i.e. should support many different types of projects so that people with many different interests and learning styles can all become engaged). The shortcomings of these earlier languages were their difficulty of use, meaning that children were simply unable to master the syntax of programming; their divergence from the interests or experiences of children, such as requiring them to generate lists of prime numbers or play with recursion; and their lack of context, meaning that guidance and encouragement for deeper exploration was not provided (ibid). To counteract these problems, Resnick's team released Scratch in 2007 as a tool that was more "tinkerable", more meaningful, and more social for children to program with.

New Curricula in Compulsory Education

Over the last decade, many educational systems have updated their curricula by incorporating CT and programming into K-9 education. Despite the shared aspects of these subjects, there are different approaches to integrating CT and programming in schools. This section demonstrates two of these approaches: a cross-curriculum strategy, and the establishment of a new subject. The following elaborates on these approaches and describes new curricula for CT and programming in some selected countries.

The UK positioned CT within a newly designed subject called simply "computing". This reform commenced in 2012, with the British Royal Society's report (2012) entitled "Shutdown or restart: The way forward for computing in UK schools". The report alarmed policymakers with its portrayal of the pressing issues and need for computing education in K-12. One year later, a national curriculum for computing was introduced in which CT was an important part. It was stipulated that all students should:

- understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms and data representation
- analyze problems in computational terms, and have repeated practical experience of writing computer programs in order to solve such problems
- evaluate and apply information technology, including new or unfamiliar technologies, analytically to solve problems
- be responsible, competent, confident and creative users of information and communication technology

(UK Department of Education, 2013)

In 2015, the Australian new curriculum for "technologies" went into effect, with a clear focus on CT. One core aspect of this curriculum is the development of CT skills in terms of problem-solving strategies and techniques that assist in the design and use of algorithms and models (Falkner et al., 2014). The following skills are now mandatory for students from Foundation Year to Year 8:

- Foundation Year to Year 2 should be able to "use computational thinking to plan, construct and evaluate digital solutions for personal need, safely using a range of appropriate software devices, functions and commands";
- Year 3 and 4 "use computational thinking to understand the nature of a problem; describe processes; and plan, construct, modify and test creative digital solutions for personal and community purposes" ;
- Year 5 and 6 "use computational thinking to analyze problems to identify key dimensions, compare common digital solutions and make decisions about fundamental design and development features… design, create, test, edit, troubleshoot and evaluate digital solutions" ;
- Year 7 and 8 "use computational thinking with increasing independence and in collaboration with others to define, analyze, research, model, test and evaluate digital solutions to problems".

(ACARA, 2012)

Although the countries mentioned above integrated CT into a new subject, others adopted an interdisciplinary approach. In Finland, CT and programming have been integrated into the national curriculum for Years 1–9 since 2016, in the subject of mathematics and crafts. The curriculum specifies learning objectives that are related to CT. In the official document, CT is listed as one of the key concepts and definitions in programming and teaching programming, and is defined as the "ability to determine which problems can be solved and which ideas can be implemented with the help of a computer program. It requires algorithmic thinking and programming to formulate a problem in a form that the computer can interpret" (Opetushallitu, 2020a, translated from Swedish). Finnish pupils should be able to program in age-suitable environments to solve problems. According to Opetushallitu (the Finnish National Education Agency):

- Students in Years 1 and 2 should familiarize themselves with the basic principles of programming and the terms, sequences of instructions, conditionals and loops. Students should be able to give instructions; automatically follow the instructional relationship between cause and effect; and identify formulas that are repeated. Students should practice testing and debugging instructions in a precise way.

- In addition to the skills trained in Years 1 and 2, students in Years 3 to 6 should learn the basics of programming in a visual programming environment, including the skills of giving precise instructions as well as engaging conditionals and loops. The students should investigate how programming is used in the surrounding community and use programming as a tool for production and creation.
- In addition to the skills included in Years 3 to 6, students in Years 7 to 9 should develop positive experiences related to programming. Students should practice using the basic structures of programming: variables, conditionals, loops and decomposition, and functions. Students should solve problems and realize their own ideas using visual or text-based programming languages. Students should investigate how programming is used in the surrounding community.

(Opetushallitus, 2020b, translated from Swedish)

Sweden has also chosen a cross-curriculum approach by embedding programming within other subjects. In 2012, the Swedish government was advised on the need to include programming in the school system as a crucial part of developing the society’s digital competence, and requested Skolverket (the Swedish National Agency of Education) to propose a national IT strategy for the school system for 2015 (Heintz et al., 2017, p. 2). The compulsory curriculum was updated by making programming obligatory from 2018. Although the term “computational thinking” is not explicitly used in the curriculum, an official supporting document commented on how programming is intended to be perceived within the curriculum. “Programming includes writing code, which has strong similarities to general problem solving. This includes formulating a problem, choosing a solution, trying and reevaluating it, and documenting it” (Skolverket, 2017a, translated from Swedish). This perspective is clearly aligned with the definition of CT used in this study. The new Swedish curriculum integrates programming mainly within the areas of mathematics and technology.

In mathematics, the following skills are taught:

- Years 1 to 3: How unambiguous step-by-step instructions can be designed, described and followed as a basis for programming. The use of symbols in step-by-step instructions.

- Years 4 to 6: How algorithms can be created and used for programming. Programming in visual programming environments.
- Years 7 to 9: How algorithms can be created and used for programming. Programming in different programming environments. How algorithms can be created, tested and iteratively improved when using programming for mathematical problem-solving.

In technology, the following skills are taught:

- Years 1 to 3: Controlling objects through programming.
- Years 4 to 6: Controlling students' own constructions or other objects through programming.
- Years 7 to 9: Controlling and regulating students' own constructions, for instance using programming. How digital tools can support the development of technical solutions, such as creating drawings and carrying out simulations.

(Skolverket, 2020b, translated from Swedish.)

According to Heintz et al. (2017), Sweden opted for the interdisciplinary mode for three reasons: the lack of space for a new subject in the already crowded curriculum; the value of the experience of using programming in different subjects and raising interest among previously underrepresented groups; and the provision of a context for students to apply CT and computing to solve increasingly complex problems in different disciplines. The new curriculum was introduced only one year ago, raising the question of whether Swedish teachers are well-prepared to deliver it. The next section reviews the CT competence of teachers.

Competence and Professional Development in Computational Thinking for In-Service Teachers

The introduction of CT into the curriculum inevitably gives rise to the demand for continuous in-service professional development. Shulman's (1987, 1986) seminal work on pedagogical content knowledge (PCK) is highly regarded by teacher educators as an appropriate framework for developing teachers' knowledge to allow them to teach CT and programming (e.g., Angeli & Valanides, 2016; Hubwieser et al., 2013; Saeli, 2012). In the context of CT education, Angeli & Valanides (2016) defined CK as an understanding of the skills described in Table 2, such as sequencing, loops, conditionals, data etc., and pedagogical knowledge (PK) as general pedagogical knowledge that is applicable to all other content domains, for instance the use of questions to promote understanding, the use of examples, explanations and demonstrations, in addition to knowledge about subject-specific pedagogical practices pertinent to CT.

According to several recent surveys, there is a pressing need to increase teachers' content knowledge of CT. As suggested by a literature review by Hsu et al. (2018), teaching staff need to receive an overall education in CT. Sentance and Csizmadia (2017) surveyed over 300 teachers who were currently teaching CT, and concluded that the most commonly mentioned challenge experienced by teachers was their content knowledge of CT. The problems in Sweden regarding teachers' CK and programming skills are also acute. According to the Swedish National Union of Teachers (Lärarnas Riksförbund, 2017), almost 70% of math teachers in Years 7–9 had never received any training in programming. Furthermore, when asked how they felt about teaching programming in classroom, 56% of the respondents replied "very uncertain" and 28% "quite uncertain." This was also confirmed by a report from the European Commission (Bocconi et al., 2016), which found that existing research has largely focused on pedagogical aspects for in-service teachers, since most primary school teachers have historically been generalists, with no training in CT nor programming.

Although CK alone is not sufficient to design effective teaching, the academic content of a subject is the cornerstone for teaching and learning. Pedagogical innovations must be steeped in academic content (Ferdig, 2001; Littlejohn & Stefani, 1999). Designers of successful classroom interventions must ensure that the content is sufficiently engaging to interest children in the world of learning (Brown, 1992, p. 173). Teachers cannot be expected to devise an interesting and engaging lesson if they are unaware of the academic content of that subject. Being unfamiliar with the subject content can also lead to severe consequences; for instance, a teacher may not be able to judge the difficulty of the content, meaning that they are unable to facilitate learning progression. In other words, if instruction is too easy for the student, they will lose interest, and if it is too hard, they will become frustrated. This can, and is likely to, affect a teacher's confidence, resulting in negative impacts on the students' attitudes towards the subject. Even worse, a teacher may try to avoid teaching it at all. It is only natural that teachers learn to develop pedagogies for a subject after they understand the subject matter. Moreover, an exploration of what teachers know can to some extent help them plan their professional development more purposefully, and more attention should therefore be given to investigating their CK. Thus far, only one study has surveyed the CK of in-service teachers. Kong and Lao (2019) developed an instrument for measuring teachers' understanding of CT, which primarily examined CT practices using Brennan and Resnick's framework. As one aim of the present study is to shed more light on teachers' CT competence, it evaluates teachers' CK of CT more systematically than previous efforts.

Since the new curriculum took effect, various initiatives, including both top-down measures and grassroots action, have been deployed to facilitate professional development in CT. Australia, Finland and Norway now prepare teachers using massively online open courses to deliver free computing content and pedagogy to teachers (Heintz et al., 2016). In the UK, a grassroots, decentralized computing community was established called Computing At School. This community not only provides resources online, but also allows teachers to meet in person at local hubs to exchange ideas and discuss issues. In this way, it can reduce their isolation and increases their engagement (Computing At School, 2016). Swedish teachers are also given the option to undergo online training in different forms. The National Education Agency offers a 16-hour web-based introductory course to all in-service teachers, and they are encouraged to enroll in tailormade academic courses at universities throughout the country. All of these courses aim to "give in-service teachers knowledge in computational thinking in general and programming in particular to be able to teach based on the National Education Agency's governing document in mathematics and technology for Grade 1–9, and provide guidance in the role of programming in teaching" (KTH, 2020). These courses can generally enhance teachers' programming skills with TBPLs or BBPLs. Teachers are often required to attend several in-person meetings at these universities for collegial discussions, although most of the work is expected to be completed independently with online support via an e-learning platform. For academic credit to be awarded, teachers need to demonstrate satisfactory programming skills, for example through a programming project. This blended mode of delivery allows for flexibility (in terms of time and location), ad hoc interactions, support and the possibility of sharing experiences. In brief, teachers' competence in CT is an indispensable component of the promotion of CT education.

Research Method

Research Context

This study was conducted as part of a national research and development (R&D) project for in-service teachers called "Programming in Subject-based Teaching". The goal of this project is to help in-service teachers to enhance their CT competence and programming skills, to allow them to deliver the new Swedish curriculum, and it assists teachers in developing instructional content and didactic methods for teaching programming. The project adopts an approach in which teachers' knowledge and experience forms the basis of joint knowledge development through the use of scientific methods and collaboration between teachers and researchers. This close collaboration between teachers and researcher includes aspects such as study design and data collection. Researchers and teachers collaborate to iteratively design, implement and evaluate learning activities that involve CT and programming. In addition, the project aims to advance research on CT in compulsory education.

This project was initiated by a non-profit research organization with a team of researchers. Between the fall of 2017 and the spring of 2020, the program enrolled approximately 130 teachers, 17 principals, and five heads of school districts from 15 compulsory schools in five Swedish municipalities. The teachers elected 19 group leaders, who periodically organized local seminars and coordinated the communication between the research team and the teachers. A plenary conference was held once per semester, where all project participants met for collegial learning, and at which the researchers would present the latest research results from the project and report on the progress of the project. The group leaders assisted the researchers in setting up workshops to allow teachers to learn about tasks and didactic methods developed by teachers and researchers through lesson study (Fernandes & Yoshida, 2012). For the principals and district heads, these conferences provided opportunities to meet and discuss organizational issues and leadership with respect to the schools' implementation of the new curriculum.

The essence of this R&D project is to resolve problems related to teachers' CT competence by applying scientific methods, and specifically based on the use of action research (Elliot, 2007) through lesson study. Action research is an approach that is commonly used to improve conditions and practices (Avison et al., 1999). In the case of curriculum change, action research enables researchers and teachers to change and reflect on the immediate problem relating to teaching CT and programming, via a joint effort. It is an iterative process that involves researchers and practitioners acting together in a particular cycle of activities, including problem diagnosis, action intervention, and reflective learning (ibid, p. 94). This approach was chosen for the project for two reasons. Firstly, action research is a form of disciplined inquiry that is used to investigate a problem or question of interest where there is presently no satisfactory answer. There currently appears to be no clear answer to the dilemma faced by teachers outlined in this research. More importantly, it was selected due to its strengths in terms of generating solutions to practical problems (ibid), i.e. helping teachers to integrate CT and programming, and implementing solutions. For professional development by teachers, such as improving teaching strategies, lesson study is a comprehensive and well-articulated process in the vein of action research. Teachers are engaged in a cyclical process in which they use primary resources and real-world information and data to inform new courses of action (Johnson, 2001). Lesson study is based on collegiality, and is characterized by the observation of live classroom lessons by a group of teachers who collect data on teaching and learning and collaboratively analyze it (Fernandez, 2002). From a motivational perspective, this helps teachers to know that their questions and perspectives matter and to pursue investigations that are of authentic concern (Ginsberg, 2011), thus empowering them. Detailed descriptions of the operationalization of lesson study can be found in Sub-study 3.

Research Approach

This thesis is based on case study research (Yin, 2018). Case studies are commonly used in many disciplines such as business, education, nursing, political science, psychology, and sociology. This approach allows the investigator to retain the holistic and meaningful characteristics of a real-life event, for instance school performance, integration process (ibid, p. 6). The rationale for conducting a case study was that the nature and circumstances of this research satisfied the three conditions (ibid, p. 9) for the choice of this approach. Yin showed that the following three conditions (Table 3) should be used to guide the choice of research method in any social science investigation: (i) the type of research question posed; (ii) the extent of the control an investigator has over actual behavioral events; and (iii) the degree of focus on contemporary as opposed to historical events.

Table 3. Three conditions for the choice of research method

Method	**Form of research question**	**Control over behavior**	**Focus on a contemporary issue**
Case study	how, why	no	yes
Experiment	how, why	yes	yes
History	how, why	no	no
Survey	who, what, where, how much/many	no	yes

The primary indicator for the adoption of a specific research method is the type of research question being asked. In general, "who" and "what" questions are more suitable for surveys, whereas "how" and "why" questions are more exploratory and explanatory, and are therefore likely to be more suited to the use of case studies, experiments and histories. The research problem in this study was based on questions related to how CT and programming have been integrated in the compulsory classroom, both of which focus on "how."

However, in addition to a case study, there was another suitable candidate for this study in the form of an experiment, and the second condition therefore needed to be applied. In experiments, researchers manipulate behaviors directly, precisely and systematically, and this was not in accordance with the nature of the present study. The intention of this research was to describe the current extent of curriculum integration, with no direct intervention from researchers, and then to provide possible explanations to allow stakeholders such as teachers to better understand what they have been doing and why this situation has arisen in CT education. Hence, there was no control or experimental group. Moreover, experiments deliberately divorce a given phenomenon from its context, whereas the integration of a curriculum is a complicated event that cannot be studied in isolation.

In terms of the third condition, this study focused on a contemporary phenomenon. The inclusion of CT and programming into the compulsory curriculum in Sweden began only in the fall of 2018; in other words, there is no precedent or documents that would allow a conventional historical study to be conducted.

In summary, a case study is an empirical inquiry that "investigates a contemporary phenomenon in depth and within its real-world context" (Yin, 2018, p.15). Unlike experiments and surveys, case studies provide a deeper understanding of a phenomenon. The results are both descriptive and theoretical, in the sense that questions are raised regarding what the instance is about and why it occurred as it did. This approach is well suited to this research, since the object of study, the integration of CT, is a contemporary phenomenon that is hard to study in isolation. Hence, the context is important and can be addressed using a case study. In this investigation, a case study can provide answers to the question of how CT has been integrated in the compulsory education so far, with a focus on the following four aspects: which CT skills are taught and assessed in classrooms, how do they progress, what is the level of teachers' CT competence, and how does it influence this integration. A case study can provide insights into the current status of CT integration, a complex issue, through a contextual analysis of the teaching of CT using so-called "thick descriptions" (Merriam, 1998). A case study approach was therefore deemed suitable for this investigation.

Research Design

In general, based on the epistemological status of the research, there are three types of case studies: explanatory, descriptive and exploratory (Yin, 2018, p.8). These three general forms of research design serve as the background for choosing a specific design for a case study. Once the research purpose has been determined, a specific design can be chosen from the two-by-two matrix shown in Figure 4, where the first dimension involves single- or multiple-case design, and the second involves single (holistic) or multiple (embedded) units of analysis (ibid, p. 47).

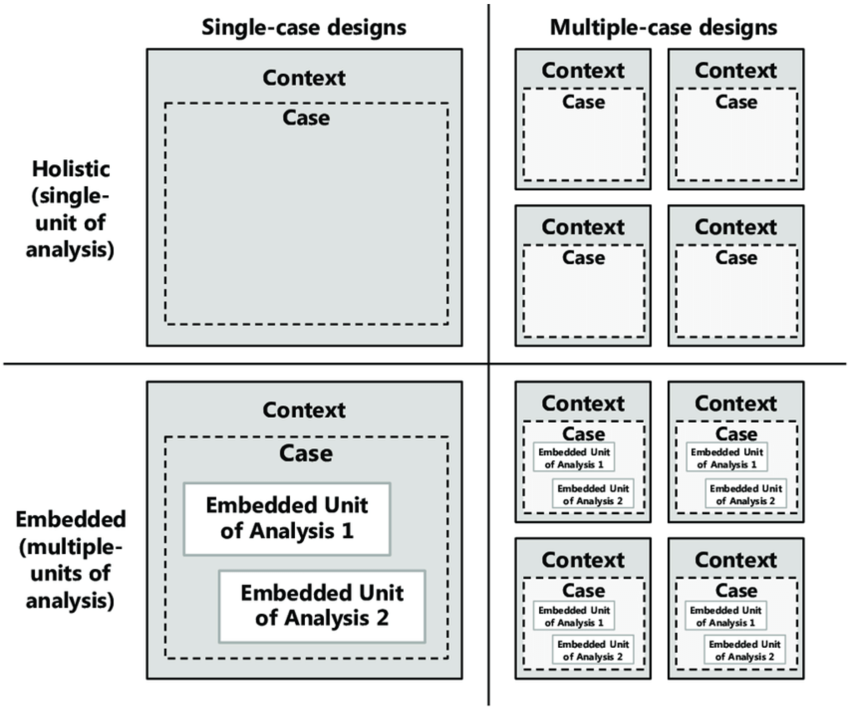
Single-case designs
Multiple-case designs
Holistic (single-unit of analysis)
Embedded (multiple-units of analysis)
Context
Case
Embedded Unit of Analysis 1
Embedded Unit of Analysis 2

Figure 4. Types of case study design

An exploratory case study aims to investigate what is happening, to seek new insights into a phenomenon, and to generate ideas and hypotheses for new research, and this approach aligned well with the nature of this study. Compulsory school teachers are integrating CT and programming into their daily teaching, and due to the novelty of this process, relevant research is not yet up to date in terms of capturing and analyzing this phenomenon. The research problem therefore involves gaining insight into a new phenomenon, and an exploratory case study can be used to chart this unknown area.

As the epistemological status has set the tone for this study, a decision then needed to be reached on whether a single or multiple-cases design would be used to address the research questions prior to any data collection. Yin (2018, p. 49) argued that a single-case design was suitable for a revelatory case, by which he meant "a situation [that] exists when an investigator has an opportunity to observe and analyze a phenomenon previously inaccessible to social science inquiry" (ibid, p. 50). Mandatory instruction in CT and programming in the compulsory curriculum in Sweden is completely new, and is therefore a phenomenon that has scarcely been observable before by researchers. This study can therefore be considered as a revelatory case, and a single-case design was therefore adopted.

It was then necessary to choose the number of units of analysis, i.e. either a single (holistic) or multiple (embedded) units. Yin (2018, p.51) suggested that a unitary unit of analysis is advantageous when no logical sub-units can be identified; however, this is not the case in the present research. Three units of analysis can be identified, with a logical link between them. Of the teachers who participated in this R&D project, the first unit of analysis was a selected group of teachers with programming experience; the second involved groups of teachers; and the third involved all the participating teachers. The logic behind this division into sub-units was that it gave greater coverage of the teachers in each step. Moreover, the choice of a unitary unit of analysis design is typically shaped by an entirely qualitative approach that relies on narrative descriptions, whereas embedded case studies are usually not limited to qualitative analysis alone. This research aimed to use multiple data sources to maximize the opportunity for extensive analysis, thus enhancing the insights that could be gained into the phenomenon.

Although the single-case design was shown to be suitable for the nature of this case study, the investigator needs to use this approach with caution, due to a potential vulnerability. Yin (2018, p. 53) highlighted that a case may later turn out not to be the case it was thought to be at the outset. Single-case designs therefore require careful investigation of the potential case to minimize the chances of misrepresentation and to maximize the access needed to collect the case study evidence. Hence, this case study was designed in a flexible and iterative way (Figure 5) to circumvent any potential problems and to ensure that the sub-units coherently covered the phenomenon of interest. More details of this flexible and iterative design are given in the case study protocol presented below.

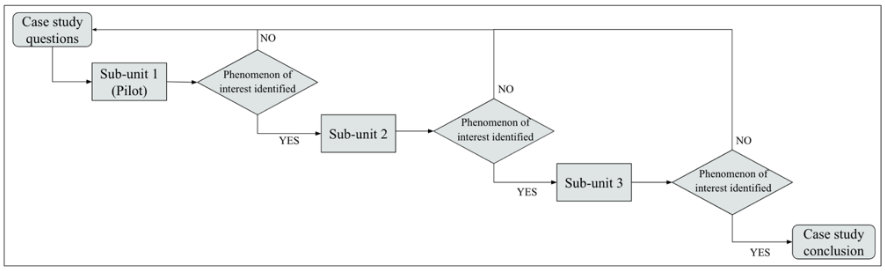
Case study questions
NO
Sub-unit 1 (Pilot)
Phenomenon of interest identified
NO
YES
Sub-unit 2
Phenomenon of interest identified
NO
YES
Sub-unit 3
Phenomenon of interest identified
YES
Case study conclusion

Figure 5. Exploratory case study design with three sub-units

To operationalize the exploratory embedded single-case study design, a case study protocol was developed to give a detailed action plan. A protocol is an important way of increasing the reliability of case study research, and is intended to guide the investigator in carrying out the study (Yin, 2009, p.46). Brereton et al. (2008) pointed out that one of the most important benefits of developing a case study protocol is that it specifies in detail how the investigator intends to answer the research questions. A protocol can also help a researcher to relate the planned data collection back to the research questions, and recall triangulation, which can be used to refine the data collection procedures. In terms of data analysis, a protocol requires the researcher to think in detail about collecting and interpreting data, and forces consideration of the different possible outcomes. The protocol for this case study contains the following components:

• Objective: A new phenomenon has emerged in Swedish compulsory education, as CT and programming are now required to be integrated into the curriculum. The objective of this case study is to explore the extent of this integration. The results are expected to provide insights into this integration to stakeholders, and particularly compulsory school teachers.

• Case: Since the integration process is a multifaceted and complex phenomenon, the primary focus of this study was on investigating the CT and programming skills applied in the classroom, and the CT competence of the teachers. To explore the case, three sub-units were established in a logical order. The three units of analysis were as follows: firstly, selected individual teachers with programming experience; secondly, groups of teachers; and finally, all teachers participating in the R&D project.

• Questions addressed in the case study:

1) Based on the current status of teachers' CT competence, what CT skills have been taught and assessed by teachers with BBPLs in compulsory education, and how do these skills progress?

2) What is the level of teachers' CT competence, and how does it influence the integration of CT?

• Theory/proposition: Teachers cannot teach and assess CT in a comprehensive manner, due to their limited CT competence. The prediction was that these limitations would be reflected in the CT skills they chose to teach. Of the possible ways of linking data to the proposition (Yin, 2018, p.33), explanation building, pattern matching and sub-study synthesis were adopted in this case study.

• Data collection and analysis: Table 4 provides an overview of the collection and analysis of data in each sub-study. It should be noted that these sub-studies do not correspond to the sub-units: four sub-studies were included in this dissertation, the first of which was a systematic literature review and was therefore not a sub-unit of the case study.

Table 4. Overview of data collection and analysis in the sub-studies

Type of study	**Unit of analysis**	**Data collection**	**Data analysis**	**Interpretation**
Systematic literature review	N/A	Qualitative and quantitative data from previous literature	Content analysis and descriptive statistics	Scrutinize previous empirical evidence on which CT skills can be obtained through Scratch
Three sub-units of the exploratory embedded single-case study	Selected teachers with programming skills	Qualitative data from semi-structured interviews	Thematic analysis	Explore whether teachers can capture and assess CT and programming in their subject teaching
	Groups of teachers	Qualitative data from documentation	Thematic analysis	Investigate which CT skills are taught and assessed by teachers, and their progression
	All teachers participating in the R&D project	Quantitative data from a survey, including a CT test and a questionnaire on the teachers' backgrounds	Descrip-tive and inferential statistics	Examine the teachers' CT competence and the relationship between this competence and background factors

The first sub-study is a systematic literature review. This is an integral part of conducting a case study (Yin, 2018, p. 27), as it narrows down the topic and since prior research findings are directly relevant to the goals of the study being undertaken. To identify which CT skills have been integrated in schools using Scratch, it was necessary to determine which CT skills could be acquired via Scratch. This review therefore collected empirical evidence from the extant literature to explore this crucial prerequisite of the study. The review also served another purpose in that it examined the applicability of Brennan and Resnick's (2012) framework to this case study. This framework was adopted to study and assess CT, as it could be used to identify CT skills, and facilitate the deductive parts of the coding and categorizing process in the subsequent sub-studies. The literature review therefore examined how well this framework incorporated the CT skills identified from previous empirical evidence. Moreover, an applicable framework at place is one measure that can be used to enhance the internal validity of a study (Gibbert et al., 2008). In addition to assisting in the identification of CT skills in the case study, the framework served as the basis for pattern matching at the data analysis stage.

The review followed the guidelines set out by Kitchenham and Charters (2007) for performing a systematic literature review in five phases: definition of a research question, design of a search strategy, selection of a study, and extraction and synthesis of data. The inductive content analysis proposed by Elo and Kyngäs (2008) was adopted in this review to identify the CT skills obtained via Scratch in the selected studies. This was executed using the following steps: selection of the unit of analysis, making sense of the data and the whole, open coding, coding sheets, grouping, categorization, abstraction, and conceptual mapping. Two coders performed a pilot analysis of five papers, working together to reach agreement on the semantics of "CT skills". Despite the inductive nature of this analysis, the coders used the common CT skills listed in Table 1 as a reference. Open coding allowed for the possibility of collecting, analyzing and categorizing other CT skills.

Due to the novelty of this phenomenon, it was not taken for granted that CT skills were integrated through programming in the classroom and that teachers could perceive and assess them, due to their limited CT competence. A pilot study was therefore conducted in Sub-study 2 to investigate which skills, in general, students could develop through programming, according to the teachers' judgments. Two possible outcomes were expected: either CT skills could be identified, or they could not. If not, in accordance with the flexible and iterative design of the study, the investigator would first reexamine the processes of data collection and analysis, and even conduct another pilot if necessary. If an absence of CT skills was found to persist after the reexamination of the data or a second pilot study, the case study would be redirected to new research questions.

Fortunately, the first pilot study revealed that the students obtained CT skills through programming from the teachers' perspective, and the next sub-unit was then introduced.

In Sub-study 2, semi-structured interviews were conducted with selected teachers based on their teaching experiences related to programming, since these were the teachers who were most likely to give a positive result. The selected teachers formed the unit of analysis. The interviews were conducted by two researchers, and were recorded and transcribed before thematic analysis was applied. Both inductive and deductive approaches were used in this sub-unit. Brennan and Resnick's framework was used to guide the deductive coding of the CT skills.

Since Sub-study 2 concluded that the selected teachers were able to capture and assess students' CT skills, Sub-study 3 examined the CT skills that were taught and assessed by teachers within a more extensive setting, in order to identify which CT skills teachers integrated into their lessons, regardless of their programming experience. As the 130 teachers participating in the R&D project learned and underwent lesson study in groups, and developed lesson plans for integrating CT and programming, this sub-unit used these teacher groups as the unit of analysis. This generated a large amount of data, which led to a fuller comprehension of the CT skills that teachers had applied in their classrooms. The progression of these CT skills was sketched based on this volume of data.

Sub-study 3 collected documentation on the lesson study undergone by these groups of teachers. Templates were provided by the research team so that the teachers could properly document their lesson studies for data collection. The aim of using these templates was to capture the data pertaining to the CT skills and the assessment of those skills. In addition to the interviews, this was the second source of data for the case study. The goal was to scrutinize, given the teachers' level of CT competence, which CT skills they had integrated and what the problems and challenges were. In this sub-unit, a hybrid approach including thematic analysis (Fereday & Muir-Cochrane, 2006) was used for data analysis that incorporated both a data-driven inductive approach and a deductive framework-assisted approach. Fereday and Muir-Cochrane (ibid, p. 83) argued that this hybrid approach allowed "the tenets of social phenomenology to be integral to the process of deductive thematic analysis while allowing for themes to emerge directly from the data using inductive coding." The framework from the systematic literature review was used to define the codebook for data analysis. Two coders analyzed the teachers' lesson study documentation according to the codebook, and all matches were recorded in the codebook. The coders also processed the data inductively, in order to recognize any words or concepts that could potentially be interpreted as CT skills. Once the coding was complete, the coders interpreted the data to identify and develop themes that were not covered by the framework. The analysis process was iterative and reflexive, and involved rereading all the documentation to ensure that any emerging themes were grounded in the original data.

After the integrated CT skills and progression had been delineated in Sub-study 3, Sub-study 4 was applied to evaluate teachers' CT competence. The competence of a teacher in a particular subject inevitably affects what can be taught. It is no mystery that teachers do not or cannot teach and assess things that they are unfamiliar with or unaware of. The teachers' competence in CT, in terms of their CK, was therefore a salient and indispensable aspect of this case study. According to Yin (2018, p. 64), when an embedded design is used, each individual case study may include the collection and analysis of quantitative data, including the use of surveys within a case. Sub-study 4 used quantitative data. While qualitative data can provide "thick" descriptions of contexts in case studies, supplementing the qualitative data with quantitative data can often result in a better understanding of the studied phenomenon (Seaman, 1999). One of the great strengths of case studies, compared with other methods, is that evidence can be collected from multiple sources. A CT test was designed and administered to survey the teachers' level of CT competence. The content of the CT test corresponded to the framework, in order to maintain coherence between the sub-units. These quantitative data therefore became the third source of data, meaning that the case study had collected data from interviews, documentation and a survey. Since the CT test was sent to all teachers participating in the R&D project, they formed the third unit of analysis. Due to the diversity among the teachers, a questionnaire on their backgrounds was constructed and attached to the CT test. This helped in gaining insights into the background factors related to teachers' CT competence, meaning that the results could provide some explanation as to why the teachers taught and assessed CT skills in the way they did. The survey data were processed using the statistical software SPSS, and analyzed using descriptive and inferential statistics.

Reliability and Validity

The four tests listed in Table 5 were used to evaluate the quality of this case study. These four tests are commonly used in empirical social research, including case studies (Yin, 2018, p. 42). Each test is recommended for use with a case study strategy, and can act as a cross-reference for the phase of research in which each type of tactic is used. To ensure construct validity, the study's conclusions were based on a chain of evidence in which data collected in each sub-study were based on the results of the previous one. The data were therefore collected in a planned and consistent manner, regardless of the type of data drawn from multiple sources. The data were also triangulated in order to strengthen the validity of the results. Data triangulation is essential to increase the precision of empirical research by taking different perspectives towards the studied object and thus providing a broader picture. The need for triangulation is evident in case studies, as they rely primarily on qualitative data. Hence, various data sources were collected for analysis in this case study, and these were mainly interviews, documentations and surveys, complemented by observations and programming artifacts.

Table 5. Tests of the quality of research

Test	**Case study tactic**	**Phase of research in which tactic is applied**
Construct validity	Use multiple sources of evidence	Data collection
	Establish chain of evidence	Data collection
Internal validity	Carry out explanation building	Data analysis
	Carry out pattern matching	Data analysis
	Apply framework	Data analysis
External validity	Apply theory in single case studies	Research design
Reliability	Apply case study protocol	Data collection
	Maintain a chain of evidence	Data collection

The internal validity was first supported through the use of a widely cited framework. Thus, the results of the case study add knowledge based on previous findings. This framework formed a common thread throughout the case study, and was used to analyze the data in all three sub-units and to synthesize the results of the sub-studies. Both explanation building and pattern matching were also carried out. Although all three sub-units were exploratory, the first two identified how CT skills were integrated by teachers in the classroom, while the third offered some explanation as to why these CT skills were integrated in these ways. Given the complexity of the research topic and the possibility of other alternative explanations, the built explanation could only reveal part of the causality. The synthesis of sub-studies also involved pattern-matching, as illustrated in Figure 6. Pattern-matching brings together disparate data gathered as part of case study research, and can combine multiple data collection and analysis methods to contribute distinct epistemological perspectives to a multifaceted and highly contextualized phenomenon (Almutairi et al., 2014).

Yin (2018, p. 21) posited that the goal of doing case study research is to expand and generalize theories (analytical generalization) rather than to extrapolate probabilities (statistical generalization). Hence, the external validity was strengthened through the use of a theory or proposition in the case study. This case study collected data to cover all components of the proposition. Yin (2018, p. 46) maintained that posing "how" and "why" as research questions can help to support the external validity, since these types of questions point to the identification of an appropriate theory or proposition.

The reliability was strengthened by using a case study protocol. A protocol not only helps to avoid a situation in which the evidence does not address the initial research questions but also describes in advance the processes necessary for conducting data collection and analysis. The use of a protocol is an important way of increasing the reliability of case study research (Yin, 2018, p. 96).

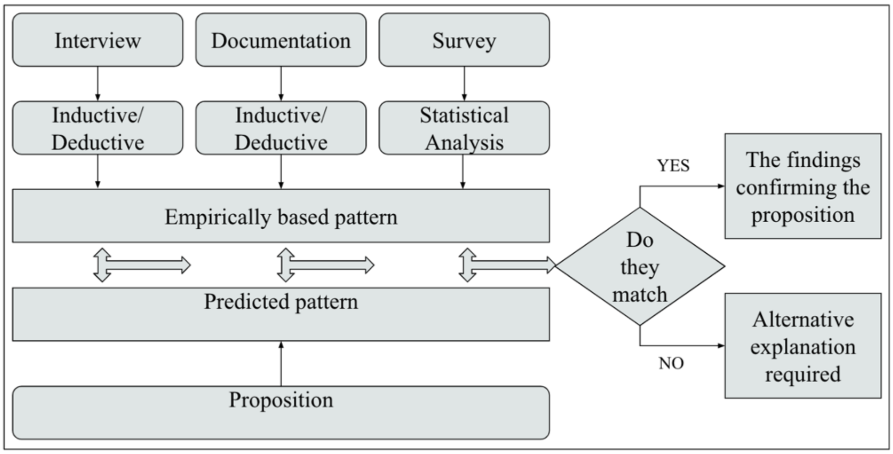

Figure 6. Flow chart for the pattern-matching process

The construct validity, internal validity, external validity and reliability were also specifically evaluated in each sub-study, and discussions of the limitations of the sub-studies were used to improve validity. To further improve the reliability, the data collection and analysis methods were described in detail each sub-study, and data analysis was performed independently and cooperatively. The combination of these measures enabled the synthesis of a multilayered description of the phenomenon of interest, giving a holistic view of the case study.

Limitations

There are certain limitations to this case study. Firstly, only public schools participated in this case study, which may have impacts on the generalizability. Flyvbjerg (2006, p. 13) argued that “[a] representative case or a random sample may not be the most appropriate strategy. This is because the typical or average case is often not the richest in information. Atypical or extreme cases often reveal more information because they activate more actors and more basic mechanisms in the situation studied.” In Sweden, the teaching resources, working conditions, and quality of teaching in private schools can differ from their public counterparts. It is therefore possible that private schools have less favorable prerequisites than public schools in terms of teacher recruiting, economics, working conditions and school management. The results of this study may therefore not be transferable, if the integration of CT in private schools lags behind that of the public sector.

Although Yin (2008, p.45) argued that case studies in general are not concerned with statistical generalization, on a micro scale, the collection of quantitative data via a survey in a case study may still be subjected to random sampling to assure its validity. It was challenging to select effective randomized samples from 130 teachers, and since participation by the teachers was voluntary, even a high response rate may not meet the statistical recommendations.

In addition, the consistency between sub-studies could be further strengthened. Although Sub-study 1 gave rise to an extended version (Zhang & Nouri, 2019) of Brennan and Resnick’s framework, it was only applied in Sub-study 3. The reason for avoiding the use of this extended scheme was to give it time and space to mature in the research community before applying it to future studies.

Ethical Considerations

The ethical considerations of this study were related to both data collection and data analysis. Since students in compulsory education are minors, the consent forms and principle of anonymity were explained to and signed by the legal guardians of the students who voluntarily participated in the study. Particular attention was paid to how digital images or videos of students were dealt with, to protect their privacy. The research group made the teachers aware of the consent and anonymity requirements when the teachers filmed or recorded students' activities in the classroom. Teachers also gave permission for the researchers to use relevant data collected from them, such as lesson study documentation, for research purposes. The storage, security, and usage of these digital data followed guidelines from "The research ethics guidebook" (Boddy, et al. 2010). The data were primarily stored on the Turing learning platform, maintained by the Department of Computer and Systems Sciences at Stockholm University. All the data were downloaded onto a USB drive as a backup, in case of data loss. Data procured through Google Forms were downloaded and stored on the researchers' hard drive, in case the data became corrupted in Google Drive, and only participants within the research project were allowed to access these data. Before the data were stored, they were anonymized, wherever possible, to ensure individual privacy. The data were allowed to be stored as long as the research team needed them for research purposes, and were only to be distributed within the research project for reasonable purposes or requests. The use of data was always consulted on within the research team. It is also ethical for the study to acknowledge the teachers' efforts in gathering relevant data. In addition, regarding data analysis, the author was conscious of possible prejudices and potential subjective influences on the study (Sanjari et al., 2014).

Research Results

This section presents the results of the case study by summarizing the key aspects of each of the four research articles. The summary of each article contains the motivation for that research paper, its research question, the method used and the findings in brief.

Research Article 1

Zhang, L. C., & Nouri, J. (2019). A systematic review of learning computational thinking through scratch in K-9. *Computers & Education*, 141, 103607.

As CT has been embraced by educational systems worldwide, researchers and teachers have considered important questions such as "what to teach" (Grover & Pea, 2013). This question is universally important in learning in all subjects, as it dictates the core content of a subject. The lack of consensus on what CT entails has led to a plethora of definitions and frameworks for CT in the educational context. This is made even more complicated by the numerous programming tools that are currently available, with dazzling affordances. It is unclear whether these foster or focus on the same aspects of CT. This paper therefore exclusively focused on a single programming tool called Scratch, a BBPL developed in the MIT media lab. It systematically reviewed the CT skills that were delivered via Scratch to compulsory school students. Thus, this paper not only addressed the lack of an up-to-date systematic overview of CT education in compulsory schools, but also provided some clarity for teachers in practice regarding "what to teach" and "what can be learned in CT". Scratch, a media-rich programming language with a theoretical basis, is popular and particularly suitable for young learners from the age of four upwards (Maloney et al., 2010; Utting et al., 2010). In this way, the paper answered the following research question:

Which CT skills can be acquired via Scratch by K-9 learners, given the empirical evidence?

This paper presented a systematic review that synthesized 55 empirical research studies. It qualitatively and quantitatively analyzed the CT skills that could be obtained through Scratch in K-9, based on previous evidence. The review adopted Brennan and Resnick's (2012) framework as the basis for defining and identifying the CT skills found in empirical studies. The major findings were related to the CT skills that K-9 students could acquire through Scratch, taking into account the progression of learning. Additional CT skills that were not captured by Brennan and Resnick's framework were also identified, including input/output, reading, interpreting, and communicating code, using multimodal media, predictive thinking, and human-computer interaction. These additional CT skills can be considered possible supplements to Brennan and Resnick's framework. The paper also discussed the difficulties in terms of the assessment and progression of the identified skills. The proposed extensions to the framework need to be validated further through both research and practice. The main contribution of this paper is that it systematically reviews the CT skills that can be obtained through Scratch, thus providing certain answers to the questions of "what to teach" and "what can be learned" in K-9 CT education.

Research Article 2

Nouri, J., Zhang, L. C., Mannila, L., & Norén, E. (2019). Development of computational thinking, digital competence, and 21st-century skills when learning to program in K-9. *Education Inquiry, 1*(17), 1–17.

An overwhelming majority of prior research on teaching and learning programming has examined the higher education context (Heintz et al., 2016). Programming as part of K-9 education is, however, much more recent, as is the corresponding research. A review of CT in K-12 education (Grover & Pea, 2013) highlighted that most of the research at lower levels of education has so far focused on definitions and tools for supporting CT, leaving significant gaps when it comes to empirical studies. Especially, studies from teachers' perspective are scarce (Portelance & Bers, 2015). To address the two gaps mentioned above, this paper investigated the following research question:

From the teacher's perspective, what skills do pupils obtain through programming activities?

To answer this research question, semi-structured interviews were conducted with 19 teachers who had been teaching programming. These teachers were selected based on their experience in integrating programming into their subjects. The transcripts of the interviews were analyzed using thematic analysis (Clarke & Braun, 2006), which led to the identification of three themes for skills related to CT and five for general skills. The identified CT skills corresponded well with the dimensions of CT proposed in Brennan and Resnick's framework (2012), namely computational concepts, computational practices, and computational perspectives. In addition to these CT skills, the thematic analysis also identified some general skills related to digital competence and 21st century skills, i.e., language skills (including programming language literacy), cognitive skills and attitudes, collaborative skills and attitudes, and creative problem-solving skills and attitudes. It also revealed some aspects of the teachers' lack of CT competence. Although all the interviewees had at least one year of experience in teaching programming, they had a limited programming vocabulary. In some cases, they were unaware of the CT skills involved. The main contribution of this paper is that it enriched the empirical data related to research into CT education in the K-9 setting and collected evidence on the development of CT skills through programming. Moreover, the focus on teachers' perspective helped to broaden the body of knowledge on teachers who have a key role in integrating CT into compulsory education.

Research Article 3

Zhang, L. C., Nouri, J., & Rolandsson, L. (2020). Progression of computational thinking skills in Swedish compulsory schools with block-based programming. In *Proceedings of the 22nd Australasian Computing Education Conference*, (pp. 66–75), February 3–7, 2020. Melbourne, Australia. Australian Computer Society, Inc. ACM

Since many countries have made CT compulsory in K-9 education, much research has been done on utilizing the prevalent block-based programming languages to teach CT in K-9. However, there is a lack of research on the optimal methods of assessing CT in primary schools (Moreno-León et al., 2017). One major obstacle to assessing learning in CT is the absence of a clear and reliable progression trajectory, since without a logical and coherent progression, teachers will find it difficult to teach and assess CT. A progression can guide and assist CT instructors in planning and designing their lessons, as it can pinpoint and make inferences about evidence collected on students' performance and any misunderstanding of specific aspects. It further serves as a necessary precursor to modifying instructions to help students to address that particular misunderstanding and advance their learning (Furtak, 2012). However, in order to establish such a progression, it is first necessary to determine which CT skills are actually engaged in daily teaching. Hence, this study answered the following research question:

Which CT skills are taught and assessed by the teachers, and how do these CT skills progress throughout compulsory education?

In a research and development project, 31 primary school teachers were organized into groups and performed 12 lesson studies (Fernandes & Yoshida, 2012) related to the teaching and assessing of CT, with approximately 500 students. During the lesson study cycles, the groups of teachers produced rich documentation on their lesson studies. This paper qualitatively analyzed this documentation and extracted the principles of how CT skills were taught and assessed with BBPLs. A hybrid approach towards thematic analysis was adopted (Fereday & Muir-Cochrane, 2006) that incorporated both a deductive method, with an a priori template of codes, and a data-driven inductive approach that allowed themes outside of the template to emerge. The deductive part applied Zhang and Nouri's (2019) framework for identifying CT skills. The results revealed the extent of involvement and the progression of these skills. The paper also discussed the problems and challenges identified in the assessment. An inventory of all of the identified CT skills and the establishment of the progression of these skills form the main contribution of this paper. It provides a base for further validation and discussion of the learning trajectory, and can be utilized by teachers to guide their instruction. Moreover, this study can enable teachers to design their curriculum in a collegial way, based on their current CT competence.

Research Article 4

Zhang, L. C., & Nouri, J. (2019). Assessing K-9 teachers' computational thinking skills through a computational thinking test. In Keengwe, J., & Wachira, P. (Eds.), *Handbook of Research on Integrating Computer Science and Computational Thinking in K-12 Education*. Hershey, Pennsylvania: IGI Global.

Although CT has recently become obligatory in the K-9 curriculum, most in-service teachers are unfortunately not equipped with an adequate CK of CT. To enable CT to permeate more subjects at the K-9 level, it is essential to provide all teachers with an adequate knowledge of CT (Yadav et al., 2017). Which CT skills can be taught to the students and assessed depends on the CT competence of teachers, who cannot assess aspects that they are not aware of or are not capable of delivering. In brief, the majority of in-service teachers are currently in dire need of support to develop their CK in CT. To point out the direction of development, teachers need to measure their CT competence first. Since the topic has become mandatory, individual teachers may have established different paces and paths for their professional development for various reasons, for example related to their background. Mapping the CT competence of teachers can enhance the efficiency of their development. This paper therefore provided teachers with a test to measure their CT competence, which can help them to identify specific areas for professional development and may assist the school management to plan teacher training in a strategic way. The implementation of the curriculum also inevitably brings teachers' background to the forefront. Both teachers and school management can benefit from an investigation of whether any factors related to a teacher's background can facilitate or hinder their development of CT competence. Hence, this study investigated the following research questions:

- *What level of CT competence do Swedish K-9 teachers possess, and which areas require improvement?*
- *Which factors related to the teachers' backgrounds are associated with the development of CT competence?*

In this paper, we constructed and administered a CT test to answer the first research question. The item design for the CT test was underpinned by Brennan and Resnick's framework (2012), and was validated by an expert review panel and a principal component analysis. This test focused on the

teachers' abilities at the three lower levels of Bloom's taxonomy (Krathwohl, 2002), i.e., remembering, understanding and applying specific CT skills. Each of these skills was tested at both the conceptual and procedural levels (Rittle-Johnson & Alibali, 1999), and the progression of CT skills was also incorporated in the test. Descriptive statistics was employed to evaluate the teachers' understanding of CT, while inferential statistical analyses were applied to examine the relationships between the teachers' background factors, their CT test scores, and their self-reported ability to teach CT. Quantitative analyses were performed in SPSS (Version 26). The results revealed the teachers' proficiency in different types of CT skills, and showed that the types of programming language mastered by teachers were associated with both their CT test scores and their self-reported ability to teach CT. This paper makes two contributions: firstly, it fills a research gap by constructing and validating a CT test for teachers, and more importantly, it reveals the teachers' current level of CT competence.

Synthesis of Sub-studies and Discussion

This section synthesizes the results of the four sub-studies. The synthesis forms a coherent whole that allows us to answer the two research questions and discuss related issues.

> 1. *Based on the current status of teachers' CT competence, what CT skills have been taught and assessed by the teachers with BBPLs in compulsory education, and how do these skills progress?*

Figure 7 presents the CT skills that were taught and assessed by teachers in this case study (shown within the yellow circle), with reference to Brennan and Resnick's framework (shown within the blue circle). The red circle shows the total inventory of CT skills generated by the systematic literature review. The CT skills identified in this case study do not precisely correspond to Brennan and Resnick's framework. Although all of the concepts and practices of CT in Brennan and Resnick's framework were identified, the perspectives were not found. However, as shown in Figure 7, the CT skills identified in this case study did capture the additional CT skills in the extended framework (Zhang & Nouri, 2019.).

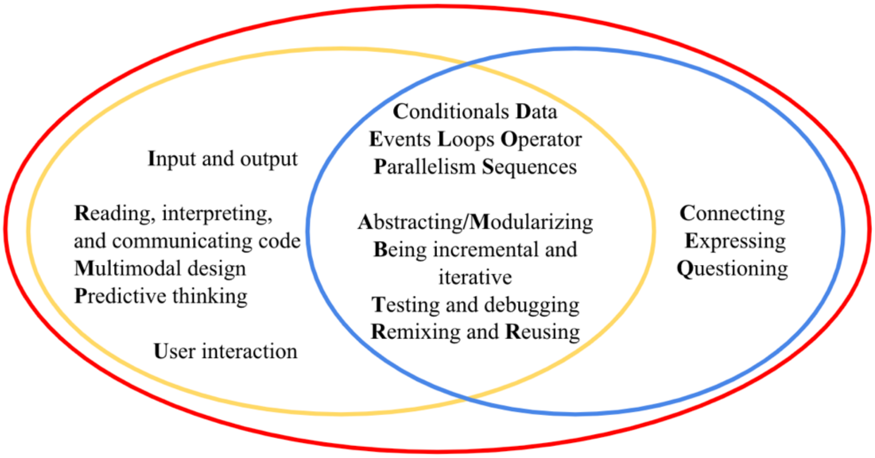

Figure 7. Comparison of CT skills.

Although the teachers applied a variety of CT skills, these skills were not given equal weight in practice, and this case study revealed the popularity of each CT skill. As shown in Figure 8, the primary focus was on CT concepts, with CT practices as a secondary focus and CT perspectives practically absent. Moreover, within each dimension of CT, some skills were paid more attention than others. In terms of CT concepts, the 19 teachers in Sub-study 2 had experience of teaching programming and were teaching Grades 1 to 9, and frequently mentioned sequences, loops and conditionals; they mentioned events and data less often, which were also identified by the teachers in students' programming artifacts. Although more CT skills were engaged by the teachers in Sub-study 3, certain CT concepts were still less prevalent than others. Events and parallelism were only introduced in Grades 1 to 3, and were absent in Grades 4 to 9. With respect to CT practices, testing and debugging were highlighted most often through the nine grades, followed by reusing and remixing. Abstracting was barely mentioned, and modularizing was rare in the programming artifacts examined in the case study. Noticeably, CT perspectives were overlooked. The only explicit mention of "Expressing" was made by some teachers in the interviews in Sub-study 2.

CT concepts	**CT practices**
Conditionals	Testing and debugging
Loops	Reusing and remixing
Sequences	Multimodal design
Data	Being incremental and iterative
Operators	Reading, interpreting, and communicating code
Event	Predictive thinking
Input and output	Abstracting and modularizing
Parallelism	

Figure 8. Heatmap of CT skills involved in the case study (the higher the number of grades in which a skill was taught, the darker the color)

A preference for certain CT skills was also seen in the systematic literature review. Table 6 shows the frequency of each the CT skills identified in the review. In a similar way to the results described above, the concepts of CT were given much more attention than the practices and perspectives. On average, each CT concept was assessed 18 times, each CT practice seven times, and each CT perspective four times. Overall, the popularity of each of the different CT concepts in the case study agreed well with the frequencies found in the review, with one exception: parallelism had somewhat higher frequency in the review, but a relatively low focus in general. Other disparities were observed between the case study and the review in terms of CT practice. While testing and debugging were commonly mentioned in the case study, reading, interpreting, and communicating code topped the list in the review. Another distinct difference is that abstracting and modularizing were assessed much more often in the review than in this case study, which reusing and remixing gained much stage in the Swedish context than in its international counterpart. Of the CT perspectives, user interaction was assessed most often, followed by expressing. One explanation is some previous studies evaluated students' programming artifacts with Dr. Scratch and user interaction is a measurement component in Dr. Scratch.

Table 6. Frequency of CT skills

CT Concepts	**CT Practices**	**CT Perspectives**
Loops (28)	Reading, interpreting, and communicating code (15)	User interaction (7)
Sequences (26)	Abstracting and modularizing (9)	Expressing (5)
Conditionals (24)	Testing and debugging (8)	Connecting (2)
Variable (20)	Multimodal design (6)	Questioning (2)
Operators (12)	Reusing and remixing (4)	
Parallelism (12)	Predictive thinking (3)	
Events (12)	Being incremental and iterative (2)	
Input and output (8)		

*Frequency is shown in brackets.

A number of factors may be associated with this uneven engagement with the different dimensions of CT. First, with regard to the near absence of CT perspectives from the assessment, it should be noted that Brennan and Resnick highlighted this in the introduction to their framework. Alongside their suggestions for evaluating the acquisition of CT skills, they expressed concerns that CT perspectives are more implicit, and hence harder to assess, regardless of the method of assessment used, whereas CT concepts and practices can be revealed and assessed more directly. As they put it, "None of the three approaches were particularly effective for understanding changes in computational thinking perspectives. It is challenging to explicitly ask a Scratcher how participation in an activity like programming with Scratch has contributed to a shift in understanding oneself or the world" (Brennan & Resnick, 2012, p. 22).

However, this does not fully explain why these aspects have been overlooked in teaching and lesson planning. Teachers in the case study had the opportunity to assess CT perspectives, and cooperative learning was either deliberately adopted by teachers or automatically and spontaneously shown by the students in this study. Sub-study 2 identified collaborative skills in terms of collaborative problem-solving skills, pedagogic communication skills, and sharing and building on others' work. Collaborative learning also predominated in Sub-study 3, as it appeared in 70% of the lesson studies. Teachers could have taken advantage of these occasions to assess CT perspectives. As they collaborated, the students used each other as learning resources and connect to each other, creating an atmosphere in which the students could debug and test each other's programs. It is likely that they questioned themselves and each other during the creation and testing of their peers' programs. During these processes, they switched between the roles of creators and consumers, and interacted with each other accordingly. This could help individuals to keep externalizing their ideas and identities, and to express themselves. This study therefore recommends that teachers should plan their lessons based on a collaborative learning approach and should use observation as a tool to assess students' development in terms of CT perspectives. More importantly, the teacher needs to discuss CT perspectives and ensure that students understand that programming involves more than constructing a bug-free program. As pointed out in the background section, Scratch was developed to promote the social aspects of programming.

Regarding the unequal attention given to the concepts of CT, it is possible that these concepts may require different levels of cognitive ability. Some of the simpler ones may have been easier to teach and assess, and hence practiced more. For instance, as shown by the results of the review, data and operators are generally harder to understand, and this may be why they were only introduced in Grades 4–9, while sequences, conditionals and loops were introduced earlier in Grades 1–3. However, the issue of cognitive difficulty does not seem applicable to parallelism, since this more complicated concept was seen in lesson plans for lower grades, but was absent from Grades 4–9. The reason for this may be related to another factor that can influence the acquirement of CT skills, namely the teacher's competence in CT. This will be discussed further in the answer to Research Question 2. The predecessors of Scratch were developed to enable parallelism (see the section entitled "History of Programming Tools for Children"). In the same vein, Scratch is a powerful tool for engaging parallel processes, and this deserves to be emphasized. Another CT concept that requires more attention relates to events. Although this concept may appear to be straightforward, such as in the statement "when the green flag is clicked", events are essential for every Scratch project. Without them, a project would not be able to begin (except by running scripts manually), and this should not be treated as being too simple for the older students and neglected. Events can progress as much as loops and conditionals do, and will be discussed further in the context of progression. As for the concepts of input and output, these were mainly taught in technology lessons, as programming is integrated into math and technology in Sweden, especially when combined with hardware such as micro:bits. While it happens more naturally in the subject of technology, it is also possible for math teachers to angle them in math lessons. It was unfortunate that some lesson plans for mathematics on functions did not apply the ideas of input and output; this may be because they did not fit with the goals of these lessons, or because there was a lack of space in an already packed lesson plan. Alternative, it may simply be because some teachers were not aware of the concepts of input and output in CT and programming.

The preferences for CT practices in this case study did not align with the results of the review. For instance, although abstracting/modularizing were frequently assessed CT skills in the review, these were the CT practices mentioned least often in the case study. Those CT practices that were overlooked should be given more attention. Being incremental and iterative is a natural way of programming; when a student debugs and tests a program, this is an inherently iterative process, since several iterations of testing are frequently required to find all the bugs. Hence, the program is improved incrementally. In addition, while teachers favored remixing and reusing, those parts being remixed or reused are possibly good examples of demonstrating what a module is. Preferences for CT practices were mirrored by the teachers' levels of CT competence, and a more detailed discussion of this issue will be presented for Research Question 2.

Two types of progression are seen in the case study: the grade in which each CT skill was introduced, and the advancement in each CT skill. Table 7 presents the grades in which the CT skills were found in this case study to be introduced. Most of the concepts and practices of CT were presented in Grades 1–3. All CT perspectives cab be fostered in Grades 1–3. CT concepts such as conditionals, loops and sequences are considered to be core programming concepts (Funke & Geldreich, 2017; Meerbaum-Salant et al., 2013) and characterize most of the solutions expressed in any imperative language (Grover et al., 2015). They may be necessary to allow students to create meaningful and interesting programming artifacts; however, due to the necessity of these particular CT skills for every program, the teacher may give more attention to the neglected CT skills. Since sequences, conditionals, and loops will probably form part of every program constructed by students, promoting underrepresented CT skills is one way to help students in their overall progress in CT skills. Table 7 shows that students in Grades 1–3 could apply these CT practices. There are also abundant opportunities to cultivate CT perspectives, as these are required for the creation of all programming artifacts. Table 7 differs from the recommendations given in Sub-study 4 in terms of input, output and user interaction. Although input and output were found in lesson plans for Grades 7–9, they had already been introduced in Grades 1–3 in the Swedish Technology curriculum.

A comparison of the progression between the case study and the review (Table 8) shows that more CT skills were integrated by the Swedish teachers in Grades 1–3. The review found that most CT skills engaged in older ages as Grades 4–6. There are certain consistencies; for example, both tables suggest that data and operators are applied at later stages.

While Table 7 implies that many CT skills can be taught at an early age, it does not describe the advancement in each CT skill, which forms the second type of progression. Both the case study and the review showed that the level of difficulty of the CT concepts can be differentiated. Table 4 in Sub-study 3 suggested a progression scheme for CT concepts consisting of three levels: introductory, intermediate and advanced. More details of the possible progression in each CT concept can be found in the review and Sub-study 3.

Table 7. Grades in which CT skills were introduced, as identified in the case study

Grades	**CT Concepts**	**CT Practices**	**CT Perspectives**
1 to 3	Conditionals Events Input and output Loops Parallelism Sequences	Being incremental and iterative Testing and debugging Reusing and Remixing Reading, interpreting, and communicating code Multimodal design	Connecting Expressing Questioning User interaction
4 to 6	Data Operators	Abstracting/modularizing Predictive thinking	-
7 to 9	-	-	-

Table 8. Progression based on the systematic literature review

Grades	CT Concepts	CT Practices	CT Perspectives
K-3	Sequences Events	Being iterative and incremental Debugging Predictive thinking Reading, interpreting and communicating the code	Connecting Expressing Questioning
4–6	Conditionals Data Events Input/output Loops Operators Parallelism Sequences	Abstracting and modularizing Being iterative and incremental Multimodal design Reading, interpreting and communicating the code Reusing and Remixing Testing and debugging	Connecting Expressing Questioning User interaction
7–9	Conditionals Data Events Input/output Loops Operators Parallelism Sequences	Abstracting and modularizing Being iterative and incremental Multimodal design Predictive thinking Reading, interpreting and communicating the code Reusing and Remixing Testing and debugging	Expressing User interaction

In regard to Tables 7 and 8, it should be noted that both should be interpreted in general rather than absolute terms. For example, although the review shows evidence for the assessment of loops in Grade 4–6, this does not exclude the possibility that students on Grades 1–3 can handle loops; in fact, the case study demonstrated that they could. These progression tables should therefore be used as a starting point for future research and for systematic refinement. More importantly, the main purpose of these findings is to help teachers determine what to teach, given that they have had hardly any training in the subject. The answer to this research question can support teachers in planning, teaching and assessing CT skills.

2. What is the teachers' level of CT competence, and how does it influence the integration of CT?

The teachers' level of CT competence was measured directly in Sub-study 4, and was also revealed to some extent in Sub-studies 2 and 3. Sub-study 4 showed that 85.5% of the teachers had never completed a course in programming, and 80% had never programmed or had scant programming experience. A survey conducted by the Swedish National Union of Teachers in 2017 (Lärarnas Riksförbund, 2017) shows that these numbers were unchanged from those of 2019. The results also showed that the mean score for the respondents was 11 out of a maximum of 20 points, and the CT test scores were associated with the types of programming language mastered. In other words, teachers who mastered TBPLs gained more points than BBPLs. More than 40% could not distinguish CT from other thinking abilities. When the teachers were asked in the CT test what CT was, 19% answered coding, and 18% thought it was arithmetic. The heat map in Sub-study 4 (left-hand column in Figure 9) shows that although the teachers were proficient in sequences, testing and debugging, they lacked a solid understanding of data, operators, parallelism and being incremental and iterative.

Teachers' CT competence	**CT skills being taught and assessed**
CT Concepts	
Sequences	Conditionals
Conditionals	Loops
Events	Sequences
Loops	Data
Data	Operators
Operators	Event
Parallelism	Parallelism
CT Practices	
Testing and debugging	Testing and debugging
Reusing and remixing	Reusing and remixing
Abstracting and modularizing	Being incremental and iterative
Being incremental and iterative	Abstracting and modularizing

Figure 9. Comparison between a heat map of the teachers' level CT competence as measured by the CT test (left) and the results in Figure 8 (right)

The case study revealed that it was difficult for students to apply data, parallelism, and abstracting/modularizing. This result may not be related solely to the cognitive level of the students, since the teachers also gained low scores on these items in the test. The teachers' results from the CT test corresponded to those of Sub-study 2, in which teachers most frequently named sequences, loops and conditionals in the interviews and did not mention parallelism or operators. One possible reason for this is that most of the teachers did not have the language to express the formal names of the CT concepts, although they explored them together with the pupils. Some of these teachers were not aware of these CT concepts at all, and did not know that they were important in programming, and were therefore unable to identify them in students' artifacts. The results of the teachers' CT tests were also reflected in Sub-study 3, which showed that some teachers may not be aware of the concept of modularizing in programming, despite the fact that it was central to the lesson plan. Furthermore, testing and debugging was applied throughout the nine grades, while being incremental and iterative was only found in lesson plans for Grades 1–3.

The resemblance between the left- and right-hand columns in Figure 9 indicates that the CT competence of teachers influences their teaching. It was obvious that the teachers' CT competence determined which CT skills they taught and assessed; a CT skill was more likely to be applied by a teacher if he or she was proficient in that skill. The unfamiliarity of teachers with certain CT skills resulted in not being able to identify them in the assessment, and hence failing to provide accurate and constructive feedback to students. Additionally, the CT competence of teachers determined the students' progression in CT skills. As argued in Sub-study 4, one possible reason for the prevalence of reusing and remixing is that the teachers' limited knowledge of CT and programming prevented the construction of programs from scratch, meaning that they had to resort to online resources offering ready-to-use programs. A deep understanding of all of the details of those programs may not be required before a teacher can reuse and remix them in a superficial way. If the teachers' understanding of CT is not improved, it is likely that students will only be exposed to CT in a superficial way. In view of this, the progression in Table 7 should be interpreted with caution. This table was the result of a collective effort by teachers, rather than at an individual level. If a CT skill is shown to be applied in Grades 1 to 9, this does not necessarily guarantee a meaningful progression in that skill. For instance, conditionals are applied throughout all nine grades, but this does not mean that the conditionals that are taught and assessed in Grade 9 are more advanced than those in Grade 6, although they should be. The progression is heavily dependent on the teacher's CT competence in terms of orchestrating the lesson planning and carrying out assessments. Only a mastery of CT skills can allow teachers to facilitate progression by students and to make sound assessments of this progression.

One drawback of Scratch is that it lacks a mechanism for providing feedback to users. Kazimoglu et al. (2012, p. 524) state that "students might create output that works by designing an inefficient programming strategy, such as a statement repeated lots of times without using a loop, because they do not possess the requisite level of knowledge to develop a better solution." As powerful as Scratch is, it can only execute what the students' algorithms dictate, and this highlights the importance of teachers possessing adequate CT competence. Teachers with limited CT competence may not capture the essence of good programming practices, and hence mislead students to linear way of practice.

Last but not least, it should be noted that programming is not equivalent to CT, as suggested by the title of the case study, “CT with Programming”. This should be made clear to teachers during their professional development. As teachers strive to become more proficient in programming, a focus on programming code in teaching practice may undermine computational ways of solving problems, such as modeling through abstraction, decomposing a problem into manageable modules, working in an incremental and iterative manner, and reusing patterns. Programming is the means of achieving CT.

Conclusion

To investigate how teachers are integrating CT into compulsory education in Sweden without adequate CT competence, an exploratory embedded single-case study was conducted. This case study was performed as part of a national R&D project involving teachers. In total, four sub-studies were carried out to investigate two core elements of this integration: the CT skills that were being taught and assessed via Scratch, and the teachers' level of CT competence. Both qualitative and quantitative methods were applied in an analysis of multiple sources of data. Three conclusions could be drawn. Firstly, a wide range of CT skills, in terms of the concepts, practices and perspectives of CT, have been taught and assessed using Scratch in Swedish schools. However, these three CT dimensions have not received equivalent levels of attention: the primary focus was on CT concepts, while CT perspectives were almost neglected. Furthermore, within each dimension of CT, skills were not practiced equally. A possible and serious consequence of this is the uneven development of students' CT skills. This case study discovered progressions of these CT skills, which recommends the time/grades for introducing and advancing the CT skills. However, these progression schemes are still in relative terms, and more systematic efforts are required. The case study surveyed teachers' current CT competence, and found that there was a great deal of room for improvement. Teachers need to sharpen their CT skills, especially in relation to loops, data, operators, parallelism, abstracting and modularizing, and being incremental and iterative. It was notable that their competence developed over the course of the R&D project, especially since most of them had never undergone any relevant training before. A teacher's level of CT competence is related to the CT skills he or she can teach and assess in lessons, and skills that the teachers are unfamiliar with are usually not promoted in teaching and assessment. The limited CT competence of teachers can therefore hinder the students' acquirement of CT skills.

The significance of this study is twofold. Firstly, this work contributes to the establishment of the body of knowledge regarding in-service teachers in CT education research. Most of the current research has focused on the students' perspective, and studies from the teachers' point of view have been scarce. By focusing on what teachers choose to or are able to teach and their competence in CT, this study brings the teachers' perspective to the forefront.

This study also has certain practical implications. Firstly, this study examined and evaluated important aspects of the integration of CT and programming in Swedish compulsory education. To the best of the author's knowledge, no previous research has provided insights into this issue since the new curriculum went into effect. This study therefore updates stakeholders (namely teachers, students, parents, school management, and policy makers) for the first time with valuable details of this integration. Secondly, due to the nature of a case study, the findings in this study can be directly applied in practice. Since the focus was on the teachers' perspective, the results can assist teachers in planning instructions and designing assessments, pinpointing CT skills that require improvement, and directing future professional development. Furthermore, each of the four sub-studies contributed to the advancement of CT education research in terms of their specific perspective and purpose.

Based on the results and limitations of this study, the following areas require future research efforts. Firstly, future work should focus on strengthening the coherence of the progression. All the CT skills identified here, with variations in the level of difficulty, should be taught at all grades to achieve a systematic and reliable progression. To be able to deliver the new curriculum effectively, teachers need to continuously improve their CT competence, and future studies should continue to evaluate this competence. As this study focused solely on teachers' CK of CT, future work should investigate the PK and PCK of CT. In addition, this study investigated only public schools, and should be complemented by an investigation of private schools. The new curriculum should also be evaluated after implementation, in terms of teaching and learning outcomes. Future work should therefore investigate the ways in which CT and programming influence students' problem-solving ability, for example. An exploration of programming language transitions could also be of value, for instance among different types of BBPLs, such as BBPLs and TBPLs. Research in this area should investigate the practices that can best support these transitions, the challenges, and the progression from one grade to the next, by examining the use of different programming languages and the transitions between them.

References

ACARA (Australian Curriculum, Assessment and Reporting Authority). (2012): The Shape of the Australian Curriculum: Technologies. Sydney, NSW: ACARA.
[Accessed on 2020-03-16.
https://docs.acara.edu.au/resources/Shape_of_the_Australian_Curriculum.pdf]

Alimisis, D. (2014). Educational Robotics in Teacher Education: an Innovative Tool for Promoting Quality Education, *in Daniela, L., Lūka, I., Rutka L., & Žogla, I. (Eds). Teacher of the 21st Century: Quality Education for Quality Teaching*. p.14–27. Cambridge Scholars Publishing.

Almutairi, A. F., Gardner, G. E., & McCarthy, A. (2014). Practical guidance for the use of a pattern-matching technique in case-study research: A case presentation. *Nursing & health sciences, 16(2),* 239–244.

Angeli, C., & Jaipal-Jamani, K. (2018). Preparing Pre–service Teachers to Promote Computational Thinking in School Classrooms. In *Computational thinking in the STEM disciplines* (pp. 127–150). Springer, Cham.

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn–Smith, J., & Zagami, J. (2016). A K–6 computational thinking curriculum framework: Implications for teacher knowledge. *Journal of Educational Technology & Society*, 19(3), 47–57.

Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems*, 75, 661–670.

Avison, D. E., Lau, F., Myers, M. D., & Nielsen, P. A. (1999). Action research. *Communications of the ACM*, 42(1), 94–97.

Barr, V. (2014). In Gonzalez, T., Diaz-Herrera, J., & Tucker, A. (Eds.). *Computing handbook: Computer science and software engineering* . (Vol. 1). CRC Press.

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

Bell, T., Andreae, P., & Lambert, L. (2010). Computer science in New Zealand high schools. In *Proceedings of the 12th Australasian Conference on Computing Education*, (pp. 15-22). January 15–22. Brisbane, Australia. Australian Computer Society. ACM

Bell, T., Duncan, C., & Rainer, A. (2017). What is coding?. In *Humble, S. (Ed.) Creating the Coding Generation in Primary Schools*. Routledge.

Berland, M. & Wilensky, U. (2015). Comparing virtual and physical robotics environments for supporting complex systems and computational thinking. *Journal of Science Education and Technology*,24 (5), 628–647.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). *Developing computational thinking in compulsory education–Implications for policy and practice* (No. JRC104188). Joint Research Centre (Seville site).

Boddy, J., Neumann, T., Jennings, S., Morrow, V., Alderson, P., Rees, R., & Gibson, W. (2010). The research ethics guidebook: a resource for social scientists. The research ethics guidebook: a resource for social scientists.
[Accessed on 2020-04-30.
http://www.ethicsguidebook.ac.uk]

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

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association* (pp. 1–25), April 13–17, 2012. Vancouver, Canada.

Brereton, P., Kitchenham, B., Budgen, D., & Li, Z. (2008). Using a protocol template for case study planning. In the *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering* (pp. 1–8). June 26 – 27, 2008. Bari, Italy. ACM

Brown, A. L. (1992). Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings. *The journal of the learning sciences*, 2(2), 141–178.

Brown, N. C. C., Kölling, M., Crick, T., Peyton Jones, S., Humphreys, S., and Sentance, S. (2013). Bringing computer science back into schools: Lessons from the UK. In *Proceedings of the 44th ACM Technical Symposium on Computer Science Education* (SIGCSE'13). pp. 269–274. March 6–9, 2013. Denver, USA. ACM

Bundy, A. (2007). Computational thinking is pervasive. *Journal of Scientific and Practical Computing,* 1(2), 67–69.

Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X., & Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Computers & Education*, 109, 162–175.

Computing at School. (2016).
[Accessed on 2020-04-30. https://www.computingatschool.org.uk/about]

CSTA & ISTE (2011). Computational Thinking in K–12 Education leadership toolkit. Computer Science Teacher Association.
[Accessed on 2020-04-30.
http://csta. acm. org/Curriculum/sub/CurrFiles/471.11 CTLeadershipt Toolkit-SP-vF. pdf]

Dasgupta, S., Hale, W., Monroy-Hernández, A., & Hill, B. M. (2016). Remixing as a pathway to computational thinking. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing* (pp. 1438–1449). February 27– March 02, 2016. San Francisco, USA. ACM.

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM,* 60(6), 33–39.

Department for Education. (2013): The national curriculum in England. Cheshire, UK: Crown.
[Accessed on: 2020-03-20.
https://www.gov.uk/government/publications/national–curriculum–in–england–primary–curriculum]

Duncan, C., Bell, T., & Tanimoto, S. (2014). Should your 8-year-old learn coding?. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education* (pp. 60–69). November 5–7, 2014. Berlin, Germany. ACM.

Elliot, C. (2007). Action research: Authentic learning transforms student and teacher success. *Journal of Authentic Learning*, 4(1), pp 34–42.

Elo, S., & Kyngäs, H. (2008). The qualitative content analysis process. *Journal of advanced nursing*, 62(1), 107–115.

Falkner, K., Vivian, R., & Falkner, N. (2014). The Australian digital technologies curriculum: challenge and opportunity. In *Proceedings of the 16th Australasian Computing Education Conference* (pp. 3–12). January 20–23, 2014. Auckland, New Zealand. Australian Computer Society, Inc. ACM

Falloon, G. (2016). An analysis of young students' thinking when completing basic coding tasks using Scratch Jnr. On the iPad. *Journal of Computer Assisted Learning*, 32(6), 576–593.

Ferdig, R. E. (2001). Psychological investigations of educational explorations with technology: understanding what makes a "good innovation". Paper presented at *the American Educational Research Association Annual Meeting* (AERA). Seattle, WA.

Fereday, J., & Muir-Cochrane, E. (2006). Demonstrating rigor using thematic analysis: A hybrid approach of inductive and deductive coding and theme development. *International journal of qualitative methods*, 5(1), 80–92.

Fernandez, C. (2002). Learning from Japanese approaches to professional development: The case of lesson study. *Journal of Teacher Education*, 53(5), 393–405.

Fernandez, C., & Yoshida, M. (2012). *Lesson study: A Japanese approach to improving mathematics teaching and learning*. Routledge.

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education*, 63, 87–97.

Flyvbjerg, B. (2006). Five misunderstandings about case-study research. *Qualitative inquiry*, 12(2), 219–245.

Funke, A. & Geldreich, K. (2017). Gender differences in scratch programs of primary school children. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education* (pp. 57–64). November 8–10, 2017. Nijmegen, Netherlands. ACM.

Furtak, E. M. (2012). Linking a learning progression for natural selection to teachers' enactment of formative assessment. *Journal of Research in Science Teaching*, 49(9), 1181–1210.

Gadanidis, G. (2015). Young children, mathematics, and coding: A low floor, high ceiling, wide walls environment. In *Cases on Technology Integration in Mathematics Education* (pp. 308–329). IGI Global.

Gadanidis, G., Cendros, R., Floyd, L., & Namukasa, I. (2017). Computational thinking in mathematics teacher education. *Contemporary Issues in Technology and Teacher Education*, 17(4), 458–477.

Gal-Ezer, J., & Stephenson, C. (2010). Computer science teacher preparation is critical. *ACM Inroads*, 1(1), 61–66.

Gal-Ezer, J., & Stephenson, C. (2014). A tale of two countries: Successes and challenges in K–12 computer science education in Israel and the United States. *ACM Transactions on Computing Education*, 14(2), 1–18.

Gibbert, M., Ruigrok, W., & Wicki, B. (2008). What passes as a rigorous case study?. *Strategic management journal*, 29(13), 1465–1474.

Ginsberg, M.B. (2011). *Transformative professional learning: A system to enhance teacher and student motivation.* Thousand Oaks, CA: Corwin Press.

Good, J., & Howland, K. (2017). Programming language, natural language? Supporting the diverse computational activities of novice programmers. *Journal of Visual Languages & Computing*, 39, 78–92.

Grover, S., & Pea, R. (2013). Assessing computational thinking in K–12: A review of the state of the field. *Educational Researcher*, 42 (1), 38–43.

Grover, S., Pea, R. & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education*, 25(2), 199–237.

Gunion, K. (2010). Fundamentals of CS: designing and evaluating computer science activities for kids. Master's thesis, University of Virginia.

Gunion, K., Milford, T., & Stege, U. (2009). Curing recursion aversion. *ACM Special Interest Group on Computer Science Education Bulletin*, 41(3), 124–128.

Heintz, F., Mannila, L., Nordén, L. Å., Parnes, P., & Regnell, B. (2017). Introducing programming and digital competence in Swedish K–9 education. In the *Proceedings of the 10th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives* (pp. 117–128). November 13–15, 2017. Helsinki, Finland. Springer.

Heintz. F., & Mannila, L. (2018). Computational Thinking for all – an experience report on scaling up teaching Computational Thinking to all students in a major city in Sweden. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (SIGCSE' 18). pp. 137–142. February 21–24, 2018, Baltimore, USA.

Heintz. F., Mannila, L., & Färnqvist, T. (2016). A review of models for introducing Computational Thinking, Computer Science and Computing in K–12 education. In the *Proceedings of 2016 IEEE Frontiers in Education Conference*. (pp. 1–9). October 12 – 15, 2016. Eire, USA. IEEE.

Hsu, T. C., Chang, S. C., & Hung, Y. T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education*, 126, 296-310.

Hubwieser, P. (2012). Computer science education in secondary schools: The introduction of a new compulsory subject. *ACM Transactions on Computing Education*, 12 (4), 1–41.

Hubwieser, P., Magenheim, J., Mühling, A., & Ruf, A. (2013). Towards a conceptualization of pedagogical content knowledge for computer science. In *Proceedings of the 9th annual international ACM conference on International computing education research* (pp. 1–8). August 12–14, 2013, San Diego, USA. ACM

Israel, M., Pearson, J. N., Tapia, T., Wherfel, Q. M., & Reese, G. (2015). Supporting all learners in school–wide computational thinking: A cross-case qualitative analysis. *Computers & Education*, 82, 263–279.

Iversen, O. S., Smith, R. C., & Dindler, C. (2018t). From computational thinking to computational empowerment: a 21st century PD agenda. In *Proceedings of the 15th Participatory Design Conference,* (7:1–7:11). August 20–24, 2018, Hasselt and Genk, Belgium

Jacob, S., Nguyen, H., Tofel-Grehl, C., Richardson, D., & Warschauer, M. (2018). Teaching computational thinking to English learners. *The journal of New York Association of Teachers of English to Speakers of Other Languages*, 5(2). 12–24.

Johnson, A.P. (2001). *A short guide to action research*. Boston, MA: Allyn & Bacon

Kafai, Y. B., & Burke, Q. (2014). *Connected code: Why children need to learn programming*. MIT Press.

Kazimoglu, C., Kiernan, M., Bacon, L., & MacKinnon, L. (2012). Learning programming at the computational thinking level via digital game-play. *Procedia Computer Science*, 9, 522–531.

Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys* (CSUR), 37(2), 83–137.

Kitchenham, B. & Charters, S. (2007). Guidelines for Performing Systematic Literature Reviews in Software Engineering, *Technical Report EBSE 2007-001, Keele University and Durham University Joint Report.*

Kobsiripat, W. (2015). Effects of the media to promote the scratch programming capabilities creativity of elementary school students. *Procedia-Social and Behavioral Sciences*, 174, 227–232.

Kong, S. C. (2016). A framework of curriculum design for computational thinking development in K-12 education. *Journal of Computers in Education*, 3(4), 377–394.

Kong, S. C., & Lao, A. C. C. (2019). Assessing In-service Teachers' Development of Computational Thinking Practices in Teacher

Development Courses. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 976-982). February 27 – March 2, 2019. Minneapolis US. ACM.

Krathwohl, D. R. (2002). A revision of Bloom's taxonomy: An overview. *Theory into practice*, 41(4), 212–218.

KTH. (2020). Programming for teachers, with focus on technology or mathematics education.
[Accessed on 2020-05-30.
https://www.kth.se/student/kurser/kurs/LL123U]

Lärarnas Riksförbundet. (2017).
[Accessed on 2020-05-30.
https://www.lr.se/opinion--debatt/debattartiklar/2017/2017-09-08-lararna-maste-fa-lara-sig-programmera]

Le Deist, F. D., & Winterton, J. (2005). What is competence?. *Human resource development international*, 8(1), 27–46.

Littlejohn, A. H., & Stefani, L. A. (1999). Effective use of communication and information technology: bridging the skills gap. *ALT-J: Research in Learning Technology*, 7(2), 66–76.

Lu, J. J., & Fletcher, G. H. (2009). Thinking about computational thinking. In Proceedings of the 40th ACM technical symposium on Computer science education (pp. 260-264). March 3–7, 2009. Chattanooga USA. ACM.

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

Maloney, J., Resnick, M., Rusk, N., Silverman, B. & Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education*, 10(4), 16.

Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., & Settle, A. (2014). Computational thinking in K–9 education. In *Proceedings of the working group reports of the 2014 Innovation & Technology in Computer Science Education Conference* (pp. 1–29). June 23–25, 2014. Uppsala, Sweden. ACM.

Maxwell, J. A. (2005). Conceptual framework: What do you think is going on. *Qualitative research design: An interactive approach*, 41, 33–63.

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2013). Learning computer science concepts with scratch. *Computer Science Education*, 23(3), 239–264.

Merriam, S. B. (1998). *Qualitative Research and Case Study Applications in Education*. Revised and Expanded from "Case Study Research in Education.". Jossey-Bass Publishers, San Francisco, CA.

Mishra, P., & Yadav, A. (2013). Of art and algorithms: Rethinking technology & creativity in the 21st century. *TechTrends*, 57(3), 11.

Moreno-León, J., Román-González, M., Harteveld, C., & Robles, G. (2017). On the automatic assessment of computational thinking skills: A comparison with human experts. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (pp. 2788–2795). May 4 – May 9, 2019. Glasgow, UK. ACM.

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

Opetushallitu. (2020a). Programmeringsbegrepp.
[Accessed on 2020-05-30.
https://www.oph.fi/sv/programmeringsbegrepp]

Opetushallitu. (2020b). Lärandeprogression inom programmering.
[Accessed on 2020-05-30.
https://www.oph.fi/sv/programmeringsbegrepp]

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc.

Parker, K. R., & Davey, B. (2014). Computers in schools in the USA: A social history. In *Reflections on the History of Computers in Education* (pp. 203–211). Springer, Berlin, Heidelberg.

Passey, D. (2017). Computer science (CS) in the compulsory education curriculum: Implications for future research. *Education and Information Technologies*, 22(2), 421–443.

Portelance, D. J., & Bers, M. U. (2015). Code and Tell: Assessing young children's learning of computational thinking using peer video interviews with Scratch Jr. In *Proceedings of the 14th International Conference on Interaction Design and Children* (pp. 271–274). June 21–Jun 25, 2015. Medford, USA. ACM.

Price, T. W. & Barnes, T. (2015). Comparing textual and block interfaces in a novice programming environment. In *Proceedings of the 11th annual International Conference on International Computing Education Research* (pp. 91–99). Neveda, US. ACM.

Repenning, A. (1993). Agentsheets: a tool for building domain-oriented visual programming environments. In *Proceedings of the INTERACT '93 and CHI' 93 Conference Companion on Human Factors in Computing Systems* (pp. 142–143). April 24–29. Amsterdam, The Netherlands. ACM.

Repenning, A. (2017). Moving beyond syntax: Lessons from 20 years of blocks programing in AgentSheets. *Journal of Visual Languages and Sentient Systems*, 3(1), 68–89.

Resnick, M., & Silverman, B. (2005). Some reflections on designing construction kits for kids. In *Proceedings of the 2005 Conference on Interaction Design and Children* (pp. 117–122). June 8–10, 2005. Boulder, USA. ACM

Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., & Kafai, Y. (2009). Digital fluency" should mean designing, creating, and remixing, not just browsing, chatting, and interacting. *Communications of the ACM*, 52(11), 60–67.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. & Kafai, Y. B. (2009). Scratch: Programming for all. *Communication of the ACM*, 52(11), 60–67.

Rittle-Johnson, B., & Alibali, M. W. (1999). Conceptual and procedural knowledge of mathematics: Does one lead to the other?. *Journal of educational psychology*, 91(1), 175.

Román-González, M., Pérez-González, J. C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking?

Criterion validity of the Computational Thinking Test. *Computers in Human Behavior*, 72, 678-691.

Rowley, J. (2002). Using case studies in research. *Management research news*, 25(1), 16–27.

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, 14(2), 131.

Saeli, M. (2012). *Teaching programming for secondary school: A Pedagogical content knowledge based approach* (Unpublished doctoral dissertation). Technische Universiteit Eindhoven, Netherlands.

Sáez-López, J. M., Román-González, M., & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two year case study using "Scratch" in five schools. *Computers & Education*, 97, 129–141.

Sanjari, M., Bahramnezhad, F., Fomani, F. K., Shoghi, M., & Cheraghi, M. A. (2014). Ethical challenges of researchers in qualitative studies: The necessity to develop a specific guideline. *Journal of medical ethics and history of medicine*, 7, 14.

Scholz, R. W., & Tietje, O. (2013). *Types of case studies. Embedded case study methods*. Thousand Oaks, California: SAGE Publications, Inc.

Seaman, C. (1999). Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, 25(4):557–572

Selby, C. & Woollard, J. (2013) Computational Thinking: The Developing Definition.
[Accessed 23-03-2018. http://eprints.soton.ac.uk/356481].

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

Shulman, L. S. (1986). Those who understand: Knowledge growth in teaching. *Educational Researcher*, 15(2), 4–14.

Shulman, L. S. (1987). Knowledge and teaching: Foundations of the new reform. *Harvard Educational Review*, 57, 1–22.

Skolverket. (2017a). Få syn på digitaliseringen på grundskolenivå – Ett kommentarmaterial till läroplanerna för förskoleklass, fritidshem och grundskoleutbildning.
[Accessed on 2020-05-30: https://www.skolverket.se/getFile?file=3783]

Skolverket. (2020b). Kursplaner för grundskolan.
[Accessed on 2020-05-30:
https://www.skolverket.se/undervisning/grundskolan/laroplan-och-kursplaner-for-grundskolan/kursplaner-for-grundskolan]

Stager, G. S., & Martinez, S. (2017). In *Humble, S. (Ed.) Creating the coding generation in primary schools: A practical guide for cross-curricular teaching*. Routledge.

Tedre, M., & Denning, P. J. (2016). The long quest for computational thinking. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research* (pp. 120–129). November 24–27, 2016. Koli. Finland. ACM

Utting, I., Cooper, S., Kölling, M., Maloney, J., & Resnick, M. (2010). Alice, greenfoot, and scratch--a discussion. *ACM Transactions on Computing Education*, 10(4), 1–11.

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies*, 20(4), 715–728.

Weintrop, D. (2015). Blocks, text, and the space between: The role of representations in novice programming environments. In the Proceeding of the 2015 IEEE Symposium on Visual Languages and Human-Centric Computing. (pp. 301–302). October 18–22, 2015. Atlanta, USA. IEEE

Weintrop, D., & Wilensky, U. (2017). Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education*, 18(1), 1–25.

Weintrop, D., & Wilensky, U. (2019). Transitioning from introductory block-based and text-based environments to professional programming languages in high school computer science classrooms. *Computers & Education*, 142, 103646.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, 25(1), 127–147.

Williams, C., Alafghani, E., Daley, A., Gregory, K., & Rydzewski, M. (2015). Teaching programming concepts to elementary students. In *Proceeding of the 2015 IEEE Frontiers in Education Conference* (pp. 1–9). October 21–24. 2015. El Paso, USA. IEEE.

Wilson, A., & Moffat, D. C. (2010). Evaluating Scratch to Introduce Younger Schoolchildren to Programming. In the *Proceedings of 22nd Annual Workshop of the Psychology of Programming Interest Group* (p. 7). Sep 9– 21, 2010. Madrid. Spain.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33–35.

Wing, J.M. (2011), Research Notebook: Computational thinking –what and why? *The Link Magazine*, 20–23.

Wing, J.M. (2017). Computational thinking's influence on research and education for all. *Italian Journal of Educational Technology*, 25(2), 7–14.

Wolz, U., Stone, M., Pearson, K., Pulimood, S. M., & Switzer, M. (2011). Computational thinking and expository writing in the middle school. *ACM Transactions on Computing Education*, 11(2), 9.

Woodside, A. G. (2010). Bridging the chasm between survey and case study research: Research methods for achieving generalization, accuracy, and complexity. *Industrial Marketing Management*, 39(1), 64–75.

Yadav, A., Gretter, S., Good, J., & McLean, T. (2017). Computational thinking in teacher education. In *Peter Rich, P. & Hodges, C. B. (Eds.) Emerging research, practice, and policy on computational thinking* (pp. 205–220). Springer, Cham.

Yadav, A., Gretter, S., Hambrusch, S., & Sands, P. (2016). Expanding computer science education in schools: understanding teacher experiences and challenges. *Computer Science Education*, 26(4), 235–254.

Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: pedagogical approaches to embedding 21st century problem solving in K–12 classrooms. *TechTrends*, 60(6), 565–568.

Yadav, A., Krist, C., Good, J., & Caeli, E. N. (2018). Computational thinking in elementary classrooms: measuring teacher understanding of computational ideas for teaching science. *Computer Science Education*, 28(4), 371–400.

Yin, R. (2018). *Case Study Research: Design and Methods*, sixth edition, Thousand Oaks, CA: Sage Publications.

Zhang, L.C., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers & Education,* 141, 103607.

Zhong, B., Wang, Q., Chen, J., & Li, Y. (2016). An exploration of three-dimensional integrated assessment for computational thinking. *Journal of Educational Computing Research*, 53(4), 562–590.