Heikkilä, Mia; Mannila, Linda

Debugging in programming as a multimodal practice in early childhood education settings

Published in:
Multimodal Technologies and Interaction

DOI:
10.3390/mti2030042

Published: 01/07/2018

Document Version
Publisher's PDF, also known as Version of record

Multimodal Technologies and Interaction

MDPI

Article

Debugging in Programming as a Multimodal Practice in Early Childhood Education Settings

Mia Heikkilä [1,*] and Linda Mannila [2]

[1] Faculty of education and welfare studies, Åbo Akademi University, 65170 Vasa, Finland
[2] Learning Environments Research Group, Aalto University, 02110 Espoo, Finland; linda@digismart.fi
* Correspondence: mia.heikkila@abo.fi

Received: 21 May 2018; Accepted: 4 July 2018; Published: 13 July 2018

Abstract: The aim of this article is to broadly elaborate on how programming can be understood as a new teaching scope in preschools, focusing specifically on debugging as one of the phases involved in learning to program. The research question How can debugging as part of teaching and learning programming be understood as multimodal learning? has guided the analysis and the presentation of the data. In this study, and its analysis process, we have combined a multimodal understanding of teaching and learning practices with understandings of programming and how it is practiced. Consequently, the multidisciplinary approach in this study, combining theories from social sciences with theories and concepts from computer science, is central throughout the article. This is therefore also a creative, explorative process as there are no clear norms to follow when conducting multidisciplinary analyses. The data consist of video recordings of teaching sessions with children and a teacher engaged in programming activities. The video material was recorded in a preschool setting during the school year 2017–2018 and consists of 25 sessions of programming activities with children, who were four or five years old. The results show how debugging in early childhood education is a multimodal activity socially established by use of speech, pointing and gaze. Our findings also indicate that artefacts are central to learning debugging, and a term 'instructional artefacts' is therefore added. Finally, the material shows how basic programming concepts and principles can be explored with young children.

Keywords: programming; early childhood education; multimodality; instructional artefacts; debugging

1. Introduction

As the society is transformed by the digitalization, there is a need to understand the underlying basics that are driving this change, as well as the possibilities and challenges involved. As a consequence, countries all over the world are adding digital competence, computer science or computing to their national curricula in order to give children this opportunity. Other concepts and terms are also used to describe the skills and knowledge needed in a digitalized society, such as computational thinking, ICT (information and communication technology) skills, the Swedish term MIK (media and information literacy) and digital literacy. The concept that has won ground in several countries during recent years is digital competence, which is commonly considered important for all citizens in our current and future society. The European Commission defines digital competence in the DIGCOMP framework [1], which has five main competence areas (information and data literacy, communication and collaboration, digital content production, security, problem solving). These areas, in turn, consist of 3–6 more detailed competences.

Digital competence can hence be seen as a broad and versatile area, which many countries, including Sweden and Finland, have now included in their national curricula for primary and lower secondary schools. For instance, in Sweden, digital competence has been added to the curriculum

for all educational levels—starting with early childhood education (ECE), throughout comprehensive education all the way up to upper secondary schools (grades 10–12). The Swedish National Agency for Education (in Swedish 'Skolverket') describes digital competence based on four aspects: (1) understanding how the digitalization affects our society, (2) being able to use and understand digital tools and media, (3) having a critical and responsible attitude towards technology, and (4) being able to solve problems and implement ideas with technology [2]. When describing programming, which is part of digital competence, Skolverket points out that programming is more than merely producing code. Rather, they argue, that programming should be seen as a problem-solving process, which can also be used to create new things, simulate phenomena as well as for controlling devices and systems. In addition, they point out that programming also has democratic dimensions. Taking such a broad view on programming, they conclude that programming should be seen as part of all four aspects of digital competence.

The Swedish government has also decided on a national digitalization strategy for the Swedish school system. The strategy states that all children and pupils at all school levels should be given the opportunity to develop an adequate level of digital competence. The government also emphasizes the need for studying the impact of digitalization on teaching and learning, in particular, as a means to support the development of school practices. This is a crucial aspect, as Swedish school legislation requires that all teaching is based on scientific grounds and proven experience. This is also one of the reasons why it is important to conduct research on new issues that arise in ECE.

An increased focus on digital competence, including the introduction of programming, in early ages is, however, still a rather new phenomenon. Consequently, there is not much previous research neither in Sweden nor internationally. For instance, we have not been able to find any study similar to the one we present here, and a review of the international literature in general on programming in ECE revealed only a few studies. Still, many preschools add programming to their everyday practice, despite the lack of supportive research and previous comprehensive experience. The rationale for doing so is multi-facetted. On the one hand, programming can be seen as a didactic tool for developing digital skills and computational thinking in different ways in preschool. Programming in preschool can also be a means of finding new ways for learning, motivating the children and increasing their interest and curiosity [3]. Children's learning and development form the basis on which planning, implementation and evaluation is designed and carried out.

In this article, we present the results from a study aiming at contributing to filling the research gap on programming in early ages. In particular, this study contributes to the knowledge base on aspects that need to be considered when planning and carrying out programming related teaching activities in a preschool setting. We have analysed video material collected during programming activities in a group of four- and five-year-olds in a Swedish ECE setting during the school year 2017/2018. Programming as an activity includes different phases and can be seen as a complex problem-solving process. In order to conduct a deeper analysis of what goes on during children's programming activities, it is therefore worthwhile to focus on a specific part of the programming process.

We have chosen to focus on debugging and how it unfolds and is carried out in an ECE context, as "the desire to correct the parts that is [*sic*] keeping their program from working correctly creates an ideal situation to develop a broad set of interconnected skills such as problem articulation, team working, persistency, and other key abilities" [4]. Debugging can hence be seen as a problem-solving activity, which involves exploration, observation, communication and reflection. These are all important elements at preschool level according to the Swedish curriculum: "Children learn through play, social interaction, exploration and creation, but also through observation, communication and reflection" [5] (p. 5). Being able to debug program code as well as to read and trace existing programs have also received particular attention in research on novice programming (e.g., [6,7]), and are considered crucial skills for learning to program.

The aim of the study is to broadly elaborate on how learning programming can be understood as a new activity in preschools. Our main research question (RQ) is the following:

How can debugging as part of teaching and learning programming be understood as multimodal learning?

We begin by discussing previous research and the theoretical perspectives related to teaching and learning programming in early ages. Next, we present the methodology used in our study, after which we present and discuss our results. The paper is concluded with some final remarks.

2. Programming in Early Ages

Introducing programming at an early age is not new (see e.g., [8,9] for examples of effort to teach programming in the 1980s), but one can argue that programming at this level lacks theoretical concepts connected to learning. Papert and colleagues introduced the LOGO programming language already in 1967 as a tool for children to learn to program. Papert [10] argued that technology should not be used to process children, but rather that children should learn to manipulate, expand and use technology in projects, thereby learning to understand and control their world. Papert's programming language LOGO is often associated with a turtle that was controlled by simple command, and many of the programming tools and environments developed for young learners today are based on the same principles (e.g., simple robots such as Bee-Bot and Blue-Bot, apps such as Kodable and Lightbot).

In preschools, the goal is not to teach programming as an intrinsic or separate subject. At preschool level the aim is also to focus on the other abilities that you train while engaging in programming activities, such as algorithms, logical thinking and debugging, as well as on the opportunities to use programming as a tool for being creative and making things as programming—by nature—is a creative activity [11]. These abilities are often collected under the umbrella term *computational thinking* (CT), which was introduced by Papert [12] almost 40 years ago but has received particular attention during the last 10 years, following an influential article by Wing [13]. Wing emphasizes the importance of learning strategies and skills, which help us use computers for what they are good at, so that we, humans, can focus on what we are good at. While CT is often framed around a set of concepts, research has shown that this is not sufficient to represent student's learning: rather CT is also about practices (experimenting and iterating: testing and debugging; reusing and remixing, and; abstracting and modularizing) as well as perspectives (expressing, connecting, and questioning) [14].

As mentioned above, the research base on programming in early ages is still rather limited. International studies on programming at preschool level emphasize the knowledge areas that children develop when engaging in activities related to robotics and programming. In a meta study, Toh et al. [15] found that this type of activities develop children's cognitive, conceptual, linguistic and social skills by focusing on, for instance, problem solving, logical thinking, collaboration and a structured way of working and presentation. Bers et al. [16] found that children from the age of four can take in programming instruction and develop their problem-solving skills related to computational thinking, coding and robotics.

While programming per se can be seen as an abstract activity, physical artefacts such as robots make the activity more concrete while simultaneously also encouraging collaboration around the programming task at hand. Bateman, Carr and Gunn [17] discuss how objects in children's learning environments are crucial for children's learning and how technological objects work as 'physical props' supporting learning processes (cf. [18]). Levy and Mioduser [19] describe how children explore the possibilities of the robot in a playful and curious manner. They also show how this, for instance, resulted in children making plans and predictable actions in order to make the robots act in a desired way. Sullivan et al. [20] show how children can design, build and program robots after a limited time of instruction. This could imply that children are susceptible to more than merely very basic instruction in programming. Sullivan et al. [20] go on to discussing how these results could aid in developing programming instruction, for instance by integrating it with mathematics and using programming for language development.

In another study, Kazakoff et al. [21] show how sequencing, which can be considered an important skill in both mathematics and early reading comprehension, can be developed through programming.

They also discuss how the sequencing skills needed when programming can be supported by children's ability to think in terms of sequences as a result of their experience from stories, which commonly build on a sequence of events. The corresponding sequential way of working is one of the fundamental concepts in programming, together with conditionals (making it possible for a program to do different things based on the current situation) and repetitions (making it possible to repeat a part of the program several times).

In addition to the focus on concrete and physical artefacts, research on programming for younger children also highlight creativity and play as important components. The design rationale behind the popular programming environment ScratchJR has been to create an easily accessible environment, where young children can, for instance, create their own digital stories [22]. Many programming applications build on gamification, that is, features used in games aiming at, for instance, increasing the motivation to use the app [23]. This also holds for the programming apps developed for or suitable for young children.

Digital tools are a natural part of children's everyday life, both at home and at preschool. As always when choosing an activity, tool or teaching approach, it is important to have a clear focus and goal. For instance, Palmér and van Bommel [24] point out that it takes thorough planning for the teaching activities to focus on the content at hand (in their case mathematics) and not technology per se. Cejka et al. [25] show the importance of teachers receiving appropriate and sufficient professional development in order to be able to develop content and form for introducing programming to younger children.

3. Theoretical Framework

When studying new and contemporary phenomena in school practice, there is often a need for combining several theoretical perspectives in order to grasp a feature, as the use of several perspectives can provide a deeper understanding of the partly unknown practice. In this study, we focus on programming in ECE, which is a practice that consists of aspects of both learning, meaning making, technology, digitalization and design. The study is hence interdisciplinary to its nature, and theoretical perspectives from these different domains are needed in order to be able to describe and analyse programming in this specific context in as good a way as possible. When knowledge development is in an initial phase, it is typically the case that several theoretical inputs are needed to establish a knowledge area.

In this study, and its analysis process, we have combined a multimodal understanding of teaching and learning practices with understandings of programming and how it is practiced. The research is interdisciplinary as it combines theories from social sciences with theories and concepts from computer science; this is central throughout the article. The study is therefore also a creative, explorative process as there are no clear norms to follow when conducting multimodal analyses.

Multimodality is here seen as a relevant analytical tool and perspective in order to deeper understand the interaction and communication going on in the sequences of teaching and learning programming [26,27]. Multimodal perspectives can contribute to analytically describing how teaching and learning as a social phenomenon is established as meaning making processes. This can be done by creating modal complexes where meaning making in a social semiotic understanding implies learning [26,28–30]. Modal complexes are never arbitrary, but vary extensively in social practice and need to be empirically studied in relation to different aspects of social life, such as teaching and learning. Empirical studies such as this one, of how modal complexes are constituted, can present the complexity of social life, and also the complexity of ECE and learning—not putting verbal or written language in the center of how communication is realized.

Pahl [31] argues that a multimodal lens on children's literacy can reveal and open up new and widened ways to look at children's lives. A multimodal lens can even result in children's activities being considered more relevant and accurate, compared to if analysis focused on verbal or written language only. Pahl's [31] research further emphasizes how language and multimodal creation of texts

become intertwined communicational activities for children. This is also brought up by Stein [32] in her studies on children's creation of multimodal practices. Both Stein [32] and Pahl [31,33] argue that multimodal analysis can create a deeper understanding of children's meaning making.

Within this approach children's meaning making is visible through communication and interaction, and how children use modes within communication and interaction. The focus in this study is on understanding how modes shape communication and vice versa within teaching and learning programming. Modal complexes that are created in communication are here related to a given content, that is, programming. The modal complexes are, as already mentioned, never arbitrary and need to evolve in different contexts where different content is in focus. This process involves transcribing and interpreting what is happening in an ECE setting around programming. Language proficiency through verbalization and written language use is not seen as the center of these processes. Instead the modes used for the task at hand become an empirical question to be answered: what modes are central to children?

We find it particularly suitable to use a multimodal perspective of learning and meaning making when the aim is to understand teaching and learning programming in ECE, since speech and written language is not necessarily the focus of young children's communication, nor in programming. Children of this age might not have learnt the norms around spoken and written language, in line with previous studies [34], which show how spoken and written languages are not the central communication means used in preschool.

By applying a multimodal perspective on programming in an ECE setting and focusing on a given aspect of the programming practice (debugging), we can gain insight into how children's processes around debugging are constituted and what multimodal processes take place when they are engaged in these activities. Multimodality is in this sense both a means and a result, but the result is never pre-given.

4. Study Settings

4.1. Study Context and Design

The study was conducted in the context of a Swedish preschool, that is, ECE for ages 1–6. In order to address our research question, we have analyzed video recordings from teaching sessions at a preschool unit in a mid-size Swedish municipality. We chose this particular unit, as it has two teachers, who are very interested in teaching programming. This can naturally also be argued to lead to potential bias, but since programming in early ages is still not very common and lack a research foundation, we do not consider the teacher selection a problem. In order to get reliable data, we needed to find engaged teachers that were committed to work systematically over a longer period of time.

The research has been designed as a case study. According to Jensen and Sandstorm [35], case studies should develop and generalize theories, resulting in a so-called *analytical generalization*. They argue that "an analytical generalization is based on the ability of one or more concepts to understand or explain events (or activities, processes) in different contexts" (p. 64). In this research, the concepts central to understanding the events and activities found were programming and debugging at preschool level. Jensen and Sandstrom [35] also point out that complex phenomena that are investigated should be contemporary and understood through concrete events. We see programming as a contemporary phenomenon and we try to deepen the understanding of how programming is a social practice by analyzing concrete events. Therefore, the case study strategy has been appropriate in relation to the aims and research question in this study and we argue that the results can be analytically generalizable.

When comparing our study with previous research, the studies by Levy and Mouser [19] and Sullivan et al. [20] share similar designs. These studies can hence be used as a basis for discussing our results and our study can also to some extent be seen as a replication of their studies. These studies have a participant oriented design focusing on how preschool children interact with and are susceptible

to different types of instruction in programming (mostly through robots) both considering content and form.

4.2. Data Collection

The data consist of video recordings of teaching sessions with children and a teacher engaged in programming activities. The activities are both analogue (also called unplugged activities), such as programming using verbal instructions or cards, and tasks that involved some digital equipment, such as small robots (Bluebots) and iPads.

The video material was recorded in the preschool during the school year 2017/2018. The video material consists of 25 programming sessions with preschool children, who were four or five years old. Since the preschool group had around 18 children, and all were not always present, there are some differences in who participates in the recordings. The children participate in the video sequences in smaller groups, ranging from pairs to groups of eight children. All in all, the video material is about 30 h long. One of the authors of this publication took care of recording the sequences at the preschool once a month, whereas the preschool teachers, guided by the researcher, recorded the other sequences. The collaboration between the researcher and the teachers was smooth. The teachers were instructed to (1) record the videos as close up to the children as possible, (2) try to get as many children as possible in the recording and (3) make sure that speech would be audible.

4.3. Analysis Process

The analysis was carried out in several steps. First, all video recordings were watched several times, with our research question in mind, in order to get an idea of the width of the data. The process of watching, taking notes, and then watching certain parts again followed, and can be seen as a process quite similar to reading interview transcripts when using interview as research method. A video recording can be seen as a form of text that shows us as researchers different aspects of a social practice. The notes constituted codes for parts of the data related to the content of the sessions and the debugging practice involved.

After watching the video recordings several times, tentative patterns were shaped as ideas of how to analytically describe both the teaching sessions in general, and the debugging situations found in these sessions in particular. We discussed the patterns found and ended up with two main themes, which make it possible to categorize and understand children's debugging activities. With these themes in mind we watched some of the video recordings once again, in order to reformulate or reshape the themes if necessary. The clearest pattern that we saw in the video recordings was how debugging was carried out differently in analogue and digital teaching sessions. One notable difference was that debugging was done more clearly and frequently when children were engaged in digital programming compared to analogue programming. The two main themes, exhibiting the largest differences in how debugging was carried out were therefore *analogue* and *digital debugging practices* and within this we elaborate on *teacher initiated/driven* debugging and *child initiated/driven* debugging. We will present and discuss these further in the results part. In order to make it easier to present and discuss our findings, we chose to transcribe one digital and one analogue session. In the first one, the teacher and children use paper cards to program a dressing robot, while the second session involves programming a floor robot (Bluebot).

When we had rewatched the material, we selected the sessions that were to be transcribed in more detail. This was done by focusing on the modes that we had found were most active in the sessions during our previous analysis. We interpreted the most recurring modes as the most active ones for children to shape meaning in this learning context. These modes were *gaze, body posture, pointing and verbal language*. Within this we noticed a very active use of *artefacts*, and we have chosen to consider the artefact as a particularly central affordance [28,29] in the constitution of the teaching sessions. Out of the many sessions that could have been selected, we chose the ones where the teacher's and the children's speech, faces, use of modes and artefacts were clear in the recordings. The purpose of the

transcripts is to exemplify how debugging is constituted multimodally and how debugging in teaching and learning programming can be considered one of the core processes of teaching and learning programming in ECE settings [36]. The transcripts and videos were rewatched by both researchers, who then discussed the material and the interpretations in order to make sure that no aspects were missed as well as to agree on potential uncertain excerpts.

The selected sessions were examples that could function as representations of the most common theme/category that we found in the material with regards to debugging. One video recording can, however, never represent another, so with that in mind, the transcription took place.

4.4. Transcription Process

The transcription was conducted with a close focus on the research question. That is, we have not followed a certain transcript convention, but rather decided to create the transcripts to meet the needs of the aim and the research question of this study. The symbol "[" represents overlapping use of modes.

According to Bucholz [37], the transcript should "respond to the contextual conditions of the transcription process itself, including the transcriber's own expectations and beliefs" pointing out how the context of transcription is central to the transcription itself. Bucholtz continues saying that there is no neutral transcription, as it always occurs in a context and is framed by a research interest, while also always being political. As a researcher, transcription is easily considered neutral because it is merely a matter of making a "copy" of what happens, from spoken or communicated language, to the writing. This is, however, an incorrect assumption according to Bucholtz, as every decision in an analysis process affects the outcome. The transcript always has evidence that choices have been made regarding interpretations of what is being transcribed, Bucholtz says. The transcript is always an interpretation, a translation—not a "copy".

Transcriptions are "partial representations" [38] (p. 173). These representations also make it possible to interpret and highlight certain meanings to a higher extent, while others are left behind. This is based on choices and decisions made in the transcription process. Green, Franquiz and Dixon (1997) also emphasize how linearly transcribed transcripts risk making reality more logical and reasonable than it actually is.

In a research review compiled by Davidson [39], she shows how transcription is an overlooked part of qualitative research and argues that this can have a negative impact on reliability and validity. Oliver, Serovich & Mason [40] highlight the same concern and say that the dilemma of seeing transcripts as something behind the stage is that they are not seen as an important part of qualitative analyses. Witcher [41] points out how the transcription process is a process of negotiation on many different levels. The listening may affect small and large decisions taken in the analysis phase, but these decisions should potentially have been taken already during the transcription phase.

In this study, transcripts are not seen as neutral descriptions of the data, but as means of bringing forward certain aspects of programming (debugging) and how it is taught and learnt in ECE. The transcriptions are important parts of the analysis, since they make it possible to discuss aspects at a micro level. In a new area where there is a lack of "established knowledge" we argue that it is important to be able to conduct analysis at a detailed level, close to a form of reality, in order to shape a common understanding of a phenomenon. The transcripts help us make analytical descriptions and analytical generalizations as a way of shaping such a common understanding. This is also close to a multimodal understanding of meaning making, which is the basis for our study. This inevitably means that an analytical description is part of the analysis of social practice, and transcripts need to be created in relation to the research questions in order to carry out the analysis [42].

4.5. Ethical Considerations

The study follows the Swedish Research Council's principles of research ethics. All research that involves people in Sweden is always based on the "individual protection requirement" [43]

(p. 5), which means that research that in any way leads to mental or physical damage, violations or humiliation must not be conducted. This requirement also includes the right to privacy without transparency in the personal circumstances of the individual.

The Swedish Research Council [44] has four main requirements concerning research ethics to ensure that research is conducted in line with these principles: information requirement, consent requirement, confidentiality requirement and the anonymity requirement. In this study, we have fulfilled these requirements by informing all participants (the municipality, teachers, legal guardians and children) about the study, what it is about, and how it would be conducted. All teachers have given their consent to participate and to be video-recorded. All legal guardians have given their consent to the video recording, and they have also been informed about the possibility to cancel their child's participation at any time without explanation. The written consent documents are stored at the university. Researchers on site also ask children at the time of recording. Children are also encouraged to give their consent during video-recording by indicating if they do not want a session to be recorded, or if they want it to be stopped (cf. [44]). They have been instructed to do this either use verbal language, a hand gesture indicating "stop" or to move out of the picture or room without notice. Children's signals were readily interpreted, also when watching the video-recordings. Children can in different ways show that they do not want to be included, and as researchers we are aware of this throughout the process. The confidentiality requirement and the anonymity requirement refer to confidentiality and anonymity regarding the participation in the research study. All material has been anonymized during the analysis and publication of the research results. The researchers do not have any list of the children, preschool teachers or legal guardians, which is one way to guarantee anonymity in the material. The collected material is only used for research purposes. All of these considerations have been followed rigorously and with accuracy when conducting this study.

5. Results

The results are presented in two main sections. The first section is a general analytical description of the results from the perspective of debugging as part of teaching and learning programming. The second section presents the result of an analysis of how debugging is carried out in analogue and digital programming sessions. In this context, we analyze how debugging can be understood as a multimodally constituted learning *process*.

5.1. Children Learning Programming—General Observations on Debugging Practice

The analysis of the video recordings shows how programming as something to learn, creates great interest amongst the children, illustrated by children's patience and willingness to follow the teacher's instructions and the content covered. Even though the children do not formulate what they do as learning, in multimodal terms programming activities can be said to create meaning making instances, with intense modal activity. The children do this through a continuation of use of modes related to the content of the sequence and related to each other, wanting to engage both with the content and their peers. One can understand this as a way of trying to include each other in the sequence.

The material also shows that pointing, gaze direction, body posture and verbal speech are the most central modes used in the communication in the videos. The programming content of the sequences is often centered around one or several artefacts, which are used to illustrate or support teaching and/or learning. Throughout the recordings, the modes mentioned are related to an artefact that is being programmed. In multimodal terms the artefacts become affordances for the creation of modes and of meaning making. In the video material of analogue programming sessions, the artefacts used are, for instance, Lego blocks, magnets, musical instruments, cards with given pictures etcetera. The modes used by children and teacher are centered around the artefact.

Some of these artefacts have an *instructional* purpose, that is, the artefacts are used to build the program. In programming terms these are called instructions, which are put together step-by-step in order to create a program that solves a given problem or task. For instance, arrows can be used

to indicate directions while cards with pieces of clothing can indicate what garment to put on. The instructional artefacts can be used to program another person, but also other artefacts. For instance, arrow cards can be used to program a toy figure's way forward in a maze and clothing cards can be used to program the way in which to dress a doll. Here the toy and doll are also artefacts, but they are *programmable*, not instructional. Instructional artefacts are a way of formalizing actions and hence make up simple programming languages. The symbols used are concrete representations of what is represented as, for instance, abstract symbols or text instructions in regular programming languages.

5.2. Debugging in Practice

The video material shows how debugging is carried out in a playful manner with laughter and play. In addition, we have found several ways in which to more systematically describe how debugging is carried out. Firstly, debugging is sometimes hard to code in a video recording due to the complexity of debugging as a phenomenon. In the first video sessions, debugging is mostly *initiated* by the teacher using verbal speech. Later on, when the children have learnt some programming practice, they are also more likely to initiate debugging themselves by pointing out problems in the solution or the proposed solution. In the excerpts below we see how debugging can be *initiated* and *driven* by both the teacher and the children.

Debugging is also done both *directly* or *indirectly*. An example of direct debugging is, for instance, when somebody states that there is an error ("oh, something went wrong") or when a programming related concept such as a bug is introduced ("what we have here now is called a bug"). Indirect debugging, on the other hand, happens for instance as the teacher uses a question in order to get the children to realize that there is a problem in the solution. The teacher can, for example, ask "what should we do instead to get there?" with 'instead' being the signal that tells the children that there is an error in the program. Other ways of indirectly notifying children of an error is, for instance, pointing, raising an eyebrow or by using a body posture that directs children's attention in a certain direction. This indirect way of trying to get the children to see or recognize a problem is also a strategy that the teachers use to teach children.

Our analysis indicates that debugging should be viewed as a process when learning to program, going from teacher initiation to children's discovery of errors. In this process, being able to recognize, understand and correct the error is important.

In the following, we describe in detail two particular cases from the video recordings. The first case illustrates how debugging is carried out in an analogue programming activity, whereas the second one illustrates the same in a digital activity using a robot. The analysis is based on the research question, that is, how debugging as part of teaching and learning programming can be understood as multimodal learning. As solutions are the results of the debugging process, we chose to also include these when describing the case studies.

Case 1: Dressing a robot using analogue programming

Seven children are sitting outdoors together with their preschool teacher Tove. Tove and the children sit on seat cushions in a ring formation on asphalt (See Figure 1). Tove has a set of cards illustrating different pieces of clothing, and she starts showing these cards, one at a time, to the children. The children collectively call out what the card shows, by shouting for instance shoes, pants, hat, and so on. Tove then goes on to tell the children that they are to develop a program that will show a robot how to dress when going outdoors. They will test the program by following the instructions themselves in order to see whether the program makes them put the clothes on in the right order. Tove uses the cards to show the children one piece of clothing at a time. Which one should they put on first? The children start building a sequence by making suggestions on what pieces of clothing to put on next.

The video is 12 min long. When identifying places where debugging took place in the video we found three sequences (about 4 min in total), and we have chosen to transcribe and present parts of these three specific sequences.

Sequence 1: The children have decided to first put on the hat and then the t-shirt, and Tove has placed the respective cards in the corresponding order in the middle of the ring. She now picks up the next card, which has underpants on it. At this point one child, Anders, objects to the programming done so far.

Figure 1. An analogue programming session using paper cards.

Anders: We [should put on the underpants first.
[Tove bends over as she prepares to lay down a new card]
[Anders starts to bend towards the middle of the ring]

Tove: [sits back up
[Should we take the underpants first?

Anders: Mmm.

Tove: Why do you think so?

Anders: When I'm [at home, I always take on the underpants first.
[continues to bend towards the middle of the ring

When Anders realizes the mistake he changes his body posture in order to get closer to the program that he feels needs to be corrected, that is, the card sequence. This particular debugging excerpt is *child-initated* and *child-driven*; that is, one of the children reacts to something being wrong in the program, explains why he thinks this is wrong and at the same time offers a solution. They then go on to discuss this a bit further and decide to put on the underpants next, since they have already decided on some steps in the program, and putting on the underpants next works fine with this order. The debugging process initiated by Anders hence led to a *negotiation*, where the participants jointly decided what to do.

Sequence 2: The children have now decided on a sequence of four cards: hat—t-shirt—underpants—trousers. They now suggest that the next card should have panties on it.

Tove: If we start the robot and he gets dressed—will he get all the clothes?

Children: Yeeees!

Tove: Will he?

Children: Yeeees!

Tove then simulates the programming, by first point to the current card sequence on the ground and then stepping through the program by making gestures of first putting on the hat, then the t-shirt, underpants and trousers. The children repeat the gestures at their own pace. After simulating the execution of the program so far, Tove looks skeptically at the children and asks "Can we put on the panties now?" The children shout "yeeees . . . noooo. Yeeees!".

Tove: So you usually have [hat, t-shirt, underpants] first, then the trousers and then
[the panties?
[claps her thighs

Child 1: I usually put on my panties first.

Tove: Aaa, [exactly
[makes a gesture towards the child

Child 2: I usually put on my underpants first.

Tove: I think we'll do like [this
[bends over, places the panties next to the underpants
Because this depends on whether you are a girl and have panties, or if you are a boy and wear underpants. So, we can have them in the same place, then we will take the trousers.

This debugging sequence is both *teacher-initiated* and *teacher-driven*. The children propose a next step that does not make sense to the teacher Tove, so she reacts and starts to ask questions, which may get the children to see the problem in their suggestion. When they still stick to the suggested next step (most likely because they think it would be a fun thing to do), Tove makes the error visible by executing the program and thus showing the problem in a very explicit and concrete manner. At this point, a child starts taking part in the debugging process, as she notes that Tove puts on the panties first. As another child responds that he puts on the underpants first, Tove solves the problem, by stating that this depends on whether you are a girl or a boy, and that these cards therefore can be placed next to each other. From a computer science perspective, they have now run into a situation where the "normal" (sequence) instruction is not enough. They now also need another fundamental type of programming concept, namely conditionals (if-statement), which make it possible for a program to do different things depending on whether a given condition (for instance, "the robot is a girl") is true or not.

Sequence 3: After having decided on the panties, the children collectively decide to put on a jacket, shoes and mittens. Next up the children suggest socks, which turns out to be problematic. Tove states "But then we have a problem, I think we have programmed wrong". At this time, the children all sit quietly paying attention. Tove goes on to describe the problem of putting on the socks at this point. She places the socks card at the end of the card sequence, after the shoe card and then looks at the children. Next, she asks if they can put on socks on top of the shoes and illustrates this by bringing out her foot as in trying to put a sock on top of the shoe. Some children also make the same gestures. The children then laugh and says nooo. Tove agrees: "No, that cannot be done." Some children start moving around and sit up straight.

Child 3: I usually put it [there!

[the child gets up and goes forward to point at the place in the card sequence where the socks should be

Tove: So before the [shoes

[takes the socks card in order to move i

[the child still stands there, points towards where the card should be

[Socks there!

You can place it here

Child: [No, [there

[again points at the same place as before

Tove: We need to simply switch [places

[switches the places so that the socks end up where the child wanted them

Tove then summarizes what they have done "We should have trousers, socks and then the jacket. Then the shoes and then the mittens." To finish the session, they all test the program by acting as robots dressing according to the instructions.

This debugging sequence is *teacher-initiated* but *child-driven*. The teacher points out the problem, but the suggestion for how to solve it comes from the children.

In addition to pointing out the difference between who initiates and drives the debugging process, the above sequences also indicate that the understanding and thereby also the solutions to analogue programming activities, particularly in familiar domains, can vary greatly. Due to the background and experiences of the people involved, what is true for one person might not be true for someone else. Debugging in this type of tasks thus takes the form of a negotiation, rather than the that of seeking a given right (or wrong) answer.

Case 2: Maths, the Bluebot and digital debugging—'bombs' as ways of teaching debugging

Two children, let us call them Frida and Oskar, are sitting together with their preschool teacher Annika at a table (See Figure 2). They have a poster in front of them on the table. The poster has a symmetrical square pattern with three types of squares: one starting square ("start"), bomb squares and number squares. The children call the poster "the map". The numbers are either written as using digits or as dots from a dice. The children will use the map together with the Bluebot, which always moves in steps of 15 cm. The squares are 15 × 15 cm, in order to fit the Bluebot's steps.

The children sit opposite to each other and the teacher Annika sits next to Frida. A deck of cards is placed next to the map. Each card has an addition task on it, which the children should calculate (for instance, 4 + 3 or 2 + 6). The deck of cards is turned upside down, so the tasks are not visible to the persons at the table. There is also a high plastic can with different colored figures in it.

The children are to take a card, complete the calculation and then program the Bluebot to move from the starting square to one of the squares with the answer in it (either to the square with a number or the one with the number represented by dots on one or several dice). The children need to find a way for the Bluebot to get to the right square without going through any of the bomb squares.

The sequence with Frida, Oskar and Annika is 11 min long and we have chosen to transcribe about 3 min where Frida will program the Bluebot, solve her task, and also conduct some debugging as part of solving the task. From this 3-min-sequence we present two specific sequences as debugging together with their respective analysis.

Figure 2. A digital programming session using a simple floor robot.

Sequence 1: The first debugging sequence starts with Frida solving her task and saying she needs to go from the starting square to the number five on the map. Annika wants her to present the route she is planning to take and program the Bluebot to go there. Frida starts moving her finger from the start but stops herself saying "uh" and looks at Annika, surprised.

Annika: (Whistling a bit) Right at that one! (Pointing at the bomb)

Frida: [Pointing with her hand to the other direction up to the same number five
[I want to go like that

Annika: [Can you go there?
[Pointing at a square with a bomb

Frida shows her intended route from start to number five by pointing and looking at the path. She does, however, not notice the bomb, which will make the Bluebot (virtually) explode. Annika's utterance "Right at that one!" as well as her pointing at the bomb can be understood as a multimodal comment to make Frida aware of the bomb in advance, before actually programming the Bluebot. Oskar is also looking at Annika's pointing. Pointing, gaze and speech are here central modes used to socially establish the understanding around the bomb. For Annika the modes can be understood as ways to relate the bomb to learning programming, while the multimodal comments for Frida can imply making meaning around the bomb.

When Frida notices the bomb, she says:

Frida: [Ahhhhh
[Pulling back her hand and pointing at the starting square

Annika: So now you'll have to rethink

Sequence 2: After this Frida starts from the beginning again and points with her hand in the other direction to the same number five. She gets a positive response from Annika who says "that is the way you can go".

Frida takes the Bluebot, puts it at the start square and turns it in the direction she wants it to move. She starts to program it by pressing the buttons on the Bluebot. It takes around 30 s, and she says that

she is going to turn around, and at the same time pressing the button, which turns the Bluebot to the right. Oskar, her fellow programmer sitting towards her, first points and then says:

Oskar: No that makes it very hard [at the bomb
[pointing at the square with the bomb
[Frida makes a grimace towards Oskar

Oskar: You will hit the [bomb
[takes his hand down and looks at the bomb

Frida looks at the Bluebot while lowering her shoulders and making a grimace, as Oskar says there are too many "one" referring to Frida having pressed the forward-arrow too many times. Frida starts counting and pointing at the squares, one at a time, while simultaneously counting "one two three four five". Oskar continues:

Oskar: You needed a turn, you ne[eded (whispering) a turn

Frida: [that one?
[looking at Annika

Annika tells Frida that she can try. Frida looks at the Bluebot keeping her hands by the buttons, and Annika tells her to press the "go" button when she feels she is ready with the program. At this point Oskar, who has not said or moved his body much so far in response to Frida's programming, notices the mistake she is about to make, having pressed the button making the Bluebot move forward too many times. Before making her aware of this verbally he points and looks at the square for which she is currently planning the next programming step. Frida's grimace can be understood as a disapproval of his utterances. Oskar continues and stresses the fact that she will touch the bomb saying—"too many one". He also continues pointing and looking, while also giving verbal comments. Despite the disapproval, Frida starts counting. At this time, Oskar is also presenting a possible solution to Frida, by saying that she needs a turn, but Frida instead turns to Annika to get her approval. Annika does not give her any particular advice and Oskar does not say anything else. Here the mistake that is about to happen creates a social situation, where Oskar multimodally tries to establish socially shared meaning around the upcoming mistake. He is making the correct analysis, but Annika's agenda is probably to let Frida make her mistakes and by doing so Frida gets to make new meaning around programming.

The excerpt above can also be seen as an example of early pair programming, which is a common way of developing programs in both learning and industry. Frida is the driver, who is controlling the Bluebot, whereas Oskar is the navigator, analysing the program, coming up with suggestions and helping Frida in solving the problem at hand (get the Bluebot to move to number five). What is missing in this particular situation, however, is a social contract, in order for Oskar's contribution to making meaning around programming to also be paid attention to.

Frida then presses the go-button.

Frida: [No!
[Lifts her hand as to say stop to the Bluebot

Annika: [Ohhh
[Lifts her hands to the Bluebot as to take it

Frida: [Laughs nervously
[Puts her hand to her mouth and putting her hand over her mouth
[Looks at Annika

Oskar stretches his body so that he can see. Annika lifts the Bluebot and imitating a bomb sound says "now it exploded", while keeping the Bluebot in her hand. Frida stretches to get the Bluebot, but Annika takes it away letting the program continue running in her hand. Then she says to Frida that they need to rethink the solution.

Here Frida's solution makes the Bluebot hit the bomb. Her saying "no!", lifting her hand, laughing nervously and looking at the teacher can be understood as if she is showing in a multimodal way that she understands that there is a problem. This is then established by Annika's comment and her taking the Bluebot. Oskar is including himself more in the sequence by stretching his body, but neither Frida nor Annika says anything with regard to Oskar's correct analysis. This might be due to the preschool culture suggesting that saying the words "you were right" or "you were wrong" is difficult. The participants in this sequence all see that the program had an error in it, but Oskar's analysis could have been confirmed and discussed—"how can we correct this"—in order for him to also strengthen his thinking.

After this, Frida changes to another strategy for programming the Bluebot, with some help from Annika. She now physically moves the Bluebot from square to square according to the route that she wants it to take, while at the same time pressing the button representing the respective move. This strategy is somewhat less abstract and makes programming more concrete and for some children this may be easier to grasp. She completes the program this way and then presses "go" from the start square to see if she has succeeded. When the Bluebot approaches the bomb that it ran into the last time she puts down her hand as to stop it from going there again, as if she did not really trust the program she had just created. The Bluebot does turn according to the plan and Frida happily looks at Annika as the Bluebot moves forward on its path and reaches number five. Annika looks at Frida saying "Well done, Frida!". Now it is Oskar's turn to complete the same kind of task.

6. Discussion and Conclusions

The aim of this article has been to broadly elaborate on how programming can be understood as a new activity in preschools, with a specific focus on debugging as one of the phases involved in learning to program. The research question *How can debugging as part of teaching and learning programming be understood as multimodal learning?* has guided the analysis and the presentation of the data.

Programming in preschool can be seen as focusing on playful learning, where children can use their imagination while developing their problem-solving skills. The children in our study showed interest in the programming activities and exhibited both patience and willingness to follow the teacher's instructions. This is in line with previous research, which found that preschool children often find programming activities attractive and engaging [3] and something they explore in a playful and curious manner [20].

The children also communicate and cooperate when solving different problems together, they explain their reasoning to each other and draw conclusions. These are all important skills according to the Swedish ECE curriculum. In their study, Levy and Mioduser [20] also showed how this, for instance, resulted in children making plans and predictable actions in order to make the robots act in a desired way. The children in our study were able to solve the programming problems at hand despite their young age, a result that corresponds to the findings by Toh et al. [15] and Bers et al. [16]. The type of planning involved became particularly notable in example 2, where the children were to develop a program that would make the robot move to a particular place on the map. Clearly, the children were able to design and build programs to control the simple robot after a limited time of instruction. Similar results were reported in [20].

Through programming, opportunities are created to develop understanding of mathematical concepts in space perception, such as forward, backward, right and left directional words. It is also about seeing and repeating patterns and breaking down structures in smaller parts. When programming, there is an advantage to see which code is repeated, in order to avoid repeating the same sequence several times.

Our study can be described as a case study, building on two examples of debugging and based on a multimodal analysis. Since debugging is considered a central feature in learning to program, focus is here put on how debugging is carried out. We found a number of debugging sequences in the material and chose to present two in the text. The reason for including these two examples was to establish a closer understanding of the multimodal micro-processes involved when engaging in programming—and in particular debugging—activities. Here a multimodal perspective has been useful in order to unravel these micro-processes, showing the complexity of communication around certain learning contents.

The results have pointed out some differing perspectives in debugging practices at preschool level. First, the person initiating and driving the debugging process differs. While some sequences were clearly *teacher-initiated*, meaning that the teacher identified the problem and made the children aware of it, some were also *children-initiated*. In the same manner, the debugging process that followed could be *teacher-driven*, *child-driven* or a combination of these. In some sequences, the children continued driving the process after the teacher had pointed out the problem, which is positive as it indicates that children can understand the workings of the program, analyze it and come up with solutions on their own or together with their peers. For future studies, it would be interesting to instruct the teacher to not interfere in the programming "too soon", but instead see whether, and if so, how and when, the children find the mistakes themselves.

Second, the results from these two examples suggest that debugging unfolds differently in analogue and digital programming sessions. Naturally, no general conclusions can be drawn from two examples, but the differences are still interesting as they, for instance, can guide teachers and teacher educators in planning and executing different types of programming activities. In the analogue session, debugging took the form of a negotiation, in order to find the answer. This might be due to the fact that personal understanding and experience of the problem domain at hand can lead to children having different views on how certain things are done. This was particularly notable in the example of getting dressed, where children are used to putting on clothes in different order. This is also the case for most of the everyday tasks we go through—we probably do most things, from brushing our teeth to making a sandwich, in slightly different ways. This is, however, also the case in "real" programming, where there is seldom only one way of solving a problem or implementing an idea. Instead, we can have different programs that lead to the same result, but the way in which this is accomplished differs. Simple activities like these, conducted at an early age, hence have similarities with "real" programming as well. Similarly, the discussion on when to put on panties and underpants ended up in the decision on having them side by side, as which one you put on depends on who you are. This also builds the basis for later talking about conditionals (if-statements) and conditions in a programming context.

The digital programming sequence, and here with a focus on debugging within the sequence, describes how the children and their teacher work both with debugging as part of learning a content, but also how debugging is established socially as a multimodal activity. The analysis shows how debugging here, similarly to in the analogue situations, is initiated by both the teacher and the children. The teacher initiates debugging by using an indirect way—by pointing or through a verbal comment. The children use pointing and verbal comment in a more direct way, focusing on the mistake that was going to take place. Compared to debugging in the analogue activity, the digital situation is more of a negotiation of how it should be handled socially. In our example of digital debugging, there is not so much focus on whether the debugging is correct or not, but rather on how it socially is handled. Nevertheless, there is less negotiation space on what is right or wrong in digital settings, such as the one here, as the children have a limited set of instructions to work with and the evaluation of whether the program is correct or not is done by a machine (the robot) and not by a peer.

Artefacts are central to programming and hence also to both analogue and digital debugging. As noted above, we can have both *instructional artefacts* (instructions in the form of arrows, images, symbols, etc.) In addition, we have *programmable artefacts*, which are controlled by the instructional

artefacts. In the analogue example, the programmable artefact was the children themselves or an imaginary robot, whereas the clothing cards were instructional artefacts. In that example, the instructional artefacts were clearly more important than the programmable one. In the digital debugging sequence on the other hand, the programmable artefact, the Bluebot, was much more visible and central to the activity. On top of its back, Bluebot has a limited set of instructional artefacts in the form of seven buttons: four arrows for directions, one goes, one empty, and one pause. These artefacts describe what the Bluebot can do, or more specifically what instructions we can make it execute. These have a certain meaning related to programming, which is central to making meaning in the video sequence. Without those instructions, meaning making around programming would be very different. Bateman, Carr and Gunn [17] also pointed out how "physical props", that is, in our terms "artefacts" support children's learning processes. In our case, both the physical programmable artefacts and the instructional artefacts connected to it are affordances for the analogue and digital debugging sequences to happen, giving the participants certain meaning-making possibilities, and also pre-creating some modes to be established.

The focus on the artefacts, and the modes made around it, have implications for teachers. The results presented in this article show that debugging is carried out multimodally, with pointing and looking as central modes accompanied by speech. The social orchestration happens around the artefact, meaning that the artefact, its affordances and the possibilities for children to make meaning multimodally around the artefact are central to debugging.

Focusing on debugging brings up quite uncommon aspects in Swedish preschools. There is a tradition in Swedish educare not to focus on children's mistakes or what is right and wrong according to certain norms. Swedish preschools are very child centred and children are encouraged to try and retry as a means of learning and developing. Children are met with patience and this can be understood historically, as the predominate paradigm is based on a strong implementation of a child perspective. To focus on debugging implies to focus on mistakes, in a context that has traditionally been very untrained in handling mistakes explicitly.

The results presented in this paper contribute to the current knowledge about programming with young children. While many incorporate programming in their preschool activities, the number of studies on programming with small children is still very limited. Empirical studies are needed in order to guide teachers in designing and planning the activities in ways, such that the children have the best opportunities to learn. Based on our analysis, we can for instance make the following quite practical observations:

- The analysis unfolds communication as multimodal, and not putting language in the center or as the 'driving force' for communication and learning. This is important for the teacher to keep in mind both when planning and conducting teaching. Knowledge on how communication is multimodally practiced gives teachers new opportunities to trust that learning can take place also without written or spoken language.
- Several programming related practices and concepts can be used and described in ECE. For instance, children in our examples were dealing with instructions, sequences and conditionals. The children in the second case also exhibited a pair programming like approach. The teachers, however, did not use programming terminology in the situations. By using the correct terms, preschool activities could play an important role in conceptualizing programming already at an early age. This, naturally, requires professional development for the teachers, as the overwhelming majority has no previous programming background.
- In order for children to find their own errors and thereby practice, for instance, their logical thinking skills, teachers should not intervene "too soon" when they realize that a program will not work. Programming is about "learning by doing", and making mistakes is an important part of this process.

Even though the sample is limited, this case study contributes valuable insights about multimodal analysis that can be used by researchers in other studies, as well as by ECE teachers when planning their programming activities and teacher educators when designing professional development for pre- and in-service ECE teachers.

Author Contributions: Conceptualization, M.H. and L.M.; Methodology, M.H.; Software, M.H.; Validation, M.H. and L.M.; Formal Analysis, M.H. and L.M.; Investigation, M.H.; Resources, M.H.; Data Curation, M.H.; Writing-Original Draft Preparation, M.H. and L.M.; Writing-Review & Editing, M.H. and L.M.; Visualization, M.H.; Supervision; Project Administration, M.H.; Funding Acquisition.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Vuorikari, R.; Punie, Y.; Carretero Gomez, S.; Van Den Brande, G. DigComp 2.0: The digital competence framework for citizens. In *Update Phase 1: The Conceptual Reference Model*; Publications Office of the European Union: Seville, Spain, 2016.
2. Skolverket. Få Syn på Digitaliseringen på Grundskolenivå. *Comment. Mater.* Available online: https://www.sigtuna.se/PageFiles/64857/it-i-skolan.pdf (accessed on 21 May 2018).
3. Fesakis, G.; Gouli, E.; Mavroudi, E. Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Comput. Educ.* **2013**, *63*, 87–97. [CrossRef]
4. Sipitakiat, A.; Nusen, N. Robo-Blocks: Designing debugging abilities in a tangible programming system for early primary school children. In Proceedings of the IDC, Bremen, Germany, 12–15 June 2012.
5. Skolverket. Förslag Till Reviderad Läroplan för Förskolan. 2018-03-23 Dnr: 2017:783. 2018. Available online: https://www.skolverket.se/om-skolverket/publikationer/visa-enskildpublikation?_xurl_=http%3A%2F%2Fwww5.skolverket.se%2Fwtpub%2Fws%2Fskolbok%2Fwpubext%2Fbilaga%2FBlob%2Fpdf695.pdf%3Fk%3D695 (accessed on 21 May 2018).
6. Lister, R.; Adams, E.S.; Fitzgerald, S.; Fone, W.; Hamer, J.; Lindholm, M.; McCartney, R.; Moström, J.E.; Sanders, K.; Seppälä, O.; et al. A multi-national study of reading and tracing skills in novice programmers. In Proceedings of the Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education (ITiCSE-WGR '04), Leeds, UK, 28–30 June 2004; ACM: New York, NY, USA, 2004.
7. McCauley, R.; Fitzgerald, S.; Lewandowski, G.; Murphy, L.; Simon, B.; Thomas, L.; Zander, C. Debugging: A review of the literature from an educational perspective. *Comput. Sci. Educ.* **2008**, *18*, 67–92. [CrossRef]
8. Clements, D.H.; Gullo, D.F. Effects of computer programming on young children's cognition. *J. Educ. Psychol.* **1984**, *76*, 1051–1058. [CrossRef]
9. Mannila, L. *Att Undervisa I Programmering I Skolan—Varför, Vad Och Hur?* Studentlitteratur AB: Lund, Sweden, 2017.
10. Papert, S. *Teaching Children Thinking (LOGO Memo)*; Massachusetts Institute of Technology, A.I. Laboratory: Cambridge, MA, USA, 1971.
11. Papert, S.; Harel, I. Situating Constructionism. Ur boken Constructionism. 1991. Available online: http://www.papert.org/articles/SituatingConstructionism.html (accessed on 21 May 2018).
12. Papert, S. *Mindstorms: Children, Computers and Powerful Ideas*; Basic Books: New York, NY, USA, 1980.
13. Wing, J. Computational thinking. *Commun. ACM* **2006**, *49*, 33–35. [CrossRef]
14. Brennan, K.; Resnick, M. *New Frameworks for Studying and Assessing the Development of Computational Thinking*; AERA: Washington, DC, USA, 2012.
15. Toh, L.P.E.; Causo, A.; Tzuo, P.W.; Chen, I.M.; Yeo, S.H. A review on the use of robots in education and young children. *Educ. Technol. Soc.* **2016**, *19*, 148–163.
16. Bers, M.U.; Flannery, L.; Kazakoff, E.R.; Sullivan, A. Computional thinking and tinkering: Exploration of an early childhood robotics curriculum. *Comput. Educ.* **2013**, *72*, 145–157. [CrossRef]
17. Bateman, A.; Carr, M.; Gunn, A.C. Children's use of objects in their storytelling. In *Interactions in Early Childhood Education: Recent Research and Emergent Concepts*; Gunn, A.C., Hruska, C.A., Eds.; Springer: Berlin, Germany, 2017.
18. Kanaki, K.; Kalogiannakis, M. Introducing fundamental object-oriented programming concepts in preschool education within the context of physical science courses. *Educ. Inf. Technol.* **2018**. [CrossRef]

19. Levy, S.T.; Mioduser, D. Approaching complexity through planful play: Kindergarten children's strategies in constructing an autonomous robot's behaviour. *Int. J. Comput. Math. Learn.* **2010**, *15*, 21–43. [CrossRef]
20. Sullivan, A.; Kazakoff, E.R.; Bers, M.U. The wheels of the bot go round and round: Robotics curriculum in pre-kindergarten. *J. Inf. Technol. Educ. Innov. Pract.* **2013**, *12*, 203–219. [CrossRef]
21. Kazakoff, E.R.; Sullivan, A.; Bers, M.U. The effect of a classroom-based intensive robotics and programming workshop on sequenceing ability in early childhood. *Early Child. Educ. J.* **2013**, *41*, 245–255. [CrossRef]
22. Flannery, L.P.; Silverman, B.; Kazakoff, E.R.; Bers, M.U.; Bontá, P.; Resnick, M. Designing ScratchJr: Support for early childhood learning through computer programming. In Proceedings of the 12th International Conference on Interaction Design and Children (IDC '13), New York, NY, USA, 24–27 June 2013; ACM: New York, NY, USA, 2013.
23. Brewer, R.; Anthony, L.; Brown, Q.; Irwin, G.; Nias, J.; Tate, B. Using gamification to motivate children to complete empirical studies in lab environments. In Proceedings of the 12th International Conference on Interaction Design and Children (IDC '13), New York, NY, USA, 24–27 June 2013; ACM: New York, NY, USA, 2013.
24. Palmér, H.; van Bommel, J. *Problemlösning Som Utgångspunkt—Matematikundervisning I Förskoleklass*; Liber: Stockholm, Sweden, 2016.
25. Cejka, E.; Rogers, C.; Portsmore, M. Kindergarten robotics: Using robotics to motivate math, science, and engineering literacy in elementary school. *Int. J. Eng. Educ.* **2006**, *22*, 711–722.
26. Kress, G.R. *Literacy in the New Media Age*; Routledge: London, UK, 2003.
27. Selander, S. *Didaktiken Efter Vygotskij: Design för Lärande*; Liber: Stockholm, Sweden, 2017.
28. Kress, G.R. *Multimodality: A Social Semiotic Approach to Contemporary Communication*; Routledge: London, UK, 2010.
29. Kress, G.; Jewitt, C. Multimodality, literacy and School English. In *The Routledge International Handbook of English, Language and Literacy Teaching*; Routledge: London, UK, 2010.
30. Bezemer, J.; Kress, G. *Multimodality, Learning and Communication: A Social Semiotic Frame*; Routledge: London, UK, 2016.
31. Pahl, K. Interactions, intersections and improvisations: Studying the multimodal texts and classroom talk of six- to seven-year-olds. *Int. J. Early Child. Lit.* **2009**, *9*, 188–210. [CrossRef]
32. Stein, P. *Multimodal Pedagogies in Diverse Classrooms: Representation, Rights and Resources*; Routledge: London, UK, 2008.
33. Pahl, K. Creativity in events and practices: A lens for understanding children's multimodal texts. *Literacy* **2007**, *41*, 86–92. [CrossRef]
34. Heikkilä, M. *Kommunikativa Resurser för Lärande: Barns Gester, Blickar Och Tal I tre Skolmiljöer*; Uppsala Universitet: Uppsala, Sweden, 2006.
35. Jensen, T.; Sandström, J. *Fallstudier*; Studentlitteratur: Lund, Sweden, 2016.
36. Bucholtz, M. The politics of transcription. *J. Pragmat.* **2000**, *32*, 1439–1465. [CrossRef]
37. Green, J.; Franquiz, M.; Dixon, C. The myth of the objective transcript: Transcribing as a situated act. *TESOL Q.* **1997**, *31*, 172–176. [CrossRef]
38. Davidson, C. Transcription: Imperatives for qualitative research. *Int. J. Qual. Methods* **2009**, *8*, 35–52. [CrossRef]
39. Oliver, D.G.; Serovich, J.M.; Mason, T.L. Constraints and opportunities with interview transcription: Towards reflection in qualitative research. *Soc. Forces* **2005**, *84*, 1273–1289. [CrossRef] [PubMed]
40. Witcher, C.S.G. Negotiating transcription as a relative insider: Implications for rigor. *Int. J. Qual. Methods* **2010**, *9*, 122–132. [CrossRef]
41. Cowan, K. Multimodal transcription of video: Examining interaction in early years classrooms. *Classr. Discourse* **2014**, *5*, 6–21. [CrossRef]
42. Vetenskapsrådet. *Forskningsetiska Principer Inom Humanistisk-Samhällsvetenskaplig Forskning*; Vetenskapsrådet: Stockholm, Sweden, 2002.
43. Vetenskapsrådet. *Good Research Practice*; Swedish Research Council: Stockholm, Sweden, 2017.
44. Ericsson, S.; Boyd, S. Children's ongoing and relational negotiation of informed assent in child-researcher, child-child and child-parent interaction. *Childhood* **2017**, *24*, 300–315. [CrossRef]