



Incorporating programming into mathematics education

How using programming shapes upper-secondary students'
mathematical understanding

Andreas Borg

Faculty of Arts and Social Sciences

Educational Work

DOCTORAL THESIS | Karlstad University Studies | 2026:16

Incorporating programming into mathematics education

How using programming shapes upper-secondary students' mathematical understanding

Andreas Borg



Incorporating programming into mathematics education - How using programming shapes upper-secondary students' mathematical understanding

Andreas Borg

DOCTORAL THESIS

Karlstad University Studies | 2026:16

urn:nbn:se:kau:diva-108477

ISSN 1403-8099

ISBN 978-91-7867-677-4 (print)

ISBN 978-91-7867-678-1 (pdf)

<https://doi.org/10.59217/eqsy6353>

© The author

Distribution:
Karlstad University
Faculty of Arts and Social Sciences
Department of Educational Studies
SE-651 88 Karlstad, Sweden
+46 54 700 10 00

Print: Universitetstryckeriet, Karlstad 2026

WWW.KAU.SE

Abstract

This thesis comprises two studies investigating upper-secondary students' use of programming as a mathematical tool. It aims to examine both the intertwined relationship between students' use of programming and their mathematical understanding, and how the design of learning activities can support the incorporation of programming into mathematics education.

The first study adopts a design-based research approach centred on a problem-solving activity involving programming. The second study examines a teacher's design of programming activities for numerical calculations and its influence on students' understanding of limits.

The Instrumental Approach provides the theoretical lens for analysing students' instrumental genesis, describing the relationship between their use of programming and their mathematical understanding. The findings indicate that, as programming is not designed as a mathematical or educational tool, its technical handling may be less intuitive for students than that of digital tools explicitly developed for mathematical purposes. A theoretical contribution of the thesis is that the analysis of students' instrumental genesis, when programming functions as a mathematical tool, must encompass not only mathematical conceptual aspects but also those required for learning to program.

The findings further suggest that using programming as a mathematical problem-solving tool, particularly when students construct their own algorithms, places considerable demands on those with limited programming experience. Conversely, providing pre-designed algorithms for numerical computations, to ease students' use of programming, may limit the development of deeper mathematical understanding. A practical contribution of the thesis is that teachers designing mathematical learning activities involving programming must balance scaffolding students' use of programming with allowing them autonomy to use the tool in ways that support their mathematical understanding.

Acknowledgements

More than eight years have passed since I began my journey as a part-time research student in 2017. It has been at times challenging, but above all an enormously rewarding experience during which I have learnt so much. Although my name appears on the cover of this thesis, its completion would not have been possible without the support of so many people.

First of all, I would like to acknowledge my excellent supervisors, Associate Professor Maria Fahlgren, Professor Yvonne Liljekvist, and Professor Emeritus Kenneth Ruthven. Your guidance and steadfast support throughout these years have meant so much to me, and you have taught me an enormous amount, not at least to believe in myself as a researcher. I could not have wished for better supervisors, and I will miss our recurring supervision meetings. Thank you for everything.

I am also deeply grateful to Karlstad University, Region Värmland, and Karlstad Municipality for funding my participation in the FUNDIG research school and for subsequently making it possible for me to continue as a doctoral student. A special thank you goes to Jorryt van Bommel and Yvonne Liljekvist at Karlstad University, as well as to Daniel Skålerud, school manager at Sundsta-Älvkullegymnasiet, who, in different ways, worked hard to ensure that I could continue my doctoral studies after completing my licentiate degree.

I would also like to thank the senior researchers at the Department of Educational Studies and the Department of Mathematics and Computer Science at Karlstad University. Through doctoral courses and seminars, you have provided valuable insights into educational research in general and mathematics education research in particular and have offered me invaluable support. I am especially grateful for the warm welcome I have received at the Department of Mathematics and Computer Science. A special thank you to my fellow doctoral students in both departments, especially Ann-Kristin Hamberg, Sofie Nilsson, Marcus Gustafsson, Jimmy Karlsson, and Annika Pettersson.

Additionally, I would like to thank Cathrine Andersson-Busch at the Department of Educational Studies for always looking after us doctoral students and for consistently having an answer to every question.

I would also like to thank Jesper Haglund, Johanna Pejlare, Ola Helensius, Kenneth Ruthven, and Alison Clark-Wilson, who provided valuable feedback and important suggestions for improvement as discussants at the mandatory seminars during the research project. A special thank you to Elisabet Mellroth and Michael Tengberg for carefully reviewing the thesis prior to publication.

Moreover, I would like to thank my principals at Sundsta-Älvkullegymnasiet, Joachim Weström and Sofia Andersson, for supporting me and for making it as easy as possible to combine my doctoral studies with my role as an upper-secondary school teacher. I would also like to thank my colleagues at Sundsta-Älvkullegymnasiet, especially my wonderful colleagues in the technology programme. A special thank you to Anna Wilk and Johan Windfäll for your support and friendship.

This thesis would not have been possible without the teachers and students who voluntarily participated in the two studies, to whom I am deeply grateful. Thank you to the three teachers who welcomed me into their classrooms and generously shared their valuable time. A special thank you to the teacher in the case study, who allowed me to participate throughout an entire mathematics course. I am also indebted to all the participating students, whose work I was able to follow through screen and voice recordings. Your work and conversations have given me so many new insights and so much new knowledge, and for that I am sincerely grateful.

Finally, thank you to my wonderful family for your constant support and belief in me. Mum and Dad—thank you for everything. And to Frida, Folke, and Tage—you mean the world to me.

Karlstad, February 2026
Andreas Borg

Table of content

LIST OF PAPERS	1
CO-AUTHORSHIP	1
1 INTRODUCTION	3
1.1 PROGRAMMING IN A HISTORICAL PERSPECTIVE.....	3
1.2 INTRODUCTION OF PROGRAMMING INTO SCHOOL MATHEMATICS.....	4
1.3 THE AIM OF THE THESIS	8
1.4 HOW THE PAPERS ADDRESS THE RESEARCH QUESTIONS.....	9
1.5 THE OUTLINE OF THE THESIS.....	12
2 THE CONTEXT OF THE THESIS	15
2.1 PROGRAMMING IN THE MATHEMATICS CURRICULA.....	15
2.2 MATHEMATICS TEACHERS' VIEWS AND USE OF PROGRAMMING.....	16
3 THEORETICAL PERSPECTIVES FOR INVESTIGATING THE INCORPORATION OF PROGRAMMING INTO MATHEMATICS EDUCATION	19
3.1 THE INSTRUMENTAL APPROACH.....	19
3.2 INSTRUMENTAL GENESIS	22
3.2.1 <i>Artefacts</i>	22
3.2.2 <i>Mental utilisation schemes</i>	23
3.2.3 <i>Instrumented techniques</i>	26
3.2.4 <i>Instrumentation and instrumentalization</i>	27
3.3 INSTRUMENTAL ORCHESTRATION.....	29
3.4 STRUCTURING FEATURES OF CLASSROOM PRACTICE.....	32
3.5 SUMMARY	37
4 EXTENDED LITERATURE REVIEW	39
4.1 KEY CHARACTERISTICS OF NOVICE PROGRAMMERS	39
4.1.1 <i>Logic errors and debugging</i>	41
4.1.2 <i>Devising a plan</i>	42
4.1.3 <i>Programming concepts</i>	43
4.2 MATHEMATICAL PROBLEM SOLVING	44
4.2.1 <i>Programming as a tool for mathematical problem solving</i>	46
4.3 A PROCEPTUAL VIEW OF MATHEMATICAL LIMITS	47
4.3.1 <i>Using programming to develop a proceptual view of limits</i>	48
4.3.2 <i>Obstacles to achieving a conceptual understanding of limits</i> ..	49
4.4 SUMMARY	49
5 METHODOLOGY	51

5.1	AN OVERVIEW OF THE TWO STUDIES	51
5.2	THE DESIGN STUDY	52
5.2.1	<i>The three phases of design research</i>	54
5.2.2	<i>Participants</i>	58
5.2.3	<i>Data generation and analysis</i>	60
5.2.4	<i>Summary</i>	62
5.3	THE CASE STUDY	63
5.3.1	<i>Characteristics of case study research</i>	64
5.3.2	<i>Participants</i>	65
5.3.3	<i>Data generation and analysis</i>	68
5.3.4	<i>Summary</i>	70
5.4	TRUSTWORTHINESS	71
5.5	ETHICAL CONSIDERATIONS	74
6	FINDINGS	77
6.1	STUDENTS' INSTRUMENTAL GENESSES WHEN USING PROGRAMMING FOR MATHEMATICAL PURPOSES	77
6.1.1	<i>Students' instrumental genesis in Paper 2</i>	78
6.1.2	<i>Students' instrumental genesis in Paper 5</i>	83
6.1.3	<i>The relationship between students' instrumental geneses in Paper 2 and Paper 5</i>	88
6.2	THE DESIGN OF THE LEARNING ACTIVITIES	90
6.2.1	<i>The instrumental orchestration in Paper 2</i>	90
6.2.2	<i>The teacher's design in Paper 4 and Paper 5</i>	94
6.2.3	<i>The relationship between the instructional design in Paper 2, Paper 4, and Paper 5</i>	98
6.3	THE ANALYSIS OF STUDENTS' INSTRUMENTAL GENESIS DURING PROGRAMMING ACTIVITIES.....	100
6.3.1	<i>Analysing schemes using scheme components</i>	101
6.3.2	<i>Analysing schemes using conceptual elements and technical elements</i>	102
6.3.3	<i>A comparison of the two analytical approaches</i>	103
6.4	SUMMARY OF KEY FINDINGS FROM THE TWO STUDIES	105
7	DISCUSSION	107
7.1	THEORETICAL CONTRIBUTIONS	107
7.1.1	<i>Two frameworks for analysing the instrumental genesis</i>	107
7.1.2	<i>Two frameworks for analysing teachers' instructional design</i>	111
7.2	PRACTICAL IMPLICATIONS	112
7.2.1	<i>Implications for in-service and pre-service teachers</i>	112
7.2.2	<i>Implications for policy</i>	117

7.3	FURTHER RESEARCH.....	120
8	REFERENCES	123
	APPENDIX: THE INTERVIEW GUIDE.....	139

List of papers

Paper 1. Borg, A., Fahlgren, M., & Ruthven, K. (2020). Programming as a mathematical instrument: The implementation of an analytic framework. *Proceedings of Mathematics Education in the Digital Age (MEDA), Linz, 16–18 September 2020*.

Paper 2. Borg, A. (2021). *Designing for the incorporation of programming in mathematical education: Programming as an instrument for mathematical problem solving* [Licentiate thesis, Karlstad University]. DiVA. Karlstad.

Paper 3. Borg, A., & Fahlgren, M. (2023). Analysing mathematical programming schemes using different lenses. *Nordic Studies in Mathematics Education*, 28(3–4), 199–219.

Paper 4. Borg, A. (2025). Incorporating programming into mathematics education: The adaptation of a teacher's craft knowledge. *Proceedings of the Fourteenth Congress of the European Society for Research in Mathematics Education (CERME14), Bozen-Bolzano 4–8 February, 2025*.

Paper 5. Borg, A., & Fahlgren, M. (2025). Students' development of instrumented action schemes for numerically determining limits using programming. *Digital Experiences in Mathematics Education*.

Co-authorship

In Paper 1, I was responsible for designing and conducting the research study, analysing the screen and voice recordings, and drafting the manuscript. All authors contributed to reviewing and revising the manuscript.

In Papers 3 and 5, I held the primary responsibility for designing and conducting the research studies, analysing the screen and voice recordings, and drafting the manuscripts. To enhance the interrater

reliability, the second author participated in the coding processes. Both authors reviewed and revised the manuscripts.

1 Introduction

This compilation thesis investigates upper-secondary school students' engagement with programming in the context of mathematical learning activities. It also explores how the design and orchestration of such activities can support and enhance this engagement.

This introduction begins by outlining aspects of the historical development of programming, followed by an examination of the incorporation of programming into school mathematics. The aim and research questions of this thesis are then articulated. Subsequently, a concise overview of the five papers is presented, accompanied by a discussion of how they are related to the researcher questions of the thesis. The chapter concludes with a description of the overall structure of this thesis.

1.1 Programming in a historical perspective

Throughout the history of mathematics, mathematicians have longed for machines able to conduct time-consuming calculations (Katz, 2009). The famous mathematician Gottfried Wilhelm Leibniz (1646–1716) argued that it was “unworthy of excellent men to lose hours like slaves in the labour of calculations, which could be safely relegated to anyone else if the machine was used” (Smith, 1959, p. 181). Leibniz himself, in 1671, designed a mechanical calculator that could conduct multiplication and division (Katz, 2009). The industrial revolution later provided the opportunity to use steam engines for powering mathematical machines. In 1833, the mathematician Charles Babbage introduces the idea of a general calculating machine whose operations were controlled by punch cards, which earlier had been used in weaving machines. Although Babbage's general calculating machine never was built, it inspired the mathematician Ada Byron King Lovelace to construct detailed instructions on how such machine could be instructed to solve a given calculation. These instructions are today referred to as the first computer program. In 1936, the young British mathematician Alan Turing published his paper *On Computable Numbers, with an Application to the Entscheidungsproblem*, in which he presented the idea of an electrical computer that could be programmed to solve any given calculation. This computer, referred to as the Turing machine,

was later used to decipher the Nazi's Enigma code. Since then, computers have become indispensable tools in mathematics, and programming has evolved into a routine practice among mathematicians.

1.2 Introduction of programming into school mathematics

The technical development of computers also meant that they were viewed as mathematical tools for educational purposes (Monaghan, 2016a). In the 1960s, Seymour Papert developed the programming language LOGO, intended to be used by younger students in school mathematics. Based on his *constructionistic* ideas¹ (Monaghan, 2016a), Papert (1980) argued that computers together with LOGO should be used as a “mathematically expressive medium, one that frees us to design personally meaningful and intellectually coherent and easily learnable mathematical topics for children” (p. 53). During the 1970s and 1980s it became affordable for schools to buy computers (Sutherland, 1994) and the idea of viewing programming as a form of experimental mathematics gained traction. Prior research has also illustrated how programming in school mathematics has the potential to evolve students' problem-solving ability (Sutherland, 1994), as well their understanding of variables (Sutherland, 1989), iterations, and recursions (Noss, 1986). Other studies have raised concerns about the impact of programming on students' conceptual understanding of mathematical concepts. In a study of undergraduate students' use of programming to determine limits of sequences numerically, Li and Tall (1993) claim that although the students were able to determine limit values, the use of programming did not foster their understanding of the formal definition of limits. Instead, Monaghan et al. (1994) argue that using programming to generate numerical values for limits may encourage a procedural interpretation of the limit concept, potentially leading to a one-sided view that emphasises the limit as a process rather than as an abstract mathematical concept. Cornu (1991) highlights the potential of programming to enhance students' conceptual understanding of mathematics. However, he argues that it is important to

¹ In constructionism, the project-based construction of meaningful objects (e.g., through programming) is regarded as a prolific way to facilitate learning (Lodi & Martini, 2021).

investigate the nature and structure of the knowledge students develop through such experiences.

During the 1990s, the use of programming in school mathematics experienced a decline. The programming languages were often perceived as difficult to use and students were not provided efficient support, partly due to mathematics teachers' limited experiences with using programming in their teaching (Resnick et al., 2009). Instead, other digital tools (e.g., advanced calculators, dynamic geometry software, and computer algebra systems) began to play more prominent roles in mathematics classrooms in the following decades. However, during the 2010s, there was a renewed interest in the use of programming in education, largely driven by educational reforms in many European countries (Bocconi et al., 2022; Tamborg et al., 2024). These reforms were motivated both by the tech industry's growing need for future workforce (Tamborg et al., 2024) and by broader calls for students to develop what in general could be referred to as digital competence (European Commission, 2019). In some countries, for example, Norway, Finland, and Sweden, programming is intended to be used in school mathematics. In Sweden, the rationale for integrating programming into the mathematics curriculum has been to ensure that all students acquire an understanding of what programming entails and develop the ability to use such technology (The Swedish National Agency for Education, 2022a). Furthermore, it is argued that the use of programming enables the exploration and application of mathematics in ways that extend beyond what is achievable through manual methods or through the use of basic digital tools (The Swedish National Agency for Education, 2025). In other countries, such as England, programming is integrated into a broader computing curriculum.

In response to the abovementioned educational reforms, the body of research concerning students' and teachers' use of programming in school mathematics has grown. The popularity of the programming environment Scratch contributed significantly to the renewed interest in programming in education, particularly at primary and lower secondary levels. In their literature review concerning teaching younger students programming in school mathematics, Holo et al. (2023) identify three major areas of research interest. First, a high proportion of the

research has focused on teachers' incorporation of programming into their mathematics classrooms and teachers' attitudes towards programming in mathematics education. Second, there has been an interest in examining the choice of programming tools and how teachers employ such tools to enhance students' learning of computer programming. Third, research has also focused on mathematical classroom activities that incorporate programming, and on learning outcomes associated with such activities. Holo et al. (2023) conclude that the teaching and learning of programming often is in the foreground whereas mathematics often remains in the background. They further emphasise the need for additional research on how programming affects students' learning within specific mathematical topics.

In contrast to the research undertaken at the lower-secondary level, the literature examining the effects of incorporating programming into upper-secondary mathematics education appears to be comparatively limited. Nevertheless, existing studies suggest that such incorporation has the potential to enhance students' mathematical understanding. For instance, Munthe (2022) has explored the role of mathematical programming problems when integrating programming into upper-secondary school mathematics. He shows that students experienced fewer difficulties related to the programming syntax when engaged with mathematical programming problems connected to familiar and well-known mathematical concepts and methods. Munthe (2024) also demonstrates that well-designed mathematical programming problems can facilitate exploratory mathematical discussions among upper-secondary school students. Taub (2024) highlights that students' use of programming to explore mathematical properties can foster mathematical reasoning. Similarly, by implying a quasi-experimental design, Psycharis and Kallia (2017) illustrate how learning programming can support upper-secondary students' mathematical reasoning skills. Moreover, their study indicates that programming instruction may also enhance students' self-efficacy in mathematics, although their problem-solving skills were not found to be statistically improved. Tossavainen et al. (2024) claim that programming code has the potential to serve as a "mediator of [upper-secondary school students'] mathematical thinking when explaining their mathematical ideas to one another" (p. 94). Furthermore, their findings suggest that programming-

based algorithms may support students' conceptual understanding of integrals and Riemann sums.

Although programming seems to have the potential to enhance students' conceptual understanding of mathematics, it is important to note that traditional programming languages and their associated software environments are neither designed as mathematical tools nor as educational tools (Buteau et al., 2020). This implies that the technical handling of programming tools may not always be straightforward for students. Furthermore, Haspekian et al. (2023) alongside Bråting and Kilhamn (2021) highlight that the different meanings of the variable notion and the equals sign in programming and mathematics risks hamper students' ability to transfer a mathematical algorithm to its computational counterpart.

In summary, during the last two decades, there has been a renewed interest in how programming can be utilised in mathematics education. Nevertheless, additional research is needed in several areas of this field to advance our knowledge of how the incorporation of programming into school mathematics can effectively support and deepen students' mathematical understanding. Hoyles and Noss (2021) emphasise the importance of conducting more detailed investigations of actual classroom practices. This includes "documentation of teacher and student interactions and output, in order to provide detail of classroom implementation" (p. 9). As highlighted by Holo et al. (2023), there is also a need for expanded knowledge on how programming influences students' learning within specific mathematical topics. Furthermore, as described in this section, the body of research examining the use and effects of incorporating programming into upper-secondary mathematics education remains limited. This thesis responds to these calls for expanded knowledge by investigating, within authentic classroom contexts, upper-secondary students' use of programming both as a problem-solving tool and as a means of numerically determining the limits of functions.

1.3 The aim of the thesis

The primary aim of the thesis is to investigate the intertwined relationship between upper-secondary students' use of programming for mathematical purposes and their mathematical understanding. Furthermore, the thesis aims to investigate how teachers' design of learning activities can facilitate the integration of programming into mathematics education.

Drijvers and Gravemeijer (2005) argue that when using a digital tool for mathematical purposes, the technical handling of the tool is closely related with the conceptual understanding of the mathematical content. This intertwined relationship is regarded as the core of a process referred to as *instrumental genesis* (Verillon & Rabardel, 1995). During the instrumental genesis, students develop mental utilisation schemes associated both with conceptual aspects concerning the mathematical content and with technical aspects concerning the handling of the artefact. Just as the use of digital tools can influence students' mathematical understanding, their existing mathematical knowledge also shapes how they engage with and utilise these tools (Trouche, 2004). The thesis places particular emphasis on students' instrumental genesis in examining how they utilise programming for mathematical purposes and how the design of learning activities shapes this developmental process. In this thesis, the design of learning activities involves the design of tasks, of technical artefacts utilised, of scaffolding provided to the students, and of the actual implementation of the activity. The above-mentioned investigations are guided by the following research questions:

1. What are the instrumental geneses of upper-secondary school students appropriating programming as an instrument for mathematical activity?
2. How can aspects of teachers' design of learning activities—in which students engage with programming as a mathematical tool—affect students' instrumental genesis?
3. How should analyses of students' instrumental genesis take account of the fact that the artefact—the programming environment—used during mathematical activities is not designed primarily as a mathematical tool?

The third research question is of interest since prior research predominately have focused on students' instrumental genesis in relation to technical artefacts designed for mathematical purposes.

1.4 How the papers address the research questions

In this section, the relationships between the research questions, papers, and studies are outlined. These relationships are also summarised in Table 1.

The research project comprises two studies. The first study is a design-based study where the design encompassed the development of mathematical problems lending themselves to programming. The design also comprised the orchestration of the learning activity in which Swedish upper-secondary school students were asked to solve the problems. Following the iterative approach of design research (Cobb et al., 2003), the study included two design cycles and aimed to develop a local instruction theory (Bakker, 2018) relating to the incorporation of programming as a problem-solving tool in upper-secondary school mathematics.

Papers 1, 2 and 3 originate from data generated during the design study. Paper 1, a conference proceeding, addresses the methodological question of how a given framework can be operationalised to analyse the intertwined relationship between students' use of programming and their mathematical understanding, described through their instrumental genesis. The findings in Paper 1 established the foundation for the analysis of students' instrumental genesis in Paper 2, which constitutes a licentiate thesis. Drawing on characteristics of educational design research, Paper 2 also examines how different aspects of the design of the learning activity influenced the instrumental genesis. Paper 3, a journal article, compares and contrasts the framework used to analyse students' instrumental genesis in Papers 1 and 2 with an alternative analytical framework.

Table 1: The relationships between research questions (RQ), papers, studies, and data sets

RQ 1. What are the instrumental geneses of upper-secondary school students appropriating programming as an instrument for mathematical activity?		
Papers addressing the research question	Study	Data set
Paper 2. Borg, A. (2021). Designing for the incorporation of programming in mathematical education: Programming as an instrument for mathematical problem solving.	Design study	Screen and voice recordings Students' programming code
Paper 5. Borg, A., & Fahlgren, M. (2025). Students' development of instrumented action schemes for numerically determining limits using programming.	Naturalistic case study	Screen and voice recordings Teacher interview
RQ 2. How can aspects of teachers' design of learning activities—in which students engage with programming as a mathematical tool—affect students' instrumental genesis?		
Papers addressing the research question	Study	Data set
Paper 2. Borg, A. (2021). Designing for the incorporation of programming in mathematical education: Programming as an instrument for mathematical problem solving.	Design study	Screen and voice recordings Students' programming code
Paper 4. Borg, A. (2025). Incorporating programming into mathematics education: The adaptation of a teacher's craft knowledge.	Naturalistic case study	Teacher interview Field notes
Paper 5. Borg, A., & Fahlgren, M. (2025). Students' development of instrumented action schemes for numerically determining limits using programming.	Naturalistic case study	Screen and voice recordings Teacher interview
RQ 3. How should analyses of students' instrumental genesis take account of the fact that the artefact—the programming environment—used during mathematical activities is not designed primarily as a mathematical tool?		
Papers addressing the research question	Study	Data set
Paper 1. Borg, A., Fahlgren, M., & Ruthven, K. (2020). Programming as a mathematical instrument: The implementation of an analytic framework.	Design study	Screen and voice recordings
Paper 2. Borg, A. (2021). Designing for the incorporation of programming in mathematical education: Programming as an instrument for mathematical problem solving.	Design study	Screen and voice recordings
Paper 3. Borg, A., & Fahlgren, M. (2023). Analysing mathematical programming schemes using different lenses.	Design study	Screen and voice recordings
Paper 5. Borg, A., & Fahlgren, M. (2025). Students' development of instrumented action schemes for numerically determining limits using programming.	Naturalistic case study	Screen and voice recordings

The second study could be described as a naturalistic case study of a Swedish mathematics teacher's incorporation of programming into his teaching of mathematics. A key aspect of the study involved examining how a class of students employed programming to explore mathematical concepts such as limits, derivatives, and integrals.

Papers 4 and 5, in this thesis, originate from data generated during the case study. Paper 4, a conference proceeding, provides an analysis of the mathematics teacher's design of programming-based learning activities, while Paper 5, a journal article, investigates the instrumental genesis of the teacher's students as they participated in these activities.

The first research question of the thesis concerns students' instrumental geneses when appropriating programming as a tool for mathematical activity, processes earlier analysed in relation to other mathematical artefacts (e.g., advanced calculators, dynamic geometry software, and computer algebra systems). This question is addressed in Papers 2 and 5. As previously described, Paper 2, presents the design study and offers an analysis of upper-secondary school students' instrumental genesis as they engage in solving mathematical problems using programming. Paper 5, building on the case study, explores students' instrumental genesis when using programming, specifically pre-designed code, to determine limit values numerically. The paper contributes to the first research question by examining how programming influences students' perceptions of limits.

The second research question concerns how teachers' design of learning activities, where programming is used as a mathematical tool, affects the development of students' instrumental genesis. This question is addressed in Papers 2, 4, and 5. In Paper 2, as part of the design study, the *instrumental orchestration* is analysed in relation to the design of a specific problem-solving activity. Paper 4 examines how the teacher in the case study integrated programming into mathematics instruction, and how this integration shaped the teacher's craft knowledge. This is examined using the *Structuring Features of Classroom Practice framework* (Ruthven, 2009), with particular attention to how the adaptation of craft knowledge relates to the teacher's design of the learning activities. Paper 5, building on the same case study,

analyses students' instrumental genesis in relation to the teacher's design of learning activities involving programming. As Bozkurt and Ruthven (2017) emphasise, the Structuring Features of Classroom Practice framework (SFCP) addresses the overarching structure of integrating digital tools into mathematics education, while the Instrumental Orchestration framework provides insights into the orchestration of specific learning activities. Nevertheless, it can be argued that these frameworks complement each other, and that aspects of overarching structures of programming integration, such as how programming is represented in the curricula, may influence the more fine-grained instrumental orchestration of particular learning activities.

The third research question explores how the analysis of students' instrumental genesis should account for the fact that the programming artefact used during mathematical activities is not primarily designed as a mathematical tool. This question is addressed in Paper 3 which examines two different approaches to operationalising the analysis of students' instrumental genesis when the artefact constitutes a programming tool. Paper 1 investigates the operationalisation of a scheme analysis using scheme components as defined by Vergnaud (1998a). In Paper 3, this analysis is contrasted with a scheme analysis that characterises students' schemes through their technical and conceptual elements, as outlined by Drijvers and Gravemeijer (2005). The first analytical approach was also applied in Paper 2, while the second was operationalised in Paper 5. The analyses of students' instrumental genesis presented in Papers 1, 2, and 5 therefore serve as valuable illustrations of how each framework can be applied.

1.5 The outline of the thesis

This first part of the compilation thesis provides a comprehensive overview of the research project², structured across seven chapters. This summary seeks to synthesise the individual papers in order to address the overarching research questions that frame the research project.

² Referred to as "kappa" in Swedish.

In Chapter 2, the context of the thesis will be presented to frame how programming has been integrated into the Swedish mathematics curricula and how such integration has been received by Swedish mathematics teachers. In Chapter 3, the theoretical foundations of the thesis are outlined, including a description of the Instrumental Approach, the Instrumental Orchestration and the Structuring Features of Classroom Practice framework. Chapter 4 presents an extended literature review that addresses specific aspects of three research areas relevant to this thesis. Chapter 5 presents a detailed account of the methodological approaches employed in the thesis. This includes an overview of the two studies, a discussion of the characteristics of educational design research and case study research, descriptions of the participants, an explanation of the data generation and analysis procedures, considerations of trustworthiness, and the ethical aspects of the research. In Chapter 6, the key findings of each paper are presented separately, alongside a synthesis that explores their interconnections with respect to the overarching research questions guiding the thesis. Finally, Chapter 7 offers a discussion of the theoretical and practical contributions of the research project, along with suggestions for future research.

The second part of the thesis comprises the full texts of the five included papers.

2 The context of the thesis

The two studies presented in this thesis (the design study related to Papers 1, 2, and 3, and the case study related to Papers 4 and 5) were conducted in two different Swedish upper-secondary schools. Since the results of these studies are shaped by the way programming has been integrated into Swedish school mathematics, this chapter provides an overview of that integration. The first section outlines the integration of programming into the Swedish mathematics curricula, while the second section examines how this integration has been perceived by Swedish mathematics teachers.

2.1 Programming in the mathematics curricula

In 2017, the Swedish Ministry of Education and Research (2017) released a strategy intended to guide the digitalisation of Swedish schools. As part of their overall digital competence, the strategy stated that students need knowledge of how programming affects their everyday life, for example, in relation to the flow of information and the technical tools students regularly use. To increase students' digital competence was regarded as a democratic issue. In line with the aims of the digitalisation strategy, programming was introduced into the revised national mathematics curricula in 2018.

In lower-secondary school, programming should be used as a mathematical tool from year four (approximately age 10), focusing on the design, interpretation, and modification of computational algorithms relating to mathematical tasks and problems (The Swedish National Agency for Education, 2022b). Whereas students initially should be introduced to visual programming (e.g., Scratch), text-based programming languages should be introduced during years 7–9. Since there is no dedicated computing subject in the Swedish compulsory school, as there is, for example, in England (Tamborg et al., 2024), Swedish mathematics teachers are expected to teach fundamental programming during mathematics lessons.

In the non-mandatory Swedish upper-secondary school, the intended use of programming in mathematics courses varies between types of

study programmes. In 2018, when incorporated into the upper-secondary school curriculum, programming was prescribed as a tool to be used during mathematical problem-solving activities. However, in 2021, the areas of use were broadened so that programming could also be used as a tool for solving mathematical tasks numerically and for data processing (The Swedish National Agency for Education, 2021). Thus, the Swedish national curriculum does not associate the use of programming to any specific mathematical content but to specific mathematical activities (Tamborg et al., 2024).

The 2021 revision of the mathematics curriculum also resulted in a slight reduction in the emphasis on programming. Prior to this change, upper secondary students enrolled in the science and technology programmes were required to use programming throughout all mathematics courses. However, following the revision, programming was formally introduced only from the third course onward. In the first two courses, the curriculum now mandates that students be presented with examples demonstrating how programming can serve as a mathematical tool.

2.2 Mathematics teachers' views and use of programming

The incorporation of programming into school mathematics has been met with concerns among Swedish mathematics teachers, where many, in 2018, had no or limited experience of programming themselves. This meant that numerous teachers felt that they lacked adequate professional knowledge to incorporate programming, in a proficient way, into their mathematics classrooms (Misfeldt et al., 2019; Vinnervik, 2022). Teachers also highlighted how vague instructions and a limited amount of time affected how they had been able to use programming in their teaching of mathematics (Humble, 2022). Although Swedish mathematics teachers have expressed concerns regarding the integration of programming into school mathematics, they nevertheless acknowledge the benefits of using programming as a mathematical tool (Misfeldt et al., 2019).

Kilhamn et al. (2021) illustrate how the arguments for incorporating programming may vary among Swedish teachers who have integrated

programming into their teaching of mathematics. In their study, 20 Swedish mathematics teachers, identified as early adopters, were interviewed about their views on programming in mathematics education. The researchers identified four different categories of arguments for using programming in school mathematics. The first category comprises arguments emphasising programming as a powerful tool once both the teachers and their students have learned to use it. As concluded by Kilhamn et al. (2021), this indicates that “some teachers trust new technology, embracing arguments about the usefulness of programming from other levels of the educational system, but without actually experiencing its usefulness themselves” (p. 174). The second category of arguments suggests that mathematical activities involving programming are perceived by students as more engaging and more closely connected to real life than traditional mathematical activities. Teachers making such arguments thus regard the use of programming as a fruitful way of engaging their students during mathematics lessons. The third category of arguments proposes that the use of programming could engage students in new ways of thinking and working (e.g., that programming offers students the possibility to test and debug). The final category of arguments emphasises that integrating programming into mathematics teaching may support students’ learning of specific mathematical content or that programming can itself be regarded as a mathematical activity. While these four categories capture the arguments advanced by early adopters, Kilhamn et al. (2021) further note that teachers who are less familiar with programming may rely on other and more general rationales when considering how, or indeed whether, to incorporate programming into mathematics education.

In her study of Swedish upper-secondary school teachers’ use of programming in their mathematics classrooms, Fuentes Martinez (2024) identified two so-called tactics for incorporating programming into mathematics education. *Dual teaching* is associated with a view of programming as an own discipline and that mathematical examples could be used to learn programming. Teachers adopt a dual-teaching tactic in order to “uphold existing course structures from previous years,—same lesson content, same textbooks—and at the same time grant their students the possibility to learn computer programming” (p. 144).

Interspersed teaching, on the other hand, is based on the view that knowledge of programming could benefit the teaching and learning of mathematics. Fuentes Martinez (2024) argues that teachers who adopt interspersed teaching “teach the normal sequence of mathematics units and recur to programming in those situations in which programming would be a reasonable tool to use within the topic” (p. 106). These two tactics have similarities to the two types of programming situations in mathematics education described by Misfeldt et al. (2019). In the first type, programming is used for tasks where mathematics is not necessarily inherently involved, such as controlling a turtle’s movement. In these cases, the teacher must actively integrate mathematical reasoning into the activity. In the second type, mathematics plays a central role, where programming is used to solve problems, explore mathematical phenomena, or construct mathematical tools. Here, the teacher’s role shifts to ensuring that students have sufficient programming knowledge to engage mathematically.

3 Theoretical perspectives for investigating the incorporation of programming into mathematics education

The overarching aims of the thesis are to investigate the intertwined relationship between upper-secondary students' use of programming for mathematical purposes and their mathematical understanding, as well as how teachers' design of learning activities can facilitate the incorporation of programming into mathematics education. To address these aims, the *Instrumental Approach* serves as the overarching theoretical framework for the two studies, and its origin and foundations will be presented in the first section of this chapter. Of particular interest within the Instrumental Approach, is students' *instrumental genesis*, which will be discussed in the second section. During this process, an artefact, in conjunction with students' mental utilisation schemes, evolves into a purposeful instrument for learning. The third section highlights the relationship between students' instrumental genesis and the teacher's *instrumental orchestration* of learning activities. In the fourth section, the *Structuring Features of Classroom Practice* framework will be introduced. This framework is operationalised to analyse how teachers' evolving craft knowledge both influences and is influenced by the incorporation of new technologies into educational settings. The chapter concludes with a short summary of the constructs which constitutes the theoretical foundations in this thesis.

3.1 The Instrumental Approach

Throughout history, the development of mathematics has been partly driven by the development of symbolic and physical tools (Artigue, 2002), for example, the decimal system, the logarithmic tables, and graphical calculators. Our use of mathematical tools has thus the potential to influence our understanding of mathematics. This intertwined relationship between the technical handling of a tool and mathematical understanding lies at the core of the Instrumental Approach. This theoretical framework serves as the overarching lens for investigating upper-secondary students' use of programming for mathematical purposes in this thesis.

The framework emerged in France during the late 1990s as a counter-balance to the idea that the technical handling of digital tools/artefacts³ (e.g., computer algebra systems) was separated from the conceptual understanding of the mathematical objects (Artigue, 2002; Ruthven, 2002). Instead, by drawing on ideas from the Anthropological Theory of Didactics, the Instrumental Approach emphasises that the technical work associated with the use of a mathematical tool is linked and intertwined with the conceptual understanding of the mathematical topic in question (Drijvers & Gravemeijer, 2005).

The Anthropological Theory of Didactics (ATD) recognises that students' understanding of mathematics is profoundly affected by social and cultural aspects within the given learning environment or institution (Artigue, 2002). Consequently, mathematical objects are not regarded as absolute entities but are developed through given activities and practices within the educational institutions. According to Artigue (2002), this implies that to “analyse the life of a mathematical object in an institution, to understand the meaning in the institution of ‘knowing/understanding this object’, one thus needs to identify and analyse the practices which bring it into play” (p. 248). In this sense, ATD has similarities with the socio-cultural perspective. In the ATD, these practices, referred to as praxeologies, involve four components: task, technique, technology, and theory (Chevallard & Bosch, 2020). The first component is the *task*, which is solved in a specific manner, using a specific *technique*. The manifested technique relies, in turn, on specific knowledge that justifies and explains the use of the specific technique for solving the task. This “discourse” (Chevallard & Bosch, 2020, p. 55) is referred to as the *technology*⁴, which in turn relies on a *theory* which “provides a structural basis for the technological discourse itself and

³ Monaghan (in Monaghan & Trouche, 2016) argues that an *artefact* should be conceptualised as a *tool* (a material object) when employed by an individual to accomplish a specific task. In this thesis, however, the two concepts will be given the same meaning to avoid ambiguities.

⁴ The ATD's notion of technology will, in this text, not be applied when analysing students' use of programming for mathematical purposes. Instead, technology will be used when referring to digital tools designed to facilitate learning and classroom management.

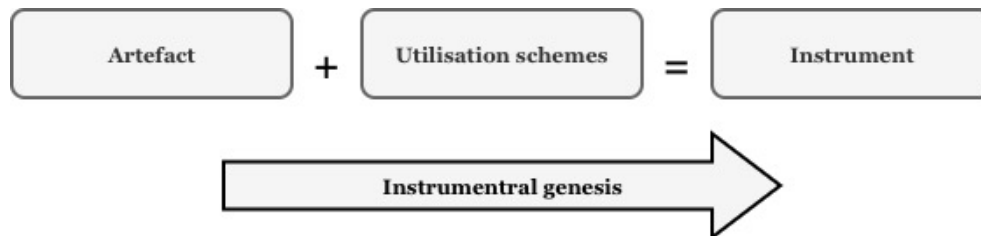
can be seen as a technology of the technology” (Artigue, 2002, p. 248). The concept of instrumented technique will be elaborated upon later in this text, particularly in relation to its role in Papers 3 and 5, where it is used to analyse students’ application of programming in solving mathematical tasks.

Since the ATD, in the late 1990s, mainly focused on traditional classroom practices, not involving the use of digital tools, the Instrumental Approach also draws on ideas from the field of Cognitive Ergonomics (Artigue, 2002) and especially the work of Verillon and Rabardel (1995). Although the field of Cognitive Ergonomics does not have a clear focus on educational settings, it focuses on human interactions involving the use of artefacts (Trouche, 2020) by building on the work of Vygotsky (e.g., the role of artefacts as facilitators of cognitive processes). In Cognitive Ergonomics, a distinction is made between an *artefact* and an *instrument* (Rabardel, 2002), where the former is regarded as an object (material or abstract) created by humans (Trouche, 2005a). An instrument, on the other hand, comprises the artefact and the mental utilisation schemes associated with the handling and use of the artefact for a given (e.g., mathematical) purpose (Verillon & Rabardel, 1995). An instrument thus consists of an artefact component and a cognitive component (Trouche, 2020), where the latter is related to specific knowledge for using the artefact in an appropriate manner for solving the given task. Trouche (2020) concludes that whereas an artefact could be described very generally (e.g., a calculator), greater precision is required when discussing an instrument since it concerns an “instrument of somebody, for performing a given type of task, at a given step of its development” (p. 407).

The process during which the instrument is formed (see Figure 1), and thus when adequate mental utilisation schemes are developed, is referred to as *instrumental genesis* (Rabardel, 2002) and is of great interest both in the field of Cognitive Ergonomics and within the Instrumental Approach. Although students’ instrumental genesis is an inherently individual process, it is important to acknowledge that this genesis is significantly shaped by the social dynamics of the classroom. Consequently, the mathematics teacher’s role is to foster a collective and productive instrumental genesis (Guin & Trouche, 1998). In the

following section, the different parts of the instrumental genesis will be described in more detail.

Figure 1: A schematic illustration of instrumental genesis, during which the artefact and the associated mental utilisation schemes develop into an instrument



3.2 Instrumental genesis

Within the Instrumental Approach, the *instrumental genesis* is of great interest since it concerns the process where students develop mental utilisation schemes associated with how the artefact could be used to accomplish the given mathematical objective (Drijvers & Gravemeijer, 2005). As aforementioned, these schemes and the artefact then form into an instrument, which should not be regarded as a physical entity but rather as a psychological construct. Schemes are defined by Vergnaud (1998b) as an “invariant organization of behavior for a certain class of situations” (p. 229), indicating that the instrument that emerges is intrinsically linked to the same class of situations. This infers that a student can develop several different instruments associated with the use of the same artefact (Drijvers et al., 2013).

3.2.1 Artefacts

What to regard as an artefact is, according to Drijvers et al. (2013), not always obvious, but rather a matter of granularity. This is also manifested by the two studies which comprise the foundation of this thesis. In the design study, associated with Papers 1, 2, and 3, the programming environment⁵ used by the students during the problem-solving

⁵ The programming environment utilised was NetBeans 8.2, an integrated development environment (IDE) primarily designed for developing applications in the Java programming language.

activity was considered the artefact. The choice of artefact was influenced by the expectation that the students would use programming to solve open-ended mathematical problems by constructing their own code from scratch. This meant that a wide range of programming concepts and functionalities within the programming environment could potentially come into play. In the case study, associated with Papers 4 and 5, students were expected to utilise numerical methods to solve mathematical tasks related to the topics of limits, derivatives, and integrals. To support students in using programming to solve these tasks, a pre-designed Python code structure was provided. This code structure, considered the artefact within this study, served as the foundation of the students' code and, to some extent, set boundaries for how the code could be developed. The different choices of artefacts were thus generally related to the type of study, including its aim and research questions. More specifically, it was related to the design of the specific learning activities involving the use of programming, developed by the researcher in the design study and by the teacher in the case study.

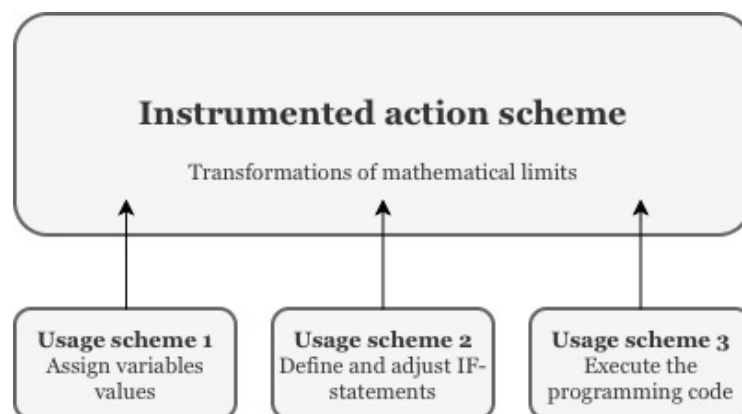
3.2.2 Mental utilisation schemes

The second part of the instrument is the mental utilisation schemes associated with the use of the artefact. Rabardel (2002) argues that such schemes could have three different functions. The epistemic function concerns understanding the given situation, the pragmatic function concerns transforming a given situation in order to achieve an appropriate result, and the heuristic function which “orient and control the activity” (p. 85).

There exist two levels of such utilisation schemes, where *usage schemes* are related to the direct handling of the artefact (Rabardel, 2002). When programming constitutes the artefact, a usage scheme may involve executing code or providing correct syntax when defining a loop. Although experienced users tend to apply usage schemes automatically, for novice users such schemes involve not only technical aspects, but also conceptual aspects (Drijvers & Gravemeijer, 2005) related to programming.

Instrumented action schemes on the other hand are associated with primary tasks, that is, actions directly related to the mathematical objectives. Consequently, applying an instrumented action scheme involves performing various transformations on mathematical objects (Drijvers & Gravemeijer, 2005). Since solving the primary tasks involves interaction with the artefact, the usage schemes serve as building blocks of the instrumented action schemes (exemplified in Figure 2). In this thesis, including its five papers, the focus will thus be on students' development and utilisation of instrumented action schemes. Although the focus of the analysis in this thesis will be on the instrumented action schemes, it is important to recognise that these schemes include components associated with given usage schemes.

Figure 2: A schematic illustration from the case study depicting the relationship between general usage schemes and a more situation-specific instrumented action scheme



Although mental schemes are developed by the individual, they also have social aspects. Rabardel (2002) stresses that although each scheme is distinctive since they are developed by a unique individual, schemes also have social dimensions as they are developed within a social context. Trouche (2004) argues that “it is impossible to distinguish, on the one hand cognitive structures (schemes) and on the other hand, cultural systems: schemes always have a social part and instrumental genesis always has individual and social aspects” (p. 288). In a mathematics classroom, students' development of utilisation schemes is, for example, influenced by choices made by the teacher (the artefact designer) and by the interactions with both the teacher and peers. Artigue

(2002) argues that the instrumental genesis thus often concerns “the appropriation of pre-existing social schemes” (p. 250).

In this subsection, it has been highlighted how schemes could have different functions (Rabardel, 2002). Vergnaud (1998a) claims that schemes also serve as theoretical constructs for describing and analysing problem-solving activities. He argues that such analysis should be operationalised by analysing four components constituting a scheme: goals and anticipations, rules of action, operational invariants, and possibilities of inferences. Every scheme is associated with *a goal (and sub-goals) and anticipations* related to the given task. Due to a scheme’s social dimension, these goals could arise from a negotiation between students and their teacher (Vergnaud, 1998a). The *rules of action* are the generative parts of a scheme which generate a given activity constituting “sequences of actions, information gathering, and controls” (Vergnaud, 2009, p. 88). These rules of action (e.g., creating nested loops in order to systematically combine variables) are in turn guided by *operational invariants* (Buteau et al., 2020). These invariants represent the epistemic aspects of a scheme (Vergnaud, 2009) and consist of *concepts-in-action* and *theorems-in-action*. Concepts-in-action (e.g., the idea of systematically combining variables) refer to concepts deemed as relevant and are activated in response to the given goal of the activity. These concepts-in-action are vital to identify and choose information (Vergnaud, 1998a) and are the building blocks of the theorems-in-action (Rabardel, 2002). While concepts-in-action can be relevant or irrelevant (or something in between), theorems-in-action can either be true or false (Vergnaud, 1998a). Theorems-in-action (e.g., systematically combining variables is a means for conducting an exhaustive trial) are thus “propositions held to be true by the subject when he or she acts” (Vergnaud, 1998b, p. 229) and influence decisions taken by students when trying to solve mathematical problems. The last scheme component is *possibilities of inferences*, which are related to the scheme’s ability to adapt to new situations (e.g., using nested loops in a new mathematical context). Such adaptation could, in turn, lead to the assimilation of new rules of action and operational invariants into the scheme.

Analysing students' instrumental genesis involves analysing the schemes developed, used, or modified by the students. In their study of university students' use of programming when engaging in mathematical investigations, Buteau et al. (2020) used the scheme components, as described by Vergnaud (1998a), as an analytical tool to examine students' instrumented action schemes. The analytical approach operationalised in the study by Buteau et al. (2020) served as an inspiration when, in Papers 1, 2, and 3 analysing the developing schemes of upper-secondary school students engaging in a mathematical problem-solving activity using programming. According to Buteau et al. (2020), describing students' instrumented action schemes through scheme components highlights the complexity of employing a programming tool, originally not intended for mathematical or educational purposes, as a means for mathematical investigations.

3.2.3 Instrumented techniques

Based on the Piagetian tradition carried forward by Vergnaud, schemes have within the Instrumental Approach served as a theoretical construct for describing students' cognitive development and their learning processes when using digital tools in mathematics education. But since schemes are mental constructs, they are not directly visible and could thus not be directly observed (Drijvers & Gravemeijer, 2005). Instead, Drijvers and Gravemeijer (2005) claim that the *instrumented techniques* used by the students should be regarded as the external and observable manifestation of an instrumented action scheme. They define an instrumented technique as a “set of rules and methods in a technological environment that is used for solving a specific type of problem” (p. 170).

In educational settings, instrumented techniques also serve as apparatuses for communicating (Lagrange, 1999). A teacher could teach students about how to solve new tasks by employing and explaining specific instrumented techniques. However, since successful application of such techniques to non-routine problems relies on well-developed instrumented action schemes, the explicit teaching of techniques also implicitly involves the cultivation of these underlying schemes. Artigue (2002) argues that instrumented techniques possess both pragmatic

values and epistemic values, where the pragmatic values are related to “efficiency, cost, field of validity” (p. 248). The epistemic value, which is of particular relevance in Paper 5, concerns how the use and development of productive instrumented techniques can foster and deepen students understanding of the mathematical objects under consideration.

Drijvers and Gravemeijer (2005) argue that the “visibility of instrumented techniques is why they, rather than the instrumented action schemes, which have a more internal and personal character, are the gateway to the analysis of instrumental genesis” (p. 169). They stress that an instrumented action scheme, manifested by its associated technique, can be described by identifying key conceptual and technical elements of the scheme. Turgut and Drijvers (2021) describe technical elements as “the more or less stable sequences of technical interactions between the user and the artefact” (p. 67) whereas conceptual elements guide the technical interaction with the artefact. The conceptual elements are shaped by the use of the artefact and its mathematical affordances and constraints (Drijvers et al., 2013).

In several studies, students’ instrumented action schemes, and thus their instrumental genesis, have been described by analysing the visible instrumented techniques (e.g., Drijvers & Gravemeijer, 2005; Fahlgren, 2017; Trouche, 2005a; Turgut & Drijvers, 2021). According to Drijvers and Gravemeijer (2005), employing technical and conceptual elements as analytical constructs to describe students’ instrumented action schemes, derived from their instrumented techniques, should be regarded as a valuable analytical approach. This approach facilitates the analysis of the often complex and non-trivial processes involved in students’ use of digital tools for mathematical purposes. In Papers 3 and 5, this analytical approach was operationalised to describe upper-secondary school students’ use of programming to solve mathematical tasks and problems.

3.2.4 Instrumentation and instrumentalization

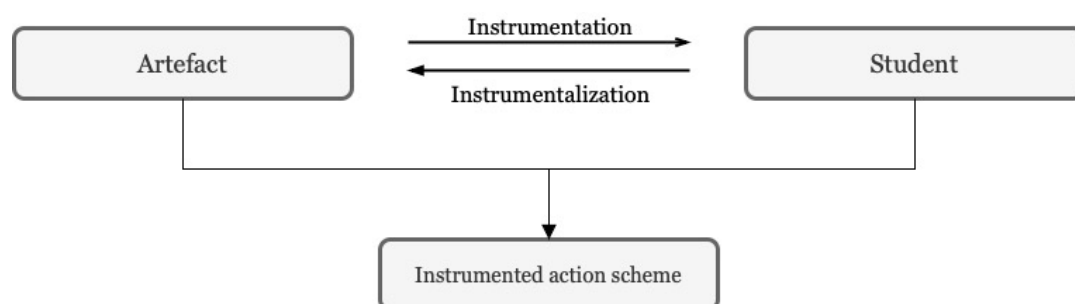
At the core of the Instrumental Approach is the notion that the ways in which students engage with a given artefact will affect how they

perceive the given mathematical objects (Drijvers & Gravemeijer, 2005). Thus, when a digital artefact is employed for mathematical purposes, it leaves a lasting impact on the user (Trouche, 2004). This is because the technical aspects involved in manipulating the artefact, in response to a given task, are closely intertwined with students' conceptual understanding of mathematics. This process, which is part of the instrumental genesis, is referred to as *instrumentation*.

If instrumentation involves a process where the use of the artefact prints its mark on the students, that is, “allows him/her to develop an activity within some boundaries (the constraints of the artifact)” (Trouche, 2004, p. 290), the students could also print their marks on the artefact when learning about its possibilities and constraints. During this *instrumentalization*, the students could customise and modify the artefact to fit their mathematical demands.

There is thus an intertwined relationship between the instrumentation and the instrumentalization as “the instrumentation process is the tracer of the artifact on the subject’s activity, while the instrumentalization process is the tracer of the subjects’ activity on the artifact” (Trouche, 2020, p. 409). Both processes should be regarded as important aspects of students’ instrumental genesis (see Figure 3) and illustrate how students’ thinking is affected by their use of the artefact and how the use of the artefact at the same time is affected by students’ thinking (Noss & Hoyles, 1996).

Figure 3: A schematic illustration of instrumental genesis adapted from Trouche (2004, p. 289)



3.3 Instrumental Orchestration

The way that mathematical objects have been portrayed and worked upon historically in schools has been deeply affected by the mathematical tools available (Haspekian et al., 2023). The use of pen and paper is associated with specific techniques, for example, when using algebraic manipulations to solve equations or when constructing a graphic representation of a function. In contrast, different techniques are required when conducting the corresponding mathematical operations using other tools. When a teacher introduces a new tool (e.g., programming) into her/his teaching, there will therefore emerge an *instrumental distance* between the two environments relating to the students' instrumental genesis. This instrumental distance is associated with technical and conceptual differences between the environments (Haspekian et al., 2023). Introducing a new digital tool into school mathematics thus place demands on teachers, as they are expected not only to teach students the techniques required to utilise the new tool, but also to reduce the instrumental distance between the new tool and the pre-existing ones. These demands can be addressed through the teachers' *instrumental orchestration* (Trouche, 2004) of the learning activity.

Although instrumented action schemes are internal and personal constructs, they are developed within a specific social context (Rabardel, 2002). From an educational perspective, students' instrumented action schemes are thus affected by the ways in which the teacher orchestrates learning activities involving the use of artefacts. Or in other words, the instrumental orchestration serves as an intentional guidance of students' instrumental genesis (Guin & Trouche, 2002; Trouche, 2005b), where choices made by the teacher "fine-tune the students' instruments and compose coherent sets of instruments" (Drijvers et al., 2009, p. 1350). Kendal and Stacey (2001) emphasise that teachers' diverse instrumental orchestrations, shaped by their instructional beliefs, may privilege different instrumented techniques and schemes. Consequently, what is considered as a productive instrumental genesis is closely intertwined with teachers' instrumental orchestrations.

Guin and Trouche (2002) outline four components of an instrumental orchestration: the set of individuals, the set of objectives, the didactical

configuration, and the exploitation mode. To further highlight the enactment of the instrumental orchestration, Drijvers et al. (2009) emphasise the necessity of also considering the didactical performance. *The didactical configuration* involves the design of the teaching settings involving decisions and considerations regarding which artefacts to use, which tasks to present and the broad structure of the lesson based on the given set of objectives (Drijvers et al., 2009; Drijvers et al., 2014). *The exploitation mode* refers to the given way a teacher intends to exploit the didactical configuration, for example, how to incorporate the artefacts alongside how to introduce, work on, and discuss chosen tasks (Drijvers, Doorman, et al., 2010). In this thesis, as presented by Drijvers, Doorman, et al. (2010), each didactical configuration will be associated with a given exploitation mode. *The didactical performance* constitutes the final component of an instrumental orchestration, taking into account how the didactical configuration and the exploitation mode are enacted (Drijvers et al., 2009). The didactical performance is also related to the ad hoc decisions taken by a teacher when teaching with technology. This could be related to which questions to pose at specific moments and how one handles students' different inputs and other unforeseen aspects during the lessons.

Drijvers et al. (2009) argue that a teacher's instrumental orchestration involves two levels. First, it is related to the idea of fostering students' instrumental genesis, that is, as described by Ruthven (2014) to "ensure that technico-mathematical development within a class follows a more collective path by means of which emergent knowledge is socialised into a shared form aligned with wider conventions and practices" (p. 381). The second level of the instrumental orchestration, part of the teacher's own instrumental genesis is, according to Drijvers et al. (2009), "instrumented by artefacts for the teachers, which may not necessarily be the same artefacts as the students use" (p. 1351).

Learning activities could be orchestrated in different ways and involve different orchestration types based on the teacher's didactical intentions (Ruthven, 2014), which in turn are related to given aspects of the didactical configuration and the corresponding exploitation mode (Drijvers et al., 2009). In their study of eighth-grade students' use of a mathematical applet incorporated in a digital learning environment,

Drijvers, Doorman, et al. (2010) highlight that various forms of instrumental orchestrations can be distinguished by the specific types of orchestrations employed. These orchestration types pertain to how teachers utilise digital artefacts, such as during whole-class explanations and during interactions with students. Tabach (2011) emphasises that the way teachers chose to utilise a digital artefact (i.e., the orchestration types they adopt) are closely linked to their own knowledge concerning the possibilities and constraints offered by the artefact. The number of orchestration types identified in the literature has increased substantially over the years (Drijvers, 2012; Drijvers et al., 2014; Tabach, 2011).

The sherpa-at-work orchestration type is of special interest in the design study described in Paper 2, as it played an important role in the design of the learning activity. The notion of *Sherpa* originates from Himalayan mountain guides⁶ who assist climbers during expeditions in the region (Guin & Trouche, 2002). Sherpas thus serve as experts by supporting other climbers, for example, by helping them navigate challenging terrain and by carrying their equipment. During sherpa-at-work orchestrations in mathematics education, originally described by Guin and Trouche (1998), chosen students (referred to as sherpa-students) are asked by the teacher to visualise (e.g., by sharing their screens) how they have used the artefact given the mathematical activity. This was also the role assigned to the sherpa-students during the design study described in this thesis. Drawing on the analogy of mountain guides, the purpose of involving sherpa-students was to support and guide their peers' work, and, in a sense, help them navigate the mathematical terrain. Sherpa-at-work orchestration could also involve allowing sherpa-students to solve a given task in front of the other students under guidance from the teacher (Guin & Trouche, 1998).

Instrumental Orchestration has previously been used as a theoretical lens to analyse both in-service and pre-service teachers' use of digital tools, such as computer algebra systems (Guin & Trouche, 2002),

⁶ The term Sherpa also refers to an ethnic group native to the Himalayan region. However, not all mountain guides referred to as Sherpas are ethnically Sherpa.

dynamic mathematics software (Bozkurt & Koyunkaya, 2022), and pre-designed mathematical applets (Drijvers, Kieran, et al., 2010; Drijvers et al., 2014). Instrumental Orchestration has also been used by Buteau et al. (2022) when describing the way programming was integrated into university courses as a tool for mathematical investigations. In this thesis, Instrumental Orchestration served as a theoretical construct in Papers 2 and 4.

In this section, the Instrumental Orchestration has been presented as a productive lens through which to examine how the orchestration of specific learning activities influences students' instrumental genesis. However, as Ruthven (2009) emphasises, teachers' orchestrations is grounded in their craft knowledge. Integrating programming into mathematics education may necessitate an adaptation of this craft knowledge, an adaptation shaped by *structuring features of classroom practice*. Bozkurt and Ruthven (2017) argue that these structuring features, along with the evolving nature of teachers' craft knowledge constitute the foundation of their instrumental orchestrations. In Paper 4, the adaptation of a teacher's craft knowledge is analysed through the lens of the Structuring Features of Classroom Practice framework (Ruthven, 2009), focusing on how the mathematics teacher adapts his craft knowledge when incorporating programming into his teaching.

3.4 Structuring Features of Classroom Practice

As described in the previous section, Instrumental Orchestration serves as a theoretical lens for describing and analysing how teachers incorporate digital tools (e.g., programming) into their teaching to support students' instrumental genesis. Ruthven (2009) argues that the incorporation of such technologies into mathematics classrooms is dependent on a professional adaptation of the teacher's craft knowledge. Cooper and McIntyre (1996) claim that such knowledge is very much formed by the teacher's classroom experiences:

[C]raft knowledge describes the knowledge that arises from and, in turn, informs what teachers do. As such, this knowledge is to be distinguished from other forms of knowledge that are not linked to practice in this direct way. Craft knowledge is not, therefore, the kind of knowledge that teachers draw on when explaining the thinking underlying their ideal teaching practices. Neither is it knowledge drawn from theoretical sources. Professional

craft knowledge can certainly be (and often is) informed by these sources, but it is of a far more practical nature than these knowledge forms. Professional craft knowledge is the knowledge that teachers develop through the processes of reflection and practical problem-solving that they engage in to carry out the demands of their jobs. (p. 76)

In order to describe and analyse how the professional adaptation of teachers' craft knowledge shapes the incorporation of new technologies, Ruthven (2009) developed the Structuring Features of Classroom Practice framework (SFCP). The framework builds on, and extends, concepts from previous research concerning the organisation of classroom practices and the development of teachers' craft knowledge. The framework identifies five structuring features of classroom practice which influence the ways teachers incorporate new technologies into their teaching: the working environment, the resource system, the activity format, the curriculum script, and the time economy.

The working environment is a structuring feature associated with the potential need to adapt the physical environment and/or the organisation of the class when incorporating a new technology. An adaptation of the working environment will thus bring about new classrooms routines compared to teaching not involving digital tools. Ruthven (2009) exemplifies how students' use of personal computers changes the working environment, for example, by providing students with (unwanted) opportunities for distraction. Craft knowledge related to the working environment can, for example, concern how to allow students access to the new technology and related material (Ruthven, 2009).

For today's mathematics teachers, there is an enormous range of digital resources⁷ and the teachers' job is not just to decide which resources to use in teaching but also to create a coherent system of resources for the given educational purposes. A *resource system* could thus be described as a "[c]ollection of didactical tools and materials in use, and coordination of use towards subject activity and curricular goals" (Ruthven, 2014, p. 387). Craft knowledge related to the creation of a coherent system of resources is associated with teachers' ability to develop

⁷ Ruthven (2009) refers to resources as "some (physical or virtual) artefact which has either been designed specifically for curricular purposes, or been the subject of educational appropriation for such purposes" (p. 146).

appropriate instrumented techniques for using the technology, as well as their ability to coordinate the use of new and existing resources. Such coordination is referred to by Ruthven (2014) as double instrumentation and constitutes part of the teachers' professional instrumental genesis (Haspekian, 2011).

Incorporating new technologies into classrooms requires an adaptation of teachers' craft knowledge relating to the *activity format* (Ruthven, 2009) and the structure of the activity. That is, the way a learning activity is organised in terms of how the content is presented and worked upon, and how students may be grouped for instructional purposes (Burns & Anderson, 1987). Adapting the activity structure involves, among other things, stating the role the new technology should play during lesson activities and its interaction with the teacher and the students (Ruthven, 2014). Bozkurt and Ruthven (2017) discuss how aspects of the activity format are related to the instrumental orchestration of the learning activity, particularly its exploitation mode, during which the teacher exploits the organisation of the instructional setting and the artefacts involved (Drijvers et al., 2009). They argue that the orchestration types identified by Drijvers, Doorman, et al. (2010), and later extended by, for example, Tabach (2011), Drijvers (2012), and Drijvers et al. (2014), could serve as a taxonomy for how classroom activities are structured around particular activity formats. In their study of teachers' incorporation of a specific dynamic digital technology, Simsek and Clark-Wilson (2024) employed the idea of networking aspects of the SFCP framework and the Instrumental Orchestration framework. They claim that the two frameworks are mutually complementary and that Instrumental Orchestration is particularly useful for examining how mathematics teachers manage classroom activities involving the use of digital technology. The Instrumental Orchestration framework thus offers "a valuable lens for a finer-grained analysis of the construct of activity structure within the broader SFCP framework" (p. 611).

The curriculum script is a structuring feature which is connected to other structuring features, for example, the resource system and the activity format, where the notion of script should be regarded as a

psychological construct (Ruthven, 2009). Ruthven (2014) provides the following description of a curriculum script:

This ‘script’ is an event-structured organisation of knowledge, forming a loosely ordered model of goals, resources and actions for teaching the topic, incorporating potential emergent issues and alternative courses of action; it interweaves mathematical ideas to be developed, appropriate topic-related tasks to be undertaken, suitable activity formats to be used, and potential student difficulties to be anticipated, guiding the teacher in formulating a suitable lesson agenda, and in enacting it in a flexible and responsive way. (p. 386)

When teaching a given topic, Putnam (1987) argues that a teacher follows the curriculum script and based on the students’ responses, adjust the teaching and the instructions. The incorporation of new digital tools affects several aspects of the teaching related to the curriculum script (Ruthven, 2014). For example, the teacher must take into consideration the possibilities and constraints associated with the new technology in relation to the mathematical learning objective. Moreover, new tools used in mathematics classrooms may require tasks fitted to be solved using the tool. The teacher must also prepare for different types of student responses and difficulties. The examples illustrate how various aspects of teachers’ curriculum scripts can influence and reshape their craft knowledge as they incorporate new technologies into their mathematics classrooms.

The fifth structuring feature is *time economy* and concerns the cost, in terms of time, of incorporating new technologies (Ruthven, 2009). A teacher must in general take into account what Assude (2005) refers to as the time capital (i.e., the actual time allocated for classroom activities) and manage the “estimated temporal cost of each activity and the global time of all activities put together” (p. 185). The incorporation of a new technology into the classroom can be made at a high cost if the time allocated to the incorporation is not in proportion to the results of the learning outcome. But the use of a new digital tool could also save time by outsourcing time-consuming calculations to the tool or easing students’ learning at a low temporal cost. According to Ruthven (2014), a teacher’s craft knowledge related to the time economy feature concerns the ability to change “the ‘rate’ at which the physical time available for classroom activity can be converted into a ‘didactic time’ measured in terms of the advance of knowledge” (p. 386). Although the time

economy is presented as a specific structuring feature, the changing of this rate is closely associated with all the other structuring features of the framework.

The five structuring features of classroom practice described in this section serve as theoretical constructs for analysing how the incorporation of new technologies into mathematics classrooms shapes the professional adaptation of teachers' craft knowledge. Ruthven (2012) argues that the SFCP framework "makes clear the scope and scale of the situational adaptation and professional learning required for teachers to successfully incorporate use of digital tools and resources in support of investigative approaches" (p. 627). Bozkurt and Ruthven (2017) claim that these structuring features not only guide teachers' orchestration of a specific lesson but also function as underlying structures that shape their overall teaching practice.

The different structuring features could, to different degrees, play prominent roles in the incorporation of new technologies into mathematics education. Drawing on their study of teachers' incorporation of a dynamic digital technology (Cornerstone Math) into upper-secondary mathematics education, Simsek and Clark-Wilson (2024) emphasise that the working environment and the time economy were less closely associated with domain-specific mathematical issues compared to the other structuring features. Instead, Simsek and Clark-Wilson (2024) argue that these two components offer a more contextualised perspective for describing the incorporation of technologies into mathematics education.

The SFCP framework has previously been operationalised as a theoretical and analytical lens when describing teachers' incorporation of different technologies, for example, dynamic mathematical software (Bozkurt & Ruthven, 2017), connected classroom technology (Chorney, 2022; Gustafsson, 2017), and dynamic digital technology (Simsek & Clark-Wilson, 2024). It has also been synthesised with the Documentational Approach to Didactics to analyse teachers' design and use of digital resources (de Moraes Rocha & Trouche, 2017). Furthermore, Buteau et al. (2019) have used the framework to highlight demands placed on university teachers when implementing a specific course in

university mathematics involving programming. In Paper 4, the SFCP framework serves as a theoretical and analytical lens to analyse the adaptation of a Swedish mathematics teacher's craft knowledge when incorporating programming into upper-secondary mathematics education.

3.5 Summary

In this chapter, the frameworks forming the theoretical and analytical foundations of this thesis have been presented. When, in Paper 2 and Paper 5, analysing upper-secondary students' use of programming as a tool in school mathematics, the Instrumental Approach serves as a theoretical lens. Of special interest is therefore the analysis of students' instrumental genesis, including the development of instrumented action schemes and the two processes of instrumentation and instrumentalization. Illustrating the instrumented action schemes could be done by using scheme components as defined by Vergnaud (1998a), operationalised in Papers 1 and 2, or by viewing the technical and conceptual elements related to students' instrumented techniques as the observable manifestation of an instrumented action scheme (Drijvers & Gravemeijer, 2005), as operationalised in Paper 5. These two different approaches to describe and analyse students' instrumented action schemes are also compared and contrasted in Paper 3.

Decisions taken by teachers to facilitate a productive instrumental genesis are part of the teachers' instrumental orchestration, which according to Drijvers, Doorman, et al. (2010) concerns three main components: the didactical configuration, the exploitation mode, and the didactical performance. The theoretical lens of Instrumental Orchestration was utilised in Paper 2 as part of the design of a mathematical problem-solving activity involving the use of programming. Ruthven (2009) argues that the incorporation of new technologies (e.g., programming) into school mathematics requires an adaptation of teachers' existing craft knowledge and that such adaptations are affected by five different structuring features of classrooms practice. An example of how the incorporation of programming calls for an adaptation of a mathematics teacher's craft knowledge is presented in Paper 4.

Each theoretical framework is accompanied by its own limitations. For instance, Instrumental Orchestration emerged as a theoretical lens for examining how individual instrumental genesis can be supported within the social dynamics of the classroom (Trouche, 2004). However, there are aspects that may influence students' instrumental genesis which fall outside the scope of the Instrumental Approach, such as students' emotions, attitudes, values, and motivation. It may also be questioned how precisely a mental scheme, existing solely in a student's mind, can be described. Nonetheless, given its demonstrated applicability across various research contexts, the Instrumental Approach is regarded in this thesis as a productive and appropriate theoretical lens for analysing students' use of programming for mathematical purposes.

4 Extended literature review

This chapter presents an extended literature review addressing specific aspects of three research areas pertinent to this thesis. Given that the students in the two studies possessed limited programming experience, the first section introduces key characteristic of novice programmers, including a progression model intended to support novice programmers in developing programming proficiency. The second section offers an overview of research related to mathematical problem solving, which informed the design of the first study, where students engaged in a mathematical problem-solving activity through programming. The third section examines the theoretical construct of *procept* in the context of mathematical limits and explores how students' development of a *proceptual* view of limits may be shaped by the incorporation of programming into the learning process. This perspective informed the analysis in Paper 5, which examined how students' use of programming to numerically approximate limit values shaped their mathematical understanding of limits. Additionally, the section reviews existing research on students' conceptions of limits. The chapter concludes with a summary of the literature review in relation to the two studies presented in this thesis.

4.1 Key characteristics of novice programmers

Moving from a novice programmer to an expert programmer is a time-consuming endeavour (Winslow, 1996) and there is an extensive body of research concerning the characteristics of novice programmers⁸. Lee et al. (2011) propose a progression model to support novice programmers in developing both programming skills and computational thinking. This model comprises three stages: Use, Modify, and Create. In the *Use* stage, students engage with pre-designed code by applying and interpreting it. The *Modify* stage involves adapting and refining existing code, which fosters the development of new skills and understandings. This stage functions as a critical bridge between consumption and

⁸ Although some of the references cited in this section are dated, they represent seminal work whose findings remain highly relevant within the research field.

creation. It catalyses the learner's transition from being a consumer of others' work to, in the *Create* stage, becoming a producer of original code. As Lee et al. (2011) note, it is through this process that "what was once someone else's becomes one's own" (p. 35).

The model will be referenced in Chapters 6 and 7, where the results of the thesis are presented and discussed. It also functions as a theoretical construct in Paper 5, which describes the case study, supporting the analysis of students' use of programming for mathematical purposes and their instrumental genesis. In the case study, the teacher's lesson design incorporated both a pre-designed code structure and provided code examples which the students were instructed to use and modify in order to determine limit values numerically.

During the lesson activity in the design study, students were expected to develop their own code. The rationale for designing mathematical problems aligned with the *Create* stage of the progression model was grounded in the Swedish upper-secondary mathematics curriculum, which at the time stipulated that programming should be employed as a tool for mathematical problem solving. Consequently, a key aspect of the design was to enable students to engage independently in an unbiased problem-solving process, requiring them to develop solutions from scratch. As previously described, the areas of application specified in the mathematics curriculum were broadened in 2021 to allow programming to also be used as a tool for solving mathematical tasks numerically and for data processing (The Swedish National Agency for Education, 2021). The expanded utilisation of programming influenced the task design presented in the case study, resulting in an emphasis on the *Use* and *Modify* stages of the progression model.

Facilitating successful progression through the model's three phases requires an understanding of the challenges commonly encountered by novice programmers. The following subsections address logic errors frequently encountered by novice programmes, as well as difficulties associated with debugging and developing programming plans. Additionally, programming concepts that novice programmers often perceive as particularly challenging are described.

4.1.1 Logic errors and debugging

When modifying, creating and executing their programs, programmers often encounter bugs and errors which means that they need to debug their programs (Brennan & Resnick, 2012). Ettles et al. (2018) define three categories of *logic errors* made by novice programmers, where a logic error is perceived as “situations where the programmer’s code compiles successfully and executes but fails to generate the expected output for all possible inputs” (p. 83). The first category concerns *algorithmic errors*, related to the algorithmic development. Bonar and Soloway (1985) argue that perceived similarities between constructing step-by-step plans using natural language and programming language often lead to algorithm errors. These difficulties may stem from *functional similarities* (e.g., the use of repeated actions in both languages) alongside *surface similarities*, since programming languages share many common words with natural language, including FOR, WHILE, IF, and THEN. The second category of errors identified by Ettles et al. (2018) arises from *misinterpretations* of the task outline, while the third category stems from *misconceptions*. These misconceptions include syntactic errors and often originate from insufficient programming knowledge.

Mannila et al. (2006) highlight that the choice of programming language can influence the frequency and types of errors made by students. In their study, they compare upper-secondary school students’ use of the programming languages Java (used by the students in the design study of this thesis) and Python (used by the students in the case study). The results indicate that students working with Python made fewer errors than those programming in Java. Given that Java syntax is generally perceived as more complex than Python’s, Mannila et al. (2006) argue that it is unsurprising that students using Java committed more syntactic errors. However, the result of the study also shows that logic errors were more prevalent among the Java users. Mannila et al. (2006) conclude that an extensive focus on a “verbose and complex syntax might result in novices finding it necessary to focus on getting the syntax correct to such an extent that the algorithm becomes a secondary concern only” (p. 218). A similar conclusion is drawn by Koulouri et al. (2014) in their study of university students enrolled in an introductory programming course. The researchers argue that the

advantage of using a less complex programming language is that it allows students to concentrate on understanding programming concepts and developing debugging skills.

Debugging is often perceived as a frustrating and challenging endeavour for novice programmers (Rigby et al., 2020) and Papert (1980) observed that many students were reluctant to engage in debugging when they encountered difficulties using LOGO. However, Papert (1980) argues that, within the context of school mathematics, debugging should be regarded as a valuable learning activity, one through which students can gain important mathematical insights and learn from their mistakes.

4.1.2 Devising a plan

Designing a program involves constructing a computational algorithm. Lahtinen et al. (2005) state that novice programmers, although they understand the programming concepts, often struggle to apply them. Even though they realise how to solve the tasks by hand, novice programmers often experience difficulties when translating their hand solutions into code (Winslow, 1996). They may also be able to express a step-by-step instruction using natural language but struggle to transfer this instruction into a computational algorithm, which according to Bonar and Soloway (1985) is not surprising due to novice programmers' limited programming knowledge. This also implies that novices struggle to link programming concepts together when devising a plan (Ginat, 2004).

When constructing their code, students must also realise the sequential nature of steps in imperative programming⁹ (du Boulay, 1989). This implies that their code is executed line-by-line and that “each instruction operates in the environment created by the previous instructions” (p. 189). du Boulay (1989) argues that this aspect is often forgotten by novice programmers and Ahmadzadeh et al. (2005) conclude that

⁹ Imperative programming refers to a paradigm where programs are expressed as ordered sequences of instructions that progressively alter the program's state.

novice programmers often perceive their programs as a set of events which occur simultaneously when executing the code.

4.1.3 Programming concepts

Programming and mathematics share various concepts, for example, variables, the equals sign, and iterations. However, the different meaning of these notions in programming and mathematics tends to cause novice programmers difficulties.

In programming, variables are used to store different types of data. This requires the user to initialise each variable and specify the type of data it is intended to hold. The variable must also be assigned a value before being used to assign other variables new values. This meaning of variables differs from the mathematical meaning of variables as something unknown (Küchemann, 1978). du Boulay (1989) claims that novice programmers often struggle to realise the difference between computational and mathematical variables since computational variables have “a kind of mathematical flavour” (p. 290). Computational variables frequently adopt names traditionally used in mathematics (e.g., x or a) and are often employed alongside other mathematical symbols, such as the equals sign.

In programming the single equals sign ($=$) is used to assign variables values. For example, writing $A = 10$ assigns the value of 10 to the variable A . In programming, unlike in mathematics, there is an asymmetry in assignment statements. Specifically, writing $10 = A$ is incorrect, as assignments must follow the form $A = 10$, where the variable appears on the left-hand side and the value being assigned on the right. In programming, $A = A + 10$ represents an operation where A is assigned a new value which equals the current value of A added by 10, whereas in mathematics it represents an equation (without solutions). du Boulay (1989) argues that novice programmers struggle to comprehend the asymmetry in such assignments. He also highlights how novices tend to regard assignments such as $A = B$ as persistent links between two variables valid throughout the code. Kohn (2017) further emphasises this point, arguing that novice programmers often interpret such assignments as mathematical definitions or formulas that apply globally

within the code. This interpretation, according to Kohn (2017) “corresponds quite directly to the way students solve mathematical problems and perform calculations on their own” (p. 349).

Using computational loops when programming in school mathematics enables students to be naturally exposed to mathematical iterations (Noss, 1986). However, in their study of novice programmers’ use of programming concepts, Cherenkova et al. (2014) highlight that students often struggle to create iterations using computational loops. These difficulties are, among other factors, due to students’ imprecise definitions of loop boundaries (Ginat, 2004). Cetin (2015) claims that novice programmers may be able to create basic loops but struggle to view the loop as an encapsulated object. Instead, when asked to describe the loop, students describe it “explicitly in a step by step manner” (p. 163). Being unable to regard a loop as an encapsulated object also affects students’ use of nested loops, that is, when an inner loop is part of the body of an outer loop. Nesting loops enables the generation of combinations of variables, a feature that is of particular relevance in Paper 2. Existing research indicates that novice programmers often struggle to understand when the use of nested loops is appropriate (Yarmish & Kopec, 2007). Furthermore, when nested loops are employed, students frequently encounter difficulties in coordinating the boundaries of the inner and outer loops (Alzahrani et al., 2018; Ginat, 2004). Yarmish and Kopec (2007) argue that novice programmers should be exposed to a variety of scenarios in which nested loops are required, in order to develop a deeper understanding of how to construct and apply such loops effectively.

4.2 Mathematical problem solving

Problem solving is a central activity for all mathematicians where “an individual (or group) [...] engage[s] in a variety of cognitive actions, each of which requires some knowledge and skill, and some of which are not routine” (Lester, 2013, p. 248). The quest among mathematicians to take on different problems has historically contributed to the development of mathematics (Santos-Trigo, 2024). Blum and Niss (1991) categorise two types of mathematical problems, *applied*

mathematical problems, related to real-world problems, and non-contextual problems referred to as *purely mathematical problems*.

Problem solving also plays a prominent role in school mathematics, as problem-solving activities have the potential to foster students' mathematical understanding (English & Sriraman, 2010). In the Swedish mathematics curricula, problem solving is described both as a mathematical activity and a mathematical ability (The Swedish National Agency for Education, 2021).

An extensive amount of research has been conducted concerning the learning and teaching of mathematical problem solving. Schoenfeld (1992a) argues that to become proficient mathematical problem solvers students must possess relevant content knowledge, familiarity with a range of problem-solving strategies, and strong metacognitive skills. Although metacognitive ability often is recognised as important to become a good problem solver (e.g., Lester, 2013; Zawojewski & Lesh, 2003), existing research provides little evidence on how best to foster this ability in students (Lester, 2013).

There is also limited evidence regarding effective teaching methods that enhance students' problem-solving ability (English & Sriraman, 2010). Notably, several attempts have been made to create models of heuristic strategies for mathematical problem solving, the most well-known of which was proposed by Pólya (1971). However, the application of such heuristic strategies has been criticised for being "descriptive rather than prescriptive" (Schoenfeld, 1992a, p. 353). Consequently, it is easy to recognise the problem-solving strategy when the problem is solved but, as highlighted by Schoenfeld (1992a), "the characterizations [do] not provide the amount of detail that [...] enable people who [are] not already familiar with the strategies to be able to implement them" (p. 353). English and Sriraman (2010) argue that merely providing students with a variety of problem-solving strategies and subsequently exposing them to non-routine problems is an ineffective means of enhancing their problem-solving ability. They also challenge the view that students must first be taught concepts and procedures, and engage in routine practice, before they can solve related problems. Instead, English and Sriraman (2010) argue that problem

solving fosters the conceptual development of mathematical ideas and should be an integral component when introducing new mathematical concepts and procedures in school mathematics.

In conclusion, existing research emphasises that mathematical problem solving is “an extremely complex form of human endeavour that involves much more than the simple recall of facts or the application of well-learned procedures”. (Lester & Kehle, 2003, p. 509). Rather, becoming a proficient problem solver in mathematics necessitates well-developed metacognitive skills and a substantial investment of time in tackling a diverse range of problems that require the application of various problem-solving strategies (Hiebert et al., 1996; Lester, 1996).

4.2.1 Programming as a tool for mathematical problem solving

In the design-based study described in Paper 2, Swedish upper-secondary school students’ use of programming as a problem-solving tool was investigated. This was of special interest since the mathematics curriculum for Swedish upper-secondary education emphasises the role of programming as a mathematical tool to support problem-solving activities. Existing research also stresses the relationship between programming and problem solving (e.g., Papert, 1980; Sutherland, 1994). In their literature review concerning younger students’ use of programming, Holo et al. (2023) emphasise that existing research suggests programming can support various aspects relating to students’ mathematical problem-solving ability. However, in a separate literature review, Popat and Starkey (2019) argue that while learning programming may involve mathematical problem-solving activities, there is no clear evidence that students become more proficient problem solvers through programming than through traditional learning activities. Furthermore, while Psycharis and Kallia (2017) illustrate how learning programming can promote upper-secondary students’ mathematical reasoning skills, they found no statistically significant differences in students’ problem-solving skills. They argue that merely learning to program is insufficient to enhance students’ problem-solving ability. In addition, students need opportunities to practice applying programming as a tool for mathematical problem solving.

4.3 A proceptual view of mathematical limits

Students' perceptions of mathematical limits are of special interest in Paper 5, where upper-secondary students utilise programming to determine the limits of functions numerically. Limits occupy a central position in mathematical analysis, forming the foundational basis for the concepts of continuity, differentiation, integration, and approximation theory. However, existing research indicate that students often struggle to fully understand the meaning of the concept and especially the formal ε - δ definition (Cornu, 1991).

Of special interest in this thesis, which concerns upper-secondary students' perceptions of limits, is the informal, or intuitive definition of limits:

$$\lim_{x \rightarrow a} f(x) = L$$

A conceptual understanding of limits of fundamental functions entails recognising that such limits involve two distinct processes (Cottrill et al., 1996). First, the domain process in which the independent variable x approaches the given number a . Second, the range process in which $f(x)$ approaches L . Moreover, students must also be able to coordinate these two processes, that is, to understand that the function $f(x)$ operates on the process of x approaching a , thereby generating the corresponding process of $f(x)$ approaching L . According to Cottrill et al. (1996), students who successfully coordinate these two processes tend to develop a coherent mathematical scheme for understanding the limit concept in relation to fundamental functions¹⁰.

Mathematical symbols often represent both a mathematical process and a corresponding mathematical concept. For example, $\lim_{x \rightarrow a} f(x) = L$ represents two dynamic processes in which $f(x)$ approaches L as x approaches a . Simultaneously, it also denotes the concept of a mathematical limit. Viewing a mathematical concept as a

¹⁰ Cottrill et al. (1996) further elaborates on a more advanced mathematical scheme required to fully comprehend the formal definition of limits. However, as the formal definition lies beyond the scope of this thesis, the development of such scheme will not be addressed.

process is, according to Gray and Tall (1994), a natural first step when introduced to a new concept. But to develop deeper mathematical understanding, students must learn to interpret mathematical symbols (such as $\lim_{x \rightarrow a} f(x) = L$) both as representations of processes (tending to a limit) and of mathematical concepts (of limit). Being able to comprehend a limit both as a process of tending to the limit and as a concept means that students have developed a *proceptual* view of limits. The notion of *procept* is thus regarded as “the amalgam of three components: a process that produces a mathematical object, and a symbol that represents either the process or the object” (Gray & Tall, 1994, p. 121).

4.3.1 Using programming to develop a proceptual view of limits

The way technology is integrated into the teaching of limits significantly shapes students’ understanding of the concept (Kumsa Beyene, 2023). Cottrill et al. (1996) highlight that programming can support students in visualising domain and range processes, thereby facilitating the coordination of these processes and contributing to the development of a mathematical limit scheme. Li and Tall (1993) also emphasise how programming could serve as a tool for numerically determine limits (specifically of sequences). However, they note that the use of programming did not foster an in-depth understanding of the concept, nor did it support students’ conceptual development of the ε - δ definition of limits. Instead, students paid more attention to the syntax of the programming language and the output of their program illustrating the stabilisation of a sequence. Monaghan et al. (1994) claim that using programming to determine limit values may primarily foster a one-sided view of limits as dynamical processes, which potentially can hinder an in-depth understanding of the limit concept and the development of a proceptual view. Cottrill et al. (1996), on the other hand, argue that the development of such process view should be regarded as a natural and important first step during the development of a coherent mathematical limit scheme and a proceptual view of limits. They further suggest that this development can be effectively supported through the use of programming.

[O]pposed to some researchers who believe that a dynamic conception may hinder progress toward the development of a formal understanding of the

limit concept, we believe that the difficulty in moving to a more formal conception of limit is at least partially a result of insufficient development of a strong dynamic conception. (Cottrill et al., 1996, p. 190)

4.3.2 Obstacles to achieving a conceptual understanding of limits

Cornu (1991) highlights how students' conceptual understanding of mathematical limits often is hindered by epistemological obstacles. These obstacles are historically rooted, having evolved alongside the historical development of the limit concept. Epistemological obstacles are linked to the notions of the infinitely large and the infinitely small, to questions regarding the existence and attainability of limits, and to the challenge that limits cannot be directly determined through a generic algebraic or arithmetic procedure. Kumsa Beyene (2023) claims that these epistemological obstacles can lead to *cognitive obstacles* among upper-secondary students, which arise when students' existing knowledge come into conflict with new concepts. In his study of Swedish upper-secondary students' understanding of limits, Kumsa Beyene (2023) identified cognitive obstacles relating to whether two functions are equal or not, to the notion of infinity, and to the informal definition of limits. Furthermore, he observed that epistemological obstacles were reinforced and transformed into cognitive obstacles through teachers' imprecise or mathematically less rigorous use of language. Cornu (1991) also stresses that when students are introduced to the concept of limits, terms such as *approach* and *limit* already carry everyday meanings. These pre-existing interpretations often persist, even after formal mathematical definitions have been presented. Juter (2009) highlights how upper-secondary students rarely develop an in-depth understanding of limits. As a result, many university students enter higher education with vague or even incorrect conceptions of limits and struggle to connect their understanding to related mathematical concepts such as derivatives and integrals.

4.4 Summary

The upper-secondary students participating in the two studies presented in this thesis are regarded as novice programmers. Existing

research indicates that novices often exhibit similar characteristics and face challenges concerning the syntax and semantics of programming languages. According to Lee et al. (2011), the *Use-Modify-Create* progression model provides a structured approach to fostering students' programming skills and their computational thinking through three stages.

In the design study, students were tasked with employing programming as a means to address mathematical problems, necessitating the *creation* of original code. Research consistently indicates that such problem-solving activities are inherently complex and that developing proficiency in mathematical problem solving is a gradual process requiring significant time and effort (Lester, 1996).

The case study examines students' use of programming as a tool for numerically calculating the limits of functions by providing a pre-designed code structure and code examples that students could *use* and *modify*. Prior research suggests that determining limits through programming can help students view limits as dynamic processes (Cottrill et al., 1996), that is, as values approaching a limit. However, developing a *proceptual* view of limits requires students not only to perceive limits as processes, but also to encapsulate these processes into a concept (Gray & Tall, 1994). Viewing limits as a *procept* is thus associated with a deeper mathematical understanding.

5 Methodology

Both studies included in this thesis aim to investigate the intertwined relationship between upper-secondary students' use of programming for mathematical purposes and their mathematical understanding. Furthermore, they aim to investigate how teachers' design of learning activities can facilitate the integration of programming into mathematics education. This chapter outlines the methodological approaches that guided the research project. The chapter begins by outlining the overarching implementation of the two studies. Subsequently, key aspects of educational design research, a methodological approach operationalised in the first study of this thesis, are presented, including an overview of the three phases of design-based research. This is followed by a description of the characteristics of case study research and the rationale for adopting this approach in the second study. Each methodological section provides a description of the participants along with the procedures for data generation and analysis relevant to the respective study. Finally, the chapter concludes with a discussion of the trustworthiness of the studies and the associated ethical considerations.

5.1 An overview of the two studies

The initial design study involved developing a single lesson activity that focused on using programming as a mathematical problem-solving tool. The design study comprised two design cycles, during which the activity was implemented in two classes of Swedish upper-secondary school students in 2019 and 2020. During both cycles, the researcher was responsible for designing the activity and for conducting the teaching experiment. According to the characteristics of educational design research, an initial local instruction theory was tested during the first design cycle. The design of the activity involved the development of tasks (Kieran et al., 2015) as well as the design of an instrumental orchestration (Drijvers et al., 2009; Trouche, 2005b). The design also took into consideration didactical variables (Ruthven et al., 2009) related to the design and students' learning of mathematics. The outcome of the data analysis of the first design cycle, formed the basis of a revised local instruction theory, which, in turn, guided the revision of the design enacted during the second design cycle. Due to the ongoing

COVID-19 pandemic, the enactment of the designed activity during the second cycle was held online. The analysis of the students' instrumental genesis across the two design cycles served as the foundation for developing a local instruction theory and design principles for incorporating programming as a problem-solving tool in upper-secondary mathematics education.

The second study, conducted in 2022, also examined students' instrumental genesis, focusing on how students appropriated and adapted a programming tool to support their understanding of various mathematical concepts. In addition, the study investigated how the craft knowledge of the participating students' teacher was adapted when programming was incorporated into school mathematics. Unlike the first study, the design and orchestration of the learning activities were carried out by the students' mathematics teacher. The second study could thus be described as a naturalistic case study (Yin, 2014) as it explores "a contemporary phenomenon (the 'case') in depth and within its real-world context" (p. 16) and moreover traces the development of issues and circumstances as they naturally arise (Abma & Stake, 2014). The case consisted of the mathematics teacher and his class of Swedish upper-secondary school students taking their third course in mathematics. Observing students' use of programming across an entire course, with particular attention to three lesson activities, enabled the researcher to analyse their instrumental genesis in greater depth than in the initial study, which entailed one single lesson activity.

5.2 The design study

The first of the two studies that form the basis of this thesis, presented in Papers 1, 2, and 3, utilised *design-based research* as its methodological approach. This design study aimed to develop design principles for integrating programming as a problem-solving tool in mathematics education by developing and testing a specific lesson activity. This section presents the foundation of educational design research¹¹ within the

¹¹ As noted by Bakker (2018), the notion of design research exists in other research fields, such as human-computer interaction, industrial engineering, and architecture. However, for the sake of readability, the term educational design research will hereafter be referred to simply as design research.

context of the design study presented in this thesis, emphasising its defining characteristics and portraying its three phases.

The idea of developing and testing pedagogical design goes back in time (Prediger et al., 2015) and has been termed as, for example, educational design, design experiments, and developmental research. The notion of design-based research (or simply design research) originates from the work of Collins (1992). Prediger et al. (2015) argue that design research should intertwine instructional design and educational research and Gravemeijer (2015) claims that educational design research should aim at developing theories that serve as reference frameworks for educators. These *local instruction theories* should consist of theories about the process of learning a given mathematical content alongside means and resources to scaffold the learning. This is also stressed by Bakker (2018) who argues that design research should generate “knowledge about which actions under what circumstances will lead to which kind of intended consequences” (p. 47).

Being *theory generative* is also one of the five characteristics of design research according to Cobb et al. (2003). Furthermore, design research should be *interventionistic* and test new educational design rather than analysing existing learning environments. Cobb et al. (2003) argue that the initial educational design should rest on clear conjectures concerning students’ learning trajectories and the learning outcomes. These conjectures should subsequently be compared to the students’ actual learning trajectories during a retrospective reflection, guiding the development of a local instruction theory (Prediger et al., 2015). Design research is thus both *prospective* and *reflective*. Based on the analysis and comparison between the hypothetical learning trajectory and the actual learning trajectory, a new and revised design forms which can be tested again. Consequently, a central characteristic of design research is its *iterative* and cyclic approach. Since design research is conducted in classrooms it accentuates *ecological validity and practice-orientation* (Prediger et al., 2015). During the design, the researcher must consider the complexity within classrooms and Prediger et al. (2015) argue that “care has to be taken that research reports describe those conditions, and the way they may have influenced the learning process” (p. 879). Cobb et al. (2003) claim that local instruction theories are

humble “not merely in the sense that they are concerned with domain-specific learning processes, but also because they are accountable to the activity of design” (p. 10).

Design research has been regarded as a sound methodological approach when investigating how the use of digital tools could foster students’ understanding of mathematics and develop a productive instrumental genesis (e.g., Drijvers, 2003; Fahlgren, 2017). Hoyles and Noss (2015) also argue that design research can serve as a valuable approach for exploring how educational tools and students’ knowledge co-evolve over time.

5.2.1 The three phases of design research

The characteristics of design research affect how this methodological approach is operationalised. Design research projects commonly consist of three phases: the preliminary design, the teaching experiment, and the retrospective analysis. In this subsection, these phases are examined in the context of the design study presented in this thesis.

The preliminary design

The initial step of the *preliminary design* is to define clear mathematical learning goals of the design study. Such goals may, but need not, be related to educational goals (Gravemeijer & Cobb, 2006). In the design study presented in this thesis, the overarching learning goal was for the students to develop a productive instrumental genesis in relation to the use of programming as a problem-solving tool.

Furthermore, an initial *local instruction theory* was elaborated based on the learning goal, existing research related to the topics (e.g., mathematical problem solving and programming), prior teaching experiences, and additional curriculum resources. The local instruction theory was intended to serve as a conjecture about the possible learning processes, specifically, students’ instrumental genesis when employing programming during a mathematical activity (Gravemeijer & Cobb, 2006). It also functioned as a conjecture about the means for

supporting these processes, particularly in relation to the design and instrumental orchestration of the learning activity.

Part of the preliminary design was also the development of a *hypothetical learning trajectory* (HLT) which according to Simon (1995) encompasses “consideration of the learning goal, the learning activities, and the thinking and learning in which the students might engage” (p. 133). The HLT is consequently informed by the initial local instruction theory (Gravemeijer, 2015). According to Prediger et al. (2015), a HLT may encompass a single activity or a sequence of activities distributed over an extended period. It is important to highlight that a HLT should not be regarded as a narrow path that all students should take, but rather as a broad learning route that students can follow at different speeds (Drijvers, 2003). A key component of the HLT in the design study was the description of an *anticipated instrumented action scheme*, as presented in Paper 2. The scheme was intended to be developed by the students during the teaching experiment. It was constructed based on Vergnaud’s (1998a) scheme components (outlined in subsection 3.2.2) and served to exemplify students’ HLT in solving the mathematical problems through programming, specifically by performing an exhaustive trial.

The initial local instruction theory and the hypothetical learning trajectory form the foundation for the design principles which will guide the initial design of the learning activity (Simon, 1995). As stated by van den Akker (1999), design principles “cannot guarantee success, but they are intended to select and apply the most appropriate [...] knowledge for specific design and development tasks” (p. 9). In this design study, the principles incorporated aspects of instrumental orchestration, which is defined as the “intentional and systematic organisation and use of the various artefacts available in a[...] learning environment by the teacher in a given mathematical task situation, in order to guide students’ instrumental genesis” (Drijvers et al., 2009, p. 1350). The principles also addressed *task design*, as well-designed tasks can foster a productive instrumental genesis (Drijvers & Gravemeijer, 2005) and shape how students perceive the mathematical topic (Watson & Ohtani, 2015). Furthermore, the design principles considered the scaffolding provided during the teaching experiments,

particularly the orchestration of how chosen students' work, referred to as sherpa-students, could support their peers' instrumental genesis.

The design of learning activities is often a complicated endeavour and Ruthven et al. (2009) highlight that no theoretical framework can cover all aspects of such design. They argue that aspects not covered by any existing framework could be referred to as *didactical variables* which function as “key levers to precipitate and manage the unfolding of the expected trajectory of learning” (p. 334). In the design study, didactical variables were initially identified in a prior analysis of students' knowledge and further variables were recognised through the actions of the students during the teaching experiments. The identified didactical variables pertained to task design, instrumental orchestration, and the scaffolding offered to the students.

The teaching experiment

The second phase of design research is the teaching experiment¹², or design experiment, where the hypothetical learning trajectory is enacted (Gravemeijer, 2015).

[T]he objective of the design experiment is not to try and demonstrate that the initial design or the initial local instruction theory works. The overall goal is not even to assess whether it works, although of course the researchers will necessarily do so. Instead the purpose of the design experiment is both to test and improve the conjectured local instruction theory that was developed in the preliminary phase, and to develop an understanding of how it works. (Gravemeijer & Cobb, 2006, p. 24)

During the enactment of the teaching experiment, the researchers should thus examine students' actions and their actual learning (Gravemeijer & Prediger, 2019). An essential component of this analysis involves comparing the outcomes of the instructional activity with the initial conjectures that informed its design. The aim of this analysis is mainly to improve the initial local instruction theory by investigating the validity of the guiding conjectures. In the design study presented in this thesis, such revisions concerned both aspects of the task design

¹² In this context, the term *experiment* does not pertain to traditional experimental designs involving treatment and control groups. Rather, it should be understood as an inquiry into the instructional design process (Drijvers, 2003).

and the instrumental orchestration (Drijvers & Gravemeijer, 2005). It also involved an analysis of the impact of the didactical variables (Ruthven et al., 2009) and lead to the identification of new didactical variables. Based on the result of the analysis, a revised design could be enacted in what Gravemeijer and Cobb (2006) refer to as the next micro design cycle¹³ in the design research methodology.

In the design study reported in this thesis, two complete micro cycles were conducted. During the intervention of both micro cycles, the researcher assumed the role of teacher and was responsible for conducting the learning activities. The role of the students' regular teachers during the activities will be outlined in subsection 5.2.2.

During the teaching experiments, data relevant for investigating the students' learning processes (Gravemeijer, 2015), based on the theoretical aim of the study (Gravemeijer & Cobb, 2006), were generated.

The retrospective analysis

The third phase of design research concerns the retrospective analysis, which seeks to contribute to the development of the local instruction theory (Gravemeijer, 2015) and the corresponding design principles (van den Akker, 1999). This implies an analysis of how the initial local instruction theory was refined during the micro cycles of the teaching experiment. Gravemeijer and Cobb (2006) emphasise that to establish trustworthiness in the retrospective analysis, it is crucial to document all aspects of the analysis process, including the formulation and evaluation of the initial conjectures. Cobb et al. (2003) argue that the “primary aim when conducting a retrospective analysis is to place the design experiment in a broader theoretical context, thereby framing it as a paradigm case of the more encompassing phenomena specified at the outset” (p. 13). This also underscores the interdependent relationship between theory generation and the advancement of knowledge in

¹³ In design research, micro cycles involve the refinement of specific design elements, whereas macro cycles encompass broader, long-term iterations of the overall instructional design (Drijvers, 2003). Since the design study described in this thesis comprised only two micro cycles within a single macro cycle, the term *design cycle* in this text will refer to micro cycles.

instructional design, which constitutes a foundational principle of design research.

In the design study reported in this thesis, the Instrumental Approach and Instrumental Orchestration served as analytical frameworks when examining students' instrumental genesis and the ways in which the instructional design influenced this process. To analyse students' instrumental genesis, data were coded according to the unit of analysis defined by the Instrumental Approach, namely schemes, instrumentalization, and instrumentation. As a researcher, I was responsible for conducting the retrospective analysis. While discussions with the students' regular teachers may have contributed to a more comprehensive understanding of the interventions, the teachers were not directly involved in the retrospective analysis itself.

5.2.2 Participants

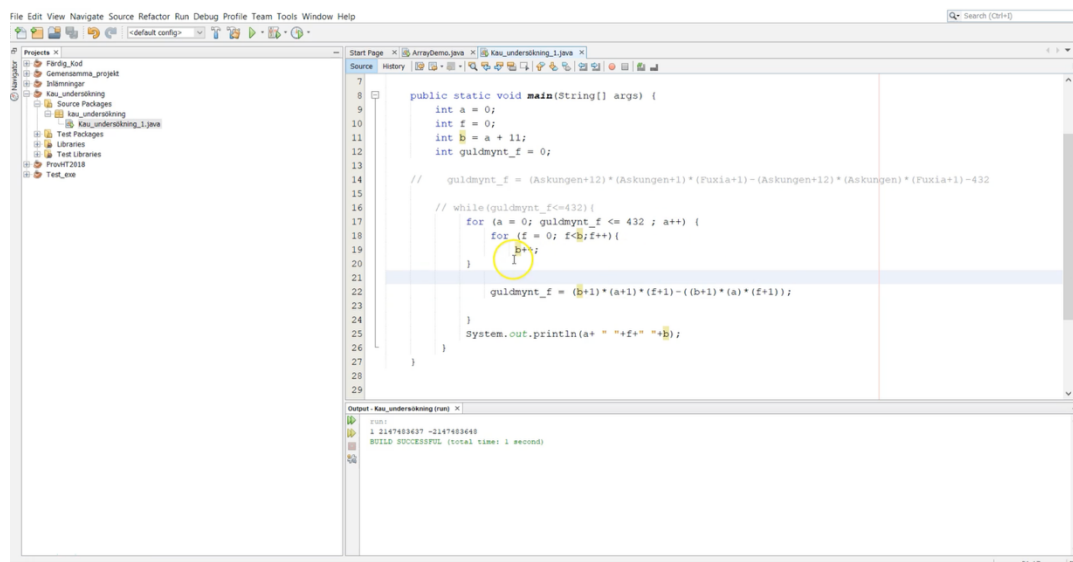
During the design study, all participating Swedish upper-secondary students (17–18 years old) were in their second year at the technology programme and were taking their third mandatory course in mathematics (Mathematics 3c). 27 students (7 girls and 20 boys) participated during the first micro cycle and 15 (3 girls and 12 boys) participated during the second cycle. All students came from the same upper-secondary school.

The rationale for choosing students from the technology programme was two-fold. First, the Swedish mathematics curriculum specified that students at the technology programme were expected to use programming as a problem-solving tool in all mathematics courses. Second, the participating students at the technology programme were taking a course in programming (Programming 1) which meant that they had been exposed to fundamental programming concepts and techniques. This affected the anticipated hypothetical learning trajectory (Simon, 1995) as it suggested that the students were less likely to encounter difficulties related to the programming syntax, allowing them to maintain a clear focus on utilising programming as a tool for mathematical problem solving. Nested loops were considered the most complex programming concept that students were expected to utilise. Existing research

highlights that novice programmers often struggle with such loops in various ways (Alzahrani et al., 2018; Ginat, 2004; Yarmish & Kopec, 2007). Consequently, the initial design took into account the need to support students in nesting loops by employing sherpa-students (Guin & Trouche, 1998). The anticipated difficulties with nested loops thus influenced both the initial local instruction theory and the corresponding design principles.

All participating students in the design study implemented their code in Java, using NetBeans 8.2 as their programming environment (exemplified in Figure 4).

Figure 4: Screenshot capturing students' Java code in NetBeans 8.2



During the teaching experiment of each micro cycle, the researcher assumed the role of teacher and was responsible for conducting the designed learning activity. In the first cycle, the students' mathematics teacher was reluctant to conduct the lesson due to his own inexperience in using programming in mathematics education. The mathematics teacher of the students who participated in the second micro cycle had good knowledge of programming. However, as the intervention took place online, the teacher was hesitant to deliver the lesson due to her limited experience of using the video-conferencing and communication platform Zoom for teaching. Although both interventions were led and conducted by the researcher (acting as teacher), the regular

mathematics teachers participated as observers. During the intervention in the first micro cycle, the students' teacher moved around the classroom, monitoring their work. The students' teacher had been instructed not to provide an extensive amount of support, as a key aspect of the design was to allow students to engage independently in a problem-solving process. In the second micro cycle, which was conducted online due to COVID-19, the teacher and the researcher (acting as teacher) were situated in the same physical room and monitored the same computer screens. As a result, the students' teacher rarely interacted directly with the students during the intervention.

It could be argued that there would have been advantages to having the students' teachers be the ones carrying out the teaching and that ethical considerations arise when the researcher is responsible for analysing his own didactical performance as part of the instrumental orchestration. However, Cobb et al. (2003) argue that it is essential for the research team, which may involve teachers, to possess "the expertise to accomplish the functions associated with developing an initial design, conducting the experiment, and carrying out a systematic retrospective analysis" (pp. 11–12). In the final section of this chapter, the ethical considerations outlined above will be discussed.

5.2.3 Data generation and analysis

During the design study, two types of data were generated for different purposes: screen and voice recordings, and students' programming code. The primary data when analysing students' instrumental genesis and the instrumental orchestration were the screen and voice recordings of students' work. This subsection provides a concise overview of the processes of data generation and analysis carried out within the design study.

Screen and voice recordings

During each cycle of the design study, the work of three student pairs was closely monitored, with their conversations and screens recorded using the screen capturing software Screencast-O-Matic. Monitoring how pairs employed programming to solve mathematical problems

enabled the researcher to gain insight into the rationale behind their decision-making. Such conversations can reveal both conceptual and technical aspects of students' instrumental genesis, aspects that may be harder to capture when observing individual students working alone with programming tools. Goos et al. (2003) further argue that interactions on the computer screen can promote "communication and sharing of knowledge in both private and public settings [as digital tools become] stimulus for, and partner in, face-to-face discussions when students work[...] together in groups" (p. 87). Tang et al. (2006) argue that the use of screen recording is less obtrusive than filming the participants using a video camera. It allows researchers to "unobtrusively collect detailed recordings focused on people's collaborations through their computer while doing their everyday work in their natural work settings" (p. 481).

Screen recordings also present certain limitations. First, the method may be regarded as intrusive, given that the researcher will monitor all on-screen activity during the analysis, including material that pertains to the participant's privacy. Another limitation is that screen recordings fail to capture the broader contextual environment in which interactions occur. Although factors related to the contextual environment could have been captured using video recordings in the classroom, this data generation method was not employed due to ethical considerations. These ethical aspects will be discussed in more detail later in this chapter.

In the first phase of the analysis of students' instrumental genesis, the researcher repeatedly viewed and listened to the screen and voice recordings while reading the corresponding transcripts. Using a thematic coding approach (Robson & McCartan, 2016), the schemes and their components (Vergnaud, 1998a) were described in detail using Microsoft Excel. At this stage, the phrasing of the scheme components was closely tied to the specific tasks and particular student pairs. The second phase of the analysis involved rephrasing and merging closely related components. This process led to the identification of more generic components, less dependent on specific tasks or students, and thus more broadly applicable. As a result of this iterative analysis, the identified scheme components were consolidated into an instrumented

action scheme, associated with the specific mathematical class of situation. To strengthen interrater reliability, parts of the data were also coded by additional researchers. Discrepancies in how the schemes were described were resolved through collaborative discussions.

Students' programming code

During the design study, all participating students were asked to submit their final programming code. This enabled the researcher to analyse the computational problem-solving strategies employed by the students and assess whether the students, in the end, were able to construct functional programs capable of solving the given mathematical problems. Since the submitted codes represented only the final product of the students' work and not the process through which it was developed, they could not be used as evidence of the students' instrumental genesis. Nevertheless, the analysis of the data provided valuable insights into how the students perceived the mathematical problems and attempted to solve them using programming. The outcomes of this analysis contributed a deeper understanding of the students' learning trajectories, which in turn informed modifications to the lesson design. A summary of the computational problem-solving strategies is presented in Appendix A in Paper 2.

5.2.4 Summary

This section has highlighted the potential of a design study to generate new knowledge both about students' learning of mathematics and about instructional design. The aim of the design study reported in this thesis was to examine how aspects of the instructional design could foster students' instrumental genesis during a programming activity. The study followed the three phases of design research and comprised two micro cycles in which the design was tested and revised.

Gravemeijer and Cobb (2006) emphasise that the micro cycles inherent in design research closely resemble what Simon (1995) describes as the mathematical teaching cycle. In this cycle, teachers design lessons based on anticipated learning trajectories, which are then enacted and iteratively revised in response to students' learning outcomes. This

parallel suggests that design research aligns with the practical approaches employed by many mathematics teachers. Consequently, the findings from this design study may hold significant potential value for both in-service teachers and teacher educators (Cobb et al., 2003).

In the design study presented in this thesis, the analysis of students' instrumental genesis is based on a single lesson activity. This limitation constrains the researcher to describe students' schemes as *proto-schemes*, since a scheme, according to Vergnaud (1998b), should be understood as an "invariant organization of behavior for a certain class of situations" (p. 229). A more longitudinal approach would have enabled a more comprehensive examination of students' instrumental genesis. However, such an approach would likely have been prohibitively time-consuming within the scope of the research project, particularly due to the iterative nature of design research (Cobb et al., 2003). Consequently, a case study was considered as a potential alternative methodological approach for the second study in this research project.

5.3 The case study

The main characteristic of case study research is its emphasis on a specific case, which Merriam (1998) describes as "a thing, a single entity, a unit around which there are boundaries" (p. 27). This definition suggests that the notion of case is broad and flexible, and could encompass a single person or a group, an educational policy, or a program.

In the case study of this thesis, as previously described, the case consists of a class of upper-secondary school students and their mathematics teacher. More specifically it focuses on how both the students and the teacher utilised programming within a specific course. Adopting a different, more naturalistic methodological approach in the second study enabled the researcher to investigate how programming could be integrated into an authentic upper secondary educational setting. It could be argued that such a case study, to a greater extent than a design study, accounts for how this integration is shaped by the inherent complexities, constraints, and affordances of a classroom environment, as well as by the intentions embedded in the mathematics curriculum. Furthermore, this approach enabled a more longitudinal investigation

of students' instrumental genesis compared to the design study, which involved a single lesson activity. It also enabled a focused examination of how students' instrumental genesis was influenced by the teacher's perception of programming in mathematics education and by his design of the learning activities. Focusing on the teacher's role, the adaptation of his craft knowledge, and his instrumental orchestration is particularly relevant in light of concerns raised by Swedish mathematics teachers regarding the integration of programming into school mathematics (Humble, 2022; Misfeldt et al., 2019; Vinnervik, 2022). The results from the case study are presented in Papers 4 and 5.

Dunbar and Blanchette (2001) argue that experimental studies (e.g., a design study) and naturalistic studies are complementary, as they provide research that is both theoretically grounded and applicable to a real-world context. Employing and integrating both approaches thus enables researchers to “overcome many of the limitations that each approach has when used alone” (p. 338). Abma and Stake (2014) emphasise that although a case is defined by certain boundaries, there are external factors, such as physical, social, and historical contexts, that lie beyond these boundaries but still influence the case. It may be argued that accounting for such external contextual factors poses a significant challenge when conducting more controlled design studies.

5.3.1 Characteristics of case study research

Although case study research is among the most widely employed research strategies in social science research, Yazan (2015) claims that seminal contributions to the field, shaped in part by the authors' epistemological orientations, adopt divergent positions regarding the design of case studies and the methods of gathering, analysing, and validating data. The methodological approach applied in the case study of this thesis has been influenced by the work of Merriam (1998). She highlights three main characteristics of a case study. First, it should focus on a specific situation (i.e., a case) that is clearly defined within established boundaries. Second, a case study should be descriptive, providing the reader with a rich and detailed account of the situation under investigation. Third, it should be heuristic, meaning that it aims to enhance understanding about the phenomenon being studied.

According to Merriam (1998), the design of a case study resembles the general structure of qualitative research. Prior to selecting the specific case, the researcher must conduct a thorough literature review, choose or construct an appropriate theoretical framework, and clearly define a research problem and corresponding research questions. During a case study, data may be generated through various methods, including interviews, observations, and document analysis (Merriam, 1998). In the case study reported in this thesis, data generation involved an interview with the teacher and direct observations of students' work in the real-world setting (Yin, 2014). This was achieved by recording students' screens and capturing the conversations between students working in pairs. Additionally, the researcher took field notes to provide a more comprehensive account of classroom activities and interactions. Unlike in the design study, the researcher in the case study avoided interactions with the students and therefore assumed the role of a marginal observer (Robson & McCartan, 2016).

The purpose of data analysis in case study research is to make sense of the data by “consolidating, reducing, and interpreting what people have said and what the researcher has seen and read—it is the process of making meaning” (Merriam, 1998, p. 178). Data generation and analysis can occur simultaneously, and the choice of analytical methods depends on the nature of the case and the type of data generated. In the case study reported in this thesis, a content analysis was performed using a thematic coding approach (Robson & McCartan, 2016). As in the design study, the data were coded based on the units of analysis defined by the Instrumental Approach, namely schemes, instrumentalization, and instrumentation. In addition, the teacher's instructional design was analysed using the Structuring Features of Classroom Practice framework (Ruthven, 2009) complemented by selected aspects of the Instrumental Orchestration framework.

5.3.2 Participants

As previously outlined, the naturalistic case study sought to explore how programming shapes students' conceptual understanding of mathematical ideas, particularly through the lens of instrumental

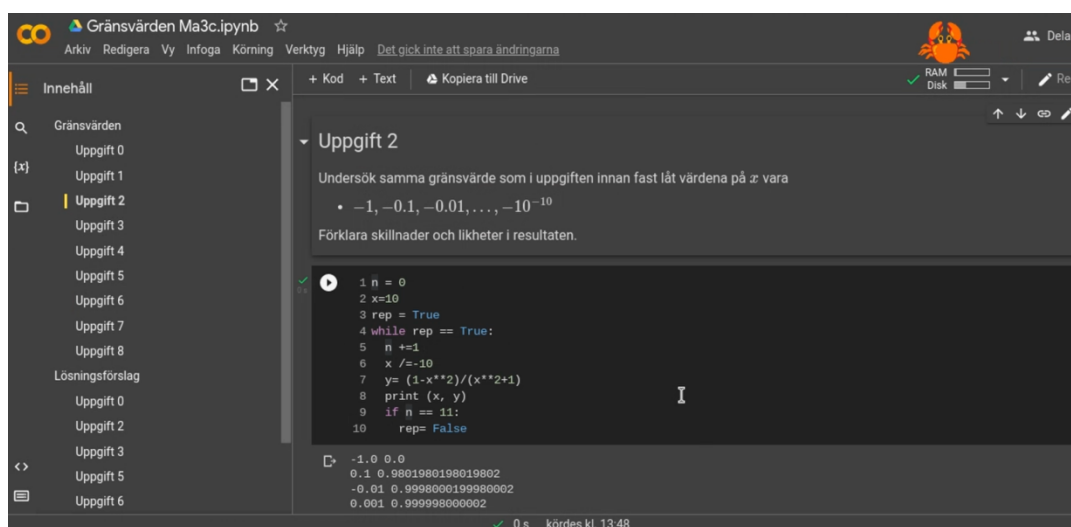
genesis. In contrast to the design study, the selection of students in this case study was influenced by the way their teacher had chosen to utilise programming in his teaching of mathematics. The participating mathematics teacher was purposefully selected due to his deliberate and reflective didactical approach to incorporating programming within upper-secondary mathematics education. As described earlier, given the concerns expressed by teachers regarding the integration of programming into the mathematics curriculum (Humble, 2022; Misfeldt et al., 2019; Vinnervik, 2022), it is particularly relevant to investigate how Swedish mathematics teachers have incorporated programming into their teaching.

The teacher in the case study was also teaching programming. His proficiency in programming meant that he had played a prominent role at his school when designing for the incorporation of programming into their mathematics courses. The teacher regarded programming as a powerful mathematical tool, offering students the possibility to conduct repeated and time-consuming calculations. He argued that programming can support students in developing a deeper understanding of numerical methods for solving mathematical tasks. Furthermore, he claimed that programming could enhance their conceptual understanding of mathematics by engaging them in structured step-by-step instructional processes. His design and orchestration involved using programming as tool for solving tasks numerically and included the development of activities, including tasks, where programming was utilised for exploring different mathematical concepts. The teacher's decision to utilise programming for solving tasks numerically was influenced by the 2021 revision of the Swedish mathematics curriculum. Consequently, programming was intended not only for problem solving (as in the design study) but also as a tool to support the application of numerical methods and data processing. An integral part of the teacher's didactical design involved decisions regarding the programming concepts and techniques that the students were expected to master, as well as strategies for supporting their technical handling of the programming environment. This led to the implementation of a pre-defined code structure, developed by the teacher. The purpose of implementing this structure was to provide students with a consistent template applicable across all tasks and reduce the occurrence of

syntactic errors. The intention behind this approach was to help maintain a clear focus on the mathematical objectives when engaging with programming. The code structure involved the design of a WHILE loop controlled by a Boolean variable. At the end of the loop's body, an IF statement was included, specifying a condition that must be met for the loop to terminate.

The participating students were in their second year at the science programme and were enrolled in their third mathematics course (Mathematics 3c). The criterion for selecting the class of participating students, besides from being taught by the participating teacher, was that the students were taking either the third-level mathematics course or a more advanced course. This criterion aligns with the revisions to the Swedish mathematics curriculum implemented in 2021 (The Swedish National Agency for Education, 2021), which stipulated that students in the science and technology programmes were not expected to engage with programming independently until the third course. Unlike the students in the design study, the students in the case study had not taken a course in programming. However, the school had decided to add about ten extra hours to their prior course in mathematics (Mathematics 2c) to ensure that the students possessed a sufficient understanding of how to utilise fundamental programming concepts and techniques in a mathematical context, including the use of the pre-designed code structure.

Figure 5: Screenshot capturing students' Python code in Google Colab



The screenshot shows a Google Colab interface with a Python notebook titled 'Gränsvärden Ma3c.ipynb'. The notebook content includes a task description and a Python code cell. The task asks to investigate the same limit value as in a previous task, but with fixed values for x ranging from -1 to -10^{-10} . The code defines a function to calculate the limit value for a given x and n . The output shows the results for $x = -1, -0.1, -0.01, \dots, -10^{-10}$.

```
1 n = 0
2 x=-10
3 rep = True
4 while rep == True:
5     n +=1
6     x /=-10
7     y = (1-x**2)/(x**2+1)
8     print (x, y)
9     if n == 11:
10        rep= False
```

```
-1.0 0.0
0.1 0.9801980198019802
-0.01 0.9998000199980002
0.001 0.999980000002
```

All participating students in the design study wrote their code in Python, using Google Colab as their online programming environment (exemplified in Figure 5).

5.3.3 Data generation and analysis

During the case study, three types of data were generated for different purposes. Consistent with the design study, screen and voice recordings of students' work constituted the primary data source for analysing the students' instrumental genesis. When, in Paper 4, analysing the adaptation of a teacher's craft knowledge while incorporating programming into mathematics education, a semi-structured interview served as the primary source of data. Both analyses were further supported by field notes generated by the researcher during the lesson activities. This subsection provides a concise overview of the data generation and analysis procedures for the case study.

Screen and voice recordings

The recording of students' screens and voices during the case study mirrored the data generation procedure used in the design study. In each lesson of the case study, four pairs of students were monitored more closely using the built-in screen capturing software on their Chromebook computers.

During the analysis of students' instrumental genesis in the case study, their instrumented action schemes were described using conceptual and technical elements related to students' instrumented techniques (described in 3.2.3). Although these schemes were analysed using a partly different analytical framework compared to the design study, the analysis was carried out in the same iterative manner as described for the design study (see subsection 5.2.3).

Teacher interview

As described earlier, the teacher in the case study made several conscious choices relating to the design of the activities involving the use

of programming. To capture the teacher's views of mathematics and the rationale for his design, a semi-structured interview was conducted after the course had been completed. Kvale and Brinkmann (2009) argue that qualitative interviews allow members of a specific social context (e.g., teachers) to debate "the reasons they have for their beliefs and actions" (p. 12).

The semi-structured interview was based on an interview guide (see Appendix) developed by the researcher. The guide was structured around specific themes (Bryman, 2016), including the teacher's perception of programming in mathematics education, his rationale for the way in which he incorporated programming into his teaching, and the five lessons in which programming was employed. The interview was conducted after the course had concluded, allowing the researcher to develop a comprehensive understanding of the social context and the implementation of the programming activities. According to Kvale and Brinkmann (2009), possessing contextual knowledge is essential for conducting meaningful and informed interviews. The contextual knowledge enabled the researcher to incorporate follow-up (secondary) questions into the interview guide, thereby enhancing the depth and relevance of the data generated.

The audio-recorded interview was conducted online using the video conferencing and communication platform Zoom. Conducting an interview online may increase the risk for technical difficulties but may also increase flexibility when it comes to planning the interview (Bryman, 2016). The interview was conducted in March 2023 and lasted 49 minutes.

To examine the adaptation of the teacher's craft knowledge, the interview was thematically coded, using the five structuring features of classroom practice identified by Ruthven (2009), and further informed by the theory of Instrumental Orchestration (Drijvers et al., 2009; Guin & Trouche, 1998).

Field notes

During the case study, the researcher adopted the role of a marginal observer (Robson & McCartan, 2016). In addition to the screen and voice recordings, field notes were taken by the researcher during the lesson activities. The field notes were intended to capture both the teacher's whole-class presentations in which he introduced the students to the mathematical and computational content, and his interaction with the students during these presentations. Furthermore, field notes were taken by the researcher to provide a description of the classroom settings, including the physical location of the students in the classroom. Finally, the field notes contained observer comments made by the researcher. Merriam (1998) highlights these types of notes (i.e., verbal descriptions of the setting, participants, and activities; direct quotations; and observer's comment) as relevant and commonly occurring content in field notes. All field notes were transcribed, refined, and partially reformulated by the researcher following each observed lesson.

The field notes served as a valuable complement by capturing contextual details that were not evident from the screen and voice recordings alone. In Paper 4, when examining the adaptation of the teacher's craft knowledge, the field notes were analysed by the researcher to investigate the teacher's *activity format* (Ruthven, 2009). A content analysis of the whole-class presentations (Robson & McCartan, 2016), documented in the field notes, enabled the identifications of different orchestration types (Drijvers, Doorman, et al., 2010). Collectively, these orchestration types defined the teacher's activity format and demonstrated how the incorporation of programming into mathematics education led to a specific adaptation of the teacher's craft knowledge with respect to this structuring feature of classroom practice.

5.3.4 Summary

To summarise, the case study adopted a naturalistic approach to investigate upper-secondary school students' use of programming in an authentic educational setting. The case focused on a mathematics teacher and the students in his course, with particular emphasis on the teacher's design of the learning activities. This methodological

approach enabled a detailed analysis of students' instrumental genesis and the teacher's instructional design. Furthermore, it allowed for an examination of how the integration of programming into his teaching of mathematics influenced the teacher's craft knowledge, particularly in relation to specific structuring features of classroom practice.

The case study was conducted over the duration of an entire mathematics course, allowing the researcher to follow the teacher's and the students' use of programming over an extended period. In contrast to the design study, this made it possible to carry out a more longitudinal investigation of the development of students' instrumental genesis. Consequently, it afforded the researcher a more comprehensive understanding of how these students engaged with programming in school mathematics.

5.4 Trustworthiness

Both studies presented in this thesis have applied qualitative methods and have included the "gathering, representation, manipulation, and interpretation of data" (Schoenfeld, 2007, p. 82). The quality of empirical studies should, according to Schoenfeld (2007), be evaluated by the trustworthiness of these processes. Adler (2022) claims that trustworthiness in qualitative research is manifested by the degree of transparency. Being transparent entails being specific not only about how the research was carried out but also about the epistemological assumptions that underpin the studies.

The trustworthiness of this thesis will be discussed based on the five aspects of trustworthiness presented by Schoenfeld (2007): descriptive and explanatory power, prediction and falsification, rigour and specificity, replicability, and triangulation.

Having *descriptive power* implies that the research "focus on what is essential for the analysis, in a way that is clear and compelling" (Schoenfeld, 2007, p. 83). The descriptions of the phenomenon of interest should thus be based on the theoretical and analytical frameworks operationalised. Adler (2022) also acknowledges the importance of highlighting how the choices of theoretical frameworks affect the

analysis and its outcome. In this thesis, the Instrumental Approach serves as the overarching framework for analysing students' use of programming in school mathematics. Such investigations are thus based on an assumption that students' technical handling of programming shapes their perceptions of mathematical concepts and vice versa. In this thesis and its accompanying papers, I have aimed to be transparent about how such assumptions guide the research. It is also important to acknowledge that employing alternative theoretical frameworks, such as the theory of semiotic mediation (Bartolini Bussi & Mariotti, 2008), would have entailed a different unit of analysis and, consequently, a different research focus.

Explanatory power is described by Schoenfeld (2007) as “the degree to which a characterization of some phenomenon explains how and why the phenomenon functions the way it does” (p. 83). In both studies, aspects of the students' instrumental genesis have been explained in relation to the researcher's and the teacher's instructional design and instrumental orchestration (Drijvers et al., 2009). Although the Instrumental Approach emphasises the interrelated nature of students' technical engagement with programming and their mathematical understanding, various contextual factors within the classroom environment, such as motivational and affective dimensions (Schunk, 2020), influence students' work to varying extents. While the concept of instrumental genesis does not explicitly address such factors, the Instrumental Approach, in conjunction with the Instrumental Orchestration and the Structuring Features of Classroom Practice frameworks, still provided substantial exploratory power in both studies.

Although the notions of *prediction and falsification* are often associated with quantitative research in natural sciences, Schoenfeld (2007) emphasises that they can also be applied in qualitative educational research. The two studies in this thesis focus on two specific mathematical domains, problem solving and mathematical limits, which in turn determined the types of problems or tasks the students were expected to solve. Since both studies are small-scale qualitative investigations, no general predictions regarding upper-secondary students' use of programming in mathematics education will be made. However, the two studies also illustrate two different approaches for incorporating

programming into school mathematics, involving two distinct applications of programming: as a tool for problem solving and as a tool for numerical computation. The analysis of these more general approaches, not tied to any specific mathematical content, examines how the design of learning activities influences students' use of programming for mathematical purposes. The outcomes of such analyses possess what Schoenfeld (2007) refers to as *potential* generality, in that the findings are reasonably applicable under circumstances similar to those of the two studies. Schoenfeld (2007) argues that even small-scale studies can offer modest predictions, in line with the idea that “in similar circumstances, similar things happen” (p. 85). Such predictions, grounded in the explanatory power of the two studies, may still be regarded as important, as they enable other researchers to test these predictions and further develop the underlying theory. Therefore, the modest predictions made in this thesis should be viewed as a small contribution which, together with similar research, may yield a deeper understanding of how the use of programming can influence upper-secondary students' mathematical understanding. As in the natural sciences, predictions in educational research can also be subjected to falsification.

The intention has been to maintain *rigour* when planning and executing the two studies to increase the overall trustworthiness (Schoenfeld, 2007) and the transparency (Adler, 2022). This entails presenting the theoretical constructs, methodological frameworks, and analytical procedures in sufficient detail to enable replication by other researchers (Schoenfeld, 1992b). Prediger et al. (2015) argue that such *specificity* is particularly important in design studies, as the methods of this methodological approach have been met with criticism for not being well defined. In the interest of transparency and rigour (Adler, 2022; Hammer & Berland, 2014), the five papers of this thesis present as much data and analysis as possible. This enables readers to independently examine the data and compare their interpretations with those of the author(s) (Schoenfeld, 1992b). Furthermore, to enhance the interrater reliability (Bryman, 2016), part of data from both studies were independently analysed by additional researchers.

It is important to acknowledge that this thesis comprise two small-scale studies, conducted in a Swedish educational context and within actual classrooms, each with their own complex ecologies (Prediger et al., 2015). These factors make it challenging to replicate each study in its entirety. However, the claims of potential generality, based on modest predictions from the two studies, may highlight aspects that could nonetheless be replicable (Schoenfeld, 2007). Therefore, the results of this thesis can provide valuable evidence within their specific context, which may inform future research, particularly studies with larger and more representative samples. To further enhance *replicability*, the study context and methodology have been described in as much detail as possible (Schoenfeld, 2007).

To enhance the trustworthiness of the two studies, different methods for generating data were employed. This approach, outlined in the previous sections, enabled the researcher to conduct *data triangulation* (Schoenfeld, 2007) during the analysis, thereby confirming the findings through multiple data sources. The use of data triangulation increases the data quality (Robson & McCartan, 2016), as well as the internal validity and reliability of case studies (Merriam, 1998).

5.5 Ethical considerations

When conducting studies involving humans, ethical considerations need to be made in relation to the risks that the project may entail for the participants (Robson & McCartan, 2016). Considering such ethical aspects also enhance the quality of the research (Bryman, 2016).

All participants in the two studies were informed both orally (twice) and in writing regarding the aim of the respective study, its implementation, and the procedures involved in data generation. Participation in the studies was voluntary, and all participants gave their informed consent before any data were generated. As all students were over 15 years of age, no informed consent from parents was required (Swedish Research Council, 2024). The participants were also informed that they could stop their participation at any given time during the study without specifying the reason for such withdrawal. During the two studies, only a small portion of students decided not to participate and only one

student (in the case study) withdraw her participation during the ongoing study. The high rate of student participation indicates that the studies, including their data-generation methods, were not perceived as intrusive. Furthermore, it suggests that the information provided to the students fostered trust in the researcher's intentions and the overall purpose of the studies.

Ethical considerations were also made in relation to the data generation. During the pilot study, described in Paper 2, a video camera was used to record the activity of the participating students. Video recordings have the potential to capture students' gestures, such as pointing at the screen or on handwritten notes. Using video cameras to record the classroom would also have enabled observation of the teachers' practices during the interventions. However, the potential benefits were deemed insufficient to justify the intrusion into personal privacy. This decision can be seen as consistent with the recommendations of the Swedish Research Council (2024), which advise that video recordings should only be employed when equivalent results cannot be obtained through alternative data generation methods. Therefore, only audio recordings of the students' conversations and recordings of the students' screens were made. Tang et al. (2006) highlight that screen recordings are perceived by students as less obstructive than video recordings. To further safeguard the privacy of the participating students, the researcher ensured that none of the screen and voice recordings were stored on any cloud-based platform. After transferring the recordings from the students' computers to a USB stick, the researcher permanently deleted the video files from the students' devices.

During the teaching experiment of each micro cycle of the design study, the researcher assumed the role of teacher and was responsible for conducting the designed learning activity. This implies that I was responsible for analysing my own didactical performance based on the intended instrumental orchestration (Drijvers et al., 2009), which give rise to ethical considerations. It is important to note, as emphasised by Gravemeijer and Cobb (2006), that "the objective of the teaching experiment is not to try and demonstrate that the initial design or the initial local instruction theory works" (p. 24). Rather, its purpose is to test the local instruction theory and, based on the outcomes of the

teaching experiment, refine it and develop a deeper understanding of how it functions. This perspective guided the analysis of the teaching experiment, which was not intended as an evaluation of the design's quality. To enhance the transparency of the analysis (Adler, 2022; Hammer & Berland, 2014) of the teaching experiments, Paper 2 presents extensive empirical material with the intention of enabling the reader to evaluate the retrospective analysis.

Both studies were conducted in accordance with the ethical guidelines of the Swedish Research Council (2024) and were reviewed by the ethics committee of Karlstad University (references HS 2018/871 and HS 2022/710). As the studies did not involve the generation of special categories of personal data¹⁴, in accordance with General Data Protection Regulation (GDPR) (Regulation (EU), 2016/679), the ethics committee of Karlstad University concluded that approval from the Swedish Ethical Review Authority (2023) was not required. In alignment with the GDPR, all data underpinning this study (i.e., voice and screen recordings, students' codes, a teacher interview, and field notes) were subject to pseudonymisation. Identifiable information, such as names, class designations, and school affiliations, was replaced with coded identifiers to safeguard participant privacy.

All data and metadata were stored on a cloud storage location recommended by Karlstad University which also provides automated backup. Since the data is governed by privacy laws and the informed consent provided by the participants, data from the two studies have not been made publicly accessible.

¹⁴ Referred to as *sensitive personal data* in Sweden (Swedish Ethical Review Authority, 2023).

6 Findings

In this chapter, the main findings from of the five papers will be presented, both separately and in relation to each other. These findings align with the thesis's overarching aims *to investigate the intertwined relationship between upper-secondary students' use of programming for mathematical purposes and their mathematical understanding, as well as how teachers' design of learning activities can facilitate the integration of programming into mathematics education*. These aims are pursued through three central research questions:

1. What are the instrumental geneses of upper-secondary school students appropriating programming as an instrument for mathematical activity?
2. How can aspects of teachers' design of learning activities—in which students engage with programming as a mathematical tool—affect students' instrumental genesis?
3. How should analyses of students' instrumental genesis take account of the fact that the artefact—the programming environment—used during mathematical activities is not designed primarily as a mathematical tool?

The first section of the chapter presents findings on students' instrumental genesis as they engaged with programming as a tool in mathematics education. The second section examines how aspects of the learning activity design in the two studies influenced this process. The third section outlines how two analytical frameworks were operationalised to assess students' instrumental genesis. The chapter concludes with a section that summarise the key findings from the two studies.

6.1 Students' instrumental geneses when using programming for mathematical purposes

In Papers 2 and 5, upper-secondary students' instrumental geneses are assessed based on two different studies. In the design-based study, presented in Paper 2, the initial stage of students' instrumental genesis is analysed since the study merely involved a single lesson activity. The case study, presented in Paper 5, involved a sequence of lessons that

enabled a more in-depth analysis of the students' instrumental genesis over an extended period.

The role of programming also varied in the two studies. Whereas programming was used to facilitate mathematical problem solving in the design study, it served as a tool for numerically determining limit values (including derivatives) in the case study.

6.1.1 Students' instrumental genesis in Paper 2

One of the aims of Paper 2 was to investigate students' instrumental genesis when using a programming environment as a tool for mathematical problem solving. In the design-based study, presented in the paper, the participating students were asked to solve two mathematical problems related to each other. The problems involved determining the ages of three sisters based on given relationships among their ages.

Problem 1

When Cinderella was wandering in the woods, she met the good fairy. The fairy said to Cinderella:

“Say what you want, and I'll fulfil your wish”.

“Then I wish to always stay at the age of today”, Cinderella responded.

When Cinderella came home and told what had happened to her two older sisters, Begonia, 11 years older than Cinderella, became furious. She exclaimed:

"You know, your stepfather gives us the product of our three ages in gold coins to share each year."

Fuxia, the mid sister, carried on:

“That's right. With your selfish wish, we will together lose 432 gold coins the next year!”

How old is Cinderella and her two sisters?

Problem 2

Suppose you did not know that Fuxia is the mid-sister. What would the solution of problem 1 then look like?

The two problems were not related to any specific mathematical content in the ongoing course but required knowledge of basic arithmetic and algebra. It was anticipated that the students would solve the problems using an exhaustive trial by constructing a program which tested all possible combinations of ages using nested loops.

As described in the previous chapter, the participating students had basic knowledge of programming but little or no prior experience of using programming in mathematics. It was thus anticipated that the students had developed usage schemes related to the handling of the artefact (Rabardel, 2002). Such schemes could be related to the handling of inputs, variables, loops (including nested loops), selection statements, and outputs. However, due to the students' limited experiences of using programming in school mathematics, it was not expected that the students had developed any instrumented action schemes (Rabardel, 2002) related to the utilisation of programming as a tool for mathematical problem solving.

Students' instrumented action scheme

During the lesson activity in each design cycle, the students developed instrumented action schemes (see an example in Table 2), which were described using scheme components defined by Vergnaud (1998a). The concepts-in-action (CiA) within the scheme function as building blocks for the theorems-in-action (TiA), propositions that students regard as true. The identified rules of action (RoA) represent the generative aspects of the scheme. These components are further associated with distinct parts of the problem-solving process: the use of algebraic formulas (AF), the application of exhaustive trials (ET), and the implementation of loops and control of variables (LC).

Since schemes are “invariant organization of behavior for a certain class of situations” (Vergnaud, 1998b, p. 229), those developed by students during a single lesson will be treated as proto-schemes or schemes-in-progress. The scheme presented in Table 2 includes components both relating to the students' conceptual understanding of the mathematical content, and to the technical interaction with the programming environment.

Table 2: An instrumented action scheme developed by the students Ian and Jacob during the teaching experiment of the second micro cycle in the design study

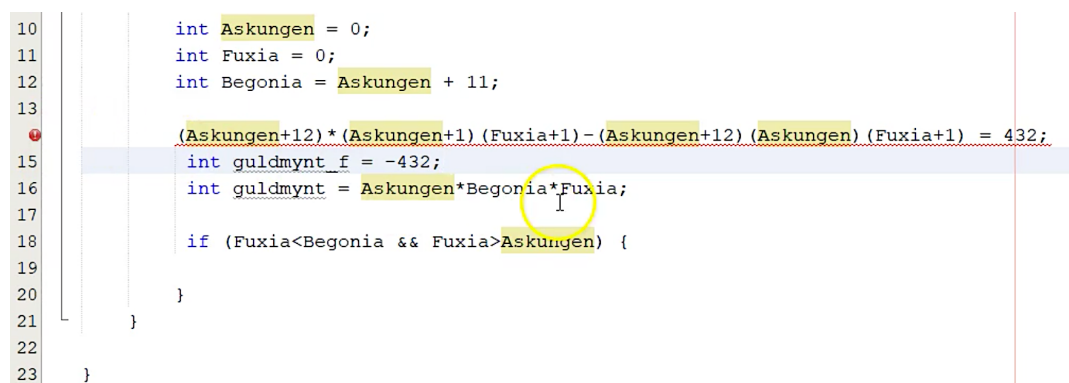
<i>Concepts-in-action (CiA)</i>	
CiA-AF1:	The idea of expressing a word problem situation in terms of (one or more) algebraic relationships between variables.
CiA-AF2:	The idea of manipulating a mathematical relationship to simplify it.
CiA-AF3:	The idea of manipulating an equation to produce an expression for an unknown variable.
CiA-ET1:	The idea of conducting an exhaustive trial.
CiA-ET2:	The idea of systematically combining variables.
CiA-ET3:	The idea of systematically varying one variable and deriving linked variables from relationships between them and it.
CiA-LC1:	The idea of establishing a loop relating to each of the variables in play.
CiA-LC2:	The idea of nesting loops (and statements within them) in order to achieve an appropriate sequence of variable-related actions.
CiA-LC3:	The idea of using conditions within loops and conditional operators in order to extract solutions within a given range during an exhaustive trial.
CiA-LC4:	The idea of establishing a loop relating to one of the variables in play and then manually altering a relationship involving the calculations of a linked variable.
<i>Theorems-in-action (TiA)</i>	
TiA-1:	Conducting an exhaustive trial is a means of solving a word problem situation involving algebraic relationships between variables.
TiA-2:	Systematically combining variables is a means for conducting an exhaustive trial.
TiA-3:	Establishing a loop for each variable in play and nesting these loops is a means for systematically combining variables.
TiA-4:	Systematically varying one variable and deriving values of linked variables from relationships between them and it is a means of conducting an exhaustive trial.
<i>Rules of action (RoA)</i>	
RoA-AF1:	Analyse the problem situation as one involving unknown quantities.
RoA-AF2:	Express the word problem situation in algebraic terms.
RoA-AF3:	(After following RoA-AF2 with apparent success) Simplify algebraic expressions.
RoA-AF4:	(After following RoA-AF2 with apparent success) Manipulate algebraic expressions towards a solution state.
RoA-ET1:	Formulate the problem situation as amenable to solution through exhaustive trial.
RoA-LC1:	Declare computational variables and (where given) assign variables values.
RoA-LC2:	Create an iteration and define conditions for the loop.
RoA-LC3:	Create nested loops in order to systematically combine variables.
RoA-LC4:	Make use of the conditional operator IF to (a) evaluate given conditions in order to (b) perform different actions based on the validity of the given conditions.
RoA-LC5:	Create an output which shows the values of variables.
RoA-LC6:	Coordinate necessary rules of action (as efficiently as possible) so as to construct a solution procedure.
RoA-LC7:	Make sure that the program takes account of all the information in the problem statement.

Instrumentalization

During the instrumentalization process, a subject (e.g., a student) adapts and modifies an artefact (e.g., a programming environment) to fit her/his (mathematical) needs (Trouche, 2004). In Paper 2, the instrumentalization involved adapting the programming environment to function effectively as a mathematical tool for problem solving. The findings suggest that such adaptations posed significant challenges for the students, indicating that the instrumentalization was less intuitive than anticipated by the researcher.

Many students, based on their inexperience of using programming for mathematical purposes, did not appreciate the possibilities (and constraints) of the programming environment (the artefact). Instead, many students, although asked to solve the problems using programming, attempted to solve the first problem analytically using pen and paper. Due to the lack of a pre-existing instrumented action scheme, many students thus turned to pre-existing mathematical schemes concerning how to solve a system of equations. A few students also tried to take advantage of another technical artefact, GeoGebra, to simplify algebraic expressions and to solve equations.

Figure 6: Screenshot from Emilia and Fredrik's use of NetBeans in the design study



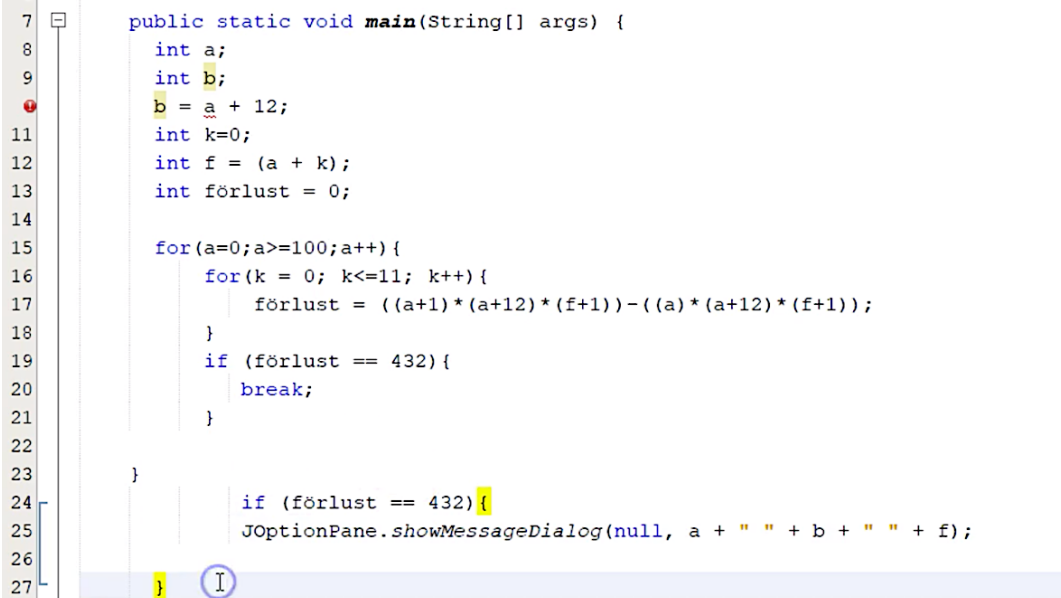
```
10     int Askungen = 0;
11     int Fuxia = 0;
12     int Begonia = Askungen + 11;
13
14     (Askungen+12)*(Askungen+1)(Fuxia+1)-(Askungen+12)(Askungen)(Fuxia+1) = 432;
15     int guldmynt_f = -432;
16     int guldmynt = Askungen*Begonia*Fuxia;
17
18     if (Fuxia<Begonia && Fuxia>Askungen) {
19
20     }
21 }
22
23 }
```

Students also encountered difficulties when attempting to perform exhaustive trials using programming. Particularly, they struggled with translating their mathematical algorithms into computational equivalents. Instead, students imposed mathematical meanings to concepts and methods utilised when programming. Many of these challenges are also documented in existing research, for example, students'

difficulties in distinguishing between the meanings of variables and the equals sign in mathematics compared to programming (e.g., Bråting & Kilhamn, 2021; du Boulay, 1989; Haspekian et al., 2023; Kohn, 2017). This is exemplified in the code snippet in Figure 6, where two students (Emilia and Fredrik) from the first cycle of the design study inserted a mathematical equation into the code at line 14, expecting the program to compute the solution.

Students also failed to acknowledge the sequential nature of steps in imperative programming (e.g., Ahmadzadeh et al., 2005; du Boulay, 1989) when trying to define mathematical relationships between variables in the code. Figure 7 presents code created by two students (Anne och Bill) from the first cycle of the design study. In a mathematically structured manner, the students define a relationship between variables b and a at line 10, which they perceive as valid throughout the entire code. This suggests that they believed the value of b , printed by the command at line 25, would consistently be 12 greater than the corresponding value of a printed by the same command. These challenges affected students' ability to instrumentalize the programming environment, as they were unable to adapt it to function effectively as a mathematical tool.

Figure 7: Screenshot from Anne and Bill's use of NetBeans in the design study



```
7 public static void main(String[] args) {
8     int a;
9     int b;
10    b = a + 12;
11    int k=0;
12    int f = (a + k);
13    int förlust = 0;
14
15    for(a=0;a>=100;a++){
16        for(k = 0; k<=11; k++){
17            förlust = ((a+1)*(a+12)*(f+1))-((a)*(a+12)*(f+1));
18        }
19        if (förlust == 432){
20            break;
21        }
22    }
23
24    if (förlust == 432){
25        JOptionPane.showMessageDialog(null, a + " " + b + " " + f);
26    }
27 }
```

Instrumentation

When students use a technical artefact for mathematical purposes, the utilisation of the artefact will affect students' perceptions of the mathematical content. This aspect of the instrumental genesis is referred to as instrumentation. The design of the learning activity presented in Paper 2, and their use of programming, exposed students to the idea of exhaustive trials. Although testing is highlighted as a heuristic problem-solving strategy by Pólya (1971), Swedish textbooks seldom encourage the use of such strategy. However, the use of programming empowers students to explore not just a limited set of cases, but the entire range of possible solutions. Therefore, it could be argued that the possibility of applying exhaustive trials is related to the students' instrumentation.

As part of this instrumentation, the students were required first to develop a mathematical algorithm and subsequently its computational equivalent. This process involved constructing an iterative procedure using nested loops, in which the values of the variables were systematically varied and combined to carry out the exhaustive trial. Consequently, the students needed to be precise when defining the relevant relationships, such as those concerning variable domains and those who defined the boundaries of the loops, not only between mathematical variables but also between their computational counterparts. The instant feedback provided by the programming environment also exposed students to relationships stated incorrectly. Although existing research highlights that novice programmers often struggle with debugging (e.g., Ettles et al., 2018; Putnam, 1987), Papert (1980) argues that debugging should be viewed as a learning activity through which students can derive valuable insights from their mistakes.

6.1.2 Students' instrumental genesis in Paper 5

Paper 5 presents findings from the case study, with particular focus on students' instrumental genesis as they employed programming to numerically determine limit values. The participating upper-secondary school students were taking their third course in mathematics and had been taught basic programming as part of their prior course. The students had also been trained in using a given code structure (regarded

as the artefact), which could be employed when solving all tasks during the classroom activities involving the use of programming. As part of the task sequence of each activity, the teacher also provided the students with pre-designed code examples which they could use and modify to complete the remaining tasks. Data were generated during three different lessons where students were expected to utilise the code structure to determine limit values numerically by printing values of the limit expression and the independent variable as the independent variable value approached a given number. The first lesson focused on introducing the limit concept, the second addressed the formal definition of the derivative, and the third explored the limit expression representing the natural logarithm. All tasks are presented in the appendix of Paper 5.

Students' instrumented action scheme

Of particular interest in Paper 5 is the instrumental genesis observed in the collaborative work of two students, Bella and David. During the three lessons (L1, L2, and L3), these students developed and utilised four instrumented action schemes, presented in Table 3. Each scheme was associated with an instrumented technique (Artigue, 2002), which is regarded as the visible manifestation of the scheme (Drijvers & Gravemeijer, 2005). The schemes comprise conceptual elements (CE), related to students' mathematical reasoning (Turgut & Drijvers, 2021), alongside technical elements (TE) relating to the technical handling of the artefact and guided by the conceptual elements. Table 3 illustrates how additional elements, associated with the mathematical topics, were assimilated into the schemes during the second and third lesson.

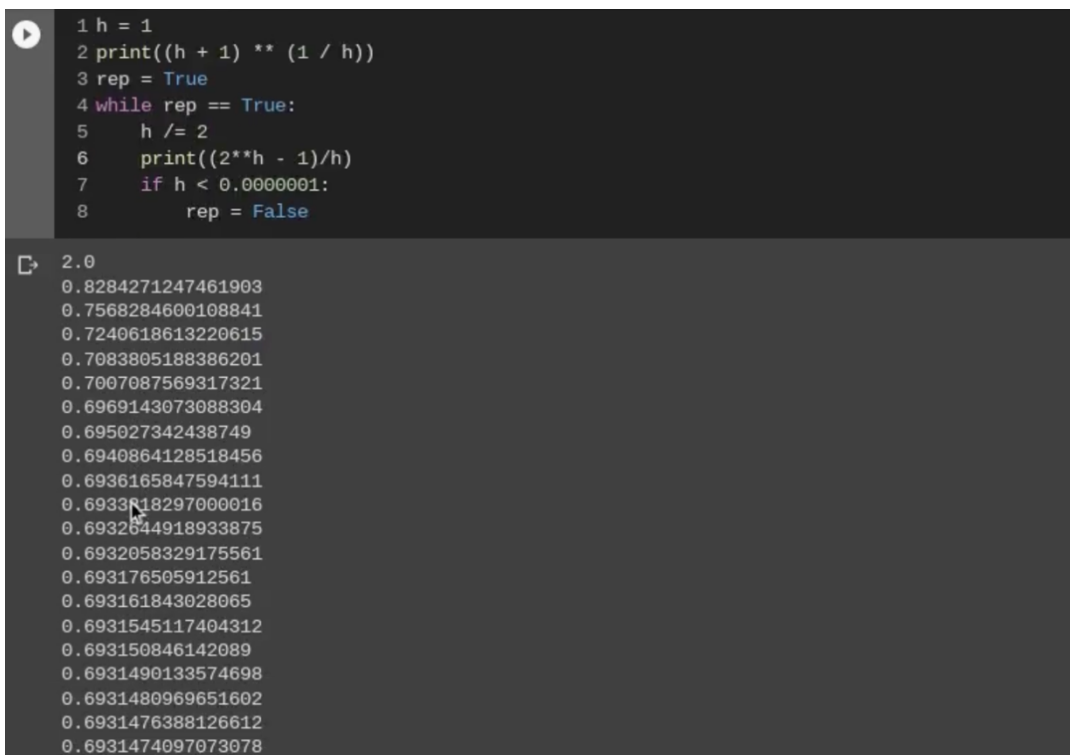
Table 3: Four instrumented action schemes developed by the students Bella and David during the case study

Instrumented action schemes (IAS)	Conceptual elements (CE)	Technical elements (TE)
IAS1: Modifying the code by adjusting the limit expression [L1, L2, L3]	CE1: The value of the algebraic expression should be calculated for each value of the independent variable (e.g., x or h) [L1, L2, L3]	TE1: Copy and paste code from previous tasks/examples [L1, L2, L3]
	CE2: The difference quotient stated in the code can be used to identify the given function [L2]	TE2: Assign computational variables values [L1, L2, L3]
IAS2: Modifying the code by adjusting the value that the independent variable should approach [L1, L2, L3]	CE3: The independent variable approaches 0 if its present value is divided by a number greater than 1 [L1, L2, L3]	TE1: Copy and paste code from previous tasks/examples [L1, L2, L3] TE2: Assign computational variables values [L1, L2, L3]
	CE4: The independent variable will approach 1 if its present value (greater than 1) is powered by a number greater than 0 and less than 1 [L2]	TE3: Use the command for modifying the value of a variable by dividing the current variable value by a given number [L1, L2, L3]
	CE5: The difference quotient stated in the code can be used to identify the x value of the point a [L2]	TE4: Use the command for modifying the value of a variable by taking the current variable value powered by a value [L2]
IAS3: Modifying the code by adjusting the condition that terminates the loop [L1, L2, L3]	CE6: By implementing an iterative process, it is possible to analyse the convergence of the function value as the independent variable approaches a given point [L1, L2, L3]	TE1: Copy and paste code from previous tasks/examples [L1, L2, L3] TE2: Assign computational variables values [L1, L2, L3]
	CE7: Increasing the number of iterations may provide a better opportunity to evaluate the numerical value of the limit [L1, L2]	TE5: Define conditions for an IF-statement used to regulate the number of iterations [L1, L2, L3]
	CE8: Relevant variable values should be printed [L1, L2, L3]	TE6: Use the command for modifying the value of a variable by adding a given number to the current variable value [L1]
IAS4: Generating an output by printing variable values [L1, L2, L3]	CE9: Analysing the output is a means to determine the limit value [L1, L2, L3]	TE1: Copy and paste code from previous tasks/examples [L1, L2, L3]
	CE10: The limit equals the value which the expression seems to approach [L1, L2, L3]	TE7: Create an output which prints values of chosen variables and/or calculations [L1, L2, L3]
	CE11: If the limit exists, the left limit should equal the right limit [L1]	TE8: Create an output which prints text [L1]
	CE12: If the function value continues to increase, without approaching a given number, it approaches infinity [L1]	

Instrumentalization

The code structure utilised by the students during the three lessons was regarded as the artefact in this study. When using programming to determine limits, the students were required to adjust and modify the code structure, an activity classified as part of students' instrumentalization. The development of the students' instrumented action schemes is thus closely related to their instrumentalization.

Figure 8: Screenshot of Bella and David's code for determining the limit $\lim_{h \rightarrow 0} \frac{2^h - 1}{h}$ in the case study



```
1 h = 1
2 print((h + 1) ** (1 / h))
3 rep = True
4 while rep == True:
5     h /= 2
6     print((2**h - 1)/h)
7     if h < 0.0000001:
8         rep = False
```

2.0
0.8284271247461903
0.7568284600108841
0.7240618613220615
0.7083805188386201
0.7007087569317321
0.6969143073088304
0.695027342438749
0.6940864128518456
0.6936165847594111
0.6933818297000016
0.6932644918933875
0.6932058329175561
0.693176505912561
0.693161843028065
0.6931545117404312
0.693150846142089
0.6931490133574698
0.6931480969651602
0.6931476388126612
0.6931474097073078

The findings reveal that the use of a code structure, together with code examples that students could use and modify, provided clear boundaries for the students' use of programming. Figure 8 illustrates how the two students Bella and David used the code structure to numerically approximate the limit representing $\ln 2$, based on the printed values of the expression $(2^h - 1)/h$ as the independent variable h approaches zero. The code structure consists of the WHILE loop controlled by the Boolean variable *rep*. At the end of the loop's body, an IF statement is included, specifying a condition that must be met for the loop to terminate. This scaffolding meant that the students seldom experienced any

technical difficulties related to programming and proficiently managed to generate an adequate output. It could be argued that students' work involved the first two stages of the progression model *Use-Modify-Create* presented by Lee et al. (2011). Progressing from the stage when you *Use* and interpret pre-designed code to the stage where you can *Create* your own code implies a shift from being a consumer of someone else's code to being a producer of your own.

However, during tasks which required conceptually challenging modifications, the students often struggled. For example, whereas they were able to take advantage of the provided code examples to construct an iteration where the independent variable approached zero, they were unable to assign the independent variable new values so that it would approach -1. It could thus be argued that the transition from the *Use* stage to the *Modify* stage, especially when it involved assimilating additional conceptual elements into students' instrumental action schemes, posed challenges for the students.

Instrumentation

Of special interest in Paper 5, relating to students' instrumentation, was how the specific way of using programming to determine limit values affected students' perceptions of the mathematical concept. When developing a conceptual understanding of limits, Cottrill et al. (1996) emphasise the significance of recognising that the function $f(x)$ operates on the process where x approaches a , thereby generating the corresponding process of $f(x)$ approaching L . The findings highlight how the two students, when using the code structure to determine limit values, foremost focused on the range process of $f(x)$ approaching L , and paid less attention to the domain process of x approaching a . This is illustrated in Figure 8, where the students print only the values of the limit expression, and not the corresponding values of the independent variable. It may therefore be claimed that, at this stage, the students struggled to coordinate the two dynamic processes, indicating that they had not yet constructed a coherent *mathematical* scheme for fully understanding the limit concept.

The findings also indicate that the specific utilisation of programming promoted a one-sided dynamic view of limits as *processes* (Monaghan, 2016b) where the mathematical expression approaches a specific value. Although Gray and Tall (1994) claim that initially viewing mathematical notions (e.g., limits) as processes is natural for students, they argue that it becomes equally important to also be able to consider limits as mathematical *concepts*. Viewing a limit both as a process of tending towards a value and as a conceptual object suggests that students have developed a *proceptual* understanding of limits (Gray & Tall, 1994). Conceiving of limits as a *procept* therefore implies that students are able to encapsulate the relevant processes into a coherent concept.

Overall, the two students were most often able to determine the limit values as the independent variable approached zero, using programming. This suggests the development of four stable instrumented action schemes for numerically determining limits, consistent with the notion of an “invariant organization of behavior for a certain class of situations” (Vergnaud, 1998b, p. 229). However, although the students successfully generated outputs and used them to determine the limit value, they occasionally struggled to interpret the conceptual meaning of the limit. This difficulty became particularly evident during the third lesson, when the students were expected to calculate and interpret limits representing natural logarithms.

6.1.3 The relationship between students’ instrumental geneses in Paper 2 and Paper 5

Both in Paper 2 and in Paper 5, upper-secondary school students’ instrumental genesis has been analysed. However, whereas programming was employed as a tool for mathematical problem solving in Paper 2, it was used for solving mathematical tasks numerically in Paper 5. Both uses align with applications of programming emphasised in the Swedish upper-secondary mathematics curriculum.

The studies of students’ instrumental geneses involved analysing the instrumented action schemes they developed. Although two different analytical frameworks were used to describe such schemes, the number of components or elements constituting the schemes in Table 2 and

Table 3 illustrates the complexity of using programming to solve mathematical tasks (Buteau et al., 2020). It should be emphasised that these tables do not represent a fixed structure of scheme components or elements associated with the two different objectives. Rather, they should be regarded as examples that, as Drijvers and Gravemeijer (2005) argue, “can help the observer to analyze the complexity of the students’ work in the technological environment” (p. 192).

The analyses of the schemes also highlight the intertwined relationship between technical and conceptual aspects associated with the instrumented action schemes. It is important to emphasise that the design study described in Paper 2 involved only a single lesson activity. As a result, students developed what may be referred to as proto-schemes or schemes-in-progress. These terms are used because a scheme is typically understood as “a more or less stable way to deal with specific situations or tasks, guided by developing knowledge” (Drijvers et al., 2013, p. 27). Given this definition, it is unlikely that a stable scheme could emerge from a single lesson activity involving only two mathematical problems. In contrast, Paper 5 analysed students’ instrumental genesis across three lessons activities conducted over a longer period. The students’ consistent and proficient use of programming to determine limits numerically suggests the development of four stable instrumented action schemes. However, the findings indicate that when students were required to adjust these schemes to solve new types of tasks, they sometimes encountered difficulties, particularly when the adjustments involved assimilating new conceptual elements.

Using programming as a tool during a mathematical problem-solving activity implied that the students had to create their own programs based on the open-ended mathematical problems. During the problem-solving process, the students were expected to create their own mathematical and computational algorithms which demanded repeated support from the researcher (taking the role of teacher) and from peer students. The way that programming was used as a tool to numerically determine limit values in the case study involved scaffolding in which the code structure formed a key part, together with code examples. This implied that students were expected to use and modify pre-existing code when solving the tasks. Together, the two studies exemplify all

phases in the progression model *Use-Modify-Create* presented by Lee et al. (2011). But where the work of the students in Paper 5 involved the first two phases, the students in Paper 2 moved straight ahead to the last phase of the model. The findings in Paper 2 reveal, as previously noted, that the students encountered difficulties when required to design their own algorithms and programs. Such demands appear to be overly challenging for students with limited or no prior experience in applying programming within mathematical contexts. This conclusion is supported by the results in Paper 5, which show that when students in the case study were expected to make more substantial modifications to their code, marking a shift from merely consuming others' work to engaging in original creation, they frequently encountered difficulties.

It could be argued that the analysis of students' different instrumental geneses has highlighted how such geneses are associated with aspects concerning the design of the learning activities, and particularly the task design. In the following section, findings related to such aspects will be described in more detail.

6.2 The design of the learning activities

Instrumental genesis is primarily conceptualised as an individual process, wherein students develop personal, mentally constructed, instrumented action schemes. However, the teacher's design of learning activities, together with other social aspects within a classroom, may affect the assimilation of such more or less socially pre-existing schemes and thus students' instrumental genesis (Artigue, 2002; Rabardel, 2002). In other words, teachers' design functions as a deliberate means of guiding students' instrumental genesis (Guin & Trouche, 2002; Trouche, 2005b). Papers 2, 4, and 5 describe aspects of the design of the learning activities within both the design-based study and the case study. This section presents these two designs alongside findings that demonstrate their influence on students' instrumental genesis.

6.2.1 The instrumental orchestration in Paper 2

One of the specified learning goals of the design study was for the students to develop a productive instrumental genesis in relation to the

use of programming as a problem-solving tool. It could therefore be argued that all aspects of the design are closely linked with the instrumental orchestration. In this subsection, special attention will be paid to the task design and the scaffolding offered to the students. Several of the *didactical variables* (DV) (Ruthven et al., 2009) identified during the design of the two design cycles DC1 and DC2 (see Table 4) were related to these aspects of the instrumental orchestration.

Table 4: Didactical variables (DV) taken into considerations in the design of the learning activity during the second micro cycle of the design study

Didactical variables

DV1: Programming language and programming environment

The participating students were to use a programming language and a programming environment familiar to the students.

DV2: Programming concepts

To solve the mathematical problems would require basic knowledge concerning variables, input/output, iterations (including nested loops), and selections.

DV3: Mathematical content

Solving the mathematical problems would require knowledge in algebra and arithmetic in order to handle variables and mathematical expressions.

DV4: Specifying the problem

Making the algebraic expression describing the loss of gold coins more straightforward compared to the original problem (DC1). Changing the age difference between Cinderella and Begonia as well as the number of lost coins in order for the first problem to have two different solutions irrespective of whether two sisters were allowed to have the same age or not (DC2).

DV5: Students working in pairs

Working in pairs was regarded as a means of sharing knowledge and served as a way of supporting the development of both instrumental genesis and problem-solving strategies. Discussions between students were also regarded as a way to detect students' instrumental geneses.

DV6: Displaying the code

Sherpa-students were able to display their code to the rest of the students (using screen sharing in Zoom).

DV7: Introductory information to the students

The participating students should be informed at the beginning of the teaching experiment that the mathematical problems could not be solved by using any analytical methods known by the students and that the problems should be solved by using the programming environment.

Task design

The findings suggest that an excessive emphasis on mathematical problem solving may have hampered students' instrumental genesis, despite the fact that the tasks were solvable using only elementary mathematical knowledge of arithmetic and algebra (DV3). As noted by Lester and Kehle (2003), mathematical problem solving is by definition a complex human activity, requiring much more than mere recall of facts. Several students in the study struggled to articulate an effective problem-solving strategy, even though they were able to express appropriate relationships among the mathematical variables. This difficulty may be attributed to their limited experience in using programming for mathematical purposes. Consequently, they struggled to recognise the affordances and constraints of the programming environment as a mathematical artefact, an essential component of instrumentalization (Trouche, 2005a).

Another design decision was to ensure that the problems could be solved using programming concepts and methods already familiar to the students from their ongoing Programming 1 course (DV2). Together with the decision to use the Java programming language and the NetBeans 8.2 development environment, both already integrated into the programming course (DV1), this design choice helped minimise difficulties with most programming concepts. However, the retrospective analysis revealed that several students encountered difficulties with nesting loops, particularly in formulating precise conditions to control the loop structures. Ginat (2004) highlights that such difficulties are common among novice programmers. The code in Figure 9 illustrates how two students (Ian and Jacob) during the second design cycle instead developed what may be described as a semi-automated program. Rather than creating a nested inner loop to vary the value of the variable f , the students manually modified the calculation at line 14, from $a + 1$ to $a + 14$, and executed the program 14 times.

In summary, the findings indicate that the task design encompassed two key aspects which may have constrained students' opportunities to use programming as a mathematical tool. First, engaging with programming as part of a problem-solving activity may have been overly demanding for students with limited experience in using the

technology for mathematical purposes. Second, the mandated use of nested loops may have further impeded students' problem-solving processes. This underscores the complex interplay between the technical handling of the artefact and the development of students' mathematical understanding, as difficulties with nested loops can negatively affect the advancement of students' mathematical problem-solving skills.

Figure 9: Screenshot from Ian and Jacob's use of NetBeans in the design study

```

12   for(int a=0; a<100; a++) {
13       double b=a+15,s=480;
14       double f = a+1;
15       double g ;
16       g = ((a+1)*(b+1)*(f+1))-(a*(b+1)*(f+1)) ;
17
18       if(g==s){
19           System.out.println("Askungen är "+a+" år gammal" );
20       }
21   }
22
23   }
24   System.exit(0);

```

Scaffolding

The main scaffolding offered to the students during the problem-solving activities was the use of so-called sherpa-students (Guin & Trouche, 1998). These students were asked on several occasions to visualise how they had used the artefact during the mathematical activities (DV6), under guidance from the researcher (acting as teacher). The intention behind using this orchestration type (Drijvers, Doorman, et al., 2010) is to guide other students' work and thus foster a collective instrumental genesis (Guin & Trouche, 1998). The analysis of other students' work indicated that the ways in which they benefited from the sherpa-students' presentations were closely related to how far they had progressed in their own problem-solving process compared to the sherpa-students, and by the degree of congruence between the ideas presented by the sherpa-students and those developed by the other students. This meant that, for some students, the sherpa-students' presentations affirmed their current approach, indicating they were progressing in the right direction. For others, who had yet to formulate a problem-solving strategy, the presentations introduced foundational ideas to support the development of such a strategy. Consequently, the findings

illustrate both that the temporal use of sherpa-students determines in which ways other students benefit from these presentations, and that it is difficult to plan when to employ sherpa-students during a lesson activity. It became a matter of balancing the need to allow students to engage independently in their problem-solving processes with the need to help them overcome potential impasses related to various aspects of those processes.

Due to the ongoing pandemic, the teaching experiment in the second design cycle was conducted online using Zoom, resulting in an unintended re-design of the original setup. Although the analysis of screen and voice recordings did not examine the consequences of this re-design in detail, and involved only a limited number of students, it suggests that the nature of student interactions, both with peers and with the teacher, could have been affected by the online format. Students appeared more reluctant to ask questions to the teacher, and being situated in different physical locations seemed to hinder communication between students working in pairs. A broad comparison of transcripts from the two teaching experiments reveals that student conversations were more detailed in the first (in-person) experiment than in the online setting. Consequently, although Zoom provided functionalities that facilitated collaboration, the lack of a shared physical space appears to have constrained student interaction.

6.2.2 *The teacher's design in Paper 4 and Paper 5*

Paper 4 presents how the teacher in the case study designed the lesson sequence involving the students' utilisation of programming to determine limit values numerically. Of particular interest was how the incorporation of programming into his mathematics classroom had required an adaptation of the teacher's craft knowledge. As argued by Bozkurt and Ruthven (2017), such adaptation is essential when teachers incorporate new digital tools (e.g., programming) into their mathematics instructions, as it shapes the nature of their instrumental orchestrations.

Drawing on an interview with the teacher, together with field notes generated during the lesson sequence, this adaptation was analysed

based on five structuring features of classrooms practice (Ruthven, 2009): the curriculum script, the resource system, the activity structure, the time economy, and the working environment. The findings reveal how the teacher's dual role as a mathematics and programming instructor influenced his pedagogical choices. Furthermore, these choices were shaped by the way programming has been integrated into the Swedish mathematics curricula.

Although Paper 5 does not explicitly analyse the teacher's design, the analysis of students' instrumental genesis provides valuable insights into the consequences of the design, given that specific design choices may shape the nature of such genesis.

The adaptation of the teacher's craft knowledge

The findings in Paper 4 indicate that several aspects relating to the design of the lessons were associated with the teacher's *curriculum script*, which stems from the teacher's professional knowledge and encompasses goals and actions related to the specific mathematical content (Ruthven, 2014). It was evident that the teacher deliberately positioned programming as a tool for solving mathematical tasks numerically rather than for mathematical problem solving. This decision was based on the argument that mathematical problem solving would require students to write code from scratch, which was presumed to be too demanding for students with limited programming experience. The principles guiding the teacher's task design were therefore grounded in the idea that programming should function as a tool for numerical computations, and that the associated tasks should aim to deepen students' understanding of key mathematical concepts, such as limits of functions, derivatives, and integrals.

To support students' technical engagement with programming in the application of numerical methods, the teacher intentionally restricted the number of programming concepts they were required to master. The concepts applied included WHILE loops, IF statements, print statements, and basic operations involving computational variables. This implied that the task design also was guided by the criterion that tasks should be solvable using the limited set of programming concepts

introduced. This aspect of the task design was partly driven by the *time economy* feature.

All tasks presented to students during the three lessons described in Paper 5 focused on the numerical determination of limit values. This was achieved using an algorithm that iteratively printed the values of the mathematical expression together with the corresponding values of the independent variable as it approached a given value (see Figure 10). Besides being asked to determine various limit values by adjusting pre-existing code, the task design involved associated questions relating the given limit value¹⁵. These questions focused on interpreting both the mathematical meaning of the numerical limit value and the mathematical reasoning underlying the construction of the code.

Figure 10: Screenshot of Bella and David's code for determining the limit

$\lim_{h \rightarrow 0} \frac{(x-1)^{10}-1}{x}$ in the case study

```

1 x=10
2 rep = True
3 while rep == True:
4     y = ((x-1)**10-1)/x
5     print (x, y)
6     x /=2
7     if x<= 0.00000001:
8         rep = False

```

```

10 348678440.0
5.0 209715.0
2.5 22.666015625
1.25 -0.7999992370605469
0.625 -1.5999120101332664
0.3125 -3.124512159192818
0.15625 -5.2296444138617915
0.078125 -7.12546412213972
0.0390625 -8.413344047315073
0.01953125 -9.165341575845384
0.009765625 -9.571797662434813
0.0048828125 -9.783110155895816
0.00244140625 -9.890848927519755
0.001220703125 -9.945246791880527

```

As previously described, the students were instructed to build their code according to a given code structure. The students also made use of pre-designed code examples, which they could modify to solve new

¹⁵ All tasks are presented in the appendix of Paper 5.

tasks. These resources were parts of the teacher's *resource system*. To ensure students had sufficient programming knowledge, the school allocated additional instructional time in the preceding mathematics course to introduce basic programming skills. This decision also reflects the *time economy* aspect of the instructional design.

During the lessons, no major changes were made in association with the *working environment*. However, during the activities, as part of the teacher's *activity format*, several different orchestration types associated with the use of programming were employed (Drijvers, Doorman, et al., 2010). The teacher employed *technical-demo orchestration* when introducing programming as a mathematical tool during the whole-class introduction; *explain-the-screen orchestration* when relating programming to the specific mathematical topic; *link-screen-board orchestration* when connecting the code displayed on the screen to algebraic manipulations on the whiteboard; and *discuss-the-screen orchestration* when initiating student discussions about the programming code.

The impact of design decisions on students' instrumental genesis

As Paper 5 examines the students' instrumental genesis during the first three lessons, the paper also provides evidence of the outcomes of the teacher's design, shaped by the structuring features of the classroom practice. The investigation of the two students' instrumental genesis in Paper 5 reveals that the students seldom encountered any technical difficulties related to the syntax of the programming language and that the code structure and the code examples provided adequate scaffolding for the students to use programming to calculate limit values numerically. These aspects of the task design and orchestration, shaped by the curriculum script, resource system, and time economy, established clear boundaries for students' engagement with programming, aligning closely with the teacher's intended outcomes for the lesson sequence. This meant that the task design and aspects associated with the orchestration of the learning activities fostered a productive instrumentalization, where the students were able to use and modify the code structure and the code examples to determine limit values.

However, the findings presented in Paper 5 indicate, as described in sub-section 6.1.2, that the teacher's task design primarily fostered a *process* view of limits among the two students. Although this may represent a natural first step in developing a *proceptual* view of limits, where students can view a limit both as a process and as a concept (Gray & Tall, 1994), this learning outcome did not fully align with the teacher's intention of developing students' conceptual understanding of limits. This became evident when the students were asked to interpret the meaning of certain limits. Furthermore, the findings indicate that the task design related to the numerical calculation of limits did not fully support students in coordinating the domain process (where x approaches a) and the range process (where $f(x)$ approaches L) into what Cottrill et al. (1996) describe as a mathematical limit scheme.

In summary, the findings reveal how the incorporation of programming into his teaching of mathematics inferred an adaptation of the teacher's craft knowledge associated with the five structuring features of classroom practice (Ruthven, 2009). These adaptations were influenced by the teacher's dual role as a programming teacher, in which deliberate choices in task design and scaffolding enabled students to use programming effectively for determining limit values. However, it could be argued that the task design did not provide students with adequate support to foster a deeper conceptual understanding of limits.

6.2.3 *The relationship between the instructional design in Paper 2, Paper 4, and Paper 5*

The papers present two markedly different approaches to introducing programming in upper-secondary mathematics education, each characterised by distinct designs and associated with different challenges. In the two studies, programming was also used for two different mathematical practices: problem solving and numerical calculations. This implied that in the design study, which involved open-ended mathematical problems, the students were expected to construct their own code from the ground up. This process involved two main stages: the development of a mathematical algorithm applicable to the problem situation, and the development of a computational algorithm for conducting an exhaustive trial. The findings presented in Paper 2 indicate

that the students encountered difficulties both in formulating an adequate problem-solving strategy and in translating their mathematical algorithms into computational implementations, while receiving only a limited amount of scaffolding. Given these findings, it could be questioned whether problem solving constitutes the most appropriate context for introducing programming to students with no, or only limited, prior experience of the technology in school mathematics.

In the case study, presented in Papers 4 and 5, the use of a code structure and code examples provided students with extensive scaffolding when using programming to determine limit values numerically. The inclusion of the code structure and worked code examples further illustrated for the students the affordances of programming as a tool for numerical calculations. However, the findings indicate how only making small adjustments to pre-designed code may not have fostered a profound mathematical understanding of the limit concept. The case study therefore highlights the delicate balance between providing sufficient scaffolding to support students' engagement with programming and granting enough autonomy to enable them to use programming as a tool for deepening their understanding of mathematical concepts.

The findings demonstrate that the design of tool-based tasks significantly influences students' engagement with programming as a mathematical instrument, thereby shaping their instrumental genesis. The two different task designs were, in several respects, shaped by the integration of programming into the Swedish mathematics curriculum. At the time of the design study, the upper-secondary mathematics curriculum stipulated that programming should be employed by students as a problem-solving tool. As the curriculum later expanded the scope of programming, the teacher in the case study opted to design tasks where programming was used to apply numerical methods.

Both task designs needed to account for students' limited prior experience with programming in mathematical contexts. This necessitated different pedagogical approaches. In the design study, the tasks emphasised the application of a pre-defined problem-solving strategy, relying only on basic mathematical knowledge (DV3 in Table 4) not directly tied to the course content. In the case study, tasks were shaped

by the practical constraint of teaching students to use the tool effectively within a limited timeframe. Consequently, the structuring feature of time economy played a pivotal role in both lesson planning and task design, for example, by providing students with a pre-defined code structure and illustrative code examples.

6.3 The analysis of students' instrumental genesis during programming activities

Over the last decades, the Instrumental Approach has been operationalised as a theoretical framework in mathematics education research to investigate students' handling of digital mathematical tools, such as computer algebra systems (e.g., Drijvers & Gravemeijer, 2005; Trouche, 2005a) and dynamic geometry software (e.g., Fahlgren, 2017; Turgut & Drijvers, 2021). In recent years, the framework has also been applied in relation to the use of programming in mathematics education (e.g., Buteau et al., 2020; Martinez, 2023; Misfeldt & Ejsing-Duun, 2015). However, a key distinction between traditional programming languages and tools such as computer algebra systems or dynamic geometry software is that the former are not primarily designed as mathematical or educational software (Buteau et al., 2020). In this section, and in relation to the third research question of this thesis, it will be elaborated on how the analysis of students' instrumental genesis might need to take into account that the artefacts used in the mathematical activities were not designed primarily for mathematical purposes. In Paper 3, students' instrumental genesis is analysed using two different analytical lenses, based on data generated during the design study. The first analytical approach was also employed in Papers 1 and 2, while the second was applied in Paper 5. Although the focus in Papers 2 and 5 was neither on comparing the respective analytical frameworks nor on evaluating their affordances and constraints, each paper includes an analysis of students' instrumental genesis. These analyses will, in this section, serve as illustrative examples of how each framework can be applied.

6.3.1 Analysing schemes using scheme components

The first approach for analysing students instrumented action schemes in the design study, presented in Papers 1, 2, and 3, involved representing these schemes using scheme components as defined by Vergnaud (1998a). According to Vergnaud (1998a), a scheme comprises *goal and anticipations, rules of actions, operational invariants, and possibilities of inferences*¹⁶. Rules of actions are defined as the generative part of the scheme whereas operational invariants are regarded as the epistemic aspects of the scheme. There are two types of operational invariants: concepts-in-action, which refer to the concepts that students consider relevant for solving a given task, and theorems-in-action, which represent propositions held to be true by students when they act. As described by Vergnaud (1998a), concepts are integral to the formulation of theorems, while theorems, in turn, give meaning and substance to the concepts. This implied that during the analysis of students' instrumented action schemes, identified theorems-in-action were formed based on relevant concepts-in-actions.

In the analysis of students' instrumented action schemes, presented in Papers 1, 2, and 3, technical and mathematical aspects of the problem-solving process are visible in numerous concepts-in-actions and theorems-in-action (see Table 2). Gueudet et al. (2020) refer to such schemes that integrate a mathematical process within programming as $p + m$ schemes, whose goals are simultaneously oriented toward both mathematical understanding and programming practice. They argue that $p + m$ schemes are bridging students' digital and mathematical competences and that "the identification of $p + m$ operational invariants in particular deepens our understanding of how mathematics and programming relate to each other" (pp. 7–8). In accordance with Gueudet et al. (2020), it could also be claimed that describing an instrumented action scheme using scheme components visualise the complexity of a $p + m$ instrumented action scheme. For example, the scheme visualised in Table 2 consists of 26 components. Although the number of scheme components may depend on the level of granularity

¹⁶ Since the analysis of students' instrumental genesis only concerned one lesson activity, it was not possible to analyse the scheme's possibilities of inferences.

applied during analysis, the resulting scheme indicates that integrating programming with mathematics activates a wide range of concepts and highlights the complex relationships among them.

6.3.2 Analysing schemes using conceptual elements and technical elements

The second approach for analysing students' instrumented action schemes, described in Papers 3 and 5, concerned representing schemes based on instrumented techniques and associated conceptual and technical elements (Drijvers & Gravemeijer, 2005; Turgut & Drijvers, 2021). In Paper 3, in which the two analytical approaches are compared, the analysis draws on data from the design study. In contrast, the analysis in Paper 5 is based on data from the case study.

Drijvers and Gravemeijer (2005) argue that since schemes are mental constructs, existing only in a student's mind, the instrumented techniques should be regarded as the visible manifestation of the schemes. Instrumented techniques are thus defined as "a set of rules and methods in a technological environment that is used for solving a specific type of problem" (p. 169). Each technique (and scheme) comprises conceptual and technical elements, where conceptual elements are linked to students' mathematical reasoning, while technical elements pertain to the operational use of the artefact (Turgut & Drijvers, 2021). The technical elements are thus guided by the conceptual elements.

In Paper 3, the two analytical approaches are compared and contrasted in line with the ideas of networking theories (Prediger et al., 2008). The findings suggest that analysing students' visible instrumented techniques is a straightforward manner of describing students' instrumented action schemes. The students' actions are closely related to the technical elements of the schemes which, in turn, are guided by mathematical conceptual elements.

The schemes presented in Paper 5 (see Table 3), concerning two students' use of programming for numerical calculations of limits, are each associated with a specific technique. In Paper 3, the scheme identified is related to several techniques. It could be argued that what

counts as a single scheme is once again a question of the level of granularity applied during analysis. It may also be argued that the four instrumented action schemes described in Table 3 are sub-schemes (Lagrange, 1999) to an overarching scheme concerning the use of programming to determine limit values numerically.

6.3.3 A comparison of the two analytical approaches

In this thesis, two analytical approaches have been operationalised to investigate students' instrumental genesis, specifically their instrumented action schemes when using programming as a mathematical tool. Before comparing and contrasting the two analytical lenses, it is important to highlight that both approaches aim to shed light on students' instrumental genesis. The two ways of analysing schemes using scheme components or instrumented techniques also reflects the historical background of the Instrumental Approach. Drijvers, Kieran, et al. (2010) emphasise that, within the context of Cognitive Ergonomics, the notion of a scheme is grounded in Vergnaud's (1998a) definition, which characterise a scheme as an "invariant organization of behavior for a certain class of situations" (p. 167). It is therefore reasonable to describe schemes using the components of schemes as defined by Vergnaud (1998a). The notion of technique stems, on the other hand, from the French Anthropological Theory of Didactics (Drijvers, Kieran, et al., 2010) and is recognised as one of the fundamental components of the praxeology (Chevallard & Sensevy, 2014). Analysing students' instrumented techniques could thus be regarded as a valid approach when investigating their instrumental genesis. Drijvers, Kieran, et al. (2010) argue that although schemes and techniques have different ontological backgrounds, they both acknowledge the intertwined relationship between technical and conceptual aspects. They further claim that this intertwined relationship constitutes the strength of the Instrumental Approach as a theoretical framework, and that it should be considered more significant than the distinction between technique and scheme.

The operationalisation of the two analytical frameworks demonstrates that both enable nuanced and detailed descriptions of instrumented action schemes. It may be argued that the two distinct descriptions of

schemes, each shaped by its respective analytical framework, exhibit shared characteristics. These commonalities enable the possibility to establishing links between the two frameworks. Rules of action are the generative parts of a scheme according to Vergnaud (1998a) and could thus be viewed as the visible manifestation of the scheme. Consequently, it could be argued that rules of action correspond to instrumented techniques which both could have pragmatic and epistemic values (Artigue, 2002). It may also be argued that concepts-in-action represent the psychological correlates of conceptual elements. On the other hand, theorems-in-action, which link together rules of actions and concepts-in-action, do not have an equivalent when describing $p + m$ schemes (Gueudet et al., 2020) based on the visible instrumented techniques. This connection between rules of actions and concepts-in-action, highlighted in Paper 3, is particularly significant in illustrating how relevant concepts, relating to both mathematics and programming, are materialised into actions using the artefact.

An important outcome of the analysis presented in Paper 3 is the call to broaden the conceptualisation of *conceptual elements* when analysing students' instrumented techniques. Unlike other digital tools designed specifically for mathematical purposes, programming may require students to construct their own programs, including the development of mathematical functionalities. It could be argued that $p + m$ schemes for using programming in school mathematics do not only encompass conceptual elements associated with the mathematical topic. The instrumented action schemes may also comprise conceptual elements associated with the code construction. Such *conceptual programming elements* may concern aspects relating to, for example, the sequential nature of steps in imperative programming (Ahmadzadeh et al., 2005) and the different meanings of variables and the equals sign in mathematics compared to programming (Haspekian et al., 2023). Consequently, including conceptual programming elements as an analytical construct when describing $p + m$ schemes offers a more nuanced account of the conceptual challenges students may encounter when using programming as a mathematical tool. However, since the students in the case study (presented in Paper 5) primarily engaged with pre-designed code examples, modifying rather than constructing them, the verbatim excerpts seldom provided explicit evidence of conceptual

programming elements. It may be claimed that the act of creating original code more prominently would foreground these conceptual programming elements.

Finally, the analysis of students' instrumental geneses in Paper 2 and Paper 5 suggests that schemes could be described at different levels of granularity. In Paper 2, the instrumented action scheme (see Table 2) comprises all aspects of the process of conducting an exhaustive trial. In Paper 5, on the other hand, students' ways of determining limit values numerically involves four different schemes (see Table 3), each of which corresponds to a specific instrumented technique. As previously noted, it could be argued that these schemes are, in fact, sub-schemes (Lagrange, 1999) within a broader scheme for determining limit values using programming. Alternatively, one could claim that the scheme presented in Table 2 could itself be decomposed into multiple sub-schemes, each representing distinct aspects of students' instrumented activity. The findings, illustrated in Table 3, show that decomposing a broader scheme into sub-schemes offers the possibility to elucidate how distinct scheme elements (or scheme components) are associated with specific instrumented techniques (or rules of actions). Moreover, this decomposition highlights the recurrence of certain elements (or components) across multiple sub-schemes.

6.4 Summary of key findings from the two studies

This chapter has presented findings from the two studies addressing students' use of programming as a mathematical tool and teachers' design of such activities. Based on the three research questions of the thesis, the findings are organised around: (a) students' instrumental genesis, (b) the design of programming-based mathematical activities, and (c) the application of two different frameworks for analysing students' instrumented action schemes. Table 5 summarise the key findings from the design study (DS) and the case study (CS) in relation to the three research questions of this thesis.

Table 5: Summary of the key findings in relation to the research questions

RQ 1. What are the instrumental geneses of upper-secondary school students appropriating programming as an instrument for mathematical activity?

Key findings For students with limited experience in using programming within school mathematics, creating their own code can be challenging, particularly when it is not clear how the programming environment can serve as a mathematical tool. This lack of clarity may impede students' instrumentalization. Moreover, the process of translating a mathematical algorithm into a computational form may be challenging due to conceptual discrepancies in notions that exist in both mathematics and programming yet are interpreted or applied differently within each domain. (DS)

Providing students with pre-designed code examples that they could use and modify helped establish clear boundaries regarding how a programming environment can function as a mathematical tool, thereby facilitating students' engagement with the artefact. While the code structure supported students in determining limits through programming, it did not always assist them in interpreting the calculated limit values. As part of the instrumentation process, the use of pre-defined code for numerically determining limits also led students to perceive the notion of a limit primarily as a process rather than as a concept. (CS)

Based on the progression model *Use-Modify-Create*, findings suggest that for students inexperienced in using programming in school mathematics, creating code from scratch may be overly demanding. Instead, students' handling of programming as a mathematical tool could be eased by introducing students to pre-designed code which they can use and modify. (DS+CS)

RQ 2. How can aspects of teachers' design of learning activities—in which students engage with programming as a mathematical tool—affect students' instrumental genesis?

Key findings Interweaving programming and mathematical problem solving can be overly demanding for students with limited programming experience, as both activities are inherently challenging. Allowing sherpa-students to guide their peers' work may, to varying degrees, support the development of a collective instrumental genesis. However, the temporal use of sherpa-students requires balancing two competing needs: enabling students to engage independently in their problem-solving processes while also providing timely assistance to help them overcome potential impasses related to different aspects of those processes. (DS)

A teacher's integration of programming into school mathematics is shaped by various structuring features of classroom practice, which are themselves influenced by how programming is embedded in the mathematics curriculum. Among these features, time economy appears particularly prominent, as it sets boundaries for aspects of lesson design, including task design. This suggests that considerations related to time economy can affect other structuring features, such as the teacher's curriculum script and resource system. (CS)

Although instrumental genesis is primarily an individual process, it is influenced by the design of mathematical activities that incorporate programming. Different forms of scaffolding (e.g., the use of sherpa-students and pre-designed code examples) affect the instrumental genesis and, consequently, students' mathematical understanding in distinct ways. (DS+CS)

RQ 3. How should analyses of students' instrumental genesis take account of the fact that the artefact—the programming environment—used during mathematical activities is not designed primarily as a mathematical tool?

Key findings The comparison between the two analytical frameworks for describing instrumented action schemes reveals that each framework offers distinct affordances yet both serve as valuable analytical lenses for characterising students' schemes. When describing $p + m$ schemes using scheme components (Vergnaud, 1998a), the theorems-in-action provide a valuable theoretical construct for linking the generative parts of a scheme (its rules of action) with its conceptual aspects, expressed through concepts-in-action. Although the abovementioned link is absent when schemes are analysed through technical and conceptual elements (Drijvers & Gravemeijer, 2005), the simplicity of describing schemes based on the observable instrumented techniques facilitates the operationalisation of the analytical framework. (DS)

The findings indicate that, since the artefacts in this thesis were not primarily designed as mathematical tools, the notion of conceptual elements should be expanded to include not only mathematical conceptual elements but also programming conceptual elements. (DS+CS)

7 Discussion

This chapter presents the main contributions of the thesis. The first section outlines and discusses its theoretical contributions. The second section examines the practical implications for teaching mathematics in upper-secondary education using programming. The final section looks ahead and discusses how the outcomes of the two studies have illustrated the need for further research.

7.1 Theoretical contributions

In this thesis, the Instrumental Approach has been operationalised as a framework for examining upper-secondary students' mathematical use of programming, with particular attention to their instrumental genesis. In addition, the construct of instrumental orchestration has been employed as a theoretical lens to illustrate how the design and orchestration of learning activities can foster both individual and collective instrumental genesis. The operationalisation of the Structuring Features of Classroom Practice framework has further enabled an analysis of how a teacher's craft knowledge was adapted when incorporating programming into mathematics teaching.

A key theoretical contribution of this thesis lies in demonstrating how the operationalisation of two analytical frameworks for examining students' instrumental genesis, together with the use of the abovementioned frameworks for analysing teachers' design of learning activities, can enrich our understanding of how programming may be meaningfully integrated into mathematics education.

7.1.1 Two frameworks for analysing the instrumental genesis

Analysing students' instrumental genesis involves describing the instrumented action schemes developed and used by students. In Papers 1 and 2, students' instrumented action schemes are portrayed using scheme components as defined by Vergnaud (1998a). In Paper 5, such schemes are described based on its conceptual and technical elements as proposed by Drijvers and Gravemeijer (2005). In Paper 3, the two analytical approaches are also compared and contrasted. The

implementation and comparison of the two analytical approaches, concerning programming as a technical artefact, serves as a theoretical contribution of this thesis. It can be argued that analysing the intertwined relationship between students' technical use of programming and their conceptual understanding of mathematical concepts is of particular importance, given that programming environments are not primarily designed as mathematical or educational tools. How this aspect influences the analysis of students' instrumental genesis has been of central relevance in this thesis.

Both analytical frameworks, illustrated by the instrumented action schemes in Table 2 and Table 3, portray the conceptual and technical complexity of utilising programming as a mathematical tool. The detailed descriptions of the schemes, as illustrated in the two tables, stem from a rigorous operationalisation of the two analytical frameworks, in which data were analysed through an iterative process. During this process, the researcher continuously revised the scheme components/elements, rephrasing, adding, or removing them as necessary. Defining these components/elements required a careful balance: on the one hand, ensuring they were sufficiently generic to identify commonalities across different schemes; on the other hand, maintaining enough specificity to capture the distinctive characteristics of each individual scheme.

The comparison between the two analytical frameworks indicates that they highlight, to varying degrees, the important relationship between technical and conceptual aspects during the instrumental genesis. Especially, theorems-in-action, defined as the propositions students regard to be true (Vergnaud, 1998a), serve to connect the technical handling of artefacts, as expressed through rules of action, with the conceptual aspects manifested by the concepts-in-action. The findings reveal that such connection is less evident in $p + m$ schemes described by its conceptual and technical elements. Describing schemes using scheme components may therefore offer a more nuanced description of the relationship between students' technical handling of programming environments and their conceptual understanding of the mathematical objects. The operationalisation and comparative analysis of the two analytical frameworks for analysing students' instrumental

genesis, particularly in relation to students' use of programming, constitutes a methodological contribution to the field.

Another theoretical contribution of this thesis lies in the proposed expansion of the analytical framework that characterises schemes through their conceptual and technical elements (Drijvers & Gravemeijer, 2005). In existing research utilising the Instrumental Approach as a theoretical framework, the artefact often constitutes a digital tool designed for mathematical purposes. For such artefacts, the mathematical affordances are readily apparent to students. However, a crucial part of students' initial instrumentalization when engaging with programming in mathematics classrooms, concerns modifying the artefact as to function effectively as a mathematical tool. This process involves, among other things, the development of a conceptual understanding of differences and similarities between the mathematical language and a programming language. I claim that these conceptual aspects linked to the technical handling of the artefact is not visible when schemes traditionally are described using conceptual and technical elements. This finding highlights the need to broaden the notion of conceptual elements. Instead of merely referring to *conceptual (mathematical) elements*, the framework should also consider *conceptual programming elements*. As Drijvers and Gravemeijer (2005) argue, usage schemes comprise not only technical aspects but also conceptual aspects. Extending the meaning of conceptual elements offers a more nuanced view of students' instrumented action schemes and has the potential to highlight important relationships between conceptual mathematical elements and conceptual programming elements. For example, the absence of crucial conceptual programming knowledge, particularly concerning the meaning of variables, can lead to the formation of a scheme in which variables are interpreted solely in mathematical terms, even when used within a programming context. When such a scheme also lacks conceptual programming elements related to the sequential nature of program execution, students tend to perceive computational assignments as static mathematical relationships. This was illustrated in the design study, where students viewed these assignments as universally valid throughout the code rather than as operations executed in a specific order.

The two analytical frameworks for analysing schemes have pros and cons. Analysing combined programming and mathematical schemes through the four main components of a scheme (Vergnaud, 1998a) may, via the theorems-in-action, more clearly illustrate the relationship between technical aspects related to the use of artefacts and conceptual aspects pertaining to both mathematics and programming. When the artefact constitutes (parts of) a programming environment, it is important to illuminate the conceptual programming aspects, as these may influence students' instrumental genesis. Therefore, as previously discussed, I propose that when analysing schemes using conceptual and technical elements (Drijvers & Gravemeijer, 2005), conceptual elements should also include elements related to students' conceptual understanding of programming. However, expanding an analytical framework may entail certain trade-offs. Drijvers and Gravemeijer (2005) point out that the terminology associated with the Instrumental Approach, such as instrument, instrumentation, and instrumentalization, is not inherently intuitive or self-explanatory. Consequently, describing schemes using two types of elements, rather than four distinct scheme components, may simplify the analysis of students' instrumental genesis. This reflects a necessary balance between providing detailed descriptions and ensuring that the outcomes of such analyses remain accessible and meaningful for both researchers and educators. It can also be questioned to what extent a mental scheme, existing solely in a student's mind, can be precisely described. Consequently, basing the analysis on students' instrumented techniques, and treating these as observable manifestations of underlying schemes (Drijvers & Gravemeijer, 2005), may be regarded as a more modest approach to characterising students' schemes.

Finally, although this thesis has demonstrated how students' instrumental genesis can be analysed through different approaches, it is important to emphasise that both approaches underscore the important intertwined relationship between technical and conceptual aspects. As Drijvers, Kieran, et al. (2010) aptly state "it is the importance of this relationship that makes instrumentation theory powerful" (p. 110).

7.1.2 Two frameworks for analysing teachers' instructional design

In this thesis, Instrumental Orchestration (Drijvers et al., 2009; Trouche, 2004) has been operationalised as a theoretical frame to explore how choices related to the orchestration of learning activities may affect a productive instrumental genesis. Moreover, also in relation to such orchestration, the Structuring Features of Classroom Practice framework (Ruthven, 2009) has been employed to analyse how the professional adaptation of a teacher's craft knowledge shaped the incorporation of programming into his teaching. Networking these frameworks, by combining them (Prediger et al., 2008), provided a multifaceted perspective on the empirical phenomenon. Extending this networked approach to the study of how programming is incorporated into school mathematics provides a theoretical contribution to the field. As emphasised by Bozkurt and Ruthven (2017), the two frameworks have different focuses. Whereas the Instrumental Orchestration offers insights into the orchestration of specific learning activities, the Structuring Features of Classroom Practice framework (SFCP) focuses on the overarching structure which set the stage for the teacher's orchestration. This conclusion is further underlined by Simsek and Clark-Wilson (2024) who argue that Instrumental Orchestration offers "a valuable lens for a finer-grained analysis of the construct of activity structure within the broader SFCP framework" (p. 1595). The analysis of how the teacher in the case study adapted his craft knowledge, framed by the five structuring features of classroom practice, illustrated how his instrumental orchestration was shaped by his professional background and personal beliefs about the role of programming in school mathematics. Moreover, the analysis demonstrated that this adaptation was affected by the way programming has been integrated into the Swedish mathematics curricula. In conclusion, this thesis has demonstrated that the SFCP framework can serve as a powerful analytical lens for investigating how and why overarching educational reforms and teacher practices are enacted through specific forms of instrumental orchestrations in mathematics classrooms.

7.2 Practical implications

This thesis has portrayed two studies that, in many aspects, adopt two different approaches concerning the use of programming as a tool in mathematics education. Both approaches are, in several ways, shaped by how programming has been integrated into the Swedish mathematics curriculum. The comparison between these approaches contributes to a more profound understanding of how different approaches for incorporating programming into mathematics education can influence students' instrumental genesis. This is of special interest for both in-service teachers, teacher educators, and policy makers.

7.2.1 Implications for in-service and pre-service teachers

In the two studies, programming was employed respectively as a means of facilitating mathematical problem solving and as a tool for performing numerical computations. To allow the students in the design study to engage independently in problem solving, they were expected to create their own algorithms. In the case study, on the other hand, the students were initially provided with given algorithms which they were expected to use and modify. It could be argued that the two different approaches took their starting points in two different ends of the progression model *Use-Modify-Create* presented by Lee et al. (2011). A key finding from the analysis of students' instrumental genesis in the two studies is that students with limited experience in using programming for mathematical purposes require appropriate support to grasp the mathematical affordances and constraints of the artefact before they can be expected to construct their own algorithms. Without such scaffolding, as evidenced in the design study, students' instrumentalization processes risk becoming primarily an unstructured exploration of these possibilities and constraints. Consequently, the process of instrumentation, through which engagement with programming shapes and deepens students' understanding of the underlying mathematical ideas, is likely to recede into the background. Omitting the *Use* and *Modify* stages of the progression model thus risks hampering students' instrumental genesis. In this sense, the task design in the design study, particularly its emphasis on mathematical problem solving, an inherently complex and demanding activity (Lester, 2013; Schoenfeld,

1992a), significantly influenced, and in times hindered, the development of a productive instrumental genesis.

Another practical implication from the design study, is that mathematics teachers must consider potential difficulties arising during students' transition from a mathematical to a computational algorithm. The findings indicate that this transition is not always a straightforward process. Existing research also highlights numerous technical and conceptual challenges met by students when introduced to programming (Ettles et al., 2018). Taub (2024) notes that although programming can foster mathematical reasoning, insufficient programming experience among students constrains the artefact's role in supporting their mathematical understanding. Due to the mathematical flavour of numerous programming concepts (du Boulay, 1989) (e.g., variables and the equals sign) students in the design study sometimes overlooked the fact that these concepts carry different meanings in mathematics and programming. Moreover, the students did not pay attention to the sequential nature of steps in programming but attributed mathematical properties to computer science calculations, for example, regarding such assignments as mathematical relationships between variables. Mathematics teachers must not overlook the fact that traditional programming languages are not designed primarily as mathematical tools (Buteau et al., 2020). As part of their design and instrumental orchestration, teachers must therefore account for the fundamental differences between the mathematical language and programming languages. Students should be made aware of the distinctions between concepts that appear in both programming and mathematics, and how these differences must be taken into account when transitioning from a mathematical to a computational algorithm.

An unintended observation from the design study was that working online, as opposed to in-person, appeared to reduce students' interactions with both their peers and the teacher. Previous research has emphasised that collaborative engagement with technology can stimulate student discussions (Brunström & Fahlgren, 2015; Goos et al., 2003) and thereby a sharing of knowledge (Goos et al., 2000). Although Zoom provides functionalities that facilitate mutual interaction with the artefact, the observations from the design study indicate that such

knowledge sharing is at risk of being constrained by the online format. It is important to note that the analysis of the screen and voice recordings did not examine the consequences of this re-design in detail. Nevertheless, this unintended observation raises important questions, given that collaborative work has the potential to foster the development of a productive instrumental genesis.

In contrast to the design study, the case study required students to engage with the *Use* and *Modify* stages of the progression model. Drawing on the teacher's task design, it may be argued that providing students with a pre-defined code structure and illustrative examples, which they could use and modify, effectively directed their attention toward exploring the artefact's potential for mathematical applications. In turn, this scaffolding supported students' instrumentalization by enabling them to concentrate on the construction of instrumented action schemes, interweaving technical aspects (relating to programming) and conceptual mathematical aspects. Thus, making only minor modifications to pre-designed code facilitated the participating students' use of programming as a tool for the numerical computation of limits. However, limiting students' autonomous use of programming may affect the extent to which such use contributes to deepen their mathematical understanding. Although the students managed to determine limit values, they did not consistently demonstrate an ability to interpret their meaning, to coordinate the domain and range processes, and to view limits as concepts not merely as processes. In the spirit of Papert (1980) it could be claimed that it is partly the construction of the code that fosters students' conceptual understanding of the mathematical topic. Making minor adjustments to pre-designed code ease students technical handling of programming but may not, as illustrated in the case study, foster an in-depth understanding of the mathematical concepts. Teachers' task design for integrating programming into mathematics education must therefore address how tasks can foster a productive instrumental genesis for using programming as a computational tool. At the same time, teachers must consider how tasks can support an instrumental genesis that promotes students' conceptual understanding by fostering the development of instrumented techniques that hold both pragmatic and epistemic values (Artigue, 2002).

In other words, the design should take into account both the processes of instrumentalization and instrumentation.

Furthermore, the findings from the case study illustrate how teachers' task design, when integrating programming into their teaching, may not only be affected by the curriculum script, but also by the structuring feature of time economy. As Ruthven (2014) argues, this structuring feature is related to teachers' ability to "changing the 'rate' at which the physical time available for classroom activity can be converted into a 'didactic time' measured in terms of the advance of knowledge" (p. 386). It could be argued that integrating programming into mathematics education places considerable demands on teachers, particularly when converting limited physical time into didactic time for programming. This challenge is compounded by the fact that programming environments are typically designed neither as mathematical tools nor as educational tools. From a Swedish perspective, where mathematics teachers must teach programming within a restricted timeframe, this becomes an even greater obstacle. The dual demands placed on Swedish teachers, introducing programming while maintaining a focus on mathematical content, likely constrain the design possibilities, limiting opportunities for students to engage with programming as a means of developing mathematical understanding. It is therefore unsurprising that the tasks designed to employ programming as a tool for numerical calculations did not effectively promote a profound mathematical understanding of limits.

In conclusion, the findings of this thesis suggest that teachers must be capable of making deliberate and informed decisions regarding their design of learning activities when integrating programming into their teaching of mathematics. In the Swedish mathematics curriculum for upper-secondary school, it is specified that programming should be used in relation to activities involving mathematical problem solving and/or the application of numerical methods (and/or data processing). The findings from the two studies suggest that introducing programming into mathematics education through mathematical problem solving may be perceived as too challenging for novice programmers who lack prior experience in applying programming within a mathematical context. Using programming as a numerical tool, could in that sense,

be regarded as a sound way of introducing programming, especially since non-problem-solving activities would allow a more direct guidance of students' use of the artefact. However, easing students' technical use of programming may come with a mathematically conceptual cost. It can be argued that mathematics teachers must strike a balance between scaffolding students' use of programming and allowing them the autonomy to independently engage with the tool in ways that support their mathematical understanding. Or in other words, teachers must find a balance between utilising programming to conduct mathematical calculations and solving mathematical problems. Achieving such balance also responds to the need identified by English and Sriraman (2010) to incorporate mathematical problem solving as a fundamental component in the introduction of new mathematical concepts. It can be argued that this complex balancing act originates from the way programming was integrated into the Swedish curriculum, originally with the overarching aim of enabling students to engage with programming and to understand its role in a digital society (The Ministry of Education and Research, 2017). Swedish mathematics teachers are not only expected to teach programming under significant time constraints, but they are also confronted with the challenge of using a tool that was not primarily designed for educational or mathematical purposes to foster students' mathematical understanding. This situation may be described as a didactical dilemma.

A successful implementation of programming in mathematics classrooms depends not only on the effective adaptation of in-service teachers' craft knowledge (Ruthven, 2014), but also on ensuring that pre-service teachers are well prepared to make informed instructional decisions. Therefore, the findings of this thesis may offer valuable insights for teacher educators, and professional developers, regarding the potential challenges faced by future mathematics teachers. The findings suggest that pre-service teachers must not only acquire knowledge about programming and its possibilities and constraints as a mathematical tool (as part of their personal instrumental genesis). They must also be equipped to make conscious didactical choices related to the integration of programming into mathematics teaching, choices that are shaped by structuring features of classroom practice.

7.2.2 Implications for policy

Although this thesis does not aim to evaluate the education reform of integrating programming into school mathematics in Sweden, the outcomes of its two studies highlight aspects related to this integration. As previously noted, in 2017 the Swedish National Agency for Education announced that, beginning in 2018, programming would be introduced as a mathematical tool from year four (approximately age 10) and onwards in Swedish lower and upper-secondary education. In lower-secondary school, the focus is on the design, interpretation, and modification of computational algorithms related to mathematical tasks and problems (The Swedish National Agency for Education, 2022b). In upper-secondary mathematics education, programming was initially introduced in 2018 as a tool for mathematical problem-solving activities. In 2021, its areas of application were broadened to also include numerical methods and data processing (The Swedish National Agency for Education, 2021). This implies that the Swedish national mathematics curricula do not associate the use of programming to any specific mathematical content but to specific mathematical activities (Tamborg et al., 2024). Furthermore, the curricula neither specify nor recommend which computational concepts students should master nor which programming languages to use. In general, the Swedish mathematics curriculum, as expressed through its overarching aims, core content, and grading criteria, sets out learning objectives, but does not prescribe specific methods for delivering the subject matter. This implies that Swedish teachers possess considerable autonomy in planning and delivering their instructions (OECD, 2024; Wermke et al., 2019). However, when programming was integrated into the mathematics curriculum, Swedish teachers expressed concerns and reported insufficient support regarding its intended application within school mathematics (Humble, 2022; Vinnervik, 2022). The findings in this thesis indicate how several aspects related to the implementation of this educational reform may have hampered Swedish mathematics teachers' use of programming in school mathematics.

Firstly, programming was incorporated into the core content of the mathematics curriculum without the removal of existing material, and teachers were expected to teach programming as an integral part of mathematics instruction. To solve this, the teacher in the case study

had allocated extra time for teaching programming in the previous course, an initiative that must be considered unusual in a Swedish context. It can therefore be presumed that teachers with limited personal experience in programming may struggle to incorporate the technology effectively into their teaching, particularly within the constraints of limited instructional time.

Secondly, although the wording of the curriculum provides teachers with considerable autonomy in how to incorporate programming into their mathematics classrooms, it does not offer sufficient guidance regarding the programming knowledge students are expected to acquire. This implied that the teacher in the case study assumed his students possessed highly varied levels of prior experience with programming from lower secondary mathematics education. Consequently, he could not expect them to possess knowledge of fundamental concepts such as loops and selection statements upon entering upper-secondary school. It was also reasonable to assume that the students had been exposed to a range of programming languages and environments.

Thirdly, findings in this thesis raise questions regarding the effectiveness of learning mathematics through programming. While using programming to solve mathematical tasks may be an efficient approach to learning programming, referred to by Fuentes Martinez (2024) as dual teaching, this does not necessarily imply that employing programming as a mathematical tool is an efficient way to enhance students' mathematical abilities and understanding. Existing research identifies several areas where students' engagement with programming can support their mathematical understanding (e.g., Munthe, 2024; Taub, 2024; Tossavainen et al., 2024). However, there is limited research investigating the effectiveness of such teaching approaches, which Fuentes Martinez (2024) refers to as interspersed teaching.

Overall, it could be argued that the concerns raised by mathematics teachers (Humble, 2022; Vinnervik, 2022) at the time of the implementation of the educational reform were very highly relevant. The accelerated implementation resulted in teachers having limited time for their own professional instrumental genesis (Haspekian, 2011). Furthermore, because programming was integrated into the mathematics

curricula without removing any existing mathematical content, teachers are required, within a restricted time frame, to teach students programming as well as the possibilities and constraint of using programming as a mathematical tool. These time constraints, combined with the limited guidance offered by the curricula, may have left many teachers fumbling in the dark. Given that national examinations do not include tasks requiring the use of programming, it is reasonable to assume that many Swedish mathematics teachers prioritise other curricular content over programming.

In England, all students should receive instruction in programming as part of the subject Computing (Tamborg et al., 2024). However, in contrast to Sweden, English students are not expected to engage with programming as part of their mathematics education. This suggests that the integration of programming into school mathematics in England is largely dependent on the initiative of individual schools and teachers. As in Sweden, initiatives involving the integration of programming into English school mathematics risk being constrained by the pressures of high-stakes testing (Noss et al., 2020).

A productive approach to integrating programming into school mathematics may be achieved through a hybrid model that draws on the respective approaches to implementation adopted in England and Sweden. Teaching programming as a distinct subject within the computing curriculum allows for a more focused exploration of the mathematical affordances of programming tools during mathematics lessons. Clearly defined learning objectives within the computing curriculum would also provide mathematics teachers with well-established expectations regarding the level of programming knowledge their students are likely to possess. From the perspective of the Instrumental Approach, outsourcing the development of usage schemes to the computing subject may enable a more focused engagement with students' instrumental genesis within mathematics education. This shift would allow mathematics instruction to concentrate on the development of instrumented action schemes aimed at transforming mathematical objects. As a result, greater emphasis would be placed more directly on using programming to enhance students' mathematical understanding.

Drawing on the findings of this study, policy makers involved in future educational reforms concerning digitalisation, should ensure that sufficient provisions are made to support both students' and teachers' technical handling of new artefacts within the school context. In this context, generative artificial intelligence is expected to play an increasingly significant role in teaching and in the school curriculum. However, as with programming, large language models (e.g., ChatGPT and Copilot) are not designed primarily as educational or mathematical tools, which implies that their mathematical affordances and constraints, from an educational perspective, may not be clear to either students or teachers. By analysing previous digitalisation reforms, such as the introduction of programming into mathematics education, we can gain a better understanding of the organisational and pedagogical conditions required for future reforms to be implemented in ways that enhance quality and promote equity within the school system.

7.3 Further research

The final contribution of this study lies in offering directions for future research in the field, which will be explored in this section.

The two studies presented in this thesis illustrate distinct approaches to incorporating programming into school mathematics. In the design study, the intention was to allow students to engage independently in problem solving, which initially led to difficulties in recognising the mathematical affordances of programming. In contrast, the case study employed a pre-defined code structure and examples, which clarified the intended mathematical focus. However, the structured design of the artefact may have limited students' opportunities to explore the mathematical concepts more deeply. Each approach presents its own didactical advantages and limitations, highlighting the need to balance structure and autonomy when incorporating programming into mathematics education. I therefore call for further research into how a balance between structured guidance and student autonomy can be effectively enacted when incorporating programming into school mathematics. Such studies could also respond to the call by English and Sriraman (2010), who advocate for problem solving to be an integral

and natural component of students' conceptual development concerning mathematical concepts in school curricula.

Both qualitative studies presented in this thesis involve only a small number of students. Consequently, while the conclusions drawn may contribute as pieces of a larger puzzle, the findings cannot be broadly generalised and therefore have limited external validity. This highlights the need for large-scale studies that examine the impact of using programming as a tool in school mathematics on students' mathematical understanding. Such studies could also evaluate the effectiveness of learning mathematics through programming and identify which mathematical abilities and aspects of students' mathematical understanding are most likely to be fostered through its use.

The case study presented in this thesis analysed how a teacher adapted his craft knowledge when integrating programming into mathematics education. The findings highlight how the teacher's professional background, along with the curricular framing of programming in the Swedish mathematics curriculum, influenced his approach to integration. Previous research in the Swedish context has shown that mathematics teachers have expressed concerns about how programming is intended to be incorporated into school mathematics (Humble, 2022; Vinnervik, 2022). These findings suggest that teachers' diverse backgrounds and attitudes toward programming may affect the adaptation of their craft knowledge and thus play a significant role in shaping how such incorporation unfolds in practice. Therefore, further research is needed that, in light of this educational reform, examines how teachers' backgrounds and beliefs, together with the ways in which programming has been integrated into the mathematics curriculum, influence their incorporation of programming into mathematics teaching. Such studies, involving both qualitative and quantitative methods, can shed light on the practical outcomes of such educational reform. They may furthermore provide valuable insights for future reforms related to digitalisation, such as the use and application of generative artificial intelligence in school mathematics.

A theoretical contribution of this thesis is that analytical frameworks for investigating students' instrumental genesis must have the

potential to take account of the inherent complexity of the artefact. More specific, such analytical frameworks must have the exploratory power to portray how conceptual programming aspects affect such genesis. While it could be argued that the students in the case study required a conceptual understanding of programming to fully comprehend the code examples, the structured guidance provided by these examples made such understanding less visible. In contrast, such conceptual understanding may become more explicit in contexts where students engage with programming more autonomously, as was the case in the design study. Consequently, I call for further research into students' instrumental genesis when using programming for mathematical purposes, research that operationalises an expanded notion of conceptual elements to include conceptual programming elements. Applying such a modified analytical framework may yield deeper insights into the intertwined relationship between students' engagement with programming and their development of an in-depth mathematical understanding.

8 References

- Abma, T. A., & Stake, R. E. (2014). Science of the particular: An advocacy of naturalistic case study in health research. *Qualitative Health Research*, 24(8), 1150–1161. <https://doi.org/10.1177/1049732314543196>
- Adler, R. H. (2022). Trustworthiness in qualitative research. *Journal of Human Lactation*, 38(4), 598–602. <https://doi.org/10.1177/08903344221116620>
- Ahmadzadeh, M., Elliman, D., & Higgins, C. (2005). An analysis of patterns of debugging among novice computer science students. *ACM SIGCSE Bulletin*, 37(3), 84–88. <https://doi.org/10.1145/1151954.1067472>
- Alzahrani, N., Vahid, F., Edgcomb, A., Lysecky, R., & Lysecky, S. (2018). *An analysis of common errors leading to excessive student struggle on homework problems in an introductory programming course* [Paper presentation]. 2018 ASEE Annual Conference & Exposition, Salt Lake City, UT, United States. <https://doi.org/10.18260/1-2--29771>
- Artigue, M. (2002). Learning mathematics in a CAS environment: The genesis of a reflection about instrumentation and the dialectics between technical and conceptual work. *International Journal of Computers for Mathematical Learning*, 7(3), 245–274. <https://doi.org/10.1023/a:1022103903080>
- Assude, T. (2005). Time management in the work economy of a class, a case study: Integration of Cabri in primary school mathematics teaching. *Educational Studies in Mathematics*, 59, 183–203. <https://doi.org/10.1007/s10649-005-5888-0>
- Bakker, A. (2018). *Design research in education: A practical guide for early career researchers*. Routledge.
- Bartolini Bussi, M., & Mariotti, M. A. (2008). Semiotic mediation in the mathematics classroom: Artifacts and signs after a Vygotskian perspective. In L. D. English & D. Kirshner (Eds.), *Handbook of international research in mathematics education* (2nd ed., pp. 746–783). Routledge.
- Blum, W., & Niss, M. (1991). Applied mathematical problem solving, modelling, applications, and links to other subjects — State, trends and issues in mathematics instruction. *Educational Studies in Mathematics*, 22(1), 37–68. <https://doi.org/10.1007/BF00302716>

- Bocconi, S., Inamorato dos Santos, A., Chiocciariello, A., Cachia, R., Kampylis, P., Giannoutsou, N., Dagienė, V., Punie, Y., Wastiau, P., Engelhardt, K., Earp, J., Horvath, M., Jasutė, E., Malagoli, C., Masiulionytė-Dagienė, V., & Stupurienė, G. (2022). *Reviewing computational thinking in compulsory education – State of play and practices from computing education*. Publications Office of the European Union. <https://doi.org/10.2760/126955>
- Bonar, J., & Soloway, E. (1985). Preprogramming knowledge: A major source of misconceptions in novice programmers. *Human – Computer Interaction*, 1(2), 133–161. https://doi.org/10.1207/s15327051hci0102_3
- Bozkurt, G., & Koyunkaya, M. Y. (2022). Using the instrumental orchestration model for planning and teaching technology-based mathematical tasks as part of a restructured practicum course. In A. Clark-Wilson, O. Robutti, & N. Sinclair (Eds.), *The mathematics teacher in the digital era: International research on professional learning and practice* (pp. 31–64). Springer International Publishing. https://doi.org/10.1007/978-3-031-05254-5_2
- Bozkurt, G., & Ruthven, K. (2017). Classroom-based professional expertise: A mathematics teacher’s practice with technology. *Educational Studies in Mathematics*, 94(3), 309–328. <https://doi.org/10.1007/s10649-016-9732-5>
- Brennan, K., & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking* [Paper presentation]. 2012 annual meeting of the American Educational Research Association, Vancouver, Canada. <http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf>
- Brunström, M., & Fahlgren, M. (2015). Designing prediction tasks in a mathematics software environment. *The International Journal for Technology in Mathematics Education*, 22(1), 3–18. https://doi.org/10.1564/tme_v22.1.01
- Bryman, A. (2016). *Social research methods* (5th ed.). Oxford University Press.
- Bråting, K., & Kilhamn, C. (2021). Exploring the intersection of algebraic and computational thinking. *Mathematical Thinking and Learning*, 23(2), 170–185. <https://doi.org/10.1080/10986065.2020.1779012>
- Burns, R. B., & Anderson, L. W. (1987). The activity structure of lesson segments. *Curriculum Inquiry*, 17(1), 31–53. <https://doi.org/10.2307/1179376>

- Buteau, C., Gueudet, G., Muller, E., Mgombelo, J., & Sacristán, A. I. (2020). University students turning computer programming into an instrument for ‘authentic’ mathematical work. *International Journal of Mathematical Education in Science and Technology*, 51(7), 1020–1041. <https://doi.org/10.1080/0020739X.2019.1648892>
- Buteau, C., Muller, E., Mgombelo, J., Rodriguez, M. S., Sacristán, A. I., & Gueudet, G. (2022). Instrumental orchestration of the use of programming technology for authentic mathematics investigation projects. In A. Clark-Wilson, O. Robutti, & N. Sinclair (Eds.), *The mathematics teacher in the digital era: International research on professional learning and practice* (pp. 289–322). Springer International Publishing. https://doi.org/10.1007/978-3-031-05254-5_11
- Buteau, C., Sacristán, A. I., & Muller, E. (2019). Roles and demands in constructionist teaching of computational thinking in university mathematics. *Constructivist Foundations*, 14(3), 294–309. <https://constructivist.info/14/3/294.buteau>
- Cetin, I. (2015). Students’ understanding of loops and nested loops in computer programming: An APOS theory perspective. *Canadian Journal of Science, Mathematics and Technology Education*, 15(2), 155–170. <https://doi.org/10.1080/14926156.2015.1014075>
- Cherenkova, Y., Zingaro, D., & Petersen, A. (2014). Identifying challenging CS1 concepts in a large problem dataset. In J. Dougherty, K. Nagel, A. Decker, & K. Eiselt (Eds.), *SIGCSE '14: Proceedings of the 45th ACM technical symposium on computer science education* (pp. 695–700). Association for Computing Machinery. <https://doi.org/10.1145/2538862.2538966>
- Chevallard, Y., & Bosch, M. (2020). Anthropological theory of the didactic (ATD). In S. Lerman (Ed.), *Encyclopedia of mathematics education* (pp. 53–61). Springer International Publishing. https://doi.org/10.1007/978-3-030-15789-0_100034
- Chevallard, Y., & Sensevy, G. (2014). Anthropological approaches in mathematics education, French perspectives. In S. Lerman (Ed.), *Encyclopedia of mathematics education* (pp. 38–43). Springer Netherlands. https://doi.org/10.1007/978-94-007-4978-8_9
- Chorney, S. (2022). Classroom practice and craft knowledge in teaching mathematics using Desmos: Challenges and strategies. *International Journal of Mathematical Education in Science and Technology*, 53(12), 3203–3227. <https://doi.org/10.1080/0020739X.2021.1931974>

- Cobb, P., Confrey, J., diSessa, A., Lehrer, R., & Schauble, L. (2003). Design experiments in educational research. *Educational Researcher* 32(1), 9–13. <https://doi.org/10.3102/0013189x032001009>
- Collins, A. (1992). Toward a design science of education. In E. Scanlon & T. O'Shea (Eds.), *New directions in educational technology* (pp. 15–22). Springer. https://doi.org/10.1007/978-3-642-77750-9_2
- Cooper, P., & McIntyre, D. (1996). *Effective teaching and learning: Teachers' and students' perspectives*. Open University Press.
- Cornu, B. (1991). Limits. In D. Tall (Ed.), *Advanced mathematical thinking* (pp. 153–166). Springer. https://doi.org/10.1007/0-306-47203-1_10
- Cottrill, J., Dubinsky, E., Nichols, D., Schwingendorf, K., Thomas, K., & Vidakovic, D. (1996). Understanding the limit concept: Beginning with a coordinated process scheme. *The Journal of Mathematical Behavior*, 15(2), 167–192. [https://doi.org/10.1016/S0732-3123\(96\)90015-2](https://doi.org/10.1016/S0732-3123(96)90015-2)
- de Moraes Rocha, K., & Trouche, L. (2017). Documentational trajectory: A tool for analyzing the genesis of a teacher's resource system across her collective work. *Proceedings of the Tenth Congress of the European Society for Research in Mathematics Education*, 3732–3739. <https://hal.science/hal-01541068>
- Drijvers, P. (2003). *Learning algebra in a computer algebra environment: Design research on the understanding of the concept of parameter* [Doctoral thesis, Utrecht University]. Utrecht University Repository.
- Drijvers, P. (2012). Teachers transforming resources into orchestrations. In G. Gueudet, B. Pepin, & L. Trouche (Eds.), *From text to 'lived' resources: Mathematics curriculum materials and teacher development* (pp. 265–281). Springer. https://doi.org/10.1007/978-94-007-1966-8_14
- Drijvers, P., Doorman, M., Boon, P., Reed, H., & Gravemeijer, K. (2010). The teacher and the tool: Instrumental orchestrations in the technology-rich mathematics classroom. *Educational Studies in Mathematics*, 75(2), 213–234. <https://doi.org/10.1007/s10649-010-9254-5>
- Drijvers, P., Doorman, M., Boon, P., & van Gisbergen, S. (2009). Instrumental orchestration: Theory and practice. In V. Durand-Guerrier, S. Soury-Lavergne, & F. Arzarello (Eds.), *Proceedings of the Sixth Congress of the European Society for Research in Mathematics Education* (pp. 1349–1358). Service des publications, INRP.

- Drijvers, P., Godino, J. D., Font, V., & Trouche, L. (2013). One episode, two lenses. *Educational Studies in Mathematics*, 82(1), 23–49. <https://doi.org/10.1007/s10649-012-9416-8>
- Drijvers, P., & Gravemeijer, K. (2005). Computer algebra as an instrument: Examples of algebraic schemes. In D. Guin, K. Ruthven, & L. Trouche (Eds.), *The didactical challenge of symbolic calculators: Turning a computational device into a mathematical instrument* (pp. 163–196). Springer. https://doi.org/10.1007/o-387-23435-7_8
- Drijvers, P., Kieran, C., Mariotti, M.-A., Ainley, J., Andresen, M., Chan, Y. C., Dana-Picard, T., Gueudet, G., Kidron, I., Leung, A., & Meagher, M. (2010). Integrating technology into mathematics education: Theoretical perspectives. In C. Hoyles & J.-B. Lagrange (Eds.), *Mathematics education and technology-Rethinking the terrain* (pp. 89–132). Springer. https://doi.org/10.1007/978-1-4419-0146-0_7
- Drijvers, P., Tacoma, S., Besamusca, A., van den Heuvel, C., Doorman, M., & Boon, P. (2014). Digital technology and mid-adopting teachers' professional development: A case study. In A. Clark-Wilson, O. Robutti, & N. Sinclair (Eds.), *The mathematics teacher in the digital era: An international perspective on technology focused professional development* (pp. 189–212). Springer. https://doi.org/10.1007/978-94-007-4638-1_9
- du Boulay, B. (1989). Some difficulties of learning to program. In E. Soloway & J. C. Spohrer (Eds.), *Studying the novice programmer* (pp. 283–314). Routledge.
- Dunbar, K., & Blanchette, I. (2001). The in vivo/in vitro approach to cognition: The case of analogy. *Trends in Cognitive Sciences*, 5(8), 334–339. [https://doi.org/10.1016/S1364-6613\(00\)01698-3](https://doi.org/10.1016/S1364-6613(00)01698-3)
- English, L., & Sriraman, B. (2010). Problem solving for the 21st century. In B. Sriraman & L. English (Eds.), *Theories of mathematics education: Seeking new frontiers* (pp. 263–290). Springer. https://doi.org/10.1007/978-3-642-00742-2_27
- Ettles, A., Luxton-Reilly, A., & Denny, P. (2018). Common logic errors made by novice programmers. *Proceedings of the 20th Australasian Computing Education Conference*, 83–89. <https://doi.org/10.1145/3160489.3160493>
- European Commission. (2019). *Key competences for lifelong learning*. Publications Office of the European Union. <https://doi.org/10.2766/569540>
- Fahlgren, M. (2017). Redesigning task sequences to support instrumental genesis in the use of movable points and slider bars. *The International Journal for Technology in Mathematics Education*, 24(1), 3–15. https://doi.org/10.1564/tme_v24.1.01

- Fuentes Martinez, A. (2024). *Practice beyond technology when programming and mathematics teaching converge* (Publication Number 61) [Doctoral thesis, University West]. DiVA. <http://urn.kb.se/resolve?urn=urn:nbn:se:hv:diva-21050>
- Ginat, D. (2004). On novice loop boundaries and range conceptions. *Computer Science Education*, 14(3), 165–181. <https://doi.org/10.1080/0899340042000302709>
- Goos, M., Galbraith, P., Renshaw, P., & Geiger, V. (2000). Reshaping teacher and student roles in technology-enriched classrooms. *Mathematics Education Research Journal*, 12(3), 303–320. <https://doi.org/10.1007/bf03217091>
- Goos, M., Galbraith, P., Renshaw, P., & Geiger, V. (2003). Perspectives on technology mediated learning in secondary school mathematics classrooms. *The Journal of Mathematical Behavior*, 22(1), 73–89. [https://doi.org/10.1016/S0732-3123\(03\)00005-1](https://doi.org/10.1016/S0732-3123(03)00005-1)
- Gravemeijer, K. (2015). Design research on local instruction theories in mathematics education. In O. Helenius, A. Engström, T. Meaney, P. Nilsson, E. Norén, J. Sayers, & M. Österholm (Eds.), *Development of Mathematics Teaching: Design, Scale, Effects* (pp. 1–3). SMDf.
- Gravemeijer, K., & Cobb, P. (2006). Design research from a learning design perspective. In J. Van Den Akker, K. Gravemeijer, S. McKenney, & N. Nieveen (Eds.), *Educational design research* (pp. 45–85). Routledge.
- Gravemeijer, K., & Prediger, S. (2019). Topic-specific design research: An introduction. In G. Kaiser & N. Presmeg (Eds.), *Compendium for early career researchers in mathematics education* (pp. 33–57). Springer. https://doi.org/10.1007/978-3-030-15636-7_2
- Gray, E. M., & Tall, D. O. (1994). Duality, ambiguity, and flexibility: A "proceptual" view of simple arithmetic. *Journal for Research in Mathematics Education*, 25(2), 116–140. <https://doi.org/10.2307/749505>
- Gueudet, G., Buteau, C., Muller, E., Mgombelo, J., & Sacristán, A. I. (2020). Programming as an artefact: What do we learn about university students' activity? *Proceedings of the INDRUM 2020*. <https://hal.archives-ouvertes.fr/hal-03113851>
- Guin, D., & Trouche, L. (1998). The complex process of converting tools into mathematical instruments: The case of calculators. *International Journal of Computers for Mathematical Learning*, 3(3), 195–227. <https://doi.org/10.1023/A:1009892720043>

- Guin, D., & Trouche, L. (2002). Mastering by the teacher of the instrumental genesis in CAS environments: Necessity of instrumental orchestrations. *ZDM*, 34(5), 204–211. <https://doi.org/10.1007/bf02655823>
- Gustafsson, P. (2017). Exploring a framework for technology integration in the mathematics classroom. *Proceedings of the Tenth Congress of the European Society for Research in Mathematics Education*, 2374–2381. <https://hal.science/hal-01942139v1>
- Hammer, D., & Berland, L. K. (2014). Confusing claims for data: A critique of common practices for presenting qualitative research on learning. *Journal of the Learning Sciences*, 23(1), 37–46. <https://doi.org/10.1080/10508406.2013.802652>
- Haspekian, M. (2011). The co-construction of a mathematical and a didactical instrument. *Proceedings of the Seventh Congress of the European Society for Research in Mathematics Education*, 2298–2307. <https://hal.archives-ouvertes.fr/hal-01273866>
- Haspekian, M., Kieran, C., Drijvers, P., Bråting, K., & Tabach, M. (2023). Algebra education and digital resources: A long-distance relationship? In B. Pepin, G. Gueudet, & J. Choppin (Eds.), *Handbook of digital resources in mathematics education* (pp. 1–33). Springer. https://doi.org/10.1007/978-3-030-95060-6_16-1
- Hiebert, J., Carpenter, T. P., Fennema, E., Fuson, K., Human, P., Murray, H., Olivier, A., & Wearne, D. (1996). Problem solving as a basis for reform in curriculum and instruction: The case of mathematics. *Educational Researcher* 25(4), 12–21. <https://doi.org/10.2307/1176776>
- Holo, O. E., Kveim, E. N., Lysne, M. S., Taraldsen, L. H., & Haara, F. O. (2023). A review of research on teaching of computer programming in primary school mathematics: Moving towards sustainable classroom action. *Education Inquiry*, 14(4), 513–528. <https://doi.org/10.1080/20004508.2022.2072575>
- Hoyles, C., & Noss, R. (2015). A computational lens on design research. *ZDM*, 47(6), 1039–1045. <https://doi.org/10.1007/s11858-015-0731-2>
- Hoyles, C., & Noss, R. (2021). Mapping a way forward for computing and mathematics: Reflections on the UCL ScratchMaths project. In C. Buteau, G. Gadanidis, G. Gannon, & A. Figov (Eds.), *Proceedings of the 2020 online seminar series on programming in mathematics education* (pp. 6–10). Mathematics Knowledge Network. <https://mkn-rcm.ca/wp-content/uploads/2021/01/OSSPME-Proceedings-January-24.pdf>

- Humble, N. (2022). Teacher observations of programming affordances for K–12 mathematics and technology. *Education and Information Technologies*, 27(4), 4887–4904. <https://doi.org/10.1007/s10639-021-10811-w>
- Juter, K. (2009). Learning analysis: Students' starting point. In C. Winsløw (Ed.), *Proceedings from NORMA08* (pp. 127–134). Brill. https://doi.org/10.1163/9789087907839_019
- Katz, V. J. (2009). *A history of mathematics : An introduction* (3rd ed.). Addison-Wesley.
- Kendal, M., & Stacey, K. (2001). The impact of teacher privileging on learning differentiation with technology. *International Journal of Computers for Mathematical Learning*, 6(2), 143–165. <https://doi.org/10.1023/A:1017986520658>
- Kieran, C., Doorman, M., & Ohtani, M. (2015). Frameworks and principles for task design. In A. Watson & M. Ohtani (Eds.), *Task design in mathematics education* (pp. 19–81). Springer. https://doi.org/10.1007/978-3-319-09629-2_2
- Kilhamn, C., Bråting, K., & Rolandsson, L. (2021). Teachers' arguments for including programming in mathematics education. In G. A. Nortvedt, N. F. Buchholtz, J. Fauskanger, F. Hreinsdóttir, M. Hähkiöniemi, B. E. Jessen, J. Kurvits, Y. Liljekvist, M. Misfeldt, M. Naalsund, H. K. Nilsen, G. Pálsdóttir, P. Portaankorva-Koivisto, J. Radišić, & A. Wernberg (Eds.), *Proceedings of the Ninth Nordic Conference on Mathematics Education* (pp. 169–176). SMDF.
- Kohn, T. (2017). Variable evaluation: An exploration of novice programmers' understanding and common misconceptions. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, 345–350. <https://doi.org/10.1145/3017680.3017724>
- Koulouri, T., Lauria, S., & Macredie, R. D. (2014). Teaching introductory programming: A quantitative evaluation of different approaches. *ACM Transactions on Computing Education*, 14(4), 1–28. <https://doi.org/10.1145/2662412>
- Kumsa Beyene, A. (2023). *Obstacles to students' learning of the limit concept: A comparative study* [Doctoral thesis, Stockholm University]. DiVA. <http://urn.kb.se/resolve?urn=urn:nbn:se:su:diva-222664>
- Kvale, S., & Brinkmann, S. (2009). *InterViews: Learning the craft of qualitative research interviewing* (2nd ed.). Sage Publications.
- Küchemann, D. (1978). Children's understanding of numerical variables. *Mathematics in School*, 7(4), 23–26. <http://www.jstor.org/stable/30213397>

- Lagrange, J. B. (1999). Complex calculators in the classroom: Theoretical and practical reflections on teaching pre-calculus. *International Journal of Computers for Mathematical Learning*, 4(1), 51–81. <https://doi.org/10.1023/A:1009858714113>
- Lahtinen, E., Ala-Mutka, K., & Järvinen, H.-M. (2005). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin*, 37(3), 14–18. <https://doi.org/10.1145/1151954.1067453>
- Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., & Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads*, 2(1), 32–37. <https://doi.org/10.1145/1929887.1929902>
- Lester, F. K. (1996). Problemlösningens natur. In R. Ahlström, B. Bergius, G. Emanuelsson, L. Emanuelsson, M. Holmquist, E. Rystedt, & K. Wallby (Eds.), *Matematik - Ett kommunikationsämne* (pp. 85–91). NCM.
- Lester, F. K. (2013). Thoughts about research on mathematical problem-solving instruction. *Mathematics Enthusiast*, 10(1), 245–278. <https://doi.org/10.54870/1551-3440.1267>
- Lester, F. K., & Kehle, P. E. (2003). From problem solving to modeling: The evolution of thinking about research on complex mathematical activity. In R. Lesh & H. M. Doerr (Eds.), *Beyond constructivism: Models and modeling perspectives on mathematics problem solving, learning, and teaching* (pp. 501–517). Routledge.
- Li, L., & Tall, D. (1993). Constructing different concept images of sequences and limits by programming. In I. Hirabayashi, N. Nobda, K. Shigematsu, & F. Lin (Eds.), *Proceedings of PME 17* (Vol. 2, pp. 41–48).
- Lodi, M., & Martini, S. (2021). Computational thinking, between Papert and Wing. *Science & Education*, 30(4), 883–908. <https://doi.org/10.1007/s11191-021-00202-5>
- Mannila, L., Peltomäki, M., & Salakoski, T. (2006). What about a simple language? Analyzing the difficulties in learning to program. *Computer Science Education*, 16(3), 211–227. <https://doi.org/10.1080/08993400600912384>
- Martinez, A. E. (2023). Using python to reason about logic and set theory: Three instrumented action schemes. *Digital Experiences in Mathematics Education*, 10. <https://doi.org/10.1007/s40751-023-00130-9>
- Merriam, S. B. (1998). *Qualitative research and case study applications in education*. Jossey-Bass.

- Misfeldt, M., & Ejsing-Duun, S. (2015). Learning mathematics through programming: An instrumental approach to potentials and pitfalls. In K. Krainer & N. Vondrova (Eds.), *Proceedings of the Ninth Congress of the European Society for Research in Mathematics Education* (pp. 2524–2530). HAL.
- Misfeldt, M., Szabo, A., & Helenius, O. (2019). Surveying teachers' conception of programming as a mathematics topic following the implementation of a new mathematics curriculum. In J. Uffe Thomas, H.-P. Marja van den, & V. Michiel (Eds.), *Proceedings of the 11th Congress of the European Society for Research in Mathematics Education* (pp. 2713–2720). HAL. <https://hal.archives-ouvertes.fr/hal-02417074>
- Monaghan, J. (2016a). Constructionism. In J. Monaghan, L. Trouche, & J. M. Borwein (Eds.), *Tools and mathematics* (pp. 181–196). Springer. https://doi.org/10.1007/978-3-319-02396-0_8
- Monaghan, J. (2016b). Tools, human development and mathematics. In J. Monaghan, L. Trouche, & J. M. Borwein (Eds.), *Tools and mathematics* (pp. 91–115). Springer. https://doi.org/10.1007/978-3-319-02396-0_4
- Monaghan, J., Sun, S., & Tall, D. (1994). Construction of the limit concept with a computer algebra system. In J.-P. de Ponte & J.-F. Matos (Eds.), *Proceedings of PME 18* (Vol. 3, pp. 279–286). University of Lisbon.
- Monaghan, J., & Trouche, L. (2016). Introduction to the book. In J. Monaghan, L. Trouche, & J. M. Borwein (Eds.), *Tools and mathematics* (pp. 3–12). Springer. https://doi.org/10.1007/978-3-319-02396-0_1
- Munthe, M. (2022). Programming in the mathematics classroom - Adversities students encounter. *Acta Didactica Norden*, 16(4), 1–22. <https://doi.org/10.5617/adno.9173>
- Munthe, M. (2024). Facilitating exploratory talk through mathematical programming problems. *Nordic Studies in Mathematics Education*, 29(1), 61–82. <https://doi.org/10.7146/nomad.v29i1.144978>
- Noss, R. (1986). Constructing a conceptual framework for elementary algebra through Logo programming. *Educational Studies in Mathematics*, 17(4), 335–357. <https://doi.org/10.1007/BF00311324>
- Noss, R., & Hoyles, C. (1996). *Windows on mathematical meanings: Learning cultures and computers*. Kluwer Academic Publishers.

- Noss, R., Hoyles, C., Saunders, P., Clark-Wilson, A., Benton, L., & Kalas, I. (2020). Making constructionism work at scale: The story of ScratchMaths. In N. Holbert, M. Berland, & Y. B. Kafai (Eds.), *Designing constructionist futures: The art, theory, and practice of learning designs*. The MIT Press. <https://doi.org/10.7551/mitpress/12091.003.0007>
- OECD. (2024). *Curriculum flexibility and autonomy: Promoting a thriving learning environment*. OECD Publishing. <https://doi.org/10.1787/eccb2-en>
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books.
- Pólya, G. (1971). *How to solve it: A new aspect of mathematical method* (2nd ed.). Princeton University Press.
- Popat, S., & Starkey, L. (2019). Learning to code or coding to learn? A systematic review. *Computers & Education*, *128*, 365–376. <https://doi.org/10.1016/j.compedu.2018.10.005>
- Prediger, S., Bikner-Ahsbals, A., & Arzarello, F. (2008). Networking strategies and methods for connecting theoretical approaches: First steps towards a conceptual framework. *ZDM*, *40*(2), 165–178. <https://doi.org/10.1007/s11858-008-0086-z>
- Prediger, S., Gravemeijer, K., & Confrey, J. (2015). Design research with a focus on learning processes: An overview on achievements and challenges. *ZDM*, *47*(6), 877–891. <https://doi.org/10.1007/s11858-015-0722-3>
- Psycharis, S., & Kallia, M. (2017). The effects of computer programming on high school students' reasoning skills and mathematical self-efficacy and problem solving. *Instructional Science*, *45*(5), 583–602. <https://doi.org/10.1007/s11251-017-9421-5>
- Putnam, R. T. (1987). Structuring and adjusting content for students: A study of live and simulated tutoring of addition. *American Educational Research Journal*, *24*(1), 13–48. <https://doi.org/10.2307/1162851>
- Rabardel, P. (2002). *People and technology: A cognitive approach to contemporary instruments*. HAL. <https://hal.archives-ouvertes.fr/hal-01020705>
- Regulation (EU). (2016/679). Regulation (EU) of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). *Official Journal of the European Union*, *L 119*. <https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=OJ:L:2016:119:FULL&from=EN>

- Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, 52(11), 60–67. <https://doi.org/10.1145/1592761.1592779>
- Rigby, L., Denny, P., & Luxton-Reilly, A. (2020). A miss is as good as a mile: Off-by-one errors and arrays in an Introductory programming course. *Proceedings of the Twenty-second Australasian Computing Education Conference*, 31–38. <https://doi.org/10.1145/3373165.3373169>
- Robson, C., & McCartan, K. (2016). *Real world research: A resource for users of social research methods in applied settings* (4th ed.). Wiley.
- Ruthven, K. (2002). Instrumenting mathematical activity: Reflections on key studies of the educational use of computer algebra systems. *International Journal of Computers for Mathematical Learning*, 7(3), 275–291. <https://doi.org/10.1023/A:1022108003988>
- Ruthven, K. (2009). Towards a naturalistic conceptualisation of technology integration in classroom practice: The example of school mathematics. *Éducation et didactique*, 3(1), 131–159. <https://doi.org/10.4000/educationdidactique.434>
- Ruthven, K. (2012). The didactical tetrahedron as a heuristic for analysing the incorporation of digital technologies into classroom practice in support of investigative approaches to teaching mathematics. *ZDM*, 44(5), 627–640. <https://doi.org/10.1007/s11858-011-0376-8>
- Ruthven, K. (2014). Frameworks for analysing the expertise that underpins successful integration of digital technologies into everyday teaching practice. In A. Clark-Wilson, O. Robutti, & N. Sinclair (Eds.), *The mathematics teacher in the digital era: An international perspective on technology focused professional development* (pp. 373–393). Springer. https://doi.org/10.1007/978-94-007-4638-1_16
- Ruthven, K., Laborde, C., Leach, J., & Tiberghien, A. (2009). Design tools in didactical research: Instrumenting the epistemological and cognitive aspects of the design of teaching sequences. *Educational Researcher* 38(5), 329–342. <https://doi.org/10.3102/0013189x09338513>
- Santos-Trigo, M. (2024). Problem solving in mathematics education: Tracing its foundations and current research-practice trends. *ZDM*, 56(2), 211–222. <https://doi.org/10.1007/s11858-024-01578-8>

- Schoenfeld, A. H. (1992a). Learning to think mathematically: Problem solving, metacognition, and sense-making in mathematics. In D. Grouws (Ed.), *Handbook for research on mathematics teaching and learning* (pp. 334–370). MacMillan.
- Schoenfeld, A. H. (1992b). On paradigms and methods: What do you do when the ones you know don't do what you want them to? Issues in the analysis of data in the form of videotapes. *The Journal of the Learning Sciences*, 2(2), 179–214. <http://www.jstor.org/stable/1466838>
- Schoenfeld, A. H. (2007). Method. In F. K. Lester (Ed.), *Second handbook of research on mathematics teaching and learning* (Vol. 1, pp. 69–107). Information Age Publishing.
- Schunk, D. H. (2020). *Learning theories: An educational perspective* (8th ed.). Pearson.
- Simon, M. A. (1995). Reconstructing mathematics pedagogy from a constructivist perspective. *Journal for Research in Mathematics Education*, 26(2), 114–145. <https://doi.org/10.2307/749205>
- Simsek, A., & Clark-Wilson, A. (2024). Adopting a framework for investigating mathematics teachers' technology-integrated classroom teaching practice: Structuring features of classroom practice. *International Journal of Science and Mathematics Education*, 23, 589–616. <https://doi.org/10.1007/s10763-024-10480-4>
- Smith, D. E. (1959). *A source book in mathematics*. Dover Publications.
- Sutherland, R. (1989). Providing a computer based framework for algebraic thinking. *Educational Studies in Mathematics*, 20(3), 317–344. <https://doi.org/10.1007/bf00310876>
- Sutherland, R. (1994). The role of programming: Towards experimental mathematics. In R. Biehler, R. W. Scholz, R. Sträßer, & B. Winkelmann (Eds.), *Didactics of mathematics as a scientific discipline* (pp. 177–187). Springer.
- Swedish Ethical Review Authority. (2023). *Guide to the ethical review of research on humans* (Revised ed.). Swedish Research Council.
- Swedish Research Council. (2024). *Good Research Practice 2024* (Revised ed.). <https://www.vr.se/english/analysis/reports/our-reports/2025-07-03-good-research-practice-2024.html>
- Tabach, M. (2011). A mathematics teacher's practice in a technological environment: A case study analysis using two complementary theories. *Technology, Knowledge and Learning*, 16(3), 247–265. <https://doi.org/10.1007/s10758-011-9186-x>

- Tamborg, A. L., Elicer, R., Bråting, K., Geraniou, E., Jankvist, U. T., & Misfeldt, M. (2024). The politics of computational thinking and programming in mathematics education: Comparing curricula and resources in England, Sweden, and Denmark. In B. Pepin, G. Gueudet, & J. Choppin (Eds.), *Handbook of digital resources in mathematics education* (pp. 1367–1392). Springer. https://doi.org/10.1007/978-3-031-45667-1_55
- Tang, J. C., Liu, S. B., Muller, M., Lin, J., & Drews, C. (2006). Unobtrusive but invasive: Using screen recording to collect field data on computer-mediated interaction. *Proceedings of the 20th anniversary conference on Computer Supported Cooperative Work*, 479–482. <https://doi.org/10.1145/1180875.1180948>
- Taub, D. (2024). *Programming as a tool for helping students understand and solve quadratic equations* (Publication Number 2024:35) [Licentiate thesis, Karlstads universitet]. DiVA. <https://doi.org/10.59217/vqbe1837>
- The Ministry of Education and Research. (2017). *Nationell digitaliseringsstrategi för skolväsendet*. Retrieved June 16th, 2025, from <https://www.regeringen.se/contentassets/72ff9b9845854d6c8689017999228e53/nationell-digitaliseringsstrategi-for-skolvasendet.pdf>
- The Swedish National Agency for Education. (2021). *Matematik [Ämnesplan]*. Retrieved February 9th, 2026, from <https://www.skolverket.se/undervisning/gymnasieskolan/program-och-amnen-i-gymnasieskolan/hitta-program-amnen-och-kurser-i-gymnasieskolan-gy11/hitta-program-och-amnesplaner-i-gymnasieskolan-gy11#/search/subjects/MAT?version=11#courses-levels>
- The Swedish National Agency for Education. (2022a). *Kommentarmaterial till kursplanen i matematik – grundskolan*. <https://www.skolverket.se/publikationer?id=9790>
- The Swedish National Agency for Education. (2022b). *Ämne - Matematik [Kursplan]*. Retrieved February 9th, 2026, from https://www.skolverket.se/undervisning/grundskolan/kursplaner-for-grundskolan#/subjects/GRGRMATo1?schoolType=GR&typeOfSyllabus=COURSE_SYLLABUS¤tListHeading=Kursplaner%20i%20grundskolan×pan=LATEST
- The Swedish National Agency for Education. (2025). *Kommentar till ämnesplanerna i matematik på gymnasial nivå*. <https://www.skolverket.se/publikationer?id=9917>

- Tossavainen, T., Johansson, C., Juhlin, A., & Wedestig, A. (2024). Programming as a mediator of mathematical thinking: Examples from upper secondary students exploring the definite integral [article]. *LUMAT: International Journal on Math, Science and Technology Education*, 12(3), 78–99. <https://doi.org/10.31129/lumat.12.3.2155>
- Trouche, L. (2004). Managing the complexity of human/machine interactions in computerized learning environments: Guiding students' command process through instrumental orchestrations. *International Journal of Computers for Mathematical Learning*, 9(3), 281–307. <https://doi.org/10.1007/s10758-004-3468-5>
- Trouche, L. (2005a). An instrumental approach to mathematics learning in symbolic calculators environments. In D. Guin, K. Ruthven, & L. Trouche (Eds.), *The didactical challenge of symbolic calculators: Turning a computational device into a mathematical instrument* (pp. 137–162). Springer. https://doi.org/10.1007/0-387-23435-7_7
- Trouche, L. (2005b). Instrumental genesis, individual and social aspects. In D. Guin, K. Ruthven, & L. Trouche (Eds.), *The didactical challenge of symbolic calculators: Turning a computational device into a mathematical instrument* (pp. 197–230). Springer. https://doi.org/10.1007/0-387-23435-7_9
- Trouche, L. (2020). Instrumentalization in mathematics education. In S. Lerman (Ed.), *Encyclopedia of mathematics education* (pp. 392–403). Springer. https://doi.org/10.1007/978-3-030-15789-0_100013
- Turgut, M., & Drijvers, P. (2021). Instrumentation schemes for solving systems of linear equations with dynamic geometry software. *International Journal for Technology in Mathematics Education*, 28(2), 65–80. https://doi.org/10.1564/tme_v28.2.01
- van den Akker, J. (1999). Principles and methods of development research. In J. van den Akker, R. M. Branch, K. Gustafson, N. Nieveen, & T. Plomp (Eds.), *Design approaches and tools in education and training* (pp. 1–14). Springer. https://doi.org/10.1007/978-94-011-4255-7_1
- Vergnaud, G. (1998a). A comprehensive theory of representation for mathematics education. *The Journal of Mathematical Behavior*, 17(2), 167–181. [https://doi.org/10.1016/S0364-0213\(99\)80057-3](https://doi.org/10.1016/S0364-0213(99)80057-3)
- Vergnaud, G. (1998b). Towards a cognitive theory of practice. In A. Sierpinska & J. Kilpatrick (Eds.), *Mathematics education as a research domain: A search for identity* (pp. 227–240). Springer. https://doi.org/10.1007/978-94-011-5194-8_15

- Vergnaud, G. (2009). The theory of conceptual fields. *Human Development*, 52(2), 83–94. <https://doi.org/10.1159/000202727>
- Verillon, P., & Rabardel, P. (1995). Cognition and artifacts: A contribution to the study of thought in relation to instrumented activity. *European Journal of Psychology of Education* 10(1), 77–101. <https://doi.org/10.1007/BF03172796>
- Vinnervik, P. (2022). Implementing programming in school mathematics and technology: Teachers' intrinsic and extrinsic challenges. *International Journal of Technology and Design Education*, 32(1), 213–242. <https://doi.org/10.1007/s10798-020-09602-0>
- Watson, A., & Ohtani, M. (2015). Themes and issues in mathematics education concerning task design: Editorial introduction. In A. Watson & M. Ohtani (Eds.), *Task design in mathematics education* (pp. 3–15). Springer. https://doi.org/10.1007/978-3-319-09629-2_1
- Wermke, W., Olason Rick, S., & Salokangas, M. (2019). Decision-making and control: Perceived autonomy of teachers in Germany and Sweden. *Journal of Curriculum Studies*, 51(3), 306–325. <https://doi.org/10.1080/00220272.2018.1482960>
- Winslow, L. E. (1996). Programming pedagogy - A psychological overview. *ACM SIGCSE Bulletin*, 28(3), 17–22. <https://doi.org/10.1145/234867.234872>
- Yarmish, G., & Kopec, D. (2007). Revisiting novice programmer errors. *ACM SIGCSE Bulletin*, 39(2), 131–137. <https://doi.org/10.1145/1272848.1272896>
- Yazan, B. (2015). Three approaches to case study methods in education: Yin, Merriam, and Stake. *The Qualitative Report*, 20(2), 134–152. <https://doi.org/10.46743/2160-3715/2015.2102>
- Yin, R. K. (2014). *Case study research: Design and methods* (5th ed.). SAGE.
- Zawojewski, J. S., & Lesh, R. (2003). A models and modeling perspective on problem solving. In R. Lesh & H. M. Doerr (Eds.), *Beyond constructivism: Models and modeling perspectives on mathematics problem solving, learning, and teaching* (pp. 317–336). Routledge.

Appendix: The interview guide

(The interview guide in this appendix has been translated from Swedish into English)

Each question is accompanied by a brief description of its purpose, explaining its relevance and providing an estimated time required for response. Depending on the teacher's responses, it may be necessary to pose probing or follow-up questions to provide an opportunity for clarification or further elaboration.

The interview is expected to take approximately 60–75 minutes to complete.

Introduction

A brief overview of the interview's purpose and structure will be provided. This is also an opportunity for the teacher to ask questions.

Information for the teacher

During the interview, I will ask you a series of questions related to how you incorporate programming into your teaching of mathematics. The primary focus will be on course 3c, as this was the course in which I had the opportunity to observe your students.

I will record only the audio of the interview using a voice recorder. Do you consent to this?

Do you have any questions before we begin the interview?

Background

The purpose of these questions is to gain an understanding of the teacher's experience, his approach to incorporating programming into mathematics teaching, and his attitude towards using programming as a mathematical tool in upper-secondary mathematics education.

Question 1

I would like to begin the interview with a few brief background questions:

- a) How many years have you worked as a teacher?*
- b) How many years have you worked at [the name of the school]?*
- c) Which subjects do you teach?*
- d) How would you describe your own programming skills?*

Purpose

To obtain a brief overview of the teacher's professional background and experience.

Intended outcome

Background information that can be used to describe the teacher's profile (e.g., for readers) and to contextualise the teacher's responses during the interview.

Estimated time required

5 minutes

Question 2

In the national curriculum for upper-secondary school mathematics, it has for several years been stated that students following the so-called c-track should be given examples of how programming can be used, or alternatively, use programming themselves as a tool for problem solving, data processing, or applying numerical methods. Could you describe your perspective on the fact that programming is now highlighted as a digital tool to be incorporated into mathematics education (specifically in the c-track)?

Probing question if the teacher's attitude is entirely negative:

Do you see any positive aspects of integrating programming into mathematics education?

Probing question if the teacher's attitude is entirely positive:

Do you see any negative aspects of integrating programming into mathematics education?

Purpose

To gain insight into the teacher's views on the use of programming as a digital tool in upper secondary mathematics education.

Intended outcome

The teacher's response may provide valuable information about how his perception influences the instrumental orchestration of learning activities, including didactic choices such as the selection of programming languages, environments, and the design of mathematical tasks.

Estimated time required

5 minutes

Question 3

Could you briefly describe how you have chosen to integrate programming across the different mathematics courses within the c-track?

Probing question if the teacher's description of the content selection in one of the courses is somewhat brief:

Could you describe the contexts in which students use programming in course x ?

Follow-up question if the teacher's response does not include a description of any overarching structure or progression related to the use of programming across the courses:

Is there an overarching structure or progression in how programming is integrated across the courses, and if so, how would you describe it?

Purpose

To gain an understanding of how the teacher has incorporated programming into various courses.

Intended outcome

The response will provide insight into how the observed students have previously worked with programming. This information can be used to describe the types of instrumental action schemes students may have developed and applied in earlier courses. The answer may also reveal whether there is a coherent progression in how programming is integrated throughout the curriculum.

Estimated time required

5–10 minutes

The Didactical configuration

The questions in this section of the interview relate to what Trouche (2004) and Drijvers et al. (2009) refer to as the teacher's didactical configuration, which constitutes the first element of an instrumental orchestration.

Question 4

Could you describe your reasoning behind the choice of programming language and the digital programming environment in which students work when solving mathematical tasks?

Probing question if the teacher does not clearly justify why the chosen programming language and/or programming environment is more suitable for the intended purpose than other available options:

Could you elaborate on what makes this programming language and/or programming environment more suitable compared to other available options?

Follow-up question if the teacher does not clearly explain how the choice of programming language and/or programming environment may facilitate students' use of programming for mathematical purposes:

Could you elaborate on how you believe the choice of programming language and/or programming environment might facilitate students' use of programming for mathematical purposes?

Purpose

To gain insight into the didactical choices regarding programming language and programming environment.

Intended outcome

Previous research has shown that the choice of programming language and programming environment can influence students' ability to use programming effectively for specific purposes (Mannila et al., 2006). Therefore, it is of interest to understand the teacher's reasoning behind the selection of these artefacts, as such choices may potentially affect students' instrumental genesis.

Estimated time required

5 minutes

Question 5

Could you describe what programming knowledge students need to possess, or need to develop, to use programming as a mathematical tool in course 3c?

Probing question if the teacher's response does not clarify what prior knowledge students are expected to have before the course begins:

Could you clarify what programming knowledge you expect students to have before the course begins?

Follow-up question if the teacher does not provide a justification for why these programming skills are considered especially relevant for using programming as a mathematical tool within the course:

Why do you consider these programming skills to be especially relevant when students are expected to use programming as a mathematical tool within the course?

Purpose

To gain insight into the prior programming knowledge students are expected to possess, as well as the competencies they are expected to develop during the course.

Intended outcome

The response will provide information about the usage schemes the teacher believes students should have already developed, as well as those they are expected to acquire throughout the course. This information, related to the teacher's instrumental orchestration, may be valuable when studying students' instrumental genesis.

Estimated time required

5 minutes

Question 6

How did you reason when selecting the types of mathematical tasks that students are expected to solve using programming?

Probing question if the teacher does not provide a general description of the types of tasks he considers particularly well-suited to be solved using programming:

Are there any types of mathematical tasks at the upper secondary level that you believe are particularly well-suited to be solved using programming?

Follow-up question if the teacher's response does not link the types of tasks to specific mathematical abilities or skills:

Are there any specific mathematical skills or mathematical abilities that you believe the use of programming has especially strong potential to develop?

Purpose

To gain insight into the teacher's considerations when selecting mathematical tasks.

Intended outcome

By understanding the teacher's rationale for task selection, it may become possible to describe, based on the teacher's intentions, the types of instrumented action schemes he aims for students to develop or apply.

Estimated time required

5 minutes

The Exploitation mode

The questions in this part of the interview are related to what Trouche (2004) and Drijvers et al. (2009) refer to as the exploitation mode, which constitutes the second element of an instrumental orchestration.

Question 7

On several occasions, you provide students with partially completed code that they can reuse to solve tasks. How do you view the balance between allowing students to write code independently and providing them with pre-written code to modify, such as code skeletons?

Follow-up question if the teacher argues that the primary focus should be on the mathematical aspects:

Is there anything that may be lost, in relation to the use of programming as a mathematical tool, when students do not create the code from scratch?

Follow-up question if nothing in the teacher's initial response indicates that there are any time-related advantages to working with pre-written code:

Are there any time-related advantages to working with pre-written code or code skeletons?

Purpose

To gain insight into the teacher's choices regarding students' creation of program code.

Intended outcome

The teacher's response may reveal his perspective on the balance between enabling students to learn how to create programs that solve mathematical problems and simultaneously develop mathematical abilities and skills. This can provide insight into the types of schemes or scheme components (both mathematical and computational) that the teacher primarily intends for students to develop or utilise.

Estimated time required

5 minutes

Question 8

During the first lesson in which programming was used in the course Mathematics 3c during the autumn term, students were asked to use programming to determine various limits. Could you describe your intentions with these tasks?

Probing question if the teacher's response does not clarify which mathematical skills or forms of mathematical understanding the students are expected to develop when working on the tasks:

What mathematical skills or forms of mathematical understanding are the students expected to develop when working on the tasks?

Probing question if the teacher's response does not clarify which programming skills or forms of programming understanding the students are expected to develop when working on the tasks:

Are there any programming skills or forms of programming understanding that the students are expected to develop when working on the tasks?

Purpose

To gain insight into the didactical choices the teacher made in designing and planning the implementation of the tasks.

Intended outcome

By exploring the teacher's didactical decisions, it becomes possible to identify the mathematical and/or computational schemes or scheme components that the teacher intends for students to develop and/or apply. This, in turn, provides an opportunity to describe the instrumental genesis that the teacher's instrumental orchestration aims to support.

Estimated time required

5 minutes

Question 9

During the second lesson in which programming was used, students were asked to use programming to determine the value of derivatives. Could you describe your intentions with these tasks?

Probing question if the teacher's response does not clarify which mathematical skills or forms of mathematical understanding the students are expected to develop when working on the tasks:

What mathematical skills or forms of mathematical understanding are the students expected to develop when working on the tasks?

Probing question if the teacher's response does not clarify which programming skills or forms of programming understanding the students are expected to develop when working on the tasks:

Are there any programming skills or forms of programming understanding that the students are expected to develop when working on the tasks?

Purpose

To gain insight into the didactical choices the teacher made in designing and planning the implementation of the tasks.

Intended outcome

By exploring the teacher's didactical decisions, it becomes possible to identify the mathematical and/or computational schemes or scheme components that the teacher intends for students to develop and/or apply. This, in turn, provides an opportunity to describe the instrumental genesis that the teacher's instrumental orchestration aims to support.

Estimated time required

5 minutes

Question 10

During the third lesson in which programming was used, students were asked to use programming to determine the value of derivatives of exponential functions with base e . Could you describe your intentions with these tasks?

Probing question if the teacher's response does not clarify which mathematical skills or forms of mathematical understanding the students are expected to develop when working on the tasks:

What mathematical skills or forms of mathematical understanding are the students expected to develop when working on the tasks?

Probing question if the teacher's response does not clarify which programming skills or forms of programming understanding the students are expected to develop when working on the tasks:

Are there any programming skills or forms of programming understanding that the students are expected to develop when working on the tasks?

Purpose

To gain insight into the didactical choices the teacher made in designing and planning the implementation of the tasks.

Intended outcome

By exploring the teacher's didactical decisions, it becomes possible to identify the mathematical and/or computational schemes or scheme components that the teacher intends for students to develop and/or apply. This, in turn, provides an opportunity to describe the instrumental genesis that the teacher's instrumental orchestration aims to support.

Estimated time required

5 minutes

Question 11

During the fourth lesson in which programming was used, students were asked to use programming to find local minimums using the gradient descent method. Could you describe your intentions with these tasks?

Probing question if the teacher's response does not clarify which mathematical skills or forms of mathematical understanding the students are expected to develop when working on the tasks:

What mathematical skills or forms of mathematical understanding are the students expected to develop when working on the tasks?

Probing question if the teacher's response does not clarify which programming skills or forms of programming understanding the students are expected to develop when working on the tasks:

Are there any programming skills or forms of programming understanding that the students are expected to develop when working on the tasks?

Purpose

To gain insight into the didactical choices the teacher made in designing and planning the implementation of the tasks.

Intended outcome

By exploring the teacher's didactical decisions, it becomes possible to identify the mathematical and/or computational schemes or scheme components that the teacher intends for students to develop and/or apply. This, in turn, provides an opportunity to describe the instrumental genesis that the teacher's instrumental orchestration aims to support.

Estimated time required

5 minutes

Question 12

During the fifth lesson in which programming was used in the course Mathematics 3c last autumn, students were asked to use programming to numerically determine the values of integrals. Could you describe your intentions with these tasks?

Probing question if the teacher's response does not clarify which mathematical skills or forms of mathematical understanding the students are expected to develop when working on the tasks:

What mathematical skills or forms of mathematical understanding are the students expected to develop when working on the tasks?

Probing question if the teacher's response does not clarify which programming skills or forms of programming understanding the students are expected to develop when working on the tasks:

Are there any programming skills or forms of programming understanding that the students are expected to develop when working on the tasks?

Purpose

To gain insight into the didactical choices the teacher made in designing and planning the implementation of the tasks.

Intended outcome

By exploring the teacher's didactical decisions, it becomes possible to identify the mathematical and/or computational schemes or scheme components that the teacher intends for students to develop and/or apply. This, in turn, provides an opportunity to describe the instrumental genesis that the teacher's instrumental orchestration aims to support.

Estimated time required

5 minutes

Conclusion

A brief concluding and general description of how the results of the interview will be used, along with an opportunity for the teacher to supplement or clarify any of his previous responses.

Information for the teacher

That was my final question. Thank you very much for taking the time to participate in this interview.

The data I have generated through this interview will hopefully allow me to compare the outcomes of how well the students were able to use programming as a mathematical tool with the intentions you had when designing the lessons. I am therefore very grateful that you took the time to participate.

Before we conclude the interview, is there anything you would like to add, or do you have any questions or reflections?

With that, we conclude the interview.



Incorporating programming into mathematics education

This thesis comprises two studies investigating upper-secondary students' use of programming as a mathematical tool. It aims to examine both the intertwined relationship between students' use of programming and their mathematical understanding, and how the design of learning activities can support the incorporation of programming into mathematics education. The findings indicate that analyses of how students use programming to support their mathematical understanding must also consider how their grasp of programming concepts shapes their mathematical use of the tool, given that programming is not designed as a mathematical or educational tool. The findings further suggest that using programming as a mathematical problem-solving tool, particularly when students construct their own algorithms, places significant demands on those with limited programming experience. Conversely, providing pre-designed algorithms for numerical computations, intended to ease students' use of programming, may restrict the development of deeper mathematical understanding. A practical contribution of the thesis is that mathematics teachers must balance scaffolding students' use of programming with allowing them autonomy to engage with the tool in ways that support their mathematical understanding.

ISBN 978-91-7867-677-4 (print)

ISBN 978-91-7867-678-1 (pdf)

ISSN 1403-8099

DOCTORAL THESIS | Karlstad University Studies | 2026:16
