MedienPädagogik
Zeitschrift für Theorie und Praxis der Medienbildung
www.medienpaed.com
ISSN 1424-3636

Themenheft Nr. 33: Medienpädagogik und Didaktik der Informatik.
Eine Momentaufnahme disziplinärer Bezüge und schulpraktischer Entwicklungen.
Herausgegeben von Torsten Brinda, Ira Diethelm, Sven Kommer und Klaus Rummler

A Design-Based Research Approach for introducing Algorithmics and Programming to Bavarian Primary Schools

Theoretical Foundation and Didactic Implementation

Katharina Geldreich, Alexandra Simon and Peter Hubwieser

Abstract

Computer Science (CS) is increasingly entering the early levels of childhood education, like primary school or even kindergarten. Although Germany has not yet developed mandatory guidelines for how to deal with these new topics, Bavaria seems to consider extending the field of computer science education to the primary sector in the long term. It is therefore becoming more and more necessary to gain insight into which teaching methods and content would be suitable for students at primary level. To investigate the characteristics of effective programming courses for primary schools, we developed a three-day introductory course following the design-based research approach. This article will set focus on both the theoretical foundation resulting from this specific research approach and the didactic implementation of the theoretical framework.

Design-Based Research als Ansatz zur Einführung von Algorithmik und Programmierung an bayerischen Grundschulen. Theoretische Fundierung und didaktische Umsetzung

Zusammenfassung

Informatik soll in den kommenden Jahren stärker in die frühe Bildung integriert werden und somit den Weg in Grundschulen oder gar in Kindergärten finden. Obwohl Deutschland noch keine verbindlichen Richtlinien für den Umgang mit diesen neuen Themen entwickelt hat, erwägt das Bundesland Bayern bereits den Ausbau der Didaktik der Informatik zur Lehrerausbildung im Bereich der Grundschule. Das Erforschen von Unterrichtsmethoden und -inhalten, die sich für diesen Bereich eignen, wird somit immer notwendiger. Um die Eigenschaften praktikabler und effektiver Informatikkurse für die Primarstufe zu untersuchen, nutzen wir den Ansatz der Design-Based Research, um einen dreitägigen Programmierkurs für Dritt- und Viertklässler zu entwickeln. Dieser Beitrag beschäftigt sich vorrangig mit der theoretischen Grundlage für die Kursentwicklung, die aus dem Forschungsansatz entwickelt wurde. Ausserdem wird die konkrete didaktische Umsetzung der theoretischen Vorarbeiten in Form des Kurskonzepts ausführlich beschrieben.

Geldreich, Katharina, Alexandra Simon, and Peter Hubwieser. 2019. «A Design-Based Research Approach for introducing Algorithmics and Programming to Bavarian Primary Schools. Theoretical Foundation and Didactic Implementation». *MedienPädagogik* 33, (Februar), 53–75. https://doi.org/110.21240/mpaed/33/2019.02.15.X.

Introduction

In the past few years, the discussion about the necessity of computer science (CS) and especially programming in primary education is growing steadily (Prottsman 2014; Bell and Duncan 2018). Introducing computer science concepts from an early age seems to have several positive consequences on the children (Duncan, Bell, and Tanimoto 2014). Learning to use computers not only as users but also as creators and gathering positive experiences in computing can strengthen their self-confidence towards CS and technology in general (Topi 2015). It may also prevent common misconceptions and prejudices towards computer science regarding the nature of the subject and the role of gender (Funke, Berges, and Hubwieser 2016). Several countries, such as Australia (Falkner, Vivian, and Falkner 2014), the United Kingdom (Brown et al. 2013) or Russia (Khenner and Semakin 2014) have already introduced computer science in their primary school curricula. Other countries like France, Poland, and New Zealand are also in the process of implementing notions of problem-solving, algorithmic thinking and basic programming in their curricula (Armoni 2018).

While Germany has not yet developed mandatory guidelines for how to deal with these new topics, the Federal Government is already pleading for the use of digital media in primary schools in order to promote the students' digital competencies (BMBF 2016). Since the federal state of Bavaria is already considering to expand the field of computer science education to the primary sector in the long term (StMWI 2017), it is all the more important to investigate which teaching methods and contents would be suitable for Bavarian primary schools.

Extending computer science to the early level of education requires practical and theoretical knowledge in a variety of disciplines (Duncan, Bell, and Tanimoto 2014). We wanted to apply a research design that takes this aspect into account and enables close interaction between researchers and practitioners. Thus, we chose the design-based research (DBR) approach to develop an effective primary school course for learning the basic concepts of programming.

The following sections will give a brief introduction to DBR and the research design we developed according to the approach. We will then set focus on the theoretical foundation and didactic implementation of the course. The findings of the literature review, the resulting theoretical framework as well as the corresponding course concept are presented. The article closes with several conclusions regarding the DBR approach and will give an outlook on our future work.

Design-Based Research

The term design-based research goes back to Brown (1992) and Collins (1992) who placed their focus on driving innovations. They developed «design experiments» as a way to carry out formative research to test and refine educational designs under

realistic and situative conditions. Based on the design-experiment, further approaches and terms have emerged, which can be partly interpreted as synonyms: design research, design-based research, design experiment, educational design research, development research (see The Design-Based Research Collective 2003; Richey and Klein 2005; van den Akker et al. 2006). They all share the aim of designing and developing interventions as solutions to a complex problem for which there are few or no validated principles or guidelines available (Plomp 2007). Wang and Hannafin see design-based research as an overriding notion and define it as

> «a systematic but flexible methodology aimed to improve educational practices through iterative analysis, design, development, and implementation, based on collaboration among researchers and practitioners in real-world settings, and leading to contextually-sensitive design principles and theories» (2005, 6).

In this continuous iterative process, design changes are made, assessed and refined to improve the design.

Different versions of this process can be found in literature, all of which contain similar phases and differ only slightly (Collins, Joseph, and Bielaczyc 2004; Reinmann 2005; Jahn 2014). Authors agree that DBR consists of the following components (Plomp 2007, 15):

- *preliminary research*: Review of literature and projects relevant to the chosen topic that results in design principles for the intervention.
- *prototyping phase:* Development of an intervention prototype that is tested, evaluated and revised.
- *assessment phase:* Evaluation whether the intervention is effective. It is also investigated how the target users can work with it and are willing to apply it.

Formative evaluation takes place in all the three phases and iterative cycles of DBR. Mixed research methods are used to maximize the objectivity, validity, and applicability of the ongoing research (Wang and Hannafin 2005).

Using DBR to develop a programming course for Bavarian primary schools

Research Design

We chose the design-based research approach because, on the one hand, we wanted to develop a practicable primary school course and, on the other hand, we wanted to investigate the students' behaviour and learning outcomes. A summary of the research design is shown in Figure 1.

The basis of our research design is a broad-based literature review which covers the field of instructional design, teaching quality, computer science education and cognitive development. Parallel to this, we conducted interviews with five primary school teachers who had no relevant previous experience with computer science (see Funke, Geldreich, and Hubwieser 2016) and examined the Bavarian primary school curriculum for possible links to computer science. At the end of this analysis phase, a conceptual framework was formulated that also considered the specific prerequisites of the Bavarian school system. The results of this analysis phase and the derived principles are described in more detail below.

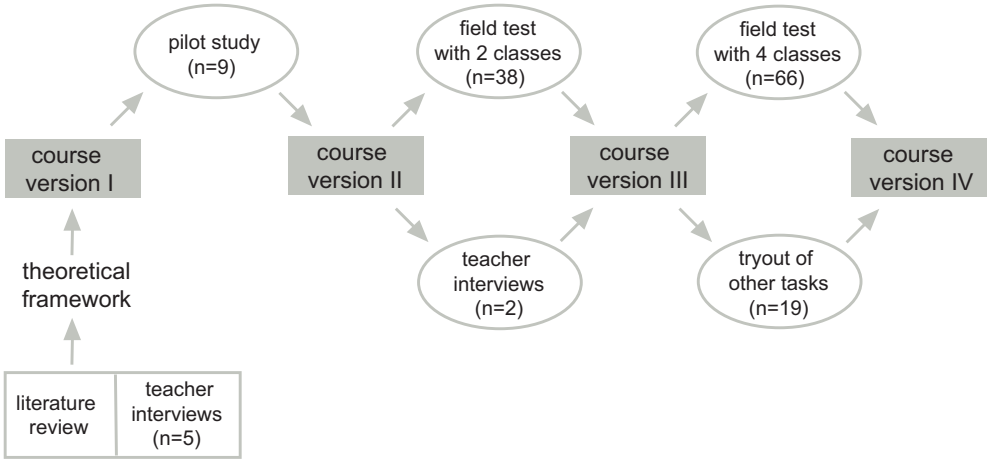

Fig. 1.: Research design.

After this phase of preliminary research, we developed a first course prototype based on the framework. The first drafts of both the theoretical framework and the resulting prototype were discussed with several experienced practitioners and an expert in the field of computer science education.

We tested the course prototype with a small group (n=9) in a pilot study that was carried out at a student research centre led by our university. After revising and optimising the tasks and some changes in the course timeframe, we conducted the course again with two fourth-grade classes (n=38) and also interviewed the accompanying teachers after the courses. We once again revised some of the tasks and made changes in the structure and design of the materials. After testing a variety of other methods with children at the student research centre (n=19), we made another field test with one third-grade and three fourth-grade classes (n=66). Since in Bavaria data may only be collected at schools in exceptional cases and with special permission, these field tests were carried out at our university. The courses were taught in German language by different instructors, including a formally educated primary school teacher and researchers in the field of CS education. The resulting course concept can be read in the further course of the article.

Research Questions and Mixed-Methods

Our research is supposed to investigate the characteristics of an effective primary school course concept for learning the basic concepts of programming. To answer this overarching question and to assess to what extent the intervention has served its purpose, we investigate several questions in our field studies:

- How do the students behave and collaborate during the learning process?
- How do the programming results relate to the learning process?
- Which differences can be identified between the behavior of girls and boys?

During the field tests, we collected data using a mixed-methods approach including the following methods:

- *Videography:* To analyze the interactions of the children with each other and with the teacher, we recorded the whole courses with at least two cameras.
- *Group interviews and questionnaires:* A variety of interviews and reflection methods were used to get an idea of the students' prior knowledge, what they think about programming and how they feel after the course.
- *Screen and audio capturing:* When the students were working on computers, screens and audio of every student were captured with special software. By doing so, we wanted to get an insight into the students working methods.
- *Scratch programming products:* All Scratch programs were saved to analyze them in detail.

Several evaluations have already been carried out, for example, an analysis of the structure and quality of the students' programming results (Funke, Geldreich, and Hubwieser 2017) and an analysis of the relationship between the students' gender and the characteristics of the program they created (Funke and Geldreich 2017).

Literature Review

Following the methodology of design-based research, we took into account theoretical principles and prior research from various disciplines, including not only computer science education but also developmental psychology, cognitive science and instructional design. The following section presents selected results of the literature review that were relevant for the design of the course concept in the subsequent DBR process. It also gives a brief insight into the Bavarian primary school curriculum.

Cognitive Development of Primary School Children

If we want to design a computer science course, we must align the content, methods and material to the developmental stage of the target group - in our case primary school children. Piaget and Inheler (1969) found out that older children think differently than younger ones and proposed a theory of intellectual development over four stages. According to this theory, primary school children are in the *concrete operational stage* (7-11 years): the students begin to think logically about actual, specific events but have difficulties in understanding abstract or hypothetical concepts. According to Piaget, the *formal operational stage* which includes the ability to think about abstract concepts is only reached at the age from 11 to 16 years. However, Neo-Piagetian theories (Case 1992; Morra et al. 2007) concluded that the development stages depend on a variety of other factors, e.g., learning area, and cannot be assigned exactly to a particular age. Apart from the classification in stages, Piaget's work valued the importance of active learning in which children are free to explore, question and try out. Teachers should also provide visual aids or props to support children's thinking. He further recommends using familiar examples to introduce more complex and abstract ideas (Piaget 1976).

Bruner (1966) argued that children could learn any material as long as the instructions are organized appropriately. Every concept should be encountered several times in a progressive spiral: at first in a concrete way or with physical actions (*enactive representation*), then within images or illustrations (*iconic representation*) and at last using codes or symbols, such as language (*symbolic representation*).

The work of Vygotsky (1978) deals, among other things, with the fundamental role of social interactions in the development of children. His notion of the *zone of proximal development (ZPD)* describes the gap between what a child can achieve alone and what it could potentially achieve with the help and guidance from someone more knowledgeable, skilled or experienced. In the interaction with others, children could get instructions or advice that allows them to develop skills they can afterward use on their own. Therefore, it follows that it's beneficial for students to work collaborative – especially in groups of mixed ability.

Instructional Design

Merrill et al. (1996) describe Instructional Design as «a technology for the development of learning experiences and environments which promote the acquisition of specific knowledge and skill by students». One aspect which is of particular importance to this development is motivation. Keller's ARCS Model (1983, 1987) defines four essential conditions, that should be met in order to get and keep students' motivation: *Attention* (getting and sustaining attention by appropriate stimuli and a variety of methods), *Relevance* (making the content count, e.g. by linking the instructions

to the students' life), *Confidence* (giving the impression that the students can succeed if they exert some effort) and *Satisfaction* (providing occasions where the students can feel good about their accomplishments). For each condition, Keller lists additional subcategories as well as strategies to fulfill them.

Mayer's cognitive theory of multimedia learning (2001, 2005) focuses primarily on the optimized integration of text and image presentation in learning contents. In this context, text can be spoken as well as written, and images can be illustrations, photos and also animations or videos. He suggests that learning occurs when we build mental representations of these various elements and formulates a variety of principles, that should support this process. According to Mayer, the most relevant multimedia instructional principles for learning opportunities are: *Signaling Principle* (highlight the organization of the material), *Coherence Principle* (avoid unnecessary content), *Spatial Contiguity Principle* (place corresponding words and pictures near each other), *Segmenting Principle* (divide lessons in user-paced segments), *Multimedia Principle* (students learn better from words and pictures than from words alone) and *Personalization Principle* (use conversational style rather than formal style).

Fundamental to Mayer's work is the cognitive load theory which has been developed by Sweller and Chandler (1991). It suggests that the construction of instructional material can influence the learning process by directing the cognitive resources to activities which are relevant to learning and not just preliminaries to it. According to Sweller et al. (1998) there exist three types of cognitive load: intrinsic, extraneous and germane. *Intrinsic cognitive load* refers to the inherent difficulty of the content being learned; *extraneous cognitive load* refers to the effort which is caused by factors that aren't central to the content, e.g., presentation methods, instructional procedures; *germane cognitive load* refers to the effort a learner has to make to understand the material by schema acquisition. It is assumed that working memory can only process a limited amount of information at one time and therefore learning is interrupted when this number is exceeded – a cognitive overload occurs (De Jong 2010). While intrinsic cognitive load can't be affected, we can prevent this overload by balancing extraneous and germane cognitive load. We can avoid the unnecessary extraneous load by, for example, removing everything that distracts the learner from the essential. Sweller et al. (1998) describe several effects, which can reduce extraneous cognitive load and can partly be found in Mayer's cognitive theory of multimedia learning (2001, 2005). The *worked example effect*, for instance, describes how we can reduce extraneous cognitive load by using step-by-step demonstrations of how to solve tasks and problems. Another effect, which is also particularly interesting for our topic is the *completion problem effect*. It states that one can reduce the size of a problem by providing a partial solution that must be completed.

Quality Criteria for Teaching and Teaching Materials

A research tradition that sets its focus on teaching quality is working with a process-product model (Dunkin and Biddle 1974; Shuell 1996) that relates the teaching process to the learning outcome. It's assumed that high-quality teaching leads to positive learning outcomes, which in turn enables us to identify characteristics of good teaching. Of course, this effect is mediated by student characteristics (e.g., pre-knowledge, self-concept of ability, interest), the classroom context (e.g., class size) and the prerequisites of the teacher which has led to more and more differentiated so-called supply-use models (Brühwiler and Blatchford 2011; Creemers and Kyriakides 2007; Pianta and Hamre 2009). Even if studies showed that we can not assume a direct effect of teaching on the learning outcome (Winne 1987; Weinert, Schrader, and Helmke 1989), these models give us further hints when looking for criteria to develop appropriate teaching methods and materials. Although the terminology differs, three recurring overarching dimensions have been identified, that are considered essential to good teaching (Klieme, Pauli, and Reusser 2009; Seidel and Shavelson 2007):

1. classroom management, clarity and structure,
2. potential for cognitive activation and,
3. learning support through monitoring of the learning process, individual feedback and adaptive instruction.

The intention of cognitive activation is engaging students in higher-level thinking. This can be achieved by challenging tasks and conveying cognitive strategies such as summarizing, questioning and predicting (Mayer 2004). The items of the 2012 PISA student questionnaire (OECD 2013) give us an idea how to create cognitive activating tasks. For example, they could encourage the students to reflect on problems, ask them to apply what they have learned to new contexts, use problems with multiple solutions or present problems with no immediately apparent method of solution. Beyond challenging the students, the tasks should also be prepared in a way that they can be adapted to different learning prerequisites and abilities of the students (Baumert et al. 2010).

Teaching Computer Science

In their *Guide to Teaching Computer Science*, Hazzan et al. (2011) emphasize the importance of the learners' active acquisition of ideas. To facilitate this throughout a lesson or a course, they introduce the *Active Learning Based Teaching Model* that consists of four successive stages: trigger, activity, discussion and summary. In the first stage (*trigger*), the topic of the class/course is supposed to be introduced in a non-traditional fashion, i.e., it should raise questions or dilemmas. In the second stage

(*activity*), the students get to work on the presented trigger. In the ensuing *discussion*, products, topics or thoughts, which originated during the activity stage, are presented to the entire class and discussed. While the teacher holds back with judging the students' contributions and confines himself only to highlight important ideas presented by the students, the classmates are encouraged to express their opinions. In the *summary*, the teacher summarizes, highlights and emphasizes important concepts that were addressed during the previous stages.

In addition to the learners' activity, they underline that the development of problem-solving skills should be the core of every introductory CS course. They propose different steps of a problem-solving process that the students should undergo. The process starts with outlining the problem requirements and understanding what the problem is about (*problem analysis*). After that, the students should think about alternative ways to solve the problem (*alternative consideration*) and select one of them (*choosing an approach*). The problem is then to be divided into subtasks (*problem decomposition*) for which algorithms are developed (*algorithm development*). In a next step, the algorithms are checked for correctness (*algorithm correctness*) and efficiency (*algorithm efficiency*). In the *reflection* at the end of the process, the students should have a chance to analyse the process they went through and conclude possible improvements.

As already described in the section *Instructional Design*, motivation plays an important role in the outcome of learning situations. Martin (2017) describes a set of learning dimensions, that can help increase personal motivation when learning to program. The dimensions go back to the results of four fieldwork studies, which were carried out with participants from pre-school to university. Regarding the design of programming tasks, open tasks seem to be more motivating (see also the work of Petre and Price 2004). Besides, students seem much more motivated, when the tasks are relevant to their daily life experience, when they can personalize their programs and when they have a chance to share the products of their learning with others.

Regarding the arrangement of the learning experience, it's recommended to use a flexible structure with sufficient opportunities for learners to iterate the introduced concepts and influence the direction of their learning process. To prevent wasted effort, learners should have the opportunity to get frequent feedback during a task. The grouping of the students (working alone, in pairs or groups) and the shape of the session regarding the physical environment can also have a strong influence on the learners' motivation – the recommendations on both aspects, however, are varying with the given circumstances and the respective course.

Caspersen and Bennedsen (2007) describe how an introductory object-oriented programming course can be designed based on cognitive science and educational psychology. They primarily focus on how the cognitive load theory, cognitive apprenticeship and the theory of worked examples can be applied. Since they work

with university students, not all suggestions can be transferred to the primary school context – still their advises how to use worked examples could also be very suitable for younger students. To help the students developing schemes and transfer the learned, it is recommended to work with multiple examples that increase in complexity and vary in the form of the problem type. Examples should not be used exclusively, and they should alternate with matching problem tasks. Gray et al. (2007) point out that the transition from complete worked examples to solving problems only works if the problems are simple. To further reduce cognitive load, they suggest using partially worked examples that contain less and less worked steps and can be completed in stages. They provide a variety of these *fading worked examples* for different concepts of programming. Several further principles to reduce cognitive load during learning computing content can be found in the work of Tuovinen (2000).

Lee et al. (2011) and Sentance and Waite (2017) are also investigating which progression students should go through when learning to program. Lee et al. propose the «Use-Modify-Create Learning Progression» in which learners first *use* existing programs, *modify* them and finally *create* their own programs. This is followed by an iterative process of testing, analysing and refining. Sentance and Waite introduce the teaching approach *PRIMM*, which is based on the work of Lee et al. among others. *PRIMM* is the abbreviation for Predict, Run, Investigate, Modify and Make. In both scaffolding approaches, students first work with already existing programs and gradually make them their own until they create their own programs.

In his computing guide for primary teachers, Berry (2013) states several aspects of meaningful and effective learning in computing which go back to the work of Howland et al. (2011). Students should actively engage in the learning process, they should be able to construct both meaning and a result, and if possible, they should have a certain choice over how they work on a task. Furthermore, the tasks should be linked to the students' own experience and allow working collaborative.

Franklin et al. (2015) also give specific advice for teaching computer science in primary schools. Their guidelines are based on their experiences in developing, piloting and evaluating a scratch-based computational thinking curriculum and refer to different topics such as providing feedback, learning environment or teaching materials. The following tips seem highly relevant for developing computer science courses for primary schools:

1. Avoid down- and uploading files from a shared computer,
2. limit or avert the use of worksheets in the computer lab,
3. if necessary, tailor writing by giving options to circle answers or drawing pictures, and
4. prepare further options for students who finish the tasks faster than others.

Links to Computer Science in the Bavarian Primary School Curriculum

In Germany, schools started to teach CS during the 1960ies, and since then, the subject underwent several drastic changes and transformations (Hubwieser 2012). However, only in few federal states of Germany computer science is a mandatory subject and cannot be found in the curricula of primary schools. In the current primary school curriculum of Bavaria, media education is mentioned as one of the overarching education goals and is described as knowledge and skills to act appropriately, self-determinedly and responsibly in the multimedia influenced society (StMBKWK 2014, 37). Even if programming or computer science in general are not explicitly mentioned, it surely can contribute to the development of media competence (Tulodziecki 2016) which includes knowledge and reflection on the structures of media (Schorb 2009). In the competence model of mathematics, the Bavarian primary school curriculum describes modelling and problem solving as one of the process-related competencies that students should develop during their time in primary school (StMBKWK 2014, 106).

Theoretical Framework

The literature findings were also very consistent with the results of our interviews with primary school teachers (see Funke et al. 2016). They suggested avoiding too much theory by a high practical relevance of the tasks and to take up the students' ideas and suggestions in order to achieve meaningful results. To show that teamwork is also very important in computer science, it was recommended that the students get opportunities to collaborate.

The results of the literature review and the interviews led to various design principles, to which we aligned the design of the course:

The course should facilitate active learning and collaborative work

Both in developmental psychology, as well as in the field of computer science education, researchers stress the importance of active learning. Also in our course, the students should have the possibility to be active from the beginning and to gradually get a deeper insight into the subject through their own actions. The focus should therefore be on student activity and hands-on experiences, avoiding long phases of front-facing teaching. Furthermore, we want to provide opportunities to work collaboratively, which is also described as beneficial for the students' skill development and their motivation. Especially when it comes to the handling of the computer, we assume there will certainly be some students who are well versed and could help other students when collaborating.

The course should go from the known to the unknown

Linking the teaching content and tasks to the students' everyday life and experience is a common practice not only in primary school. In our course, we want to build on the students' previous knowledge and provide examples and tasks to which the students can relate. This can refer to the form of the tasks as well as to the content or context. Especially at the beginning of the course, tasks should be created which they know in a similar form from other school subjects or everyday situations.

The course should enable the students to undergo different problem-solving steps and to solve problems independently

The development of problem-solving skills is not only important in terms of algorithms and programming but is also seen as a broadly applicable thinking competence required in various domains and subjects (Grover and Pea 2018). Since *modeling* and *problem-solving* can already be found in the Bavarian primary school curriculum, it could also be a way to cover algorithms and programming in school without having an own subject of computer science.

The course concept itself and the design of the tasks should put problem-solving in the foreground and give the students the opportunity of going through the different steps of the process. In addition, we want to prepare visual aids to help the students solve problems and guide them when working independently.

The course should give the students the opportunity to encounter the fundamental algorithmic structures in different ways

Following Bruner's argumentation that children should have the opportunity to encounter new concepts several times in various manners, we decided to use different teaching approaches. At first, they should get to know the computer science concepts with physical actions, which can be met very well with the CS unplugged approach that is distinguished by social activities, group problem-solving and high engagement of the students without actually working on computers (Bell, Witten, and Fellows 2015). After that, the students work within a programming environment on a computing device. To provide a child-friendly programming environment and avoid frustrating syntax errors, we decided to use the block-based language Scratch (Maloney et al. 2010).

Furthermore, we want to take into account the Use-Modify-Create Framework (Lee et al. 2011) and the PRIMM approach (Sentance and Waite 2017) and offer tasks where the students first work with existing programs and gradually make them their own until they create their own programs.

The course should reduce cognitive load for the students by focusing on the fundamental algorithmic structures and principles of programming

As the students already get familiar with the CS concepts in the unplugged activities, we also want to reduce the students' germane cognitive load while working in the programming environment. To further simplify working with Scratch, we already introduce the characteristic programming blocks during the unplugged part. Corresponding to the blocks in Scratch, we will manufacture haptic programming blocks for the processing of the tasks. Since the printed and laminated blocks are equipped with magnets and velcro straps, they can be used both on the blackboard and on sheets of fabric, e.g. felt. The idea is to improve orientation in the unfamiliar programming environment and decrease extraneous cognitive load, which could be caused by the numerous features and new impressions within Scratch.

The course should provide motivating and cognitive activating programming tasks

Our literature review revealed that motivation and cognitive activation have a major impact on students' learning results. In order to increase the students' personal motivation, we want to choose a specific context for the course that runs through the entire course, the tasks and materials. This context is to be linked with the students' personal experiences, it should enable a lively and entertaining communication of content and offer a broad spectrum of tasks. Furthermore, neither boys nor girls should be predominantly associated with the topic.

To develop motivating tasks, we also want to follow the results of Martin (2017), who concluded that motivating programming tasks are open, relevant to daily life experience, can be personalized and shared with others. In addition, the design of the tasks should focus on the cognitive activation of the students.

Course Design

During the course, the students are supposed to learn the basic principles of programming. They should understand how a computer program works, get familiar with the programming environment Scratch and be enabled to program their own multimedia projects. At the same time, they ought to enjoy the course and should have the opportunity to put their creativity into practice.

The course takes place over three days, on which we spend four hours a day with the children. The theme «circus» runs through the course and most of the tasks as a consistent motif. Figure 2 shows a schematic overview of the «programming circus», its content and learning objectives.

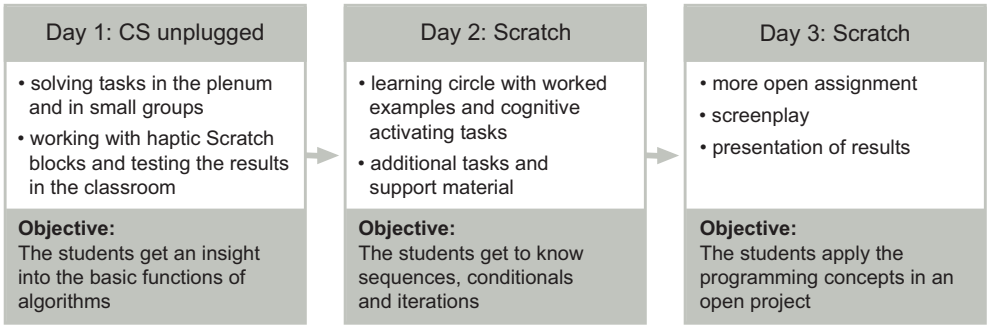

Fig. 2.: Schematic overview of the course.

Day 1: Computer Science unplugged

Since most of the students do not have any prior knowledge in programming or general in computer science, the goal of the first day is to give them a basic idea of how a computer program works. As a first step, the students' ideas and associations which they relate to programming are collected in a short query and noted down on the board or a poster. Depending on how much previous knowledge the students have, it is possible to continue working with images on which different situations are displayed in which programming plays a role.

As an introduction to giving precise and clear instructions, the students get the task of programming the teacher to do a certain task (e.g. open the window, walk into a certain corner of the classroom). The teacher acts like a robot and only moves if given the right commands. How to communicate with the robot must be found out by the students. To further practice giving commands, the students have to convert a pictorial craft instruction for a name tag into unambiguous language-based commands. Afterwards, they build their name tags following one of the two instructions. In these first exercises, it quickly becomes clear that it is not easy to formulate something in such a way that it means the same for everyone. The teacher discusses with the students that this is one reason why there are defined programming languages for Computers, e.g. Scratch. To get to know the first programming blocks, we once more use the «human robot». Instead of using everyday commands, the students have to use haptic Scratch blocks to solve a short task.

In order to become familiar with the haptic Scratch blocks, the students work in groups to solve more complex tasks in a grid that is built up from carpet tiles (see. Fig. 3). To take up the circus theme, they program circus characters to complete different missions, e.g., a monkey that wants to be led to his banana or a forgetful clown who wants to find his missing belongings (see Fig. 4). They create the programs using the haptic Scratch blocks, execute it in the grid and check them for errors. If necessary, they debug their programs.

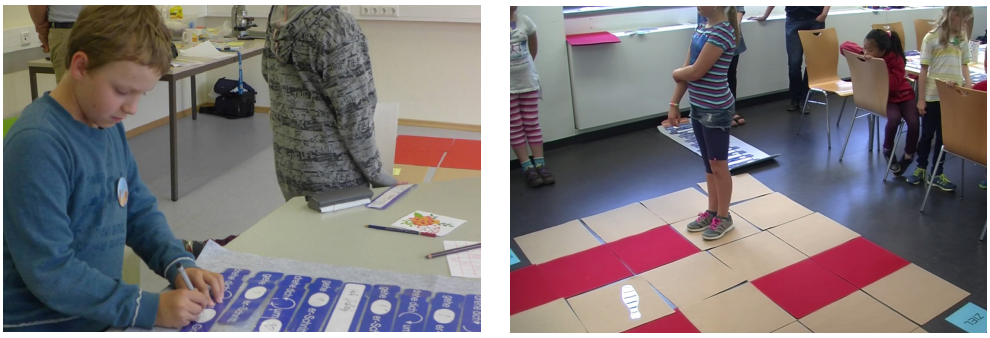

Fig. 3.: Working with the haptic programming blocks (left); testing the solutions in the field (right).

We designed the tasks that they can be solved by using selections and iterations, but also by sequences. To help the students – particularly at the beginning – with their tasks, we prepared signs that show the different steps that must be carried out during the problem-solving process. They can be hung up in the classroom and help the students when they are working on a task.

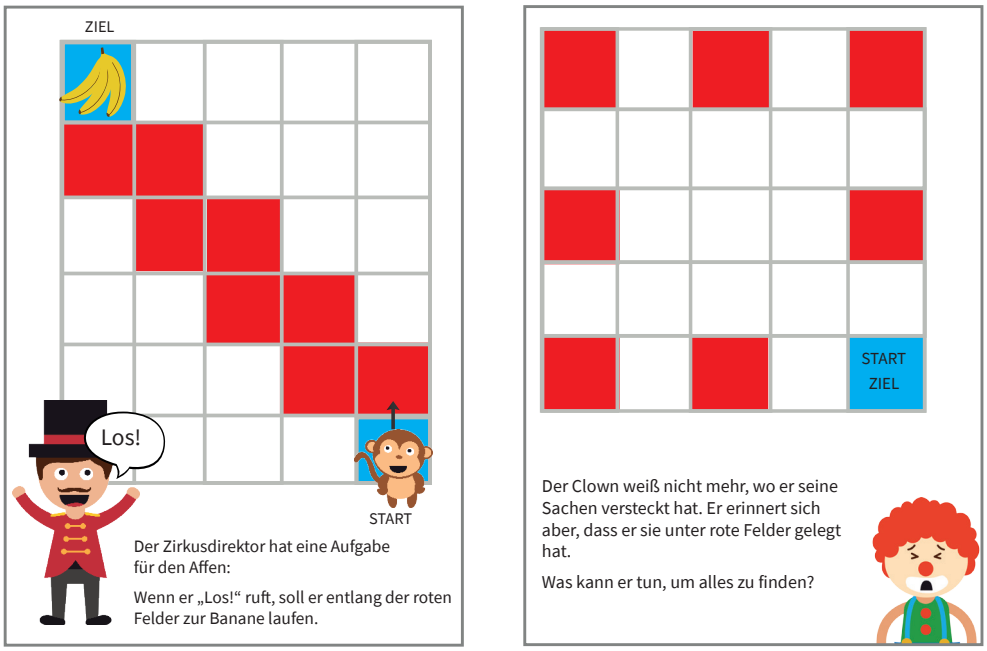

Fig. 4.: Sections from the students' task sheets.

Day 2: Scratch Learning Circle

On the second day, we want to enable the students to create simple multimedia products in the block-based programming environment Scratch. Thus, every student works on a computer. We composed a learning circle in which the core operations

of Scratch get gradually introduced and that every child can handle at its own pace. Starting from questions regarding the software handling, the individual circle cards lead from simple sequences to the implementation of iterations and conditionals. For example, the students program the welcome greeting from the circus director (see Fig. 5), a joke-telling clown, a tiger crossing the arena and a dancing bear.

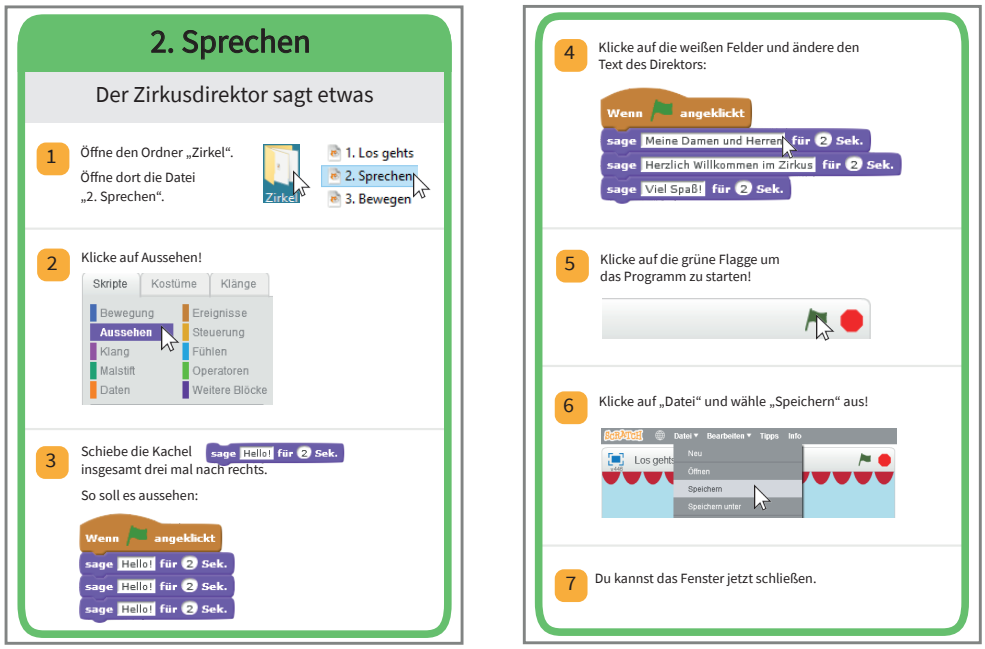

Fig. 5.: Front (left) and back (right) of a circle card.

To reduce extraneous cognitive load, we provide worked examples that the students recreate on their own computing devices. After every example, they work on an exercise that has been designed to comply with the principles of cognitive activating tasks: the introduced function must be slightly modified or used in a different context. To face the students' expected variety in knowledge and learning pace, we prepared additional exercises as well as helpful tips for the more complex tasks.

When designing the materials for the learning circle, we tried to realize several principles of Mayer's cognitive theory of multimedia learning. For example, we used recurring colors to highlight the organization of the cards, we addressed the students directly and tried to use both, words and pictures, and placed them near to each other.

Day 3: Open Scratch Projects

Our goal for the third day was to get an impression of what the students have learned so far and what they can apply in more open tasks. The students should think of their own Scratch stories, write these down in a script (Figure 6, left) and implement these in Scratch (Figure 6, right). To get comparable results, we set the following mandatory requirements for the students' projects. The programs should 1) work on more than one sprite 2) move the sprites during execution 3) comprise at least one loop and 4) include at least one conditional statement. After meeting these requirements, the students should continue their programming work without any further guidelines.

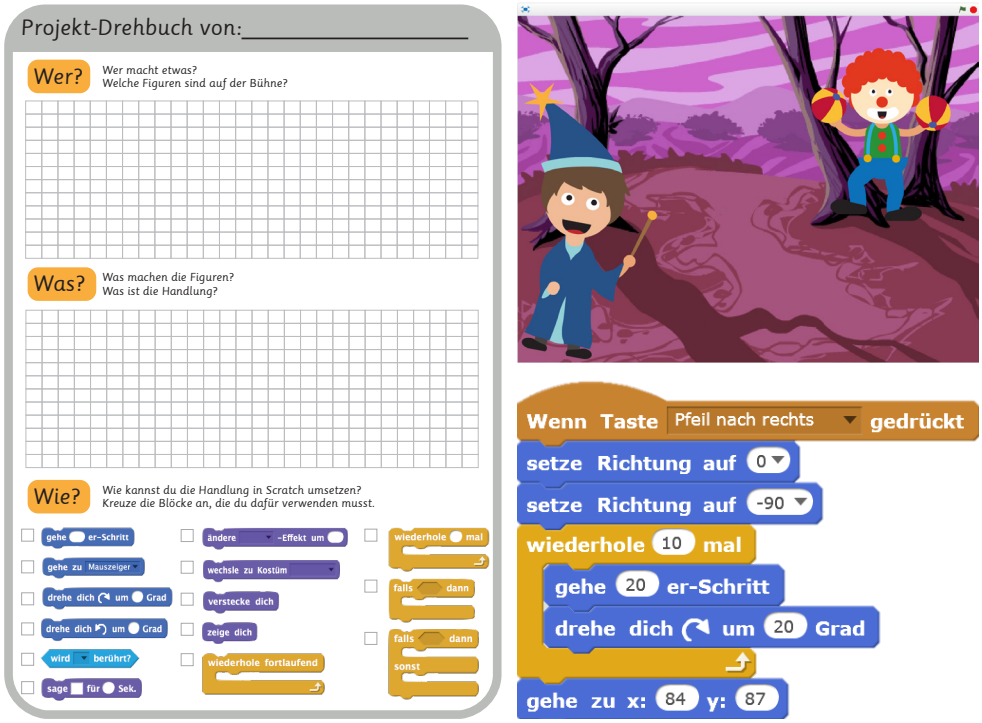

Fig. 6.: Project script (left) and part of a student Scratch project (right).

At the end of the third course day, the students present their programs in front of the class and have the opportunity to comment on their project. There is an applause for each project, and the students get a certificate for the participation in the course. The programming circus ends with a reflection on what the students think they have learned, what they liked the best and what they didn't like the previous days.

Conclusion and Future Work

The design-based research approach provided a viable research framework for the development of a programming course concept for primary schools. It enabled us to take into account relevant theories from various disciplines and best practices in the field of computer science education. The iterative design cycles and evaluations made it possible to focus on a variety of course elements and to apply multiple research methods. In this way, aspects could also be investigated that were not planned in the first place and the course concept could be continuously improved.

However, the openness of the methodology can also be a challenge. Regarding the phase of *preliminary research* that is described in this article, there are no specific guidelines on how to carry out the literature review. The fact that in our case researchers with different professional backgrounds were involved in the selection of the relevant thematic areas and theories was very helpful. By drawing on the expertise of a trained primary school teacher and a researcher in computer science education, we were able to perform a very comprehensive and varied literature review. This interdisciplinary collaboration was also advantageous in the development of the theoretical framework and the course concept.

A shortage of skills or knowledge among the researchers can have a very limiting effect on the intermediate results of the DBR process. If they lack sufficient knowledge and experience, they consequently are not able to achieve optimal solutions. Especially when, as in our case, the researchers are also evaluators and implementers, it is important to make research open to professional scrutiny and criticism by people outside the project (McKenney, Nieveen, and van den Akker 2006). We were fortunate that we regularly had the opportunity to discuss both the theoretical framework and the course concept within our team that consists of people with diverse knowledge and levels of experience.

Despite these challenges, we consider DBR to be an important innovation opportunity for didactic research since the transfer between theory and practice is one of the most important mission when being a researcher in computer science education. Through the synergy of theory and practice, design-based research can lead to development and improvement in both areas.

Our future work will focus on further assessments and analyses of the data we collected in the field tests. In addition, we will investigate whether primary school teachers can work with the course concept and are willing to apply it in their teaching.

Based on the course concept, we already developed an in-service professional development workshop for primary school teachers (Geldreich, Talbot, and Hubwieser 2018). Thirty-nine teachers from 20 primary schools all over Bavaria have already participated in the teacher training and will try out the course concept with their students. In doing so, we want to ensure that our preliminary results, which

are very setting-specific, can be generalized and replicated with a larger sample and real-life teaching conditions. The results of our future research will be used to further specify the design principles for primary school programming courses and thus will hopefully make a research contribution not only on a practical but also on a theoretical level.

References

Akker, Jan van den, Koeno Gravemeijer, Susan McKenney, and Nienke Nieveen, eds. 2006. *Educational Design Research*. New York: Routledge.

Armoni, Michal. 2018. «Training Teachers for K-6 Computing Education». *ACM Inroads* 9 (3): 18–19. https://doi.org/10.1145/3231600.

Baumert, Jürgen, Mareike Kunter, Werner Blum, Martin Brunner, Thamar Voss, Alexander Jordan, Uta Klusmann, Stefan Krauss, Michael Neubrand, and Yi-Miau Tsai. 2010. «Teachers' Mathematical Knowledge, Cognitive Activation in the Classroom, and Student Progress». *American Educational Research Journal* 47 (1): 133–180. https://doi.org/10.3102/0002831209345157.

Bell, Tim, and Caitlin Duncan. 2018. «Teaching Computing in Primary Schools». In *Computer Science Education*, edited by Sue Sentance, Erik Barendsen, and Carsten Schulte. London and New York and Oxford and New Delhi and Sydney: Bloomsbury Academic.

Bell, Tim, Ian H. Witten, and Mike Fellows. 2015. *CS Unplugged: An Enrichment and Extension Programme for Primary-Aged Students*. 3rd ed.

Berry, Miles. 2013. *Computing in the National Curriculum: A Guide for Primary Teachers*. Bedford: Newnorth Print, Ltd.

BMBF. 2016. *Bildungsoffensive für die digitale Wissensgesellschaft: Strategie des Bundesministeriums für Bildung und Forschung*. Berlin: Bundesministerium für Bildung und Forschung.

Brown, Ann L. 1992. «Design Experiments: Theoretical and Methodological Challenges in Creating Complex Interventions in Classroom Settings». *The Journal of the Learning Sciences* 2 (2): 141–178. https://doi.org/10.1207/s15327809jls0202_2.

Brown, Neil C. C., Sue Sentance, Tom Crick, and Simon Humphreys. 2013. «Restart: The Resurgence of Computer Science in UK Schools». *ACM Transactions on Computing Education* 1 (1). https://doi.org/10.1145/2602484.

Brühwiler, Christian, and Peter Blatchford. 2011. «Effects of Class Size and Adaptive Teaching Competency on Classroom Processes and Academic Outcome». *Learning and Instruction* 21 (1): 95–108. https://doi.org/10.1016/j.learninstruc.2009.11.004.

Bruner, Jérome. 1966. *Towards a Theory of Instruction*. Cambridge: Harvard University Press.

Case, Robbie. 1992. «Neo-Piagetian Theories of Child Development». In *Intellectual Development*, edited by Robert J. Sternberg and Cynthia A. Berg. New York: Cambridge University Press.

Caspersen, Michael E., and Jens Bennedsen. 2007. «Instructional Design of a Programming Course – A Learning Theoretic Approach». In *Proceedings of the Third International Workshop on Computing Education Research*. New York, NY: ACM. https://doi.org/10.1145/1288580.1288595.

Chandler, Paul, and John Sweller. 1991. «Cognitive Load Theory and the Format of Instruction». *Cognition and Instruction* 8 (4): 193–332. https://doi.org/10.1207/s1532690xci0804_2.

Collins, Allan. 1992. «Towards a Design Science of Education». In *New Directions in Educational Technology*, edited by Eileen Scanlon and Tim O'Shea, 15–22. New York: Springer. https://doi.org/10.1007/978-3-642-77750-9_2.

Collins, Allan, Diana Joseph, and Katerine Bielaczyc. 2004. «Design Research: Theoretical and Methodological Issues». *The Journal of the Learning Sciences* 13 (1): 15–42. https://doi.org/10.1207/s15327809jls1301_2.

Creemers, Bert, and Leonidas Kyriakides. 2007. *The Dynamics of Educational Effectiveness: A Contribution to Policy, Practice and Theory in Contemporary Schools*. London, New York: Routledge.

Duncan, Caitlin, Tim Bell, and Steve Tanimoto. 2014. «Should Your 8-Year-Old Learn Coding?» In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education*, 60–69. New York, NY, USA: ACM. https://doi.org/10.1145/2670757.2670774.

Dunkin, Michael J., and Bruce Biddle. 1974. *The Study of Teaching*. New York: Holt, Rinehart and Winston.

Falkner, Katrina, Rebecca Vivian, and Nickolas Falkner. 2014. «The Australian Digital Technologies Curriculum: Challenge and Opportunity». In *Proceedings of the Sixteenth Australasian Computing Education Conference*, edited by Jacqueline Whalley and Daryl D'Souza. New York: ACM.

Franklin, Diana, Charlotte Hill, Hilary Dwyer, Ashley Iveland, Alexandria Killian, and Danielle Harlow. 2015. «Getting Started in Teaching and Researching Computer Science in the Elementary Classroom». In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, 552–557. SIGCSE '15. New York, NY, USA: ACM. https://doi.org/10.1145/2676723.2677288.

Funke, Alexandra, Marc Berges, and Peter Hubwieser. 2016. «Different Perceptions of Computer Science». In *2016 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)*, edited by Sridhar Iyer and Neena Thota, 14–18. IEEE. https://doi.org/10.1109/LaTiCE.2016.1.

Funke, Alexandra, and Katharina Geldreich. 2017. «Gender Differences in Scratch Programs of Primary School Children». In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education – WiPSCE '17*, 57–64. Nijmegen, Netherlands: ACM Press. https://doi.org/10.1145/3137065.3137067.

Funke, Alexandra, Katharina Geldreich, and Peter Hubwieser. 2016. «Primary School Teachers' Opinions about Early Computer Science Education». In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research - Koli Calling '16*, 135–139. New York, NY, USA: ACM. https://doi.org/10.1145/2999541.2999547.

Funke, Alexandra, Katharina Geldreich, and Peter Hubwieser. 2017. «Analysis of Scratch Projects of an Introductory Programming Course for Primary School Students». In *Proceedings of the 2017 IEEE Global Engineering Education Conference (EDUCON)*, 1229–1236. IEEE.

Geldreich, Katharina, Mike Talbot, and Peter Hubwieser. 2018. «Off to New Shores: Preparing Primary School Teachers for Teaching Algorithmics and Programming». In *Proceedings of the 13th Workshop in Primary and Secondary Computing Education on – WiPSCE '18*, 1–6. Potsdam, Germany: ACM Press. https://doi.org/10.1145/3265757.3265783.

Gray, Simon, Caroline St. Clair, Richard James, and Jerry Mead. 2007. «Suggestions for Graduated Exposure to Programming Concepts Using Fading Worked Examples». In *Proceedings of the Third International Workshop on Computing Education Research*, 99–110. New York, NY: ACM. https://doi.org/10.1145/1288580.1288594.

Grover, Shuchi, and Roy Pea. 2018. «Computational Thinking: A Competency Whose Time Has Come». In *Computer Science Education*, edited by Sue Sentance, Erik Barendsen, and Carsten Schulte, 19–38. London and New York and Oxford and New Delhi and Sydney: Bloomsbury Academic.

Hazzan, Orit, Tami Lapidot, Noa Ragonis. 2011. *Guide to Teaching Computer Science: An Activity-Based Approach*. London: Springer. https://doi.org/10.1007/978-0-85729-443-2.

Howland, Jane L., Jonassen, David H., and Rose M. Marra. 2011. *Meaningful Learning with Technology*. 4th ed. London: Pearson Education.

Hubwieser, Peter. 2012. «Computer Science Education in Secondary Schools – The Introduction of a New Compulsory Subject». *ACM Transactions on Computing Education* 12 (4): 1–41. https://doi.org/10.1145/2382564.2382568.

Jahn, Dirk. 2014. «Durch Das Praktische Gestalten von Didaktischen Designs Nützliche Erkenntnisse Gewinnen: Eine Einführung in Die Gestaltungsforschung». *W&E* 66 (1): 3–15.

Jong, Ton de. 2010. «Cognitive Load Theory, Educational Research, and Instructional Design: Some Food for Thought». *Instructional Science* 38 (2): 105–134. https://doi.org/10.1007/s11251-009-9110-0.

Keller, John M. 1983. «Motivational Design of Instruction». In *Instructional-Design Theories and Models: An Overview of Their Current Status*, edited by Charles M. Reigeluth, 383–434. Mawah: Lawrence Erlbaum Associates Inc.

Keller, John M. 1987. «Development and Use of the ARCS Model of Instructional Design». *Journal of Instructional Development* 10 (3): 2–10. https://doi.org/10.1007/BF02905780.

Khenner, Evgeniy, and Igor Semakin. 2014. «School Subject Informatics (Computer Science) in Russia». *ACM Transactions on Computing Education* 14 (2): 1–10. https://doi.org/10.1145/2602489.

Klieme, Eckhard, Christine Pauli, and Kurt Reusser. 2009. «The Pythagoras Study. Investigating Effects of Teaching and Learning in Swiss and German Mathematics Classrooms». In *The Power of Video Studies in Investigating Teaching and Learning in the Classroom*, edited by Janik Tomás and Tina Seidel, 137–160. Münster: Waxmann.

Lee, Irene, Fred Martin, Jill Denner, Bob Coulter, Walter Allan, Jeri Erickson, Joyce Malyn-Smith, and Linda Werner. 2011. «Computational Thinking for Youth in Practice». *ACM Inroads* 2 (1): 32. https://doi.org/10.1145/1929887.1929902.

Maloney, John, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. 2010. «The Scratch Programming Language and Environment». *ACM Transactions on Computing Education* 10 (4): 1–15. https://doi.org/10.1145/1868358.1868363.

Martin, Christopher, James. 2017. *Designing Engaging Learning Experiences in Programming: Dissertation*. Dundee: University of Dundee.

Mayer, Richard E. 2001. *Multimedia Learning*. New York: Cambridge University Press.

Mayer, Richard E. 2004. «Should There Be a Three-Strikes Rule Against Pure Discovery Learning?». *American Psychologist* 59 (1): 14–19. https://doi.org/10.1037/0003-066X.59.1.14.

Mayer, Richard E. 2005. «Introduction to Multimedia Learning». In *The Cambridge Handbook of Multimedia Learning*, edited by Richard E. Mayer, 1–26. New York: Cambridge University Press.

McKenney, Susan, Nienke Nieveen, and Jan van den Akker. 2006. «Design Research from a Curriculum Perspective». In *Educational Design Research*, edited by Jan van den Akker, Koeno Gravemeijer, Susan McKenney, and Nienke Nieveen, 67–90. New York: Routledge.

Merrill, David M., Leston Drake, Mark J. Lacy, and Jean Pratt. 1996. «Reclaiming Instructional Design». *Educational Technology* 36 (5): 5–7. http://m.firstprinciplesofinstruction.com/Papers/Reclaiming.PDF.

Morra, Sergio, Camilla Gobbo, Zopito Marini, and Ronald Sheese. 2007. *Cognitive Development: Neo-Piagetian Perspectives*. New York: Psychology Press.

OECD. 2013. *PISA 2012 Results: Ready to Learn: Students' Engagement, Drive and Self-Beliefs (Volume III)*. 2013. Paris: OECD.

Petre, Marian, and Blaine Price. 2004. «Using Robotics to Motivate `Back Door' Learning». *Education and Information Technologies* 9 (2): 147–158. https://doi.org/10.1023/B:EAIT.0000027927.78380.60.

Piaget, Jean. 1976. *The Child and Reality*. New York: Penguin Books.

Piaget, Jean, and Bärbel Inheler. 1969. *The Psychology of the Child*. New York: Basic Books.

Pianta, Robert C., and Bridget K. Hamre. 2009. «Conceptualization, Measurement, and Improvement of Classroom Processes: Standardized Observation Can Leverage Capacity». *Educational Researcher* 38 (2): 109–119. https://doi.org/10.3102/0013189X09332374.

Plomp, Tjeerd. 2007. «Educational Design Research: An Introduction». In *An Introduction to Educational Design Research*, edited by Tjeerd Plomp and Nienke Nieveen, 9–35. Enschede: SLO.

Prottsman, Kiki. 2014. «Computer Science for the Elementary Classroom». *ACM Inroads* 5 (4): 60–63. https://doi.org/10.1145/2684721.2684735.

Reinmann, Gabi. 2005. «Innovation Ohne Forschung? Ein Plädoyer Für Den Design-Based Research-Ansatz in Der Lehr-Lernforschung». *Unterrichtswissenschaft* 33 (1): 52–69.

Richey, Rita C., and James D. Klein. 2005. «Developmental Research Methods: Creating Knowledge from Instructional Design and Development Practice». *Journal of Computing in Higher Education* 16 (2): 23–38. https://doi.org/10.1007/BF02961473.

Schorb, Bernd. 2009. «Gebildet Und Kompetent. Medienbildung Statt Medienkompetenz?» *Medien+ Erziehung*, no. 5: 50–56.

Seidel, Tina, and Richard J. Shavelson. 2007. «Teaching Effectiveness Research in the Past Decade: The Role of Theory and Research Design in Disentangling Meta-Analysis Results». *Review of Educational Research* 77 (4): 454–499. https://doi.org/10.3102/0034654307310317.

Sentance, Sue, and Jane Waite. 2017. «PRIMM: Exploring Pedagogical Approaches for Teaching Text-Based Programming in School». In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education – WiPSCE '17*, 113–14. Nijmegen, Netherlands: ACM Press. https://doi.org/10.1145/3137065.3137084.

Shuell, Thomas. 1996. «Teaching and Learning in a Classroom Context». In *Handbook of Educational Psychology*, edited by David C. Berliner and Robert C. Calfee, 726–764. New York: Macmilan.

StMBKWK. 2014. *LehrplanPLUS Grundschule: Lehrplan für die Bayerische Grundschule*. München: Bayerisches Staatsministerium für Bildung und Kultus, Wissenschaft und Kunst.

StMWI. 2017. *Bayern Digital II: Investitionsprogramm für die Digitale Zukunft Bayerns*. München: Bayerisches Staatsministerium für Wirtschaft, Landesentwicklung und Energie.

Sweller, John, Jeroen J. G. van Merriënboer, and Fred G. W. C. Paas. 1998. «Cognitive Architecture and Instructional Design». *Educational Psychology Review* 10 (3): 251–296.

The Design-Based Research Collective. 2003. «Design-Based Research: An Emerging Paradigm for Educational Inquiry». *Educational Researcher* 32 (1): 5–8.

Topi, Heikki. 2015. «Gender Imbalance in Computing». *ACM Inroads* 6 (4): 22–23. https://doi.org/10.1145/2822904.

Tulodziecki, Gerhard. 2016. «Konkurrenz Oder Kooperation? Zur Entwicklung Des Verhältnisses von Medienbildung Und Informatischer Bildung». *MedienPädagogik: Zeitschrift Für Theorie Und Praxis Der Medienbildung* (25: Medienbildung Und Informatische Bildung – Quo Vadis?): 7-25. https://doi.org/10.21240/mpaed/25/2016.10.25.X.

Tuovinen, Juhani E. 2000. «Optimising Student Cognitive Load in Computer Education». In *Proceedings of the Australasian Conference on Computing Education*, 235–241. New York, NY, USA: ACM. https://doi.org/10.1145/359369.359405.

Vygotsky, Lew Semjonowitsch. 1978. *Mind in Society: The Development of Higher Psychological Processes*. Cambridge: Harvard University Press.

Wang, Feng, and Michael J. Hannafin. 2005. «Design-Based Research and Technology-Enhanced Learning Environments». *Educational Technology Research and Development* 53 (4): 5–23. https://doi.org/10.1007/BF02504682.

Weinert, Franz E., Friederich-W. Schrader, and Andreas Helmke. 1989. «Quality of Instruction and Achievement Outcomes». *International Journal of Educational Research* 13 (8): 895–914. https://doi.org/10.1016/0883-0355(89)90072-4.

Winne, Philip H. 1987. «Why Process-Product Research Cannot Explain Process-Product Findings and a Proposed Remedy: The Cognitive Mediational Paradigm». *Teaching and Teacher Education* 3 (4): 333–356.