Changing Computer Programming Education; The Dinosaur that Survived in School

An explorative study of educational issues based on teachers’ beliefs and curriculum development in secondary school

LENNAERT ROLANDSSON

Changing Computer Programming Education; The Dinosaur that Survived in School

An explorative study of educational issues based on teachers’ beliefs and curriculum development in secondary school

LENNART ROLANDSSON

Licentiate Thesis
Stockholm, Sweden 2012
This licentiate thesis consists of a synthesis of two papers, a summary in Swedish, and the following papers:


II Rolandsson, L. Teachers’ Beliefs Regarding Programming Education. Accepted for publication in Skogh & de Vries (eds.) (201x) Technology Education - Practicing Teachers Researching Teachers Practice. Series: International Technology Education Studies. Sense Publishers (Published here with kind permission.)

Department for Learning
KTH School of Education and Communication in Engineering Science
SE-100 44 Stockholm
Sweden

Typeset with \LaTeX by the author. Written in Emacs & Word.
Printed by E-print, Stockholm.

ISBN 978-91-7501-559-0
TRITA-ECE 2012-02

© Lennart Rolandsson, 2012
“Commonly teachers believe that if something is taught it is automatically learned. If it is not learned, then the problem lies in the inadequacy of the student’s ability, motivation, or persistence, not in the ineffectiveness of the instruction” (Nuthall, 2004, p. 278)
Abstract

With the intention to contribute to research in computer programming education the thesis depicts the mind-set of teachers and their beliefs in relation to the early enactment of the informatics curriculum in Swedish upper secondary school. Two perspectives are covered in the thesis. Based on original documents and interviews with curriculum developers, the enactment of the informatics/programming curriculum during the 1970s and 1980s is explored (Paper 1). This historical perspective is supplemented with a perspective from the present day where current teaching practice is explored through teachers’ statements (seminars with associated questionnaires) regarding their beliefs about teaching and learning programming (Paper 2).

The historical data reveals that experimental work within the informatics curriculum was initiated in the mid-1970s. In the early stages of the curriculum development process a contemporary post gymnasium programme in computing was used as a blueprint. The curriculum relied on programming as well as system development, wherefore a question of importance was raised early in the process; should the subject matter of informatics be taught by ‘regular’ Natural Sciences and Mathematics teachers or by contemporary vocational education teachers in ADP? The question was initially solved using stereotypical examples of how to apply system development, which was later suggested as a replacement for programming activities. The initial incitement to offer informatics education during the 1970s was discovered in the recruitment of a broader group of students within the Natural Science Programme and the perception that it would contribute to the development of students’ ability to think logically and problem solving skills.

The thesis unravels an instructional dependence among today’s teachers where students’ logical and analytical abilities (even before the courses start) are considered crucial to students’ learning, while teachers question the importance of their pedagogy. Teachers in the study commonly express the belief that their instructions hardly matter to the students’ learning. Instead, these teachers perceive learning programming as an individual act. The inquiry also discover two types of instruction; a large group putting emphasis on the students’ experiences of learning concepts of computer science (not necessarily to do with syntax), which corresponds with the existence of two groups of teachers during the 1980s; the partisans who perceived learning as based on repeating sequences in a behaviouristic manner, and defenders who perceived learning as based on discovery and self-teaching.

In summary the inquiry depicts an instructional tradition based on teachers’ beliefs where the historical development of the subject sets the framework for the teaching. Directly and indirectly the historical development and related traditions govern what programming teachers in upper secondary school will/are able to present to their students.

Keywords: programming education, teachers’ beliefs, curriculum development, upper secondary school.
Acknowledgements

Like other doctoral students in the final state of a licentiate, I am indebted to a huge amount of people. In the following section I express my gratitude as distinctively and correctly as I can, helped by my recollection of the journey.

First and foremost I wish to thank my supervisors; Professor Inga-Britt Skogh and Dr. Sirkku Männikkö-Barbutiu. I deeply appreciate Inga-Britt Skogh’s gentle guidance through the educational research domain and her encouragement and confidence in what I wrote. She was the facilitator who opened my eyes to research in relation to what a community of teachers’ can offer.\(^1\)

With the guidance and support of my second supervisor, Sirkku Männikkö-Barbutiu, I discovered the beauty of writing. The writing process has been a discovery in itself, as words and sentences slowly emerged into something meaningful that mirrored my own learning process, as well as something that would be useful to the teacher/researcher community. She gave me self-respect, and confidence in my way of thinking in relation to text for a deeper understanding.

Thanks also go to my colleagues at the Royal Institute of Technology (KTH), Stockholm University and the University of Gävle, all involved in the cooperation TUFF:s (Technology education for the future, Swedish TeknikUtbildning För Framtiden) graduate school. To be a part of the development in School of Education and Communication in Engineering Science (ECE) is stimulating and rewarding in many aspects.

In initial phase of identifying researchers with similar interest I was invited by Anna Eckerdal, Anders Berglund and Michael Thune to share and discuss in their research group. A step that became vital and beneficial to this work. The inquiry has been funded by the Swedish government and the municipality of Nynäshamn through Lärarlyftet, a program for teachers’ continuing professional development. Without that incitement I suppose this journey would never have been completed. Their support is therefore gratefully acknowledged.

The hours spent in archives have made me appreciate the considerable work invested to make documents available and searchable. I therefore acknowledge the staff of The National Archives (RA) in Arninge who did so kindly discuss my area of interest and help me whenever a document was needed. Gunnar Haglund, archivist

\(^1\)See Swedish Informatics Teachers’ Network (SITSNET).
ACKNOWLEDGEMENTS

at the Swedish National Agency of Education (SNAE) always handled my requests with respect, and suggested further references (people or material) that may be able to answer my questions.

When looking back it seems that the journey has been a successful experience where I have been fortunate to meet people engaged in school management and development. Of course they did not automatically enter my scene of investigation, but when I researched their existence and presented my intentions, I was astonished by their willingness to help and share memories in interviews. The list of today's programming teachers involved in the research would be at least 150 teachers. I deeply appreciate the time and effort they contributed. This thesis would have been nothing without their involvement.

Rolf Nilsson, one of the Swedish computer pioneers within secondary education, to whom I owe a debt of gratitude, willingly answered my questions. He became a witness of what it means to initiate computer technology in the school system.

Christina Selander Ekström, principal at Nynäshamns gymnasium, who encouraged me to express and ask questions about some of the many implicit dilemmas schools face. I deeply appreciate her taking my questions seriously and opening the door for further doctoral studies.

Gusten Rolandsson, my father, who has been an inspiration over the years. He offered at least one answer of my school organizational questions, based on his experience and his knowledge of facts and history regarding the Swedish school system.

Finally, I would like to thank my family, who has shown support, patience and confidence throughout these years. My love to Christina, Jonathan, Rebecka and Jakob.
Preface

It took me almost 35 years with a diversity of teachers, instructional methods and students (20 years of teaching) to realize that the problem of teaching and learning was far more complex than passing my earlier experiences on to my students or embracing exemplary teaching.

During my university studies, a search for the foundations of knowledge began. Commonly, concepts in Physics were offered, “as they are” without references or concern for their origins or historical evolution. I specifically remember how a course in the History of Physics opened my eyes to the importance of societal context, i.e. how concepts used could be seen as products of their time and the people behind them. As an upper secondary school teacher, I worked hard to implement these ideas as they (to me) seemed rich and useful to learning and understanding.

When working as a project manager for a vocational education in data and telecommunication a few years later, I learnt from experience that “softer concepts” (context, culture, values and norms) matter to education. While teaching computer programming in different contexts I found myself teaching differently in different context, and this became an important experience.

Some years later, after my Master thesis, the issue appeared in a new guise. I was heavily involved in instructional design for learning, based on collaboration. The idea was to design collaborative learning environments in computer programming and databases, for teachers and students. The aim was to use IT/ICT to enhance the learning situation in classrooms and on-line. In the search for co-teachers, we found that applicants commonly expressed interest, but our intention to collaborate with schools became almost impossible to achieve. An experience that set the wheels in motion for this study.

Nynäshamn, 2012-11-20
Lennart Rolandsson
Contents

Acknowledgements v
Preface vii
Contents viii
1 Introduction 1
1.1 Setting the scene . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
1.2 Concepts and how they are interpreted/used in the thesis . . . . . 3
1.3 The Swedish school system and ADP . . . . . . . . . . . . . . . . . . 5
1.4 Purpose and Research Questions . . . . . . . . . . . . . . . . . . . . 6
1.5 Thesis outline . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
2 Theoretical framework 9
2.1 Fixed instruction strategies . . . . . . . . . . . . . . . . . . . . . . . 9
2.2 Theories and models for learning and teaching . . . . . . . . . . . . 10
2.3 Curriculum development of informatics . . . . . . . . . . . . . . . . 11
2.4 Defenders and partisans . . . . . . . . . . . . . . . . . . . . . . . . . . 11
2.5 Teachers’ perception of ‘How to teach and learn programming’ . . 13
2.6 Beliefs and teachers’ beliefs . . . . . . . . . . . . . . . . . . . . . . . 14
3 Methodology 17
3.1 The historical perspective . . . . . . . . . . . . . . . . . . . . . . . 17
3.2 Today’s teachers’ perspective . . . . . . . . . . . . . . . . . . . . . . 21
3.3 Ethical considerations and trustworthiness . . . . . . . . . . . . . . 24
4 Summary of Articles 27
4.1 Paper 1: Informatics and programming in Swedish upper secondary school - vision and experimental work during the 1970s and 1980s . . 27
4.2 Article 2: Teachers’ beliefs regarding programming education . . . 29
5 Results 31
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>The historical development of the informatics curriculum during the 1970s</td>
<td>31</td>
</tr>
<tr>
<td>5.2</td>
<td>The historical development of the informatics curriculum during the 1980s</td>
<td>33</td>
</tr>
<tr>
<td>5.3</td>
<td>Today's teachers' practice</td>
<td>34</td>
</tr>
<tr>
<td>5.4</td>
<td>Summary of results</td>
<td>37</td>
</tr>
<tr>
<td>5.5</td>
<td>The curriculum development process of computer programming - a time line</td>
<td>38</td>
</tr>
<tr>
<td>6</td>
<td>Discussion</td>
<td>41</td>
</tr>
<tr>
<td>6.1</td>
<td>Curriculum and teachers' beliefs</td>
<td>41</td>
</tr>
<tr>
<td>6.2</td>
<td>The evolution of the curriculum - a top-down or bottom-up process</td>
<td>42</td>
</tr>
<tr>
<td>7</td>
<td>Conclusions and final remarks</td>
<td>45</td>
</tr>
<tr>
<td>7.1</td>
<td>Informatics curriculum evolution</td>
<td>45</td>
</tr>
<tr>
<td>7.2</td>
<td>Beliefs</td>
<td>46</td>
</tr>
<tr>
<td>7.3</td>
<td>Messages to teachers and curriculum developers</td>
<td>46</td>
</tr>
<tr>
<td>8</td>
<td>Further studies</td>
<td>49</td>
</tr>
<tr>
<td>8.1</td>
<td>In educational technology</td>
<td>49</td>
</tr>
<tr>
<td>8.2</td>
<td>In teacher's associations and interest groups</td>
<td>50</td>
</tr>
<tr>
<td>8.3</td>
<td>In teacher's perceptions of teaching and learning</td>
<td>50</td>
</tr>
<tr>
<td>8.4</td>
<td>In the school system</td>
<td>51</td>
</tr>
<tr>
<td>9</td>
<td>Sammanfattning (summary in Swedish)</td>
<td>53</td>
</tr>
<tr>
<td>9.1</td>
<td>Inledning</td>
<td>53</td>
</tr>
<tr>
<td>9.2</td>
<td>Forskningsfokus</td>
<td>54</td>
</tr>
<tr>
<td>9.3</td>
<td>Tidigare forskning</td>
<td>54</td>
</tr>
<tr>
<td>9.4</td>
<td>Metod</td>
<td>55</td>
</tr>
<tr>
<td>9.5</td>
<td>Resultat</td>
<td>56</td>
</tr>
<tr>
<td>9.6</td>
<td>Diskussion</td>
<td>57</td>
</tr>
<tr>
<td>9.7</td>
<td>Slutsatsar</td>
<td>58</td>
</tr>
<tr>
<td>Bibliography</td>
<td></td>
<td>59</td>
</tr>
<tr>
<td>A Swedish School System</td>
<td></td>
<td>75</td>
</tr>
<tr>
<td>B Questionnaires</td>
<td></td>
<td>77</td>
</tr>
<tr>
<td>Papers</td>
<td></td>
<td>93</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Setting the scene

All students can learn

School is constantly transforming, as new philosophical imperatives and political educational decisions shape our thinking about how education should or could be practiced. Resnick (2010) elaborates on what school is able to offer “through social processes that include participation in certain forms of high-demand learning (Resnick, 2010, p.186) instead of as an “entity” or as something that people have a fixed amount of time to accomplish (Resnick and Gall, 1997; Dweck and Molden, 2005). In the 1990s “the public agenda of raising educational levels for all has been promoted under the banner of the standards movement, often accompanied by the phrase ‘All children can learn’” (Resnick, 2010, p.184). Resnick suggests a new way of school organization where teacher instruction and professional development should aim for and secure a “thinking curriculum” that has high cognitive demands on students’ and teachers’ conceptual learning, reasoning, explaining and problem solving.1

Computer programming as a solution to enhance problem solving in schools

The history of programming offer a divers picture favoring a multitude of different solutions and development environments (Bergin and Gibson, 1996). They all share ambitions to offer easy access to languages and efficient management of concepts within computer science and/or informatics. For many years computer programming has been perceived as school’s “new Latin”. It is believed to foster problem solving skills, thinking skills and understanding of computers (Urban-Lurain and Weinshank, 2011). However the transferability of skills and knowledge be-

1 See also Nisbeta (1993) and Resnick and Klopfer (1989).

Chapter 1

Introduction

1.1 Setting the scene

All students can learn

School is constantly transforming, as new philosophical imperatives and political educational decisions shape our thinking about how education should or could be practiced. Resnick (2010) elaborates on what school is able to offer “through social processes that include participation in certain forms of high-demand learning (Resnick, 2010, p.186) instead of as an “entity” or as something that people have a fixed amount of time to accomplish (Resnick and Gall, 1997; Dweck and Molden, 2005). In the 1990s “the public agenda of raising educational levels for all has been promoted under the banner of the standards movement, often accompanied by the phrase ‘All children can learn’” (Resnick, 2010, p.184). Resnick suggests a new way of school organization where teacher instruction and professional development should aim for and secure a “thinking curriculum” that has high cognitive demands on students’ and teachers’ conceptual learning, reasoning, explaining and problem solving.1

Computer programming as a solution to enhance problem solving in schools

The history of programming offer a divers picture favoring a multitude of different solutions and development environments (Bergin and Gibson, 1996). They all share ambitions to offer easy access to languages and efficient management of concepts within computer science and/or informatics. For many years computer programming has been perceived as school’s “new Latin”. It is believed to foster problem solving skills, thinking skills and understanding of computers (Urban-Lurain and Weinshank, 2011). However the transferability of skills and knowledge be-

1 See also Nisbeta (1993) and Resnick and Klopfer (1989).
Research indicates that students’ understanding of computer programming concepts often remain firm and barely change (Kurland et al., 1989; Mayer et al., 1986). The understanding is not deep enough. Syntax and semantics in computer programming demands an abundance of experience before it can be used for problem solving purposes (Urban-Lurain and Weinshank, 2011). The process of learning programming is considered by some as a life-long adventure, going from novice to professional (Winslow, 1989; Kurland et al., 1989). In fact Resnick (2009) even states that computer programming languages are too difficult to use in education. According to Resnick (ibid.) programming languages are either introduced in such a way that it does not appeal to students’ experiences or in a tutoring context where teachers fail to provide both sufficient guidance and appropriate challenges.

Computer technology and computer programming education

Educational technology aimed at enhancing learning has existed since the 1960s in different guises (e.g. Murray, 1983; Thomas and Kobayashi, 1987; Kollerbaur, 2005). Computer programming education benefits considerably from what computer technology, programming paradigms and environments offer the learning process. Specific programming languages (like Python, Pascal and Basic) and environments (like Alice, BlueJ, Arduino and Raspberry Pi) are developed for educational purposes to lower the knowledge threshold for computer programming. Today applications, literature and services are made available at an evenly increasing pace. The artefact computer and the technology around it being the only environment for informatics education could therefore be questioned. The ideas behind computer science concepts - the thoughts underpinning the art of programming - could be taught in alternative ways without implicit access to computer technicalities. Instead instructions could be taken from settings where logic, structure and algorithms are exclusively taught (Thiers and Vahrenhold, 2012; Bell et al., 2011; Feaster et al., 2011; Bell et al., 2012; Haberman, 2006) by emphasising collaboration, role-play gaming and settings outside of classrooms.

Teaching programming

According to Shulman (1987) there is a difference between knowing a topic and being able to teach it. Computers and programming languages constantly evolves, and this raises both practical and didactical issues for teachers to deal with. The understanding is not deep enough. Syntax and semantics in computer programming demands an abundance of experience before it can be used for problem solving purposes (Urban-Lurain and Weinshank, 2011). The process of learning programming is considered by some as a life-long adventure, going from novice to professional (Winslow, 1989; Kurland et al., 1989). In fact Resnick (2009) even states that computer programming languages are too difficult to use in education. According to Resnick (ibid.) programming languages are either introduced in such a way that it does not appeal to students’ experiences or in a tutoring context where teachers fail to provide both sufficient guidance and appropriate challenges.

Computer technology and computer programming education

Educational technology aimed at enhancing learning has existed since the 1960s in different guises (e.g. Murray, 1983; Thomas and Kobayashi, 1987; Kollerbaur, 2005). Computer programming education benefits considerably from what computer technology, programming paradigms and environments offer the learning process. Specific programming languages (like Python, Pascal and Basic) and environments (like Alice, BlueJ, Arduino and Raspberry Pi) are developed for educational purposes to lower the knowledge threshold for computer programming. Today applications, literature and services are made available at an evenly increasing pace. The artefact computer and the technology around it being the only environment for informatics education could therefore be questioned. The ideas behind computer science concepts - the thoughts underpinning the art of programming - could be taught in alternative ways without implicit access to computer technicalities. Instead instructions could be taken from settings where logic, structure and algorithms are exclusively taught (Thiers and Vahrenhold, 2012; Bell et al., 2011; Feaster et al., 2011; Bell et al., 2012; Haberman, 2006) by emphasising collaboration, role-play gaming and settings outside of classrooms.

Teaching programming

According to Shulman (1987) there is a difference between knowing a topic and being able to teach it. Computers and programming languages constantly evolves, and this raises both practical and didactical issues for teachers to deal with. The understanding is not deep enough. Syntax and semantics in computer programming demands an abundance of experience before it can be used for problem solving purposes (Urban-Lurain and Weinshank, 2011). The process of learning programming is considered by some as a life-long adventure, going from novice to professional (Winslow, 1989; Kurland et al., 1989). In fact Resnick (2009) even states that computer programming languages are too difficult to use in education. According to Resnick (ibid.) programming languages are either introduced in such a way that it does not appeal to students’ experiences or in a tutoring context where teachers fail to provide both sufficient guidance and appropriate challenges.
1.2. CONCEPTS AND HOW THEY ARE INTERPRETED/USED IN THE THESIS

strategy to focus primarily on general principles and concepts has therefore become common and successful (e.g. Thies and Vahrenhold, 2012; Bell et al., 2011, 2012). The idea that concepts and principles do not need to be taught using technology as a mediating tool for learning has opened up for new ways of teaching computer programming that are appropriate to a broader group of students.

There is research indicating (McGettrick et al., 2005; Gries, 2006; Linn and Clancy, 1992; Sloane and Linn, 1988) that new pedagogical ideas, the development of new educational environments, new educational programming languages and the introduction of new informatics curricula have had little or no influence on the instructional pattern (content and work methods) offered in programming education in upper secondary school.

Learning programming

The process of learning programming - going from novice to professional - is by some considered to be a lifelong adventure (Winslow, 1989; Kurland et al., 1989). Kurland et al (1989) shows that students with two years of programming instruction commonly have not yet reached what would call a deeper understanding: "Many students had only a rudimentary understanding of programming". Winslow concludes that "One wonders [...] about teaching sophisticated material to CS1 students when study after study has shown that they do not understand basic loops ..." (Winslow, 1996, p.21, in Robins et al., 2003). A more optimistic view is expressed by Guzdial (2004) who claims that since computer programming has evolved considerably, and gradually diffused into different domains and gadgets the problems in teaching and learning, could be expected to diminish (diSessa, 2001).

This study

This thesis focuses on teachers' and curriculum developers' ambitions to foster computer programming skills in secondary school students. The study is an attempt to explore if and how the ambitions of yesterday's curriculum developers' are reflected in today's programming education practice in upper secondary school.

1.2. CONCEPTS AND HOW THEY ARE INTERPRETED/USED IN THE THESIS

Epistemology

Epistemology is the philosophical study of knowledge and belief. The word originates from the Greek word, episteme, which could be translated as the understanding of knowledge. Philosophy typically investigates epistemology while studying the nature of knowledge and its limitations.

Similar questions as exist among epistemology philosophers are also prevalent among teachers when asking the with the w-questions (why, what, how and when);
to what extent is it possible for a given subject or entity to be known? What is knowledge in my subject domain? How is knowledge assessed and acquired? When should I discuss the complexity of the knowledge at focus?

The first of the w-questions is probably the most compelling while it demands reflection and experience of a specific knowledge domain within a school context. This question is at focus throughout the investigation in order to enable study of whether teachers are constrained by school, and what computer technology offers in the educational context.

**Instruction and instructional design**

Instruction and instructional design (used as equivalent to the concept of didactics) elaborates on didactic issues like “what to teach” and “how to teach”. The concept of instructional design could be traced back to teaching machines in instructional settings and programmed instruction (Seel and Dijkstra, 2008). Today though, the concepts of instruction and instructional design are used among constructivists as well as among behaviorists (Mayer, 1999).

The word “instruct” comes from the Latin word “instruere”: to set up, furnish, kit out and teach. The thesis adopts the concept of instruction in a way similar to Brockenbrough (1993). He adds the prefix in to *stuere* which could be interpreted as “to build within”. The conclusion of the construction elicits the fact that instruction “cannot occur unless it occurs inside people’s heads and that no amount of external activity or material can substitute for the existential fact that knowledge must be (re)created by each individual”. (Brockenbrough, 1993, p.184)

**Informatics or Computer Science**

School systems worldwide embrace the content of computer programming differently, and it is offered under labels such as “computing”, “computer science” or “informatics” (Dagiene, 2006). However the computer science education research community seems to make no distinction between the two terms “computer science” and “informatics” (Saeli et al., 2011). During the 1970s, 1980s and 1990s, the Swedish informatics curriculum was labelled differently depending on context, educational level and decade; “Datalära”, “Datateknik”, “Datakunskap”, “Datorkunskap” and “ADB” which in English translates as Computing, Information technology, Information knowledge, Computer knowledge and Automatic Data Processing (ADP).4

The word “Data” is obscure, as it could nowadays be interpreted to mean “the computer” and/or “the information in the computer”. However history and the original meaning, which correlates to the international discourse, infer

---

4Word from Latin which translates to place up together

4Figure 5.2 depicts a historical overview of informatics education in Swedish upper secondary school.

---
that “information” is the most appropriate translation. Throughout the study I have chosen to use the word “informatics” whenever some of the Swedish labels mentioned above is used in association with the curriculum.

1.3 The Swedish school system and ADP

To offer an insight into the educational context of the inquiry, a brief presentation of the Swedish school system and the subject matter of informatics, will be given in the following section.

The Swedish upper secondary school has undergone a series of changes during the past 40 years (Linderojaso and Lundgren, 2000). Three different curricula (Lgy70, Lpf94 and GY2000) for gymnasium (upper secondary school) have passed, and today a fourth curriculum (GY-11) is on its way. The basic structure follows the same principles now as it did then; compulsory school (nine years) followed by two, three or four years of upper secondary school.

However, disruption emerged during the 1970s in the monolithic Natural Science Programme (NSP). The programme experienced a decline in student numbers, after the introduction of the new curriculum. Some believed NSP was too theoretical, wherefore a computing alignment within the NSP was suggested in 1976 by the Swedish National Board of Education (NBE). The disruption is worth mentioning as vocational education in ADP and natural sciences became an issue in the enactment of the informatics curriculum.

The current design of upper secondary school was established during the 1970s. The organisation of upper secondary school education was designed to facilitate the separation of students aiming for higher education from students not aiming for higher education. In conjunction with this ‘main system’, special higher education programmes (TEP) were expected to start their professional career after a fourth year with specialising in chemistry, construction, machinery or electricity. The organisation of upper secondary school education was designed to facilitate the separation of students aiming for higher education from students not aiming for higher education. In conjunction with this ‘main system’, special higher education programmes (TEP) were expected to start their professional career after a fourth year with specialising in chemistry, construction, machinery or electricity. The organisation of upper secondary school education was designed to facilitate the separation of students aiming for higher education from students not aiming for higher education. In conjunction with this ‘main system’, special higher education programmes (TEP) were expected to start their professional career after a fourth year with specialising in chemistry, construction, machinery or electricity. The organisation of upper secondary school education was designed to facilitate the separation of students aiming for higher education from students not aiming for higher education. In conjunction with this ‘main system’, special higher education programmes (TEP) were expected to start their professional career after a fourth year with specialising in chemistry, construction, machinery or electricity.
CHAPTER 1. INTRODUCTION

situation today is somewhat different as all national programmes, except the TEP, are three years long.10

1.4 Purpose and Research Questions

Over the years, the subject matter of computer programming has transformed and diffused into the school subject in Sweden as well as internationally. Programming teachers all over the world face a number of challenges. The development of new programming curricula is one concern. In Sweden for example The Swedish National Agency of Education releases a new curriculum approximately every tenth year. Any revision of policy documents (regardless of frequency) means that teachers are affected by and dependent on not only the context and the considerations that have brought forward the current curriculum but also the considerations that brought/will bring forward previous and future curricula. The changing educational frameworks together with the rapid development of technology make programming teaching a challenging task (Haberman, 2006).

Another disturbing issue is the decline in numbers of students in computer science/informatics noticed internationally (Haberman, 2006; Syslo and Kwiatkowska, 2008). A report (Wilson et al., 2010) published two years ago by the Association for Computing Machinery (ACM), “Running On Empty” pictures a crisis in computer science for the K-12 education. According to Wilson there seems to be a “move away from physical hardware and computer programming towards the application of computers in real-world situations and the use of generic software packages to solve problems” (Woollard, 2005, p.190).

In summary computer programming education is demanding. Teachers are expected to have acquired digital fluency and pedagogical ability to offer appropriate content and instruction. In this thesis the characteristics of computer programming education in Sweden are focussed but since the situation in Sweden is not unique it is believed to be of interest in an international context.

To understand the intention, continuous development and the enactment of curriculum, the thesis describes the informatics curriculum development in relation to the following two strands:

1. A historical strand with the question: How was the informatics curriculum developed in Swedish upper secondary school during the 1970s and the 1980s?

2. A present time strand where Swedish upper secondary teachers’ beliefs about programming and educational constraints are focussed on, with the question: What beliefs do programming teachers express regarding teaching and learning computer programming in upper secondary school?

10The TEP has been through two revisions since 2000, to raise the numbers of students in tertiary studies in the field of technology. A fourth year was re-introduced as an experiment offering different alignments for those interested in a professional career instead of further studies.
1.5 Thesis outline

The thesis is based on two papers, and has the intention to contribute to research in informatics curriculum theory. For those who are interested in the preceding work, I refer to the papers:

1. Informatics and programming in Swedish upper secondary school - Visions and experimental work during the 1970s and 1980s
2. Teachers’ beliefs regarding programming education

The thesis covers a summary and a synthesis of these two papers. Below a short description of each chapter and the appendices.

The first chapter is an introduction to the theme of the thesis. The aim is to provide a platform to appreciate the didactical constraints that teachers of today face when teaching computer programming in upper secondary school.

The second chapter shows a theoretical framework of perspectives to position the investigation. Teachers’ instructions in relation to informatics are in focus, and tradition and beliefs are discussed.

The third chapter presents the methodologies used in the two sub-studies. Ethical considerations are also addressed.

The fourth chapters give a summary of findings presented in the two papers; a history oriented strand picturing the informatics curriculum development process, and an education-oriented strand picturing the epistemological beliefs of today’s teachers.

In chapters five and six a synthesis of the two papers is presented. In the sixth chapter specific characteristics in computer programming education are discussed.

The seventh chapter concludes the study with some suggestions for further development in curriculum for computer programming.

The eighth chapter offers some suggestions for further studies within the field of upper secondary school computer programming education.

The ninth chapter presents a Swedish summary of the thesis.

The appendicies offer each an overview of the Swedish school system and questionnaires used in the thesis.
Chapter 2

Theoretical framework

The following chapter presents the research domain of how computer programming has been taught and its underpinnings (teachers’ beliefs). To achieve this, the intersection between computer programming curriculum and pedagogy has been scrutinised. The curriculum is approached as a dynamic entity which is changed, enacted and experienced in a continuous process by different parties (Goodson, 1993; Linde, 1993). The intersection of curriculum history in informatics and today’s teacher’s beliefs is a way of studying the curriculum as a continuous process, from a macro and a micro-level, where existing determinants in the transformation of a school subject could be discovered (Linde, 1993).

2.1 Fixed instruction strategies

According to Guzdial teachers are tasked “to make computation available to thinkers of all disciplines”. He pinpoints that the educational discourse about computer programming started early in the 1960s “… [where] programming was an exploration of process, a topic that concerned everyone, and that the automated execution of process by machine was going to change everything” (Guzdial, 2008, p.25). However, it seems that instructions in computer programming have remained the same since the 1960s. According to Kaplan (2010, p.1) computer programming “is largely taught today the way it was taught 60 years ago” while he states that research has not informed the computer programming teaching community.

The study of teaching and learning computer programming seems complex by nature; the researchers could approach the educational problem from at least five different perspectives: 1) The use of development environments and visualisation, 2) The development of an understanding of tasks/problems in a context, 3) The development of an understanding of tasks/problems in a context, 4) The development of an understanding of tasks/problems in a context, 5) The development of an understanding of tasks/problems in a context.

---

1 The number of perspectives is an observation made during investigation.
2 See the work conducted within e.g. conferences such as: European Conference on Object-Oriented Programming (ECOOP); and Object-Oriented Programming, Systems, Languages & Applications (OOPSLA).
3 See the work conducted within e.g. The International Computing Education Research (ICER)
CHAPTER 2. THEORETICAL FRAMEWORK

The development of programming languages for use in education, \(^4\) Teachers’ instruction (Sloan and Linn, 1988; Linn and Clancy, 1992; Postareff and Lindblom-Yläne, 2008; Trigwell et al., 1994; Kember and Kwan, 2000; Boulton-Lewis et al., 2001), and finally 5) The understanding of students learning (Meyer and Land, 2003; Boustedt et al., 2007).

The study at hand however, approaches the complexity of computer programming education in a somewhat different way compared to the above mentioned five perspectives, as it is based on the assumption that teachers’ beliefs influence their instruction and interactions with students.

2.2 Theories and models for learning and teaching

According to McCormick (1992) research in informatics curriculum for secondary level is rare. The absence of research papers dealing with established theories or models of learning and teaching programming is obvious (Sheard et al., 2009). This problem is brought up by Holmboe et al. (2001) as they investigate the nature and the scope of computer science education (CSE) research.

\(^{4}\)We argue that there has been a lack of reference to pedagogical theory, underlying most past research studies. This has resulted in a failure to provide teachers with ‘pedagogical content knowledge’, critical to gaining useful insights into cognitive and educational issues surrounding learning” (Holmboe et al., 2001)

Despite the troublesome lack of theories there is some researches related to teachers’ instruction, from different disciplines. A phenomenographical investigation by Postareff and Lindblom-Yläne (2008) discovered two existing approaches to teaching, learning-focused and content-focused. Similar findings were reported by Kember et al. (2000) and Trigwell et al. (1994) who also exposed the existence of different teaching approaches; student- and teacher centred instruction. The outcomes of these three investigations are not exactly comparable, but they do show the dichotomy between existing teaching instructional strategies. In a phenomenographical study performed by Boulton-Lewis et al. (2001), with 24 upper secondary school teachers, four instructional approaches were identified

1. Transmission of content/skills – focus is on the content and the students are somewhere in the background.

2. Development of skills/understanding – teachers direct the learning process and students are perceived as participants.

4) Teachers’ instruction (Sloan and Linn, 1988; Linn and Clancy, 1992; Postareff and Lindblom-Yläne, 2008; Trigwell et al., 1994; Kember and Kwan, 2000; Boulton-Lewis et al., 2001), and finally 5) The understanding of students learning (Meyer and Land, 2003; Boustedt et al., 2007).

The study at hand however, approaches the complexity of computer programming education in a somewhat different way compared to the above mentioned five perspectives, as it is based on the assumption that teachers’ beliefs influence their instruction and interactions with students.

2.2 Theories and models for learning and teaching

According to McCormick (1992) research in informatics curriculum for secondary level is rare. The absence of research papers dealing with established theories or models of learning and teaching programming is obvious (Sheard et al., 2009). This problem is brought up by Holmboe et al. (2001) as they investigate the nature and the scope of computer science education (CSE) research.

\(^{4}\)We argue that there has been a lack of reference to pedagogical theory, underlying most past research studies. This has resulted in a failure to provide teachers with ‘pedagogical content knowledge’, critical to gaining useful insights into cognitive and educational issues surrounding learning” (Holmboe et al., 2001)

Despite the troublesome lack of theories there is some researches related to teachers’ instruction, from different disciplines. A phenomenographical investigation by Postareff and Lindblom-Yläne (2008) discovered two existing approaches to teaching, learning-focused and content-focused. Similar findings were reported by Kember et al. (2000) and Trigwell et al. (1994) who also exposed the existence of different teaching approaches; student- and teacher centred instruction. The outcomes of these three investigations are not exactly comparable, but they do show the dichotomy between existing teaching instructional strategies. In a phenomenographical study performed by Boulton-Lewis et al. (2001), with 24 upper secondary school teachers, four instructional approaches were identified

1. Transmission of content/skills – focus is on the content and the students are somewhere in the background.

2. Development of skills/understanding – teachers direct the learning process and students are perceived as participants.
2.3. CURRICULUM DEVELOPMENT OF INFORMATICS

3. Facilitation of understanding – teachers and students work together to construct personal meaning

4. Transformation – the teacher organizes the situation to provide enough stimulus for students to take action, while the teacher fades into the background.

These investigations pinpoint a distinction between teachers’ instruction and students’ engagement or ability. The second and third categories suggested by Boulton-Lewis et al. (ibid.) include approaches of importance for this study that will be further discussed.

2.3 Curriculum development of informatics

Informatics education/curricula obviously vary from country to country regarding content and ambitions (Dagiene, 2006; Micheuz, 2006; Dagiene, 2005; Hubwieser et al., 2011). Research from the domain of computing history however, reveals some commonality between different countries; the UK (Woollard, 2005), Austria (Micheuz, 2005), the Ukraine (Spirin, 2005), Lithuania (Dagiene, 2005), Poland (Syslo and Kwiatkowska, 2005), Sweden (Rolandsson, 2011). In many of these studies, the implementation of curriculum development in schools is described in relation to investments in hardware and software.

Some studies elicit the fact that computer technology raises considerable problems for instruction and pedagogy as technological optimism commonly entails the innovation of ICT (Tapscott, 1998; Segal, 1996; Pedersen, 2001; Cuban, 2001, 1986; Pelgrum, 2001; Donaldson and Knupfer, 2001). In the field of education, ICT is believed to offer the solution to a number of pedagogical problems (Segal, 1996 in Karlsohn, 2009, p.353) similar to the optimism entailing the innovation of educational technology in the 1960s. Pedersen and Cuban (Pedersen, 2001; Cuban, 2001) raise a question about the implicit technological determinism within informatics education.

The tension between the technicalities within informatics and the didactics has been a long lasting process with many backdrops (Pelgrum, 2001; Donaldson and Knupfer, 2001). Explicit technicalities in computer technology which commonly dominates instructional design, wherefore Cuban underlines the necessity to discuss and elaborate on methodological questions (Cuban, 1986, 2001).

2.4 Defenders and partisans

During the 1980s two main divisions of educators existed ‘side by side’, perceiving learning and teaching in programming differently. Teaching and learning computer programming was debated by the defenders of programmed teaching and the partisans of learning through discovery and self-teaching (Solomon, 1986 in Mendlesohn et al., 1990). In the spirit of behaviourism, the defenders conceived programmed teaching through repeated sequences as an effective learning tool (Suppes, 1979).
In the spirit of constructionism the partisans advocated learning through discovery as a way of supporting children’s own knowledge building (Papert, 1980). Today, the debate has faded while these two groups of defenders and partisans still exist among teachers. This will be discussed further.

In an attempt to find an instructional theory for computer programming education, Linn and Sloan studied “naturally occurring instructions” (Sloane and Linn, 1988):

“Historically, programming classes built on experiences of expert programmers who taught themselves. Students were provided with assignments and access to computers and were expected to learn through trial and error and unguided discovery”. (Sloane and Linn, 1988, p.208)

Teachers commonly used discovery learning, problem-solving procedures or extensive feedback. An effective method for teaching programming seems hard to discover (Linn, 1985; Linn and Dalbey, 1985). A quotation from Linn and Clancy (1992) explains the situation that many students faced during the 1990s:

“Programming instructors often assume that students can take their general problem-solving skills and discover specific programming design skills on their own. Thus, students learn program design through unguided discovery . . . In programming, the acquisition of design skills is further impeded . . . For example, texts frequently feature what is called ‘top-down design’, the process of designing a program by breaking the high-level statement of the problem into parts and then continuing with this process until the program is completed . . . Instruction that suggests program design proceeds in an uncomplicated, top-down fashion confuses and frustrates students . . . [while] teachers often describe the features of completed programs or the characteristics of the language syntax . . . As a result, students may think they should know how to design solutions to problems without actually learning the skills. Programming instruction often implies that design skills are available to everyone. Students lacking a clue as to how to proceed in designing the solution to programming problem may conclude they are incapable of learning, when in fact, they are actually unaware of how to proceed” (Linn and Clancy, 1992, p.125-126)

The quotation describes a situation where teachers expect learning to emerge through unguided discovery and students to be able to draw from their former abilities or skills, for success or failure. There is reason to believe the situation remains still today. In 1996, East and Wallingford pinpoints the same instructional problem:

“There is indeed little discussion of the teaching of programming that relates to pedagogy and almost none that addresses how the process of learning might or should affect instruction.” (East et al., 1996, p.1)
2.5 Teachers’ perception of ‘How to teach and learn programming’

Approximately 10 years later, the problem still remains on the agenda, as teaching programming was discussed as one of the seven challenges in computing education (McGettrick et al., 2005). Gries (2006) takes it one step further when he pinpoints the difference between teaching facts and teaching for ‘deeper’ understanding which correlates with Resnick (2010) and a thinking curriculum.

“We need to look seriously at how we teach programming. The purpose of an education should not simply be to pour facts into students, but rather to teach them to think.” (Gries, 2006, p.82)

Bruner (1996) and Lister (2008) emphasise the importance of teachers liberating themselves, as teachers from “folk” pedagogy (Bruner, 1996; Lister, 2008), where you commonly teach the content based on intuition and your own educational experiences. In general, folk pedagogy and educational beliefs are seen as resistant to external influences and accordingly very difficult to change (Kagan, 1992; Luft and Roehrig, 2007; Olson, 1981; Yerrick et al., 1997), as they depend on the individuals’ personal growth, ability to reflect upon and understand her/his own teaching practice (Baird et al., 1991; Brookfield, 1995; Schön, 2003).

If a group of teachers show similar patterns in teacher knowledge and beliefs, it is of importance to examine the implications of these patterns. The underpinnings of such patterns could be the reasons for the difficulties in learning that computer programming offers. In that case the implicit constraints in education depend on the messengers (teachers) as well as on the content itself (Thompson, 1984).

In studies by Pajares (1992) and Schraw and Olafson (2002) the transformation of knowledge in classrooms is scrutinised as dependent on teachers’ epistemological beliefs in relation to teaching computer programming. The concept of belief is described using two different dimensions; beliefs about classroom practices and ontological beliefs. Our beliefs influence our understanding of the world (Abelson, 1986; Alexander and Dochy, 1995). Accordingly, teachers’ beliefs will influence their perception of learning environments, instructional materials and different instructional approaches available to them (Alexander and Dochy, 1995; Calderhead, 1996; Bangum, 2003). Kagan postulated, “most of a teacher’s professional knowledge can be regarded more accurately as a belief” (Kagan, 1992, p.73). Thompson (1984) argues for the importance of teachers’ conception (their beliefs, views, and preferences) in research about education, as it constitutes a primary mediator between the subject and the learners. This is supported by research in for instance mathematics when studying teachers’ interpretation and implementation of mathematics curricula (Clark and Peterson, 1986; Romberg and Carpenter, 1986; Thompson, 1984).
### 2.6 Beliefs and teachers’ beliefs

Thompson (1992) distinguish between beliefs and knowledge, where the concept of belief holds specific features:

1. Beliefs are held with varying degrees of conviction and
2. There is no consensus as there is no need to satisfy a truth condition.

The concept of knowledge is attained in consensus within a cultural context holding a specific belief (Abelson, 1979; Ernest, 1991). It should however be noted that beliefs sometimes move to the status position of knowledge and vice versa. Teachers’ epistemology in relation to their daily work and subjects they are teaching is commonly labelled as epistemological beliefs or personal epistemology.

Beliefs oriented research can be traced back to the 1920s, but it was not until the 1970s when it was fuelled by a shift in paradigms, from a focus on teachers’ behaviours to a focus on teachers’ thinking and decision-making processes (Thompson, 1992), that it became widespread among scholars of education.

#### Beliefs and teachers’ knowledge

In an overview by Pajares (1992) it is obvious that research concerning teachers’ epistemological beliefs is perceived as “a messy construct”. The messiness is partly due to the fact that different concepts like construct, personal theories, attitudes, beliefs and knowledge (Hashweh, 2005; Kagan, 1992; Pajares, 1992) are used interchangeably among researchers.

Alexander et al. (1995) studied differences between peoples’ beliefs and peoples’ knowledge, linking it to the number of years in education. The outcome of their research showed that people use the words (belief and knowledge) interchangeably and therefore, they could be perceived as the same.

> “Are knowledge and beliefs, in actuality, synonyms marking the same semantic territory, or are they antonyms denoting orthogonal dimensions of human understanding? Or, is it possible that the concepts of knowing and believing share a common ground, while still retaining some unique and unshared terrain?” (Alexander and Dochy, 1995, p.415)

They suggested a cross-cultural investigation to uncover diverse perspectives on knowledge and beliefs to uncover similarities within communities, while they emphasized the existence of dissimilar views of these two concepts within the same cultural group.

---

*Ernest (1991) dissolves the distinction between knowledge and beliefs as he re-labels the two concepts in relation to the social aspect, individual constructions (subjective knowledge) or social constructions (objective knowledge).

*For an explanation of the origin and flavours of personal epistemology I refer to Hofer (2001).
Belief systems and classroom practices

The need to cluster different beliefs into systems is linked to the discovery that the same individual can hold contradictory beliefs (Leatham, 2006). A belief system could be described as a metaphor for an individual’s organisation of beliefs in similar ways as conceptual knowledge is conceived in cognitive structures (Green, 1971, in Thompson, 1992). Some research claims that belief systems are more episodic in nature than knowledge systems, as they tend to connect to specific situations or experiences (Abelson, 1979), which would explain why knowledge is situated and dependent on context (Leatham, 2006).

A great deal of empirical evidence has established the significance of beliefs for understanding teacher behavior (e.g., Calderhead, 1996; Clark and Peterson, 1988; Fajares, 1992). Several scholars suggest that teaching practices are strongly associated with teachers’ beliefs about teaching and learning (e.g., Hofer and Pintrich, 1997; Kagan, 1992; Hofer, 2004), including interaction with students, instructional materials, and instructional design (Kagan, 1992; Song et al., 2007).

However, research also describes inconsistencies between teachers’ beliefs and their classroom practices (e.g., Schraw and Olafson, 2002; Ertmer, 2005; Fang, 1996), which could be explained by contextual factors; teacher’s ability to apply their beliefs in practice do not match answers to a self-report conducted in a research context (Fang, 1996).

Teachers’ epistemological beliefs

Research about teachers’ epistemological beliefs in general seems to have a more holistic approach compared to research about students’ epistemological beliefs, which is why world views were introduced (Schraw and Olafson, 2008). Therefore multiple dimensions, to depict teachers’ beliefs or beliefs system, have been suggested (Schommer-Aikins, 2004; Schraw and Olafson, 2008; Olafson and Schraw, 2006) to differ between epistemological beliefs, in relation to specific subject domains, and holistic epistemological stances.

According to Schraw and Olafson

“it is important to distinguish clearly between epistemological beliefs and epistemological world views. The former consist of specific beliefs about a particular dimension of knowledge such as its certainty, simplicity, or origin. The latter consist of a set of beliefs that collectively define one’s attitudes about the nature and acquisition of knowledge. Each adult presumably has a set of epistemological beliefs that are included within an epistemological world view” (Schraw and Olafson, 2002, p.102)

Schraw and Olafson (2002) identify/compare three epistemological world views in their study of the instructional implications of teachers’ beliefs about knowledge and learning. The world views are 1) The realist, 2) The contextualist and 3) The
relativist, all with their associated nine beliefs; about knowledge, about curriculum, about pedagogy, about assessment, about reality and standards for judging truth, about constructivism, about the role of the teacher, about the role of the student, about the role of peers. Below is a short summary of each distinctive world view is presented:

1. In the realist world view there is an objective body of knowledge that is acquired via transmission and reconstruction. Teachers with the realist world view perceive students as passive recipients. The knowledge at stake is pre-established and agreed upon by experts. To acquire high levels of skill you have to work systematically under the governance of the teacher.

2. In the contextualist world view learners construct understanding in a collaborative context where the teacher acts as a facilitator. The learning process is more important than the knowledge it constructs as knowledge will change over time, wherefore emphasis is on students’ skills so they learn to acquire knowledge on their own. Teachers holding this world view encourage peer work and expert scaffolding. Authentic co-operative assessment is desirable.

3. In the relativist world view each learner constructs their individual understanding of the same subject content. Teachers with the relativist world view commonly design environments where students are encouraged to think independently. Self-regulation is an objective in itself, as peers are important as promoters of self-regulation. Criterion-based assessment is used on an individual basis.

In later publications Schraw and Olafson (2006) summarise these three world views into the concept of ontological beliefs. The model they suggest consists of two dimensions and a four-quadrant scale, depicting the teachers as relativists or realists in each dimension; one for teachers’ epistemological belief and one for teachers’ ontological belief. In the study at hand the model is used to propose the existence of commonality and specific beliefs caused by the subject domain.

7See Schraw and Olafson (ibid.) for a description of each world-view.
Chapter 3
Methodology

The investigation combines two perspectives; a historical perspective and today’s teacher perspective. The methodologies in these two perspectives differ, as the historical perspective shows findings of preserved documents, and today’s teacher perspective is a result of what four seminars and associated questionnaires revealed. In the following chapter the methodologies will be presented and discussed. The reader is referred to the papers Informatics and programming in Swedish Upper Secondary School and Teachers’ Beliefs Regarding Programming Education, for a thorough description of details.

3.1 The historical perspective

The following section describes methods used in the reconstruction of the curriculum. Considerations and delimitations are discussed, as they have defined the discoveries as much as the search for documents and people involved. The section begins with a short introductory text about the use of arenas as a way of structuring the historical data.

Arenas - a way to structure the data

The intersection of curriculum history in informatics and today’s teachers’ beliefs is in this study explored through studies of the continuous curriculum development process where both the macro and the micro levels are addressed in the search for existing determinants in the transformation of a school subject (Linde, 1993). This has been a common approach in educational research in Sweden since the 1960s, which e.g. brought forward the frame-factor theory (Dahllöf, 1969; Lundgren, 1972, 1977, 1983). The distinction between micro and macro levels were later made visible by the distinguishing of different arenas (Linde, 1993; Lindensjö and Lundgren, 2000; Klasander, 2010). The arena concept offers a rational way to describe the development process and reification of a curriculum at different levels;
the arena of enactment, the arena of transformation and the arena of realisation. Linde (1993) refer to the following arenas:

- **The arena of enactment** concerns the curriculum development process, from the initial steps to the final enactment of a curriculum document.
- **The arena of transformation** particularly concerns teachers’ reification work of concepts described in the curriculum. Teachers are considered gatekeepers of what can be offered and established in the classroom. Principals and school publishers also belong to this arena.
- **The arena of realisation** concerns the acts in classrooms according to the intentions written in the curriculum.

In this study the first and the second arenas are the ones primarily focused on.

**Document analysis and interviews with key persons**

The National Archives (RA) and Stockholm City Archive (SSA) have been valuable resources in the reconstruction of the development process, as they possess an abundance of documents from the time of interest. Specific effort has been made to understand on what arena/level, and by whom the Swedish informatics curriculum was processed for the first time. Historical documents from archives depicting the curriculum development were scanned and compiled to digital volumes to enable study of the interaction between and within different groups at the National Board of Education (NBE) and upper secondary schools. Contemporary educational journals from the 1970s, 1980s and the 1990s were scrutinized to understand the overall process and communication within the development process.

Parties involved in the curriculum development process were identified and interviewed if still alive; around ten interviews were conducted. Resources (proposals, government bills, decrees, submissions, reports and written communications) from the governmental library, Riksdagen Library in-house and on-line, have been studied to understand in what way the Ministry of Education (ME) was involved in the development process. The interconnection between the ME and the personnel in schools has been observed to be of a complex nature, as some teachers and principals worked at many different levels, including at the level of the ME. According to Marklund (1987, p.70-71) many countries in Europe cannot exhibit anything like the Swedish NBE (except Finland), as countries commonly commission curriculum development to the administrative department of their ME.

In this study the first and the second arenas are the ones primarily focused on.

**Document analysis and interviews with key persons**

The National Archives (RA) and Stockholm City Archive (SSA) have been valuable resources in the reconstruction of the development process, as they possess an abundance of documents from the time of interest. Specific effort has been made to understand on what arena/level, and by whom the Swedish informatics curriculum was processed for the first time. Historical documents from archives depicting the curriculum development were scanned and compiled to digital volumes to enable study of the interaction between and within different groups at the National Board of Education (NBE) and upper secondary schools. Contemporary educational journals from the 1970s, 1980s and the 1990s were scrutinized to understand the overall process and communication within the development process.

Parties involved in the curriculum development process were identified and interviewed if still alive; around ten interviews were conducted. Resources (proposals, government bills, decrees, submissions, reports and written communications) from the governmental library, Riksdagen Library in-house and on-line, have been studied to understand in what way the Ministry of Education (ME) was involved in the development process. The interconnection between the ME and the personnel in schools has been observed to be of a complex nature, as some teachers and principals worked at many different levels, including at the level of the ME. According to Marklund (1987, p.70-71) many countries in Europe cannot exhibit anything like the Swedish NBE (except Finland), as countries commonly commission curriculum development to the administrative department of their ME.

---

1. See Figure 5.2 in Results for an overview of the development process of the subject matter of informatics.
2. See Figure 5.2.
3. A resume of the most prominent people involved in the curriculum development is found in Emanuel (2009a; 2009b).
Methodological considerations

3.1. THE HISTORICAL PERSPECTIVE

Archived studies

According to personnel at the RA, the NBE archives are very crudely organised, and would need a re-organisation to live up to present-day standards. Therefore, specific attention in the investigation of documents has been given in relation to upper secondary school and vocational education in ADP (VADP). During the 1970s and 1980s, administration of the Swedish NBE expanded into different bureaus and divisions, whereas the following bureaus have been at focus:

1. S-division for education in school (1972 to 1982) with Bureau S3, for upper secondary school with responsibility for the technical and the natural science domains.
2. G-division for upper secondary school (1982 to 1991) with Bureau for curriculum development (LPU), Bureau for course development and syllabus development (KPU and KPU 2).

Other archives and sources

The ME archives have been another source of information. The ME archives show the curriculum process, in particular from 1986 onwards when two working groups focusing on informatics education were initiated within the ME:

1. The informatics application group (Swedish: Dataprogramgruppen, DPG), 1985 to 1988. A group initiated by the ME, with intentions to facilitate computer technology in school (Emanuel, 2009b; Lindberg, 1988, Utbildningsdepartementet, 1988) and supply educational software in different subject domains. The group were later transferred to the NBE.
2. The informatics education group (Swedish: Datautbildningsgruppen, DUG). In 1986 they proposed a program of action (Datautbildningsgruppen, 1986) for the future of informatics education, with some important considerations for how school would relate to informatics, involving vocational experts in the formation of future informatics curriculum and its associate modules. The unanimous recommendations show the emphasis the ME places on the inclusion of vocational aspects in the informatics curriculum.

Documents from the CSB archives revealed information on why two specific schools (Berzeliuskolan and Bandhagens gymnasium) and associated teachers became influential in the beginning of the curriculum development process. The SCB

Methodological considerations

3.1. THE HISTORICAL PERSPECTIVE

Archived studies

According to personnel at the RA, the NBE archives are very crudely organised, and would need a re-organisation to live up to present-day standards. Therefore, specific attention in the investigation of documents has been given in relation to upper secondary school and vocational education in ADP (VADP). During the 1970s and 1980s, administration of the Swedish NBE expanded into different bureaus and divisions, whereas the following bureaus have been at focus:

1. S-division for education in school (1972 to 1982) with Bureau S3, for upper secondary school with responsibility for the technical and the natural science domains.
2. G-division for upper secondary school (1982 to 1991) with Bureau for curriculum development (LPU), Bureau for course development and syllabus development (KPU and KPU 2).

Other archives and sources

The ME archives have been another source of information. The ME archives show the curriculum process, in particular from 1986 onwards when two working groups focusing on informatics education were initiated within the ME:

1. The informatics application group (Swedish: Dataprogramgruppen, DPG), 1985 to 1988. A group initiated by the ME, with intentions to facilitate computer technology in school (Emanuel, 2009b; Lindberg, 1988, Utbildningsdepartementet, 1988) and supply educational software in different subject domains. The group were later transferred to the NBE.
2. The informatics education group (Swedish: Datautbildningsgruppen, DUG). In 1986 they proposed a program of action (Datautbildningsgruppen, 1986) for the future of informatics education, with some important considerations for how school would relate to informatics, involving vocational experts in the formation of future informatics curriculum and its associate modules. The unanimous recommendations show the emphasis the ME places on the inclusion of vocational aspects in the informatics curriculum.

Documents from the CSB archives revealed information on why two specific schools (Berzeliuskolan and Bandhagens gymnasium) and associated teachers became influential in the beginning of the curriculum development process. The SCB
archives revealed few documents of importance to this study. However, further investigation into economical documents would most likely reveal information about how different schools invested in computer technology, which in turn could show the dependence on technology for education in informatics. This study does not include such an analysis.

Considerations regarding the delimitation of the study

In order to make the research task manageable, a limitation of the investigation has been made regarding the level of education and the timespan. The delimitation in time of this study is linked to the revision of curriculum that started at the end of the 1980s, which brought about the curriculum Lp94, later published in 1994. Previous research regarding informatics in schools was scrutinised (Nissen and Riis, 1985; Riis, 1987, 1991; Nissen et al., 1991; Jedeskog et al., 1991; Riis, 2000; Nissen et al., 2000). It was found that previous research mostly focused on the subject matter of computing (Swedish: Datalära). The research about the rise and fall of computing/Datalära, mainly highlighted the diffusion of computer hardware in school. These findings were not considered relevant to this study as the computing studied was offered during a period of four years at lower secondary school during the mid-1980s, hence not mirroring the content of upper secondary school computer programming.

The transfer of VADP to tertiary education has also been a concern. Literature and archive documents show the existence of VADP as higher special courses in 13 different cities (Huvudmannaskapskommittén, 1980) during the 1970s, as they were changed from higher special courses in secondary school, to be part of tertiary school. This transfer was made during the same time as experimental work in upper secondary education was implemented. Articles have been published in the teachers’ association membership journal (Swedish: Allmannsriktade meddelande för lärarföreningen ALF) to investigate former educators’ experiences of VADP.

Ethical considerations

Names of people involved in the curriculum development process have been used, while they were officially employed by NBE and government. Whenever people were interviewed, they were also informed about intentions of how information will be processed and used for research. To fulfill the ethical principals of information and use (Vetenskapsrådet, 2002) each informant were asked whether their information could be used for research. However, information from interviews have been used to the extent that it confirmed what archive documents describe.

The CSB in Stockholm (SCSB) was scrutinised in relation to Bandhagens gymnasium in order to understand what sort of documents that were preserved.

The journal is offered only to members of the teachers’ association.
3.2 Today’s teachers’ perspective

The following section shows methods used to investigate programming education from a teachers’ perspective. To facilitate the understanding of the selection of informants the first sub-section however, shows statistics picturing the number of computer programming teachers in Sweden today.

Computer programming education in upper secondary school - number of schools and teachers

In the year 2000, computer programming was offered to all students. According to statistics, a peak was reached in 2002. Statistics from the Swedish National Agency of Education (SNAE) show that the number of students taking computer programming during 2006 to 2008 was about 6,000 students each year, coming from different programmes such as Electricity (ECP), Technology (TEP) and Natural Science (NSP). This investigation focuses on specific teachers with experiences from at least two levels of programming courses.

There is no official register that offers the correct number of practicing programming teachers in upper secondary school. Based on the assumption that one school with at least six students attend a course in programming the number of teachers has been estimated to 300; correlates to one or two teachers per school.9

The number of schools, in 2006 to 2008, that offered one or more of the three programming courses, is shown in Table 3.1; this means that there are more than 300 computer programming teachers in Sweden. Statistics from the SNAE databases show the existence of 341 schools offering courses in computer programming. However, a thorough investigation exposes that a majority of schools have small numbers of students attending the courses.10

3.2.1 Today’s teachers’ perspective

Table 3.1: The number of schools that offered computer programming courses on the levels A, B and C, during 2006 to 2008.

<table>
<thead>
<tr>
<th>A</th>
<th>A+B</th>
<th>A+B+C</th>
</tr>
</thead>
<tbody>
<tr>
<td>341</td>
<td>233</td>
<td>158</td>
</tr>
</tbody>
</table>

9There are other students attending courses in programming, from other programmes, but they are in minority.
10Swedish schools are not obliged to offer programming courses with a teacher. It depends on the number of attending students. The data from SNAE’s database reveals the existence of 50 schools with 30 or more students, which most presumably would imply two teachers at each school. There is an economical breaking point however, where it is not applicable to offer a teacher-led course. The number of six students is assumed based on information from discussion with a school principal. The breaking point implies the existence of 250 schools that offered regular courses with a teacher in programming.

11Few schools reach high numbers of students. Out of the 341 schools only 35 schools have more than 30 students and 5 schools have more than 70 students each year.
12Swedish schools are not obliged to offer programming courses with a teacher. It depends on the number of attending students. The data from SNAE’s database reveals the existence of 50 schools with 30 or more students, which most presumably would imply two teachers at each school. There is an economical breaking point however, where it is not applicable to offer a teacher-led course. The number of six students is assumed based on information from discussion with a school principal. The breaking point implies the existence of 250 schools that offered regular courses with a teacher in programming.

13Few schools reach high numbers of students. Out of the 341 schools only 35 schools have more than 30 students and 5 schools have more than 70 students each year.
CHAPTER 3. METHODOLOGY

The transformation of informatics curriculum

Research in teachers’ beliefs is commonly based on the assumption that teachers’ deeds is a product of their beliefs (Hofer and Pintrich, 1997; Kagan, 1992; Hofer, 2004; Song et al., 2007), while it should be remembered that the interplay between those two is of a complex nature (Beswick, 2005a). In the study, questionnaires have been used to depict the intentions and underpinnings for teachers’ transformation of curriculum to teaching and practice in classroom, which is not considered as the actual outcome on the arena of realization.

The diversity of existent beliefs among teachers was of great value for the investigation, wherefore questionnaires was considered a proper choice of instrument to picture the mindset of the teachers community. Major rationales to offer seminars were discovered while teachers showed an interest to participate in forthcoming seminars. Another rational for seminars was found in the distribution and management of questionnaires.

Selection of informants and the mail list

In the early stages of the project there was a need to establish contact (an e-mail list) with computer programming teachers in Sweden. Approximately 1050 school webpages were scrutinised. Based on the following two criteria, a list of 196 potential upper secondary schools and 206 teachers was drawn up:

- Schools offering the Natural Science Programme (NSP) and/or the Technology Programme (TEP).
- Schools that offered computer programming courses in at least in two out of three levels.

The number of Swedish programming teachers is (as mentioned above) estimated to be about 300 individuals. The original list of computer programming teachers was limited in its sample size, but it could be expected to contain teachers holding significantly differing beliefs about teaching and learning computer programming. However, the list was expanded in successive seminars with teachers (from approx. 200 to 300).

In total, 90 individual teachers attended one or more of the four seminars and 45 teachers attended two or more of the seminars. Each seminar was attended by 20 to 50 teachers. That initial list continuously evolved and today embraces approximately 250 computer programming teachers, from all over Sweden, who together constitute the Swedish Informatics Teachers’ Network (SITSNET).

These webpages were identified based on information from the school database offered by the SNAE and Gymnasieguiden at http://www.gymnasieguiden.se. Retrieved 2012-07-10.
A series of seminar

From the very start of the project, the task to capture the diversity of teachers’ instruction was considered to be a main issue for the study. For this purpose, a series of seminars was initiated. The four seminars were offered regularly from 2009 to 2011 focusing on didactic/pedagogical aspects of programming education, with the intention to vitalize teachers’ reflection about programming education. The seminars were offered in partnership with Stockholm University, the Royal Institute of Technology (KTH), Uppsala University, researchers from UpCERG, Microsoft Sweden AB, curriculum experts associated to the Swedish National Agency of Education (SNAE) and teachers. For a detailed description of each seminar the reader is referred to the second paper, Teachers’ Beliefs Regarding Programming Education. The seminars became a valuable resource and an inspiration in the search for future seminar activities (e.g. content of importance to the teachers as well as educational problems faced by teachers).

Questionnaires

The diversity of existing beliefs among the participating teachers was of great value to the investigation, therefore questionnaires were considered a proper choice of tool to reveal the mind-set of the teachers. The design of the questionnaires (See Appendix B) was a result of the seminars, as they became important to better understand teachers’ perceptions of teaching and learning programming. The main questionnaire (Quest1) was sent to 250 teachers nationwide, to 1) reveal what it is like to teach programming, 2) understand teachers’ interest in participating in in-service training seminars and/or 3) collaborate in networks. It was designed to discover the teachers’ perception in relation to four domains; programming/programming languages, education, learning and networks.

The first questionnaire (Quest1) became crucial, as it showed the collective mind-set of programming teachers. Data from teachers’ responses was analysed to show themes of interest in successive seminars.

The other three questionnaires (Quest2 to 4) were administrated in relation to the seminars. The first seminar (Sem1) became a start and offered information about the research project and the development of a new curriculum in computer programming. The second questionnaire (Quest2) was presented and managed during the second seminar (Sem2). The outcome of the second and third questionnaires (Quest3), was presented during the third seminar (Sem3), focusing on the question “Why programming at upper secondary school?” and “What type of programming?” The fourth seminar (Sem4) was designed to offer a picture of what higher education values as important and what programming knowledge upper sec-

12 The analysis were inspired by inductive analysis (Lincoln and Guba, 1985; Miles and Huberman, 1994).
13 The third questionnaire was distributed to the participants, and collected by mail beforehand.
3.3 Ethical considerations and trustworthiness

The research design is built on trustworthiness and its relation to applicability for the individual teacher, the community of teachers and/or society as a whole. Ethical considerations are accounted for in accordance with Gustafsson, Hermén, and Peterson (2005) who distinguish two aspects of considerations; researcher-ethical considerations and research-ethical considerations. These two aspects will be discussed in relation to the concepts embraced by trustworthiness (Denzin and Lincoln, 2000) and the ethical principles suggested by the Swedish Research Council (Vetenskapsrådet, 2002) have been followed; information, consent, confidentiality and use.

Ethical considerations

The following text shows the principles

1. Information: During the whole research process (during seminars and when distributing the questionnaires) cover letters and information were presented to inform about the anonymity and the confidentiality that would be used when processing the information.  

2. Consent: The principle of consent could be questioned, but there never existed any sort of obligation to submit answers to the questionnaires for participation in seminars. Each questionnaire was offered in relation to a seminar, except the second (Quest2).  

3. Confidentiality: It is always easier to be critical with hindsight, wherefore I as a researcher have had to be careful with how I handle the informant’s values and experiences. During the whole investigation process I have been in contact with a huge amount of people who are/were committed to what they believe is/was necessary and important. These peoples’ stories and witness accounts of engagement made an impression, which has been of great value to me and my research. Therefore, it has been important to live up to, and

---

1See Appendix B.  
2The second questionnaire was distributed during the seminar while informants were informed of the research intentions and the confidentiality that would be used when processing information.
pay back the trust informants showed me. During the whole research process.
(in seminars as well as within questionnaires) letters and information was
delivered to inform about the research intentions and the confidentiality that
would be used when processing the information.

4. **Use**: The information offered by informants has been used for research pur-
poses, this study, papers and future research work. The informants were
informed of this.

### Trustworthiness

A common denominator among computer programming teachers is the need for
professional development, as this is sparsely offered at schools. The investigation
therefore became dependent on teachers' willingness and ability to explore their
epistemological beliefs and shortcomings. This has been a methodological poten-
tial for the study at hand, as it was also identified as an opportunity to do something
more than just conduct the research process with recruitment of informants, data
collection and analysis. The need for such a stance emerged early in the research
process. However, such a research design also offers ethical considerations where
"the interconnectivity between production of knowledge at the ethics of production"
(Trainora and Bouchardb, 2012, p.3) becomes important. According to Lincoln
and Guba (Denzin and Lincoln, 2000) trustworthiness research involves establish-
ing:

1. **Credibility**: In this study the research design was based on the idea that
teachers hold a diversity of different beliefs about the educational situation,
but commonality would appear if the beliefs were compared. In that process,
a diversity of beliefs was believed to be the research outcome. However, the
methodology used will never explain any individual's beliefs, as reflections are
collected during different seminars. One third of the teachers community
expressed a sincere interest in participating in the seminars early on in the
research process. It was therefore assumed that these informants represented
those with a specific interest in teaching and learning as well as those in-
terested in socialising with peers with similar questions and experiences. A
group of teachers on site would therefore fulfil a social need as well as offering
opportunities to share what was considered important.

It could be questioned whether or not the participants' beliefs mirror di-
versity that is broad enough, and if the data mirrors teachers' practice, as
the research design opened up for investigation as much as the participants
could share. In a research review done by Beswick (2003; 2007), mismatches

14 The discussion is influenced by reciprocity and rigorous reflexivity (Trainora and Bouchardb, 2012) in research design.
17 The individuals did not participate in all seminars, while some joined later and some only attended the first seminar.

between provided data from teachers concerning their beliefs and their practices were discovered. Beswick suggests that differences in context matters to the research outcome. The contextual nature of teachers’ beliefs is therefore considered important (Beswick, 2002, 2004) to this study. All questionnaires except Quest2 were done beforehand, which was considered beneficial because of the school context. Some attempts were made to enhance the contextual aspects of the need to make the methodology used trustworthy.

2. Transferability: The methodology used in this study elicits the fact that curriculum development in informatics is the outcome of work carried out by many teachers and experts, and is officially engaged in a continuous process that started at least 40 years ago. An understanding of the development and the underpinnings that shaped today’s curriculum and instruction was believed to be of huge value to teachers’ reflection in practice and to future curriculum developers. A blog was therefore published to communicate results to the community. During one of the first seminars, the teachers expressed their disinterest in the research field of didactics in computer science, while social aspects and exchange of instructional experiences was considered of higher value.

3. Dependability: The methodology used depends on the interplay between teachers, researcher, and context of different sites, where the study as well as the researcher’s learning is the outcome. Examples of instruction were searched for among individual participants, especially those that brought something different to the teachers’ community. The seminars offered potential situations in which such differences could be revealed.

4. Confirmability: The seminars were believed to offer an environment for research in interaction with teachers as well as it would take too much effort and time to dispute my own beliefs regarding programming education. To stimulate reflection, the four seminars were offered at different sites; two academic ones (Stockholm University and KTH), an industrial one (Microsoft) and one upper secondary school. With such a design, teachers were exposed to at least three different perspectives; industry, university, and school. The researcher and author of this study also became the initiator of the seminars. This was considered problematic, wherefore board members for the formation of a teacher’s network that would manage the seminars were searched for. It was considered important in the building of trust between teachers and researcher.

---

18The blog never became an instrument of importance. It was however considered an important step in the interaction with the teacher community.
19At the time I had been teaching computer programming for at least 10 years.
Chapter 4

Summary of Articles

The following chapter gives a brief presentation of each paper and some of the results. Conclusions and results are discussed in the next chapter. For a comprehensive understanding the reader is referred to the papers.

4.1 Paper 1: Informatics and programming in Swedish upper secondary school - vision and experimental work during the 1970s and 1980s

Submitted to Computer Science Education.

The development process, the final enactment and the early implementation of the informatics curriculum is described in the first paper. The research question addressed reads: How was informatics curriculum developed in Swedish upper secondary school during the 1970s and the 1980s? To unravel the interaction between schools, teachers and the NBE, documents regarding the curriculum development process available at the National Archives (RA) and city archives have been studied.

Findings reveal that the Natural Science Programme (NSP) was remodelled during the mid-1970s. While holding a pragmatic educational approach, the new informatics curriculum was believed to attract more students. The new curriculum was also believed to offer enhanced learning in Mathematics and Physics, as logical computer technology and these two subject matters were believed to hold similar rational intentions.

The interaction between and within different groups at the National Bureau of Education (NBE) was studied to better understand the early steps taken towards the enactment of the informatics curriculum. Some teachers and principals worked at many levels in the school system. Three upper secondary schools in particular, had previous experience of vocational education in ADP; Hedbergska skolan in Sundsvall, Carlbergskolan in Västerås and Berzeliuskolan in Linköping, whereof the last one, Berzeliuskolan, became significant in the enactment of the informatics curriculum.
A pre-existing vocational education during the 1960s and 1970s became the blue-print in the formation of the new subject matter. However, other schools also contributed to the development. At e.g. Sunnerboskolan in Ljungby (Björk et al., 1975; Björk, 1974, 1977, 1983; Kollerbaur, 2005), a working group of enthusiastic teachers teaching different subject matters showed a tremendous driving force in how to use computer technology for interaction and enhanced learning. This enthusiasm could be traced back to the early 1970s when curriculum developers in Mathematics and Physics (Utbildningsdepartementet, 1971; Kollerbaur, 2005) saw computers as tools for a dynamic curriculum. Computer programming was first introduced to the NSP within a specific alignment for informatics. Later it was introduced in other programmes as well, within experimental work. The outcome of this introduction was commonly perceived as unsatisfactory, as computer programming was time-consuming and not appropriate for students with a small interest in the techno-centric environment that computers could offer during the 1980s.

The implementation of the school subject informatics exposed another problem, caused by the heritage from vocational education in ADP. The societal concern and system development were believed to be important for understanding the true value of computer programming. This concern became an issue for the curriculum development group as teachers generally did not have the appropriate competence to teach system development/systemisation. Besides, the content of system development/systemisation was experienced as vocational and non-technical, which made the content appear somewhat strange and not appropriate for the NSP. However, these opinions seemed to vanish with time, as the computer application market offered educational software more appropriate for subject matters like Mathematics and Physics. In 1985 the Swedish association of computer teachers (Swedish: Svenska datalärarföreningen) stated that

“...technologically oriented modules like computer programming should decrease in size in favour of information technology, system development and project collaboration.” (Datautbildningsgruppen, 1986, Appendix 3, p.5)***

The time when computer programming was believed to foster logical and analytical reasoning in schools had now passed. Computer technology itself offered new educational technologies with new hopes for enhanced learning and better understanding of concepts in Natural sciences and Mathematics. Computer programming became an optional knowledge domain for students with a specific computer technology career in mind. In the curriculum reform of 1994, Lp94, computer programming was re-introduced as a subject matter on its own, after the trade organisation of electricity put pressure on the Swedish National Agency of Education (SNAE) to offer computer programming courses.
4.2 Article 2: Teachers’ beliefs regarding programming education

Accepted for publication in Technology Education – Practicing Teachers Researching Teachers Practice (201x).

The second paper explores the beliefs of today’s programming teachers from the following research question: What beliefs do programming teachers express regarding teaching and learning computer programming in upper secondary school? The article revolves around data collected during a series of seminars focusing on upper secondary programming education. Between 20 and 50 of the teachers in the study attended one or more of the seminar/s. At each seminar, a questionnaire designed to elicit teachers’ beliefs about aspects of importance for their instructional design and students learning was given to the teachers/informants.

The responses were analysed (Lincoln and Guba, 1985; Miles and Huberman, 1994), to find common themes underpinning the work of computer programming teachers. The analysis showed four themes in relation to teachers’ beliefs about learning and teaching:

1. Students’ individual connective time
2. Teachers’ pedagogy
3. Students’ abilities
4. Students’ interest and motivation.

Further analyses reveal that the assessment process is crucial to teachers’ choice of instruction strategies. This is particularly valid in the beginners’ course, where collaboration among students (peer-learning) is often practiced, and where skills essential to working in groups are commonly considered not to be important. A common practice among secondary school teachers of technology (Sidawi, 2009; Hennessey and Murphy, 1999). In conclusion it could be said that two distinctive instructional patterns exist among teachers: individual support, and instruction for experience of learning. This fully corresponds to previous research (Boulton-Lewis et al., 2001).

The analysis of data exposed that a majority of the teachers in the study express a number of expectations concerning their students’ abilities; specific abilities such as logical and analytical thinking are emphasized as important for successful learning, while the ability to work in a group and to communicate is perceived as beneficial but not of any concern during the assessment process. The paper raises the question of whether teachers perceive abilities as fixed and inborn (naïve belief) or something that students could acquire with some effort (sophisticated belief). Findings suggest that a majority of the teachers hold a naïve belief. Findings also show that the teachers in the study focus on the individual, constructivist based learning which, according to Schraw and Olafson (2008), indicate that the teachers...
in the study commonly hold on to relativistic world-views. To understand whether
computer programming teachers hold a realistic or a relativistic epistemology, fur-
ther investigation would be needed.

A specific feature appeared in the study regarding the concept of pedagogy.
Depending on how the questions were phrased, teachers’ belief in pedagogy varied
from question to question. When teachers were asked “Do you consider pedagogy
as important for programming education” 95% perceived pedagogy as important,
while in another session only 5% perceived pedagogy as an important factor for
students’ learning. On the other hand only 23% of the teachers perceived a degree
in pedagogy as important when asked “What kind of education do you perceive as
necessary for being a teacher in programming?” An interpretation of these different
percentage numbers reveals the existence of a relationship between instruction and
pedagogy, but it does not necessarily coincide with teaching (Nuthall, 2004).
Chapter 5

Results

In this chapter the results from the two papers are presented. Findings of importance are described and summarised while details and empirical groundings are described in the papers.

5.1 The historical development of the informatics curriculum during the 1970s

Teachers' involvement

There were obviously a number of different parties involved in the curriculum development process. This investigation primarily focuses on initiatives suggested by experts within the NBE and/or by selected teachers. Findings reveal the existence of a small number of schools and teachers who, of their own initiative, partook in the development when informatics was first implemented as a school subject during the 1970s (Rolandsson, 2011). Most of the teachers involved in the curriculum development process were teachers within Natural Sciences and/or Mathematics who embraced visions regarding how to develop and transform their subjects with informatics and computer programming. In e.g. 1969, a local work group of teachers made a proposition regarding how teaching and learning could be changed within educational technology (Fagerström, 1973; Utbildningsdepartementet, 1971). Their vision included the use of desktop calculators in Mathematics and Physics education:

1 We [the teachers] presume that better understanding would occur if students were systematically trained to work with mathematical forms.

2 From Berzeliuskolon in Linköping, Pauluskolon in Malmö and Östra gymnasiet in Umeå.
Subject content and work methods

In the development process of the first informatics curriculum, system development/systemisation became an issue in computer programming education. During this period few schools offered expertise in informatics (which at the time was mainly about computer programming). Commonly, only schools with established links to (local) industry could offer knowledgeable teachers with competence in computer programming. Teacher competence therefore became a critical factor.

When experimental work was initiated at upper secondary school during the 1970s experiences from vocational ADP (VADP) were at focus. An issue emerged as teachers from Natural Sciences and Mathematics would substitute teachers from VADP. The investigation reveals that the complexity of teaching system development/systemisation brought up the need for stereotypical examples to make the knowledge applicable to instruction. In the beginning, system development/systemisation became associated with its existing vocational counterpart, wherefore the NSP was sometimes considered vocationally influenced. Later, system development/systemisation was believed to replace computer programming in schools.

Subject content and work methods

In the development process of the first informatics curriculum, system development/systemisation became an issue in computer programming education. During this period few schools offered expertise in informatics (which at the time was mainly about computer programming). Commonly, only schools with established links to (local) industry could offer knowledgeable teachers with competence in computer programming. Teacher competence therefore became a critical factor.

When experimental work was initiated at upper secondary school during the 1970s experiences from vocational ADP (VADP) were at focus. An issue emerged as teachers from Natural Sciences and Mathematics would substitute teachers from VADP. The investigation reveals that the complexity of teaching system development/systemisation brought up the need for stereotypical examples to make the knowledge applicable to instruction. In the beginning, system development/systemisation became associated with its existing vocational counterpart, wherefore the NSP was sometimes considered vocationally influenced. Later, system development/systemisation was believed to replace computer programming in schools.

In the final version of the curriculum, approximately 200 hours were designated to instruction in informatics, and system development would take up half that time. The problems related to the fact that instructions were to be given by teachers who had never experienced the concepts of system development, were not solved. The initial idea discussed during the earlier phases of the development process was that informatics should be taught by teachers competent in system development (e.g. someone from VADP or a teacher with in-service training). This vision did for obvious reasons have to be adjusted to the ‘real world’ situation in schools where few teachers were qualified. Instead most teachers had to rely on stereotypical examples in school literature. Berzeliussskolan, a secondary school with experienced teachers from industry, is one of the exceptions where informatics education had been offered since the late 1960s. The following schools collaborated with local industries: Hedbergska skolan in Sundsvall with forest industries, Carlérska skolan and Zimmermankskolan in Västerås with ASEA and Berzeliussskolan in Linköping with DataSach. One of the first Swedish initiatives to offer computer programming education at upper secondary school was identified as the collaboration between a gymnasium in Västerås and ASEA Education AB (Rolandssen, 2011).
5.2. THE HISTORICAL DEVELOPMENT OF THE INFORMATICS CURRICULUM DURING THE 1980s

The potential of what computers could offer besides computer programming and system development was also recognised by teachers at Sunnerboskolan in Ljungby, where a group of teachers shared a vision of education enhanced by computer technology (Fagerström, 1973). A third secondary school, Bandhagens gymnasium, showed a special interest, during the mid-1970s, in transforming the Natural Science Programme (NSP). This became an important incitement for the implementation of a vocational subject in informatics for upper secondary school.

5.2 The historical development of the informatics curriculum during the 1980s

Teachers’ involvement

The 1983 informatics curriculum was the result of efforts made within different NBE work groups. The work groups based their conclusions on experimental work performed during the 1970s. Therefore it is interesting to read the response from the Swedish association of computer teachers (Swedish: Svenska datalärarföreningen), where they suggest that computer programming should be replaced by system development:

“...technologically oriented modules like computer programming should decrease in size in favour of information technology, system development and project collaboration.” (Datautbildningsgruppen, 1986, Appendix 3, p.5)**

This is a significant and important marker from the teachers themselves. A pioneer in informatics education (Rolandsson, 2011; Kollerbaur, 2005) and also the NBE’s expert in informatics curricula, Lars-Eric Björk wrote a year later, in 1987, that

“Should programming remain in the curriculum [at all]? The pendulum concerning the [problem] has swung from one extreme to another. There are many voices today who state that programming should be extracted from school.” (1987, p.14) ***

---

1 According to an interview with the then principal, Torsten Odmark the school was under heavy pressure and felt they had to offer a concept that was appealing to a broader group of students.
2 According to interviews with teachers and the local principal active at Berzeliusskolan. Apart from teachers with experience from vocational education in ADP it was common to use teachers from DataSaab in technical subjects.
In research (Kansanen, 1995), according to the majority of teachers in the study, the concept of pedagogy is broad and unspecific, wherefore it is hard to adopt teachers' pedagogical approaches and abilities.

Subject content and work methods
During the 1980s system development/systematisation was once again brought up in the arena of enactment. This time it was suggested to replace computer programming, as programming was perceived as too technical and time consuming and not appropriate for all students. The evolution of computer technology that took place during the 1980s caused the pendulum to swing from one extreme to the other; from a subject with practical epithets practised in a vocational way, to a subject matter dependent on computer applications, with minimal need for knowledge of computer programming.

In a report published by the Data Education Committee (Datautbildningsgruppen, 1986), the necessity to offer computer programming, at least in supplementary courses, to students aiming at industry and a professional career was presented. The content of computer programming was therefore moved to higher realms in the school system. Accordingly, computer programming programming was therefore considered too time consuming and technical for the average student (ibid.).

Subject content and work methods
During the 1980s system development/systematisation was once again brought up in the arena of enactment. This time it was suggested to replace computer programming, as programming was perceived as too technical and time consuming and not appropriate for all students. The evolution of computer technology that took place during the 1980s caused the pendulum to swing from one extreme to the other; from a subject with practical epithets practised in a vocational way, to a subject matter dependent on computer applications, with minimal need for knowledge of computer programming.

In a report published by the Data Education Committee (Datautbildningsgruppen, 1986), the necessity to offer computer programming, at least in supplementary courses, to students aiming at industry and a professional career was presented. The content of computer programming was therefore moved to higher realms in the school system. Accordingly, computer programming gradually vanished from the curricula and computer applications within different subjects replaced the initial idea of fostering skills in algorithm construction appropriate for problem solving.8

5.3 Today's teachers' practice
In the following subsection two specific factors in relation to teachers' beliefs will be discussed; teachers' pedagogy (including students' time on task) and students' abilities.

Teachers pedagogical approach
The concept of pedagogy is broad and unspecific, wherefore it is hard to adopt in research (Kansanen, 1995). According to the majority of teachers in the study, the skills of computer programming vanished at the end of the 1980s and were later reintroduced as a subject matter in the educational reform, Lp94. It was the trade organisation of electricity that put pressure on the Swedish National Agency of Education (SNAE) to offer courses in computer programming. Later, in the 2000s, computer programming courses became electable for every student in upper secondary school. The circumstances that brought about a specific curriculum for computer programming; the third step in the process, are not described in the thesis because of time constraints. It will therefore become a matter for future investigations.

According to Björk, the impact of computer programming was minimal in the design of subject matters like Mathematics.7 Schools and teachers seemed to adapt more easily to premade material and computer applications than to self-made constructed algorithms for problem solving (Björk, 1987). Among teachers, computer programming was therefore considered too time consuming and technical for the average student (ibid.).

Subject content and work methods
During the 1980s system development/systematisation was once again brought up in the arena of enactment. This time it was suggested to replace computer programming, as programming was perceived as too technical and time consuming and not appropriate for all students. The evolution of computer technology that took place during the 1980s caused the pendulum to swing from one extreme to the other; from a subject with practical epithets practised in a vocational way, to a subject matter dependent on computer applications, with minimal need for knowledge of computer programming.

In a report published by the Data Education Committee (Datautbildningsgruppen, 1986), the necessity to offer computer programming, at least in supplementary courses, to students aiming at industry and a professional career was presented. The content of computer programming was therefore moved to higher realms in the school system. Accordingly, computer programming gradually vanished from the curricula and computer applications within different subjects replaced the initial idea of fostering skills in algorithm construction appropriate for problem solving.8

5.3 Today's teachers' practice
In the following subsection two specific factors in relation to teachers' beliefs will be discussed; teachers' pedagogy (including students' time on task) and students' abilities.

Teachers pedagogical approach
The concept of pedagogy is broad and unspecific, wherefore it is hard to adopt in research (Kansanen, 1995). According to the majority of teachers in the study, the skills of computer programming vanished at the end of the 1980s and were later reintroduced as a subject matter in the educational reform, Lp94. It was the trade organisation of electricity that put pressure on the Swedish National Agency of Education (SNAE) to offer courses in computer programming. Later, in the 2000s, computer programming courses became electable for every student in upper secondary school. The circumstances that brought about a specific curriculum for computer programming; the third step in the process, are not described in the thesis because of time constraints. It will therefore become a matter for future investigations.
pedagogy barely contributes to students’ learning. However, two types of pedagogy were identified among the teachers in the study: (1) teachers who support students individually for enhanced learning and (2) teachers involved in making methods instead of teaching technicalities within programming languages.

The factor of students’ time on task is considered a part of teachers’ pedagogy. According to the study, time on task seems to be important. More than 70% of the teachers explicitly express the need for brief instructions followed by students’ work on tasks. Two quotations show the dichotomy in the classroom between ‘time on task’ and instruction:

“I give brief instructions every lesson. I demonstrate sometimes, on the computer, about hardware and programming etc. The main part of the time, however, students spend engaged in programming by themselves.” ***

“Students who pass with particular distinction are able to independently solve complex tasks. They solve these tasks in a qualitative excellent way and they are able to independently correct errors and deviations to create as good code as possible. They are able to find and use information appropriately, on their own. They are comfortable with the basics and they accomplish their tasks in a limited time frame.” ***

A minority of teachers state that they use peer-learning as a method for students’ learning. The following quotation shows how one teacher used collaboration among his/her students in favour of self explaining code. The quotation also shows the interconnectivity between assessment and students’ independence, which was common among teachers. The quotation below is supported by a small group of teachers

“I don’t use examination papers at all, as they are clumsy and hard to interpret. I prefer continuous assessment in projects, where I’m looking for the independence and group engagement, individually and in conversation while solving tasks. The code gives an impression of their understanding, and their development in different domains. I even ask for commented code which facilitates my work [as a teacher].” ***

Use of peer-learning and collaboration in tutoring could be seen as a feature of computer programming education. However, the feature is also found in technology education (Sidawi, 2009; Hennessy and Murphy, 1999) wherefore it could just as well be something dependent on assessment or other circumstances implicit in the school environment.
Students’ abilities

The investigation reveals the existence of two strands concerning teachers’ views about their students’ ability to learn computer programming. Some teachers express confidence in their students’ ability to learn and some express the opposite. However, both groups of teachers perceive difficulties for most/all students learning programming skills. According to Marzano (2007) this lack of confidence in students’ possibilities and abilities is troublesome, as students’ achievement is influenced by teachers’ actions with students, and teachers’ beliefs about students’ chances to succeed.

The investigation depicts computer programming teachers as having high pre-requisites on students’ abilities (logical and analytical). This issue was further investigated during a specific seminar in order to unravel whether teachers ex-
press high expectations of their students in relation to all courses/course levels in programming. The outcome was extraordinary (See Figure 5.1) as a majority of teachers considered the analytical and logical abilities as (very) important for successful learning, in all three courses.

5.4 Summary of results

Experimental work with the informatics curriculum was initiated in Sweden in the mid-1950s. A contemporary post gymnasium programme 9 in computing was used as a blueprint in the early stages of the curriculum development process. The subject matter of informatics had to embrace systems development as it was believed to offer a societal aspect of computer programming. From this original setting, enthusiastic teachers, in particular Natural Science and Mathematics teachers, started to teach the subject of informatics. There was an incitement of major significance, as knowledge in computer programming and algorithm construction was believed to offer personal development of logical and analytical abilities for problem solving. However, practice in a school context revealed computer programming as problematic, as students experienced it as hard to learn, and teachers found the teaching of it to be time consuming.

During the 1980s, software evolved to such an extent that self-developed software and computer programming was questioned and later replaced. The need for computer programming skills was overtaken by computer applications developed for subject matters like e.g. Natural Sciences and Mathematics.

The incitement for offering computer programming in schools has changed during the studied period from vocational purposes, to attracting a broader group of students in a two-step process: (1) supplying industry with programming experts during the 1960s and (2) offering an alternative alignment within the NSP, which would appeal to a broader group of students.

The importance of teachers’ pedagogy for an enhanced learning outcome was brought up during all four seminars included in the study. The study shows that learning computer programming is commonly perceived, among teachers, as an individual act, where technicalities in computer programming languages are perceived as important. The phenomenon is underlined by Pears et al. (2007) and Nuthall (2004) where teachers are shown to perceive their teaching to be of minor importance to students’ learning outcome.

The investigation shows two distinctive groups of teachers; a large group of supportive, tutoring teachers (concerned with the syntax of programming language) and a smaller group of teachers engaged in instructional design for the experience of learning (not necessarily with syntax). Finally, the need for students to have acquired logical and analytical abilities when beginning the courses is considered (very) important to the students’ chances of learning computer programming.

5.4. SUMMARY OF RESULTS

A one-year higher special course for vocational purposes.

5.5 The curriculum development process of computer programming - a time line

A number of different parties were involved in the curriculum development process. Figure 5.2 is an attempt to present an overview of the presented findings with the help of a time-line starting from the beginning of the 1970s to the present day. The findings presented in this thesis concern the period from the 1970s to the end of the 1980s and today (2009-2011). Three arenas (Lindensjö and Lundgren, 2000) of particular interest to the study have been identified. From the top, the arena of realisation with students and teachers is accounted for. The next arena is the arena of transformation consisting of events/actions related to teachers and the syllabuses. The third arena – the arena of enactment – covers the establishment of authoritative documents and when they were implemented. The arena of enactment (bottom arena) also shows projects of importance initiated by the NBE and the ME.
Figure 5.2: Different projects, labels and parties involved in the curriculum development process for computer programming in upper secondary school 1970-2010.
Chapter 6

Discussion

In this chapter selected findings regarding the intersection of teachers’ beliefs (about pedagogy and students’ abilities), students and computer programming and the historical development process of programming education are discussed.

6.1 Curriculum and teachers’ beliefs

Teachers’ instruction is in this thesis described from a historical perspective to picture the necessity of re-thinking the curriculum. In line with Lister’s (2008) suggestion, this study highlights the necessity for teachers to avoid instructing in the same way that they themselves have been taught.

According to Mendelsohn et al. (1990) teachers have limited influence on the political intentions and the computer technology development; “they passively follow developments in hardware and software and adapt to political choices concerning computer equipment more than participating in decisions” (Mendelsohn et al., 1990, p.175). The study at hand questions this statement while findings reveal the existence of teachers involved in experimental work who significantly have influenced the implementation of computer programming and the enactment of the informatics curriculum. Of course their visions, ambitions and enthusiasm would not have been possible without the support from politicians regarding necessary decrees and economical resources.

Teachers’ beliefs are, according to research (Kagan, 1992; Luft and Roehrig, 2007; Olson, 1981; Yerrick et al., 1997) perceived as resistant to change but the findings of this study show that teachers can make a difference if they are provided with the right conditions. Teachers are allowed to elaborate in three different areas, to achieve successful computer programming education; the level of the computer programming language, the intended transferability of the language and the overall teaching style, where beliefs about teaching influence the teachers’ choice of approach.

He raised the issue of liberation as a teacher from ‘folk-pedagogy’ in computer programming.

---

Chapter 6

Discussion

In this chapter selected findings regarding the intersection of teachers’ beliefs (about pedagogy and students’ abilities), students and computer programming and the historical development process of programming education are discussed.

6.1 Curriculum and teachers’ beliefs

Teachers’ instruction is in this thesis described from a historical perspective to picture the necessity of re-thinking the curriculum. In line with Lister’s (2008) suggestion, this study highlights the necessity for teachers to avoid instructing in the same way that they themselves have been taught.

According to Mendelsohn et al. (1990) teachers have limited influence on the political intentions and the computer technology development; “they passively follow developments in hardware and software and adapt to political choices concerning computer equipment more than participating in decisions” (Mendelsohn et al., 1990, p.175). The study at hand questions this statement while findings reveal the existence of teachers involved in experimental work who significantly have influenced the implementation of computer programming and the enactment of the informatics curriculum. Of course their visions, ambitions and enthusiasm would not have been possible without the support from politicians regarding necessary decrees and economical resources.

Teachers’ beliefs are, according to research (Kagan, 1992; Luft and Roehrig, 2007; Olson, 1981; Yerrick et al., 1997) perceived as resistant to change but the findings of this study show that teachers can make a difference if they are provided with the right conditions. Teachers are allowed to elaborate in three different areas, to achieve successful computer programming education; the level of the computer programming language, the intended transferability of the language and the overall teaching style, where beliefs about teaching influence the teachers’ choice of approach.

He raised the issue of liberation as a teacher from ‘folk-pedagogy’ in computer programming.
6.2 The evolution of the curriculum - a top-down or bottom-up process

In the intersection of political initiatives, existence of manageable computer technology for schools and teachers' beliefs, there are problems to address. According to the study, one issue of concern appeared early in the implementation of the informatics curriculum. The introduction of system development/systematisation demanded specific knowledge or experience from the vocation, a competence that was not available in most schools. This problem was never solved properly, as computer applications overtook the need for these skills in schools.

The study has discovered agents (teachers and experts) of importance to the development of the curriculum. The existence of teachers embracing multiple roles in different arenas was a surprise, as this is not common in today's enactment of curricula in upper secondary school. The study therefore has found a bottom-up development process in the enactment of informatics curriculum during the 1970s.

Instructional strategies

The study shows two main types of instructional strategies influencing what education can offer. Boulton-Lewis et al. (2001) suggest a distinction similar to these two instructions; teaching for development of skills/understanding and teaching for facilitation of understanding. This corresponds to what the study discovered among computer programming teachers; individual scaffolding and instruction for facilitation of understanding, development of skills/understanding, and teaching for facilitation of understanding.

The investigation process at hand has probably influenced the development of today's curriculum (GY-11) in computer programming to some extent. Some discussions were further discussed on the internet forums organised by the Swedish National Agency of Education. The new curriculum for computer programming was released in the autumn of 2011.
6.2. THE EVOLUTION OF THE CURRICULUM - A TOP-DOWN OR BOTTOM-UP PROCESS

experience of learning. These two instructional strands also correspond to the debate during the 1980s between the partisans who emphasised the importance of repetition in small steps, and the defenders who emphasised the advantages of ‘discovery-learning’ where the students construct their own learning.

Findings from this study however, suggest a third way of describing instruction with the adoption of beliefs systems (Schraw and Olafson, 2008). From the instructional perspective, computer programming teachers seem to hold a relativistic ontology (Schraw and Olafson, 2008) which has similarities to that of the partisans during the 1980s and that of today’s teachers, who support students individually. However, a relativistic ontology, as described by Schraw and Olafson (ibid) also supports the collaboration between students/learners.

Pedagogical aspects

The importance of teachers’ pedagogical approach is questioned in the study. According to Pears et al. (2007) some teachers delivering beginners’ programming courses at university level, believe their teaching has little, if any, impact on the learning outcome of the students, which seems to hold even among teachers at upper secondary level. The concept of pedagogy is vague (Canning, 2007; Firmin et al., 2012; Postareff and Lindblom-Ylänne, 2008; Kansanen, 1995), as it holds important features in the interaction with the teacher community. The teachers in the study commonly refer to pedagogy instead of didactics or instruction when expressing their way of working in the classroom.

This study suggests the existence of a diversified picture regarding the importance of pedagogy; a minority of the teachers viewed pedagogy as important for improved student learning in general, a quarter of the teachers perceived a degree in pedagogy as important for teaching programming, and the vast majority perceived pedagogy as important for learning programming. A contradiction therefore seems to exist among teachers, but this could be interpreted as teachers holding a distinction between students with logical/analytical abilities and students who lack the prerequisites for learning computer programming, where pedagogy matters only to those who already possess the required abilities.

Students’ abilities

Early on in the research process a dichotomy was found regarding students’ abilities. Some of the teachers express confidence in their students’ abilities to learn (“I am one of those that believe any student can learn programming”) while others express

6See the discussion in Theoretical framework.
7Individual students construct their knowledge, in their own time, supported by a teacher.
8Kansanen (1995, p.34) explains “… teachers’ answers must be interpreted in the context of their own thinking and not through the language of the research and the researchers”. This makes the concept of pedagogy unspecific.
the opposite. The dichotomy between the two groups of teachers holding different confidence in students’ abilities to learn is an interesting and important discovery. The discovery that teachers hold beliefs about the capacities of various groups of students is supported by research made in mathematics education (Beswick, 2002, 2004, 2005b; Schoen et al., 2003). In the study at hand teachers have high expectations on students’ logical and analytical abilities, which could be a consequence of how they perceive computer programming, or a lack of appropriate teacher training. The instructional phenomenon of high expectations and the relevance of pedagogy according to the teachers is awkward as students are supposed to learn these skills during course. The study therefore suggest computer programming courses does not offer a ‘thinking curriculum’ in correlation with what Resnick (2010) envisage for school.
Chapter 7
Conclusions and final remarks

Over the years many attempts have been made to open up computer programming for education and lower the barriers for novices (Holt and Cordy, 1988; Kelleher and Pausch, 2005).  

According to research (Pelgrum, 2001; Donaldson and Knupfer, 2001; Haberman, 2006) technicalities within informatics commonly dominate today’s instruction, which according to Cuban (1986; 2001) and Pedersen (2001) raises important questions about the implicit technological determinism within informatics education. The question raised in this thesis concerns the nature and art of computer programming education and whether instruction has evolved enough for a diversity of students studying at upper secondary school.

7.1 Informatics curriculum evolution

The incitements for offering computer programming in upper secondary schools have changed from an vocational stance in supplementary courses. During the 1970s and 1980s it was offered at NSP as an alternative alignment for recruitment of more students to tertiary education in Mathematics and Natural Sciences. At the end of the 1980s computer applications were more appropriate wherefore programming was marginalized. Instead system development/systemization was suggested by ME as a replacement for computer programming in favoring logical thinking. Today’s courses in Swedish computer programming curriculum are electable for all students, as it is obligatory for some students at NSP.

Commonly, such attempts imply the construction of programming languages where one “avoids the languages that enforce many new abstract ideas, or have too many ‘programming tricks’ to be learnt, or give too little immediate concrete feedback” (Mendelsohn et al., 1990).

A description of the work established, 1963 to 2005, by the International Federation for Information Processing (IFIP) and the technical committee in education, TC3, was made by Watson (2006).

45
7.2 Beliefs

The study elaborates on teachers' beliefs in relation to the curriculum, where two specific beliefs have been discovered as problematic in the curriculum for upper secondary school. The existence of a relativistic ontology in teachers' instruction is suggested. However, teachers do not draw on students' differences for education in collaboration and peer-learning. The study has come to the following conclusions about today's teachers:

1. Teachers commonly perceive education as dependent on students' logical and analytical abilities.
2. Teachers perceive their pedagogy as insufficient for students learning.

7.3 Messages to teachers and curriculum developers

Today's teachers seem unaware of historical aspects of computer programming curriculum and its legacy from a vocational stance. I therefore hope that the conclusions from this study will add a useful perspective to teachers and curriculum developers to be used in future revisions of the computer programming curriculum for teachers as well as curriculum developers.

Concepts instead of technicalities

In one interview the informant expressed concern about computer programming while he spontaneously said:

"Programming, the dinosaur that survived!"

I found that remark interesting, as it describes computer programming education as archaic. Perhaps it also mirrors how instruction has been perceived over the years. I strongly suggest a re-imagining of the curriculum where the concept of computer science/informatics is more in focus than the implicit technicalities of programming languages. Today, teachers are dependent on students' intrinsic abilities in logic and analysis, which is awkward for a thinking curriculum where students are supposed to learn these skills (Resnick, 2010). So, where shall we start? Does in-service training to raise teachers' understanding of "softer values" involved in teaching and learning such an advanced subject matter as computer programming suffice? I would say no! I suggest in-service and pre-service teacher training focusing more on the conceptual learning instead of technicalities in programming languages.

Message to curriculum developers

Historical aspects of informatics are essential when reflecting on teaching and learning informatics (Akerha and Aspray, 2004; Micheuz, 2006). An instructional design
which draws on the historical context would most probably offer new pedagogical perspectives and learning possibilities for a broader group of students. A bridge between the abstract and the real could be built, if computer pedagogy was influenced by human activities, which in fact was what gave rise to computer technology (Dijkstra, 1975, in Böszörmenyi, 2005) in the first place. Education that offers such a perspective would hold a non-deterministic view of computers, where questions such as why, when and within which context could be a part of the education and understanding of computer science concepts. To consider historical aspects in future revisions of informatics curriculum is therefore strongly advised.
Chapter 8

Further studies

Different incitements in curriculum development have been noticed during investigation, each holding their own specific perspective on computer programming education; industry in need of people with skills, industry with interest to sell hardware, politics, the school-system and finally the teacher. The study is based on the school-system and teacher perspective to show the informatics curriculum development process in schools. It would however be pertinent to study the same development from the perspectives of politics or industry. The following headlines summarise what I have found to be relevant for further studies in different domains: Educational technology, Teacher’s association and interest groups, Teacher’s perception about teaching and learning and The school system.

8.1 In educational technology

1. The subject matter of computer programming is not manageable (teachable) in a school context with a diversity of learning styles and abilities. An international comparative study would be of interest, to understand whether the phenomenon is bound to Swedish circumstances or if it is bound to computer technology itself. Such a study would possibly show whether computer programming education is determined by its technicalities and the prerequisites it demands of students.

2. To deepen our understanding of the practical and theoretical aspects of computer programming, the dependence on computers and development environments in schools should be investigated further. The research perspective therefore becomes a question of how well the educational research model describes the interaction of computer technology and teachers’ instruction, for companies like Luxor, Norsk Data and IBM were involved in computers for the school market during the 1980s, which probably had an impact on teachers’ in-service training.

2 In the spirit of the Darmstadt model Hubwieser et al. (2011).
students’ learning. This is further elaborated by for instance Lee (2008) and Newson (1999) in the concept of techno-pedagogy.

8.2 In teacher’s associations and interest groups

1. During the investigation it become obvious other interest groups outside the educational system existed, like teachers’ associations (ALF, SIGYM, ASKigruppen) and manufacturers (Luxor, Norsk Data and IBM). The documents that were scrutinised reveal a heavy focus on computer technology hardware as prerequisites to offer computer programming in schools, instead of pedagogical aspects. A comparative study between today’s movement of technologically enhanced learning and the 1980s when computer resellers were surfing on the optimistic wave, would be of interest.

2. The software and hardware industry, with companies’ interest to expand into schools. Educational technology is commonly believed to offer new possibilities for enhanced learning (Cuban, 2001, 1986; Karlsohn, 2009). Parties interested in this perspective are typically found in teacher associations, companies within the school market and organisations involved in educational enhancement.1

8.3 In teacher’s perceptions of teaching and learning

1. Research in computer science education suggests (Saeli, 2012; Ragonis and Hassanz, 2008; Saeli et al., 2011; Woolard, 2005) we shall study the intersection of teachers’ pedagogical knowledge and content knowledge (PCK).2 to prevent the reinvention, year after year, of teaching computer programming in old patterns (Gundial, 2008). According to research in the domain of PCK (Saeli, 2012, p.19), teachers’ knowledge develops with teaching experience. It would therefore be of interest to conduct a similar study with the perspective of PCK.

2. The concepts of pedagogy, being logical, being analytical etc. need to be problematized in further studies, to understand teachers’ conceptualisation. A question of importance for reciprocity between research and teachers’ practice concern whether these concepts matter are significante for teachers’ in computer programming. Perhaps computer programming knowledge is perceived among teachers as solely tacit, as research constructs an overhead of concepts not appropriate for their practice. The research approach of Grounded theory is therefore suggested for further studies.

8.2 In teacher’s associations and interest groups

1. During the investigation it become obvious other interest groups outside the educational system existed, like teachers’ associations (ALF, SIGYM, ASKigruppen) and manufacturers (Luxor, Norsk Data and IBM). The documents that were scrutinised reveal a heavy focus on computer technology hardware as prerequisites to offer computer programming in schools, instead of pedagogical aspects. A comparative study between today’s movement of technologically enhanced learning and the 1980s when computer resellers were surfing on the optimistic wave, would be of interest.

2. The software and hardware industry, with companies’ interest to expand into schools. Educational technology is commonly believed to offer new possibilities for enhanced learning (Cuban, 2001, 1986; Karlsohn, 2009). Parties interested in this perspective are typically found in teacher associations, companies within the school market and organisations involved in educational enhancement.1

8.3 In teacher’s perceptions of teaching and learning

1. Research in computer science education suggests (Saeli, 2012; Ragonis and Hassan, 2008; Saeli et al., 2011; Woolard, 2005) we shall study the intersection of teachers’ pedagogical knowledge and content knowledge (PCK).2 to prevent the reinvention, year after year, of teaching computer programming in old patterns (Gundial, 2008). According to research in the domain of PCK (Saeli, 2012, p.19), teachers’ knowledge develops with teaching experience. It would therefore be of interest to conduct a similar study with the perspective of PCK.

2. The concepts of pedagogy, being logical, being analytical etc. need to be problematized in further studies, to understand teachers’ conceptualisation. A question of importance for reciprocity between research and teachers’ practice concern whether these concepts matter are significante for teachers’ in computer programming. Perhaps computer programming knowledge is perceived among teachers as solely tacit, as research constructs an overhead of concepts not appropriate for their practice. The research approach of Grounded theory is therefore suggested for further studies.
8.4 In the school system

1. Investigations about VADP were conducted in different archives in Solna, Växjö and Örebro municipalities and interviews with principals and personnel from Linköping, Umeå and Stockholm were carried out. These elicit the fact that VADP schools were self-governed to a high extent (finances, personnel and computers), while hosted by upper secondary school until the mid-1970s. The investigation, based on archive documents, suggests that people from VADP were only involved in the initial phase, until the mid-1970s, when VADP was incorporated into tertiary education (Huvudmannaskapspartietén, 1980). Could the reason be found in the technological evolution of computers?

2. A historical study that concerns the 1960s and 1970s and the transformation within specific upper secondary schools, going from a vocational subject in ADP to a subject matter of informatics. A comparative study would tell whether local industries played a part in the development of informatics education and teachers’ competence, which would be based on documents from school archives and associated County School Boards. It would unravel the interplay between regular education in upper secondary school and higher special courses in VADP.

3. A work group started by the Ministry of Education, DPG, initiated a new programming language suitable for Swedish circumstances, DPG-PROLOG (Utbildningsdepartementet, 1988; Ekstig et al., 1989) and a computer for the school market, COMPIS (Emanuel, 2009b). It is believed that the formation of the group and the incitements to develop a new programming language held much of the discourse at that time, in relation to the intersection of education and computer technology.

4. Teachers becoming NBE experts is suggested a phenomenon for Swedish circumstances, while the pattern from the 1970s remains in curriculum work during the 1990s and 2000s, in the two curriculum reforms, Lpf94 and GY-2000. That pattern is considered interesting while NBE experts could as well originate from university or industry. A comparative study with other countries would therefore be interesting to conduct.

*However, not much if any research is done in relation to VADP.*
Kapitel 9

Sammanfattning
(summary in Swedish)

9.1 Inledning


1På regeringskansliet den 21/2 2011. Rundabordsamtalen ingick i en seminarierese inom Digitala nödet.

2På grund av en viss begreppsförvirring och olika namn på kurser inom datorteknik/informationsteknik har jag valt att använda begreppet Informatik. Se begreppsdefinitionerna i första kapitlet, Introduction. Inom Sveriges kommuner och Skolverket används företrädesvis begreppen IT och/eller IKT, vilket enligt mig ligger betoningen vid användandet av teknik i undervisningen samma än betoning på undervisning om teknikbegreppen själva. I programminnehavning finns utbildning med teknik och utbildning om teknik, vilket kan göra nämnda distinktionen svårhanterlig.
9.2 Forskningsfokus


9.2 Forskningsfokus


9.3 Tidigare forskning


Programmering på gymnasiet

Under en stor del av 1970- och 1980-talet var undervisningen i Informatik syno

nymt med programmering. Från flera auktoriteter (Dahland, 1990; Sloane och Linn, 1988; Cuban, 1986; Pea och Kurland, 1984; Reed och Palumbo, 1991; Clements och Gull, 1984) højdes kritiska röster om lämpligheten av programmering i skolan; innehållet var alltfor fokuskänns och teknik till sin natur. Sverige förde en ut

bildningspolitik som syftade till att så många som möjligt skulle erhålla datorunder

ervisning (Datadelegationen, 1985), vilket innebar att merparten av eleverna fick
9.4 Metod

Historiskt perspektiv

Historiska källdokument från Riksarkivet och olika Stadsarkiv har använts. Ett totaltantal personer som var engagerade i kursplanutvecklingen intervjuades. Även källor från Riksdagens bibliotek, som belönar politiska intentioner och investeringsättning på datorer i skolan har använts.

Data av relevans för kursplanutveckling, från Riksarkiv och olika stadsarkiv, skannades och sammanställdes i digitala volymer för att granskas i efterhand; interaktionen mellan olika arbetsgrupper inom SO och mellan SO och olika skolor/lärare studerades speciellt.

Nutidsperspektiv

För att nå kunskap om erfarenheter av undervisning och lärande i programmering startades ett lärarnätverk däremot. Inom ramen för en seminarseries med fyra tillfällen, under 2009 till 2011,
betsmetoder. Diskuterades olika didaktiska teman/frågeställningar, lärares erfarenheter och arbetsmetoder.\[5]

I samband med seminariums tillfrågades lärarna via enkäter om sina erfarenheter och didaktiska överväganden i programmeringsundervisningen. Insamlade data analyserades och kategoriserades i en två-stegsprocess. En preliminär analys visade på två olika synsätt vad gäller elevers förutsättningar att lâra sig programmering; det fanns lärare som ansåg, och inte ansåg, att alla elever kan lâra sig programmering. I nästa steg genomfördes en induktiv dataanalys där teman arbetades fram (Lincoln och Guba, 1985; Miles och Huberman, 1994).

9.5 Resultat

Programmering i skolan – ett historiskt perspektiv


I Sverige valde man att utveckla ett unikt programmeringsspråk, DPG-PROLOG, och en skoldator, COMPIS (Utbildningsdepartementet, 1988), för svenska förhållanden. Datorter och datorapplikationer har dock fortsatt och utvecklats och därmed behovet av annorlunda utbildning. Undervisning i programmering ansågs därför

55Om möjligt ges finansiellt och resursmässigt kommersiella seminarier fortgå. Träffarna genomfördes i samarbete med Teknikum/Stockholms universitet, Upparks universitet, KTH och Microsoft Sweden AB.

56Om möjligt ges finansiellt och resursmässigt kommersiella seminarier fortgå. Träffarna genomfördes i samarbete med Teknikum/Stockholms universitet, Upparks universitet, KTH och Microsoft Sweden AB.
9.6 Diskussion


Lärarnas förståelser – ett muttidsperspektiv


57
denna satsning inte igenom i Sverige.

9.7 Slutsatser
Samtidigt som lärare ifrågasätter betydelsen av deras pedagogik för elevers lärande så visar studien att lärare förutsätter logiska och analytiska färdigheter/förmågor för elevers lärande. Programmeringsundervisningen uppvisar därför starkt exkluderande drag. Man kan därför fråga sig om undervisningen har utvecklats i tillräcklig grad för att erbjudas vid gymnasiet. Uppsatsen pekar därför på en tydlig uppfattning om ämnets svårigheter som funnits under lång tid.

I Finland var man betydligt tydligare med behovet av att satsa på specifika utbildningsspråk inom programmering. Enligt Saarikoski (2011) hade den Finländska regeringen uttryckligen markerat att LOGO skulle användas för att undvika teknikaliteter i programmeringspråk. Sveriges utbildningsdepartement gjorde några forsök med PROLOG, vilket aldrig dog igenom inom skolan.

Liknande undervisningsmönster som exkluderar elever återfinns även inom andra skolämnen som t.ex. matematik (Ball and Bass, 2000), varför det är svårt att påstå att detta är någonting karaktäristiskt för mötet mellan programmering, eleven och läraren, varför det kan finnas ett annat problem i botten. Det ligger dock utanför denna uppsats.


---

1 I Finland var man betydligt tydligare med behovet av att satsa på specifika utbildningsspråk inom programmering. Enligt Saarikoski (2011) hade den Finländska regeringen uttryckligen markerat att LOGO skulle användas för att undvika teknikaliteter i programmeringspråk. Sveriges utbildningsdepartement gjorde några forsök med PROLOG, vilket aldrig dog igenom inom skolan.

2 Liknande undervisningsmönster som exkluderar elever återfinns även inom andra skolämnen som t.ex. matematik (Ball and Bass, 2000), varför det är svårt att påstå att detta är någonting karaktäristiskt för mötet mellan programmering, eleven och läraren, varför det kan finnas ett annat problem i botten. Det ligger dock utanför denna uppsats.


Bell, T., P. Curzon, Q. Cutts, V. Dagiena, and B. Haberman (2011). Introducing students to computer science with programmes that don’t emphasise programming. In *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education, ITiCSE ’11*, New York, NY, USA, pp. 391–391. ACM.


BIBLIOGRAPHY


### The Swedish school system

<table>
<thead>
<tr>
<th>Level</th>
<th>The management</th>
<th>Jurisdiction</th>
<th>Example of decisions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The parliament</td>
<td>The ground law</td>
<td>General objectives for society</td>
</tr>
<tr>
<td>2</td>
<td>The parliament</td>
<td>Specific laws</td>
<td>School law</td>
</tr>
<tr>
<td>3</td>
<td>The government - Ministry of education</td>
<td>Decrees</td>
<td>Decrees and curriculum</td>
</tr>
<tr>
<td>4</td>
<td>The National Bureau of Education, NBE</td>
<td>Administrative decisions and decisions for execution</td>
<td>Central planning</td>
</tr>
<tr>
<td>5</td>
<td>County School Board</td>
<td>The region</td>
<td>Regional planning</td>
</tr>
<tr>
<td>6</td>
<td>School Board</td>
<td>Local</td>
<td>Organization plan</td>
</tr>
<tr>
<td>7</td>
<td>Principal</td>
<td>Local</td>
<td>Working plan</td>
</tr>
<tr>
<td>8</td>
<td>Teacher</td>
<td>Local</td>
<td>Education</td>
</tr>
<tr>
<td>9</td>
<td>Student</td>
<td>Local</td>
<td></td>
</tr>
</tbody>
</table>

Figure A.1: Translated from the original version summarized in Swedish by Marklund (1987, p. 69).
KONSTEN ATT UNDERVISA I PROGRAMMERING


Som ett led i datainsamlingen har jag tagit fram följande enkät med frågor som jag ber Dig besvara. Enkäten vänder sig till lärare som anmält intresse till programmeringsolympiaden. Den information Du lämnar kommer att behandlas konfidentiellt och data kommer att avidentifieras när resultatet avrapporterats.

I ett senare skede kommer jag att kontakta några av dem som besvarat enkäten för ett uppföljande samtal. I formuläret kan Du med ett kryss ange att Du vill vara med också i detta datainsamling.

Den besvarade enkäten skickas lättast till <min.epostadress@xxx.se>

För dig som önskar skicka den besvarade enkäten med posten kommer ett separat utskick att göras. Om så är fallet vill jag att du meddelar mig per e-post.

Tack för Din medverkan - Dina synpunkter är viktiga!

Har du frågor kring studien eller enkäten får du gärna kontakta mig.

<table>
<thead>
<tr>
<th>1. Personuppgifter</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Namn</strong></td>
</tr>
<tr>
<td><strong>E-post</strong></td>
</tr>
</tbody>
</table>

Vill Du medverka i en uppföljning av Dina svar på enkäten?
(via telefon alt. via personligt möte)

JA ☐ NEJ ☐
KONSTEN ATT UNDERVISA I PROGRAMMERING

2. Programmering och programmeringsspråk

A. Vilken typ av utbildning har Du i programmering?

B. Vilken utbildning tycker Du att man behöver för att undervisa i programmering?

C. Hur länge har Du undervisat i programmering?

D. Vilka årskurser och gymnasieprogram har Du undervisat i den senaste femårsperioden?

E. Vilka av följande kurser har du undervisat i den senaste femårsperioden? Fyll även i det programmeringsspråk du använt.
   - a) Programmering A  JA NEJ Progr.språk:
   - b) Programmering B  JA NEJ Progr.språk:
   - c) Programmering C  JA NEJ Progr.språk:
   - d) Webbdesign B  JA NEJ Progr.språk:
   - e) Databashantering  JA NEJ Progr.språk:

F. Vilket programmeringsspråk föredrar Du?

G. Vilken inställning har Du till produkter med öppen källkod?

H. Har Du arbetat utanför skolan med programmering under den senaste femårsperioden?
   - JA NEJ
   - Om svaret är JA på föregående fråga ber jag Dig ange typ av arbete som Du haft.

3. Undervisningen

A. Har Du och Dina elever arbetat tillsammans med andra skolor, eller klasser/grupper i programmering?
   - JA NEJ

B. Om svaret är JA på föregående fråga ber jag att Du beskriver Dit syfte med detta samarbete.
KONSTEN ATT UNDERVISA I PROGRAMMERING

C. Brukar Dina elever använda Internet (forum, chatt, skype, wiki m.fl.) för interaktion med andra elever och andra programmeringsintresserade för att lösa uppgifter i programmering?

JA ☐  NEJ ☐  VET INTE ☐

D. Använder DU andra bedömningsgrundar än prov för att värdera elevens kunskaper? Om svaret är JA ber jag att Du kort beskriver det underlag Du använder vid betygssättningen.

E. Använder Du film (video på nätet, youtube) i Din undervisning?

JA ☐  NEJ ☐

F. Använder Du facebook, twitter, blogg eller wiki i Din undervisning?

JA ☐  NEJ ☐

G. Om Du använder utveckling av spel i undervisningen, vilken typ av spel?

H. Brukar Dina elever arbeta i huvudsak enskilt eller tillsammans (parvis eller grupper)?

I. Vilken typ av undervisning anser Du är bäst för att elever skall lära sig programmering?

J. Beskriv vilka utvecklingsmiljöer (ex. Notepad, Eclipse, BlueJava eller Dreamweaver) Du använder i programmeringsundervisningen?

K. Vilken är enligt Dig den tydligare kopplingen mellan matematik och programmering?

L. Finns det, enligt Dig, någon koppling mellan programmering och annat ämne i gymnasieskolan?

M. Arbetar Dina elever föruträdesvis med enkla, mindre uppgifter eller med komplexa och större programmeringsuppgifter?

4. Lärande

A. Vilka kvalitéer anser Du att en elev behöver för att bli duktig i programmering?
KONSTEN ATT UNDERVISA I PROGRAMMERING

<table>
<thead>
<tr>
<th>B. Vad innebär det enligt Dig att lära sig problemlösning?</th>
</tr>
</thead>
<tbody>
<tr>
<td>C. Anser Du att ALLA kan lära sig programmering? Motivera varför.</td>
</tr>
<tr>
<td>D. Lägger Du in estetiska värden i programmering?</td>
</tr>
<tr>
<td>JA</td>
</tr>
<tr>
<td>E. Om svaret är JA på föregående fråga ber jag att Du beskriver egenskaper som gör koden estetiskt tilltalande?</td>
</tr>
<tr>
<td>F. Hur många timmar/vecka förväntar Du Dig att Dina elever arbetar hemma med programmering?</td>
</tr>
<tr>
<td>G. I Programmering B och C finns flera begrepp som syftar mot en objekt-orienterad programmering. Vilket eller vilka är, enligt Din mening, det svåraste begreppet/begreppen då elever skall lära sig objekt-orienterad programmering?</td>
</tr>
<tr>
<td>H. Många elever kan lära sig syntax och programmeringsspråkets funktion, men stöter på svårigheter då de skall lösa större uppgifter. Vad är enligt Din mening, den vanligaste orsaken till elevers problem?</td>
</tr>
<tr>
<td>I. Vilka kvalitéer skiljer, enligt Din mening, elever som får betyget G respektive betyget MVG i programmering?</td>
</tr>
</tbody>
</table>

S. Nätverk

| A. Använder Du nätverksgemenskaper för att lära Dig mer om programmering? Om svaret är JA får Du gärna dela med Dig av Dina kontakter. |
| B. Har eleverna på Din skola en grupp med programmeringsintresserade? |
| C. Är Du intresserad av att ingå i ett nätverk med lärare för att |
|   a) utbyta idéer i skolläget programmering? |
|   b) utbyta programmeringsuppgifter? |
|   c) utveckla samarbetsmöjligheter mellan elevgrupper och skolor? |
|   JA | NEJ | |

---

KONSTEN ATT UNDERVISA I PROGRAMMERING

<table>
<thead>
<tr>
<th>B. Vad innebär det enligt Dig att lära sig problemlösning?</th>
</tr>
</thead>
<tbody>
<tr>
<td>C. Anser Du att ALLA kan lära sig programmering? Motivera varför.</td>
</tr>
<tr>
<td>D. Lägger Du in estetiska värden i programmering?</td>
</tr>
<tr>
<td>JA</td>
</tr>
<tr>
<td>E. Om svaret är JA på föregående fråga ber jag att Du beskriver egenskaper som gör koden estetiskt tilltalande?</td>
</tr>
<tr>
<td>F. Hur många timmar/vecka förväntar Du Dig att Dina elever arbetar hemma med programmering?</td>
</tr>
<tr>
<td>G. I Programmering B och C finns flera begrepp som syftar mot en objekt-orienterad programmering. Vilket eller vilka är, enligt Din mening, det svåraste begreppet/begreppen då elever skall lära sig objekt-orienterad programmering?</td>
</tr>
<tr>
<td>H. Många elever kan lära sig syntax och programmeringsspråkets funktion, men stöter på svårigheter då de skall lösa större uppgifter. Vad är enligt Din mening, den vanligaste orsaken till elevers problem?</td>
</tr>
<tr>
<td>I. Vilka kvalitéer skiljer, enligt Din mening, elever som får betyget G respektive betyget MVG i programmering?</td>
</tr>
</tbody>
</table>

S. Nätverk

| A. Använder Du nätverksgemenskaper för att lära Dig mer om programmering? Om svaret är JA får Du gärna dela med Dig av Dina kontakter. |
| B. Har eleverna på Din skola en grupp med programmeringsintresserade? |
| C. Är Du intresserad av att ingå i ett nätverk med lärare för att |
|   a) utbyta idéer i skolläget programmering? |
|   b) utbyta programmeringsuppgifter? |
|   c) utveckla samarbetsmöjligheter mellan elevgrupper och skolor? |
|   JA | NEJ | |
KONSTEN ATT UNDERVISA I PROGRAMMERING

d) fortbilda Dig i programmering? JA □ NEJ □

Stort tack för Din medverkan!
Personuppgifter

<table>
<thead>
<tr>
<th>Lärarutbildning eller pedagogisk påbyggnadsutbildning</th>
<th>Ja</th>
<th>Nej</th>
</tr>
</thead>
</table>

Antal år med undervisning i Programmering

Under de senaste fem åren har Du undervisat följande kurser:

- Programmering A
- Programmering B
- Programmering C

Kan alla elever lära sig kunskaper motsv. Programmering A?

- Ja
- Nej

Kan alla elever lära sig kunskaper motsv. Programmering B?

- Ja
- Nej

Kan alla elever lära sig kunskaper motsv. Programmering C?

- Ja
- Nej

I vilken omfattning förutsätter Du att eleven har följande förkunskaper i Programmering?

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lite</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mycket</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Att kunna kommunikera kunskaper och programmeringssstrategier

1 2 3 4 5 att "spara" och återanvända en del av koden
1 2 3 4 5 att analysera och byta ut eller byta plats på olika delar
1 2 3 4 5 att också en del av koden
1 2 3 4 5 ett speciellt tabellformat
1 2 3 4 5 ett allmänt tabellformat

Att kunna arbeta i grupp

1 2 3 4 5 att kunna samarbeta med andra, dela idéer och arbeta gemensamt

Att kunna se helheter och skapa strukturer

1 2 3 4 5 att kunna se helheter och skapa strukturer

Att "bryta ner" ett problem i detaljer (analytiskt tänkande)

1 2 3 4 5 ett logiskt tänkande

Att kunna kommunicera och utveckla matematiskt

1 2 3 4 5 ett abstrakt tänkande

i det aktuella programmeringsspråket (syntaxkunskaper)
I vilken omfattning arbetar Du med att eleven skall utvecklas i följande kunskaper i Programmering?

<table>
<thead>
<tr>
<th>Lite</th>
<th>Mycket</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4 5</td>
<td>att kunna kommunicera kunskaper och programmeringsstrategier</td>
</tr>
<tr>
<td>1 2 3 4 5</td>
<td>förmågan att arbeta i grupp</td>
</tr>
<tr>
<td>1 2 3 4 5</td>
<td>att se helheter och skapa strukturer</td>
</tr>
<tr>
<td>1 2 3 4 5</td>
<td>att &quot;bryta ner&quot; ett problem i detaljer (analytiskt tänkande)</td>
</tr>
<tr>
<td>1 2 3 4 5</td>
<td>ett logiskt tänkande</td>
</tr>
<tr>
<td>1 2 3 4 5</td>
<td>ett matematiskt tänkande</td>
</tr>
<tr>
<td>1 2 3 4 5</td>
<td>ett abstrakt tänkande</td>
</tr>
<tr>
<td>1 2 3 4 5</td>
<td>i det aktuella programmeringsspråket (syntaxkunskaper)</td>
</tr>
</tbody>
</table>

I vilken omfattning anser Du att följande kunskaper är viktiga för att eleven skall få högsta betyg i Programmering?

<table>
<thead>
<tr>
<th>Lite</th>
<th>Mycket</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4 5</td>
<td>att kunna kommunicera kunskaper och programmeringsstrategier</td>
</tr>
<tr>
<td>1 2 3 4 5</td>
<td>förmågan att arbeta i grupp</td>
</tr>
<tr>
<td>1 2 3 4 5</td>
<td>att se helheter och skapa strukturer</td>
</tr>
<tr>
<td>1 2 3 4 5</td>
<td>att &quot;bryta ner&quot; ett problem i detaljer (analytiskt tänkande)</td>
</tr>
<tr>
<td>1 2 3 4 5</td>
<td>ett logiskt tänkande</td>
</tr>
<tr>
<td>1 2 3 4 5</td>
<td>ett matematiskt tänkande</td>
</tr>
<tr>
<td>1 2 3 4 5</td>
<td>ett abstrakt tänkande</td>
</tr>
<tr>
<td>1 2 3 4 5</td>
<td>i det aktuella programmeringsspråket (syntaxkunskaper)</td>
</tr>
</tbody>
</table>

Kommentarer (skriv på kanten om det inte räcker):
I vilken grad är det viktigt för Din undervisning?

Einkäten speglar ett antal faktorer som troligen påverkar dina val som lärare i programmering. Syftet med enkäten är att undersöka hur dessa faktorer värderas på en skala mellan "Inte alls" och "Mycket". Resultatet kommer att sammanställas och presenteras på seminariet den 8 oktober. Hoppas därför på din möjlighet att hjälpa till.

Sammanställningen kommer användas för en seminariediskussion om begränsningar och möjligheter inom programmeringsundervisning.

För att resultatet skall kunna sammanställas i tid, behöver jag ditt svar senast, torsdag förmiddag. Även andra svar som kommer senare är välkomna, då sammanställningen är av betydelse i framtida seminariediskussioner. Den information som Du lämnar behandlas konfidentiellt och data kommer att avidentifieras.

Den besvarade enkäten skickas lättast till <min.epostadress@xxx.se>

Tack för Din medverkan - Dina synpunkter är viktiga!

Har du frågor kring studien eller enkäten får du gärna kontakta mig.

1. I vilken grad är följande faktorer viktiga för elevers lärande?

<table>
<thead>
<tr>
<th>Faktor</th>
<th>Inte alls</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>Mycket</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Programmeringsspråket har enkel syntax</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B. Det finns färdiga moduler/klasser/bibliotek</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C. Språkets syntax är generaliserbart till andra språk</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D. Användargränssnittet är lättanvändligt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E. Det finns funktioner för programdesign</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F. Det finns verktyg för felsökning av buggar mm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G. Programvaran är gratis o. tillgänglig via Internet</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H. Det finns grafiska representationer av objekt (typ spellinna miljöer som Alice, Scratch)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I. Programköring går att visualisera stegvis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J. Språket och/eller miljön används inom industrin</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K. Möjlighet att arbeta tillsammans via Internet</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L. Möjlighet att arbeta nära datosystemet (komplexa, länka och bygga skript)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M. Du kan programmera objektorienterat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N. Du kan programmera funktionsorienterat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O. Det finns bra pedagogisk litteratur/material</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P. Programmeringsspråket kräver ett strukturerat arbetssätt vid implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q. Man kan programmera externa enheter (LEGO robotar, styr-och mätlaborationer mm)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Forts på nästa sida
2. I vilken grad är följande faktorer viktiga för din undervisning?

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Elevers önskemål</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B. Elevers tidigare erfarenheter</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C. Elevers intresse och motivation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D. Eleverna som sammanhänger tid vid datorn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E. Eleverna som nytta av språkutvecklingsmiljön i högre studier</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F. Eleverna som välja att samarbeta med andra elever</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G. Elevers logiska förmåga</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H. Elevers strukturella förmåga</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I. Eleverna som bestå de av NV - elever</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J. Eleverna som bestå de av TE - elever</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K. Eleverna som bestå de av EC - elever</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L. Eleverna som läser programmering som individuellt val</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3. I vilken grad är följande faktorer viktiga för att utveckla din undervisning?

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Dina kunskaper i liknande språkutvecklingsmiljön</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B. Dina möjligheter (tid, vilja mm) till fortbildning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C. Ett nätverk med kunniga personer i programmeringsutvecklingsmiljön</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D. Skolledningens stöd och engagemang i Din utveckling</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E. Samarbeten mellan lärare för att utveckla nytt undervisningsmaterial (bok och mjukvara)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F. Endagars seminarier under en längre tid</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G. Kortare kurser</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H. Längre kurser</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I. En koppling till lärarutbildning med lärarkandidater som praktiserar hos Dig</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
En enkät för lärarens perspektiv på undervisning i
programmering

Mitt namn är Lennart Rolandsson och jag är doktorand vid forskarskolan
Teknikutbildning för framtiden, TUFF vid Stockholms universitet. Mitt forskningsämne
är programmeringslärares perspektiv på undervisning. Har själv undervisat i programme-
ring i ett antal år, vilket väckt frågor som jag undersöker inom ramen för mitt
avhandlingsarbete som beräknas vara klart i slutet av 2011.

Som ett led i datainsamlingen har denna enkät utarbetats med frågor som jag ber Dig
besvara. Den bygger på tidigare enkäter, seminarier och intervjuer med enskilda lärare.
Informationen du lämnar kommer att behandlas konfidentiellt och data kommer att
avidentifieras när resultatet avrapporteras.

I seminariet, den 1 april 2011, kommer en samlad bild av enkätsvaren diskuteras. I ett
senare skede kommer slutsatserna från seminariet och enskildtävaren ingå i en
licentiatavhandling. Om Du önskar avhandlingen i digitalt format ber jag Dig kryssa i
rutan här nere. Personligen kommer jag att borga för att Du får en pdf-fil som tåg för
hjälpen.

Den besvarade enkäten skickas snarast möjligt till <min.epostadress@xxx.se> för en
sammansättning inför seminariet.

Tack för Din medverkan - Dina synpunkter är viktiga!
Har du frågor kring studien eller enkäten får du gärna kontakta mig.
1. Personuppgifter

Namn Ålder Man Kvinna

E-post Skola

Är du intresserad av en pdf-kopia av avhandlingen? (via e-post)
JA NEJ

2. Lärares pedagogisk utbildning

A. Hur många universitetspoäng har Du (på ett ungefär) i

   Fysik Programming

   Matematik Pedagogik

B. Om Du har genomgått lärarutbildning eller motsvarande. Vilken typ av
   pedagogisk examen har Du?

C. Har du en lärarexamen med kurser i programmering? Vilka kurser?

D. Vanifrån har du fått Dina ämneshuvudskapar i programmering?

E. På vilken (högsta) nivå har Du studerat programmering?

3. Lärares pedagogisk erfarenhet

A. Hur många år har Du undervisat i programmering?

### ATT UNDERVISA I PROGRAMMERING  
Quest 4

<table>
<thead>
<tr>
<th>B. Vilka kurser i programmering har Du undervisat den senaste femårsperioden?</th>
</tr>
</thead>
<tbody>
<tr>
<td>C. Har Din syn på undervisning i programmering förändrats sedan Du började undervisa i ämnet?</td>
</tr>
<tr>
<td>D. Anser Du att pedagogik är viktigt för programmeringsundervisning?</td>
</tr>
<tr>
<td>E. Vad har påverkat Dig i valet av undervisningsmetod och val av resurser (Internet, Litteratur mm) i Din programmeringsundervisning?</td>
</tr>
</tbody>
</table>
| F. Hur ställer Du Dig till följande läraricitat?  
"Programmering är mer matematik än vad matematikämnet i skolan är"  
(Håller INTE med) | 1 | 2 | 3 | 4 | 5 | 6  
(Håller med) |
| G. Hur ställer Du Dig till följande läraricitat?  
"Vi lärare lägger för mycket tid på syntax och för lite tid på struktur. När elever får en större uppgift 'drunknar' de i mängden av kod, istället för att dela upp den i delproblem."  
(Håller INTE med) | 1 | 2 | 3 | 4 | 5 | 6  
(Håller med) |
| H. Hur ställer Du Dig till följande läraricitat?  
"Bättre om (eleverna) kunde strukturera problemet med papper och penne (flödesschema) och tänka igenom lösningen innan man kodar."  
(Håller INTE med) | 1 | 2 | 3 | 4 | 5 | 6  
(Håller med) |
| I. Hur ställer Du Dig till följande läraricitat?  
"Man måste hjälpå eleverna att se behovet av en strategi istället för att angripa hela problemet på en gång"  
(Håller INTE med) | 1 | 2 | 3 | 4 | 5 | 6  
(Håller med) |
| J. Hur ställer Du Dig till följande läraricitat?  
"Elever har svårt att dela upp uppgiften/problemets i mindre delar och lösa dem var och en för sig."  
(Håller INTE med) | 1 | 2 | 3 | 4 | 5 | 6  
(Håller med) |
ATT UNDERVISA I PROGRAMMERING

**Quest 4**

4. För Dig som undervisar/undervisat i Programmering A

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A.</strong> Anser Du att ALLA elever som börjar läsa Programmering A kan bli godkända?</td>
<td>JA</td>
<td>NEJ</td>
<td>VET INTE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I vilken omfattning spelar följande faktorer en roll för lärandet i Programmering A?</td>
<td>(Lite)</td>
<td>(Mycket)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Lärarens pedagogik</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>• Tiden som eleven arbetar på egen hand</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>• Elevers motivation och intresse</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>• Elevers förmåga att strukturera koden</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>• Elevers analytiska och logiska förmåga</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>B.</strong> Brukar Dina elever, i Programmering A, arbeta i huvudsak enskilt eller tillsammans (parvis eller grupper)?</td>
<td>(Enskilt)</td>
<td>(Tillsammans)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>C.</strong> Arbetar Dina elever företrädesvis med enkla och mindre uppgifter eller med komplexa och större programmeringsuppgifter i Programmering A?</td>
<td>(Enkla/Mindre)</td>
<td>(Komplexa/Större)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>D.</strong> Vilken typ av undervisning (teoretisk eller praktisk) anser Du är bäst för att elever skall lära sig Programmering A?</td>
<td>(Teoretisk)</td>
<td>(Praktisk)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>D.</strong> Vilken typ av undervisning (teoretisk eller praktisk) anser Du är bäst för att elever skall lära sig Programmering A?</td>
<td>(Teoretisk)</td>
<td>(Praktisk)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ATT UNDERVERSÅ I PROGRAMMERING

5. För Dig som undervisar/undervisat i Programmering B

A. Anser Du att ALLA elever som börjar läsa Programmering B kan bli godkända?

<table>
<thead>
<tr>
<th></th>
<th>JA</th>
<th>NEJ</th>
<th>VET INTE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

I vilken omfattning spelar följande faktorer en roll för lärandet i Programmering B?

- Lärarens pedagogik
- Tiden som eleven arbetar på egen hand
- Elevers motivation och intresse
- Elevers förmåga att strukturera koden
- Elevers analytiska och logiska förmåga

<table>
<thead>
<tr>
<th>Faktor</th>
<th>(Lite)</th>
<th>(Mycket)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lärarens pedagogik</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Tiden som eleven arbetar på egen hand</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Elevers motivation och intresse</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>Elevers förmåga att strukturera koden</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Elevers analytiska och logiska förmåga</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

B. Brukar Dina elever, i Programmering B, arbeta i huvudsak enskilt eller tillsammans (parvis eller grupper)?

<table>
<thead>
<tr>
<th>(Enskilt)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>(Tillsammans)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

C. Arbetar Dina elever företrädesvis med enkla och mindre uppgifter eller med komplexa och större programmeringsuppgifter i Programmering B?

<table>
<thead>
<tr>
<th>(Enkla/Mindre)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>(Komplexa/Större)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

D. Vilken typ av undervisning (teoretisk eller praktisk) anser Du är bäst för att elever skall lära sig Programmering B?

<table>
<thead>
<tr>
<th>(Teoretisk)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>(Praktisk)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6. För Dig som undervisar/undervisat i Programmering C

<table>
<thead>
<tr>
<th></th>
<th>JA</th>
<th>NEJ</th>
<th>VET INTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Anser Du att ALLA elever som börjar läsa Programmering C kan bli godkända?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>I vilken omfattning spelar följande faktorer en roll för lärandet i Programmering C?</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Lite)</td>
<td>(Mycket)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lärarens pedagogik</td>
<td>1 2 3 4 5 6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tiden som eleven arbetar på egen hand</td>
<td>1 2 3 4 5 6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Elevers motivation och intresse</td>
<td>1 2 3 4 5 6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Elevers förmåga att strukturera koden</td>
<td>1 2 3 4 5 6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Elevers analytiska och logiska förmåga</td>
<td>1 2 3 4 5 6</td>
<td></td>
</tr>
</tbody>
</table>

| B. Brukar Dina elever, i Programmering C, arbeta i huvudsak enskilt eller tillsammans (parvis eller grupper)? |
|---|---|---|---|---|---|---|
| (Enskilt) | 1 | 2 | 3 | 4 | 5 | 6 |
| (Tillsammans) |     |     |     |     |     |     |

| C. Arbetar Dina elever företrädevis med enkla och mindre uppgifter eller med komplexa och större programmeringsuppgifter i Programmering C? |
|---|---|---|---|---|---|---|
| (Enkla/Mindre) | 1 | 2 | 3 | 4 | 5 | 6 |
| (Komplexa/Större) |     |     |     |     |     |     |

| D. Vilken typ av undervisning (teoretisk eller praktisk) anser Du är bäst för att elever skall lära sig Programmering C? |
|---|---|---|---|---|---|---|
| (Teoretisk) | 1 | 2 | 3 | 4 | 5 | 6 |
| (Praktisk) |     |     |     |     |     |     |

Stort tack för Din medverkan!
