Technische Universität München
TUM School of Social Sciences and Technology

# Programmieren in der Grundschule
## Eine Design-Based-Research-Studie

Katharina Geldreich

Vollständiger Abdruck der von der TUM School of Social Sciences and Technology der Technischen Universität München zur Erlangung des akademischen Grades einer

**Doktorin der Philosophie (Dr. phil.)**

genehmigten Dissertation.

Vorsitz:

Prof. Dr. Claudia Nerdel

Prüfer*innen der Dissertation:

1. Prof. Dr. Peter Hubwieser
2. Prof. Dr. Uta Hauck-Thum
3. Prof. Dr. Johannes Krugel

Die Dissertation wurde am 05.04.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Social Sciences and Technology am 28.06.2023 angenommen.

Für Marianne und Manuel

Für Monika, Mama und Papa

# Zusammenfassung

Da Kinder in einer durch die Digitalisierung geprägten Welt aufwachsen und somit immer früher mit neuen Technologien und Medien in Kontakt kommen, sollten diese Aspekte auch in der Grundschule berücksichtigt werden. Der Aufbau von informatischen Kompetenzen wird im Rahmen des Grundschulunterrichts jedoch in der Regel nicht ermöglicht. Da sich das Programmieren für das Verstehen und aktive Gestalten der digitalen Welt im besonderen Maße als Unterrichtsgegenstand zu eignen scheint, wird der Themenbereich in dieser Arbeit für den Einsatz in der Grundschule aufbereitet. Dabei werden verschiedene Ebenen berücksichtigt: der *Unterricht*, die *Lehrerfortbildung* und entsprechende *Rahmenbedingungen*. Als Forschungsansatz für die Studie wird der Ansatz der *Design-Based Research* gewählt, der sich besonders für die Entwicklung nachhaltiger Innovationen für den Unterricht eignet.

Zunächst werden ein Unterrichtskonzept und Materialien für das Programmieren in der Grundschule entwickelt. Dabei wird nicht nur auf die Informatik und Informatikdidaktik Bezug genommen, sondern auch auf Erkenntnisse der Entwicklungspsychologie sowie Grundschuldidaktik und -pädagogik. Das Unterrichtskonzept wird im Rahmen einer Pilotstudie und zwei weiteren Erprobungszyklen mit 115 Schülerinnen und Schülern der dritten und vierten Jahrgangsstufe in unterrichtsnahen Lehr-Lernsituationen erprobt und weiterentwickelt. Anschließend wird die Unterrichtssequenz von 41 Grundschullehrkräften in ihrem Unterricht erprobt, angepasst und erweitert. Um Letzteres zu ermöglichen, wird ausgehend von der Unterrichtssequenz und entsprechenden Befunden im Bereich der Lehrerbildung ein passendes Fortbildungskonzept für Grundschullehrkräfte entworfen, erprobt und evaluiert.

Um zu untersuchen, wie das Konzept an den Schulen implementiert wird und welche Auswirkungen dies auf die Schülerinnen und Schüler hat, werden explorative Interviews mit den Lehrkräften geführt und von ihnen erstellte Unterrichtsmaterialien ausgewertet. Die Interviews werden zusätzlich im Hinblick auf notwendige Rahmenbedingungen für das Programmieren in der Grundschule und das Fortbildungskonzept ausgewertet. Um die Auswirkungen des Fortbildungskonzepts auf die Lehrkräfte zu beleuchten, wird eine Fragebogenstudie mit vier Messzeitpunkten durchgeführt. Dabei stehen unter anderem die Selbstwirksamkeitserwartung der Lehrkräfte in Bezug auf ihre Programmierfähigkeiten und die Umsetzung der Fortbildungsinhalte im Unterricht im Fokus. Die Ergebnisse werden zudem durch Aussagen aus den Interviews ergänzt.

Die Ergebnisse der Zyklen und Untersuchungen münden schließlich in konsolidierte Leitlinien für die Umsetzung des Programmierens in der Grundschule. Diese beziehen sich sowohl auf die unterrichtliche Umsetzung als auch auf die Fortbildung von Lehrkräften. Darüber hinaus werden mögliche Bezüge zum Lehrplan der bayerischen Grundschule aufgezeigt.

# Abstract

As children grow up in a world shaped by digitization and thus come into contact with new technologies and media at an increasingly earlier age, these aspects should also be taken into account in primary school. However, the development of computer science competencies is not usually facilitated within primary school teaching. Since programming seems to be a particularly suitable skill for understanding and actively shaping the digital world, the subject area is prepared in this thesis for implementation in primary school. Thereby, different levels are considered: *teaching*, *in-service teacher training*, and required *prerequisites*. The research approach chosen for the thesis is *Design-Based Research*, which is particularly suitable for developing sustainable innovations for teaching.

First, a teaching concept and materials for programming in primary school are developed. The concept is based not only on computer science and computer science education but also on findings from developmental psychology and primary school pedagogy. The designed teaching sequence will be tested, evaluated, and further developed within a pilot study and two test cycles with 115 third and fourth-grade students in extracurricular learning situations. Subsequently, the teaching sequence will be tested, adapted, and extended by 41 primary school teachers in their lessons. To make the latter possible, a suitable in-service training concept for primary school teachers will be designed, tested, and evaluated based on the teaching sequence and corresponding findings in the field of teacher education.

In order to investigate how the concept is implemented at the schools and its effects on the students, exploratory interviews are conducted with the teachers and teaching materials created by them are evaluated. The interviews are additionally analyzed with regard to necessary prerequisites for programming in primary schools and the training concept. To investigate the effects of the in-service teacher training on the teachers, a questionnaire study with four measuring points will be conducted. Here, the focus is, among other things, on the teachers' self-efficacy expectations regarding programming and its implementation in the classroom. The results will also be supplemented by statements from the interviews.

The results of the cycles and surveys finally lead to consolidated guidelines for implementing programming in primary schools. These refer both to the implementation in class and in-service teacher training. In addition, possible connections to the curriculum of the Bavarian primary school are presented.

# Danksagung

Diese Arbeit wäre ohne die Unterstützung zahlreicher Personen nicht möglich gewesen, denen ich an dieser Stelle danken möchte. Zuallererst bedanke ich mich aufs herzlichste bei meinem Doktorvater Prof. Dr. Peter Hubwieser, der mich in allen Stadien meines Forschungsvorhabens mit seiner fachlichen Expertise sowie konstruktivem Feedback und viel Vertrauen in meine Arbeit begleitet hat. Zudem danke ich meinen GutachterInnen Prof. Dr. Uta Hauck-Thum und Prof. Dr. Johannes Krugel für ihre Unterstützung und Anregungen.

Bedanken möchte ich mich außerdem bei Alexandra Simon, die mich auf einem großen Teil meiner Forschungsreise begleitet hat, für ihr Feedback zu der in dieser Arbeit entwickelten Unterrichtssequenz und für ihre Hilfe bei der Organisation, Durchführung und Auswertung der ersten Erprobungen. Ein weiterer Dank geht an Prof. Dr. Marc Berges, Christine Lutz und Mike Talbot, die mich tatkräftig in meinem Forschungsvorhaben unterstützt haben und meine Zeit an der TUM zu etwas ganz Besonderem gemacht haben. Danken möchte ich zudem Hannah Ertl und Christina Ioanna Pappa, die mich während ihrer Zeit als Hilfskräfte bei der Aufbereitung und Auswertung der Interviews und Fragebögen unterstützt haben.

Weiterer Dank gilt allen Schülerinnen und Schülern, die an den Erprobungen teilgenommen haben und sich dem Programmieren voller Neugierde und Lernfreude genähert haben. Ein großer Teil dieser Arbeit wäre zudem nicht möglich gewesen ohne die engagierten und wagemutigen Lehrkräfte, die am *AlgoKids*-Projekt teilgenommen haben. Ihre Offenheit, Erfahrungen und Ideen haben diese Arbeit enorm bereichert.

Besonders bedanken möchte ich mich bei meinem Lebensgefährten Manuel, der mich seit Beginn meiner Arbeit unterstützt hat und ohne den ich diese Arbeit nicht hätte fertigstellen können. Mein besonderer Dank gilt außerdem meiner Familie und meinen Freunden, die immer an mich geglaubt und ein offenes Ohr für mich gehabt haben.

# Inhaltsverzeichnis

# Veröffentlichungen

Teile dieses Forschungsvorhabens wurden bereits vor der Einreichung der vorliegenden Dissertation veröffentlicht. Die folgende Liste gibt einen Überblick über die Veröffentlichungen und die entsprechenden Kapitel.

- Kapitel 14
  - Funke, Alexandra, Katharina Geldreich und 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). New York, NY, USA: ACM, S. 135–139.
  - Geldreich, Katharina, Alexandra Funke und Peter Hubwieser (2016). „A Programming Circus for Primary Schools". In: Informatics in Schools: Improvement of Informatics Knowledge and Perception. Hrsg. von Andrej Brodnik und Françoise Tort. Cham: Springer, S. 46–47.
  - Geldreich, Katharina, Alexandra Funke und Peter Hubwieser (2017). „Willkommen im Programmierzirkus – Ein Programmierkurs für Grundschulen". In: Informatische Bildung zum Verstehen und Gestalten der digitalen Welt (INFOS). Hrsg. von Ira Diethelm. Bonn: Gesellschaft für Informatik, S. 327–334.
  - Geldreich, Katharina, Alexandra Simon und Peter Hubwieser (2019). „A Design-Based Research Approach for introducing Algorithmics and Programming to Bavarian Primary Schools: Theoretical Foundation and Didactic Implementation". In: MedienPädagogik: Zeitschrift für Theorie Und Praxis der Medienbildung 33 (Februar), S. 53-75.
- Kapitel 15
  - Geldreich, Katharina, Mike Talbot und Peter Hubwieser (2018). „Off to new shores: Preparing Primary School Teachers for Teaching Algorithmics and Programming". In: Proceedings of the 13th Workshop on Primary and Secondary Computing Education (WiPSCE), S. 139-144.
  - Geldreich, Katharina, Mike Talbot und Peter Hubwieser (2019). „Aufgabe ist nicht gleich Aufgabe – vielfältige Aufgabentypen bewusst in Scratch einsetzen". In: Informatische Bildung zum Verstehen und Gestalten der digitalen Welt (INFOS). Hrsg. von Arno Pasternak. Bonn: Gesellschaft für Informatik, S. 181-190.

- Kapitel 16
  - Funke, Alexandra, Katharina Geldreich und 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). IEEE, S. 1229–1236.
  - Funke, Alexandra und 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). Hrsg. von Erik Barendsen. New York, NY: ACM, S. 57–64.
  - Simon, Alexandra, Katharina Geldreich und Peter Hubwieser (2019). „How to Transform Programming Processes in Scratch to Graphical Visualizations“. In: Proceedings of the 14th Workshop in Primary and Secondary Computing Education on (WiPSCE). Hrsg. von Quintin Cutts und Thorsten Brinda. New York, NY: ACM, S. 1–9.
- Kapitel 17 und 18
  - Geldreich, Katharina und Peter Hubwieser (2020). „Implementierung einer Unterrichtssequenz zu Algorithmen und Programmierung in der Grundschule. Eine qualitative Interviewstudie mit Grundschullehrkräften“. In: Digitale Bildung im Grundschulalter. Grundsatzfragen zum Primat des Pädagogischen. Hrsg. von Mareike Thumel, Rudolf Kammerl und Thomas Irion. München: kopaed.
  - Geldreich, Katharina und Peter Hubwieser (2020). „Programming in Primary Schools: Teaching on the Edge of formal and non-formal Learning“. In: Non-Formal and Informal Science Learning in the ICT Era. Hrsg. von Michail Giannakos. Singapore: Springer.
  - Hubwieser, Peter und Katharina Geldreich (2021): „Algorithmen für Kinder (AlgoKids). Abschlussbericht eines Kooperationsprojekts der Technischen Universität München und des Bayerischen Staatsministeriums für Unterricht und Kultus“. Online verfügbar unter https://mediatum.ub.tum.de/1715848.

# Abkürzungen

| | |
|---|---|
| AG | Arbeitsgemeinschaft |
| ALP | Akademie für Lehrerfortbildung und Personalführung |
| ATL | *Aspects of Teaching and Learning* |
| BMBF | Bundesministerium für Bildung und Forschung |
| BMDV | Bundesministerium für Digitales und Verkehr |
| BMWi | Bundesministerium für Wirtschaft und Energie |
| Bsp. | Beispiel |
| bspw. | beispielsweise |
| bzw. | beziehungsweise |
| bzgl. | bezüglich |
| CC | *Creative Commons* |
| CAS | *Computing at School* |
| CDC | *Curriculum Development Council* |
| CS | *Computer Science* |
| CSTA | *Computer Science Teachers Association* |
| CSU | Christlich-Soziale Union |
| CT | *Computational Thinking* |
| DBR | *Design-Based Research* |
| DBRC | *Design-Based Research Collective* |
| d. h. | das heißt |
| DIVSI | Deutsches Institut für Vertrauen und Sicherheit im Internet |
| DP | *Design Principle* |
| ebd. | ebenda |
| etc. | et cetera |
| f. bzw. ff. | folio bzw. folio (Dopplung) |
| FB-LK | Fragebogen für Lehrkräfte |
| GDSU | Gesellschaft für Didaktik des Sachunterrichts e. V. |
| GeRRI | Gemeinsamer Referenzrahmen Informatik |
| GFD | Gesellschaft für Fachdidaktik e. V. |
| ggf. | gegebenenfalls |
| GI | Gesellschaft für Informatik e. V. |
| GS | *General Studies* |
| HdkF | Stiftung Haus der kleinen Forscher |
| ICT | *Information and Communications Technology* |

| | |
|---|---|
| i. e. | id est |
| ifib | Institut für innovative Bildung |
| ISB | Staatsinstitut für Schulqualität und Bildungsforschung |
| ISTE | *International Society for Technology in Education* |
| IT | Informationstechnik |
| KMK | Kultusministerkonferenz |
| KUI | Kompetenzen für das Unterrichten in Informatik |
| LB | Lernbereich |
| LFB(s) | Lehrerfortbildung(en) |
| LK | Lehrkraft bzw. Lehrkräfte |
| MINT | Mathematik-Informatik-Naturwissenschaft-Technik |
| MIT | *Massachusetts Institute of Technology* |
| MNU | Verband zur Förderung des MINT-Unterrichts |
| MOOC | *Massive Open Online Course* |
| MSB NRW | Ministerium für Schule und Bildung des Landes Nordrhein-Westfalen |
| MZP | Messzeitpunkt |
| NRW | Nordrhein-Westfalen |
| o. ä. | oder ähnliche(s) |
| OECD | *Organization for Economic Co-operation and Development* |
| PCK | *Pedagogical Content Knowledge* |
| PD | *Professional Development* |
| SFZ-BGL | Schülerforschungszentrum Berchtesgadener Land |
| sic | sic erat scriptum |
| sog. | sogenannte/sogenannter |
| SMK Sachsen | Staatsministerium für Kultus Sachsen |
| StMBKWK | Bayerisches Staatsministerium für Bildung und Kultus, Wissenschaft und Kunst |
| SWK | Ständige Wissenschaftliche Kommission der Kultusministerkonferenz |
| SuS | Schülerinnen und Schüler(n) |
| TB | Teilbereich |
| TE | *Technology Education* |
| TUM | Technische(n) Universität München |
| TUM-DDI | Professur für Didaktik der Informatik der Technischen Universität München |
| u. a. | und andere |
| Ustd. | Unterrichtsstunde(n) |
| vgl. | vergleiche |
| VP-System | Visuelles Programmiersystem |
| z. B. | zum Beispiel |
| z. T. | zum Teil |

# 1 Einleitung

## 1.1 Motivation

Die Grundschule als erste gemeinsame Schule hat den Auftrag, allen Kindern eine grundlegende schulische Bildung zu ermöglichen. Laut den *Empfehlungen zur Arbeit in der Grundschule* der Kultusministerkonferenz (KMK) bietet sie ein Bildungsangebot, welches einerseits auf Wissen und Können zur Bewältigung von alltäglichen Lebenssituationen ausgerichtet ist, und andererseits die Grundlage für eine verantwortliche Teilhabe am gesellschaftlich-kulturellen Leben sowie für lebenslanges Lernen schaffen soll (KMK 2015, S. 5). Das Wissen und Können soll dabei nicht nur grundlegend, sondern auch anschlussfähig sein (ebd., S. 12). Die MINT-Bildung (Mathematik-Informatik-Naturwissenschaft-Technik) – so die KMK – soll in der Grundschule die Weiterentwicklung von Präkonzepten sowie den Aufbau tragfähiger Lernmotivation ermöglichen und wird vorrangig in den Fächern Mathematik, Sachunterricht, Werken und in der Medienbildung avisiert (ebd., S. 17). Letztere findet in allen Fächern statt und bereitet die Schülerinnen und Schüler auf eine selbstbestimmte Teilhabe an der multimedialen Welt vor, indem sie ihnen ermöglicht „Medien aller Art sachgerecht und produktiv zu nutzen, sich Informationen zu erschließen, eigene Darstellungen und Medienbeiträge zu gestalten und zu präsentieren, multimediale Kommunikationswege zu nutzen sowie den Einsatz und die Wirkung von Medienbeiträgen zu verstehen, zu bewerten und kritisch zu reflektieren“ (ebd.). In dem Ausblick der Empfehlungen hält die KMK außerdem fest, dass die Grundschule adäquat auf gesellschaftliche Veränderungen und wissenschaftliche Erkenntnisse reagieren muss:

> „Kinder leben in einer schnelllebigen, durch technische Entwicklungen bestimmten Welt. Die Grundschule reagiert darauf entwicklungsoffen und ermöglicht jedem Kind unter Berücksichtigung seiner individuellen Bedürfnisse eine flexible Lern- und Bildungszeit. Lehrerinnen und Lehrer bieten einen Erfahrungsraum für einen selbstverantwortlichen, kompetenten und kreativen Umgang mit Medien in einer digitalisierten Welt. So entsteht eine Grundschule, die Chancen- und Bildungsgerechtigkeit gewährleistet und die Kinder befähigt, ihre Zukunft in einer sich stetig entwickelnden Gesellschaft aktiv zu gestalten.“ (KMK 2015, S. 29)

Auch wenn die Informatik als Teil der MINT-Bildung zwar indirekt erwähnt wird, sucht man sie in den meisten Lehrplänen der Grundschulen vergebens. Selbst im Rahmen der Medienbildung, deren Bezugsdisziplin die Informatik ist, werden informatische Inhalte in den meisten Bundesländern bislang nicht explizit ausgewiesen. Der Einsatz digitaler Medien im Unterricht der Grundschule

beschränkt sich in erster Linie auf das Schreiben von Texten, das Recherchieren im Internet sowie das Nutzen von Lernprogrammen (MPFS 2019, S. 51). Die *Dagstuhl-Erklärung zur Bildung in der digitalen vernetzten Welt* (GI 2016b) weist jedoch darauf hin, dass Digitale Bildung neben einer anwendungsbezogenen und gesellschaftlich-kulturellen Perspektive immer auch eine technologische Sichtweise auf die Erscheinungsformen der Digitalisierung ermöglichen sollte. Die *Charta Digitale Bildung*[1], die 2019 von der Gesellschaft für Informatik ins Leben gerufen und von Vertreterinnen und Vertretern aus Politik, Wirtschaft, Wissenschaft und Gesellschaft unterzeichnet wurde, verweist ebenfalls auf eine technologische Perspektive der Digitalen Bildung und stellt das Gestalten der digitalen Welt ins Zentrum. Um Schülerinnen und Schülern eine kompetente, selbstbestimmte Teilhabe an der immer stärker digitalisierten Welt zu ermöglichen, scheinen bestimmte Grundkenntnisse und Fertigkeiten aus der Informatik somit unumgänglich.

Für die aktive Gestaltung der digitalen Welt scheint sich die Programmierung im besonderen Maße als Unterrichtsgegenstand zu eignen. Zwar gibt es dafür bereits einige Unterrichtsmaterialien, diese sind jedoch oftmals nicht in ein didaktisches Konzept eingebettet oder umfassend evaluiert. Die Frage, wie Grundschullehrkräfte befähigt werden können, Materialien zu informatischen Themen im Unterricht einzusetzen und sich dies auch zuzutrauen, wird in der Regel vernachlässigt. Ungeklärt ist ebenfalls, welche Rahmenbedingungen notwendig sind, um das neue Themengebiet in der Grundschule zu implementieren und zu etablieren. Diese Arbeit verfolgt daher eine mehrperspektivische Zielsetzung: Zunächst sollen ein Unterrichtskonzept und Materialien für das Programmieren in der Grundschule entwickelt werden. Dabei wird nicht nur Bezug auf die Informatik und Informatikdidaktik genommen, sondern auch auf die Entwicklungspsychologie sowie Grundschuldidaktik und -pädagogik. Das Unterrichtskonzept soll darüber hinaus erprobt, evaluiert und weiterentwickelt werden. In einem nächsten Schritt soll ein passendes Fortbildungskonzept für Grundschullehrkräfte entwickelt und erprobt werden. Die entsprechende Begleitforschung befasst sich mit der Überarbeitung der beiden Konzepte, den Auswirkungen auf die Schülerinnen und Schüler bzw. Lehrkräfte sowie den notwendige Rahmenbedingungen für das Programmieren in der Grundschule.

## 1.2 Aufbau der Arbeit

Die vorliegende Arbeit gliedert sich in sechs Teile die jeweils mehrere Kapitel umfassen (siehe Abbildung 1.1).

Im ersten Teil wird zunächst die Ausgangslage der Arbeit beschrieben und die Relevanz des Forschungsvorhabens begründet. Dabei wird zunächst allgemein auf informatische Bildung im Primarbereich eingegangen und die Rolle des Programmierens sowie Fortbildungsmaßnahmen zur informatischen Bildung thematisiert. Im Anschluss wird das Forschungsvorhaben skizziert und entsprechende Forschungsfragen formuliert.

[1] https://charta-digitale-bildung.de/

Im zweiten Teil wird der forschungsmethodische Bezugsrahmen aufgespannt, der für die Arbeit relevant ist. Ausgehend von Ausführungen über die Gestaltungsorientierung in der Bildungsforschung wird der Forschungsansatz der *Design-Based Research* beschrieben, der in dieser Arbeit realisiert wurde. Es folgt die Herleitung des weiteren Vorgehens im Forschungsprozess.

In Teil III werden die theoretischen Grundlagen für die Entwicklung des Unterrichts- und Fortbildungskonzepts zum Programmieren in der Grundschule dargestellt. Dabei werden zum einen die entsprechenden informatischen Fachkonzepte bzgl. der Programmierung eingeführt, zum anderen werden relevante Grundlagen aus der Entwicklungspsychologie, Grundschulpädagogik und -didaktik, Fachdidaktik der Informatik sowie der Lehrerbildung skizziert.

Teil IV der Arbeit beschäftigt sich mit der Entwicklung von Leitlinien und konkreten Konzepten zur Umsetzung des Programmierens in der Grundschule. Basierend auf den vorausgegangenen Ausführungen werden zunächst Lernziele zum Programmieren in der Grundschule und der entsprechenden Lehrerbildung abgeleitet. Dem Vorgehen der *Design-Based Research* folgend, werden im Anschluss vorläufige theoriebasierte Gestaltungsprinzipien für die Entwicklung eines entsprechenden Unterrichtskonzepts sowie eines Fortbildungskonzepts erarbeitet. Auf dieser Basis werden anschließend ein konkretes Unterrichtskonzept und ein Fortbildungskonzept für die Umsetzung des Programmierens in der Grundschule vorgestellt.

In Teil V wird die Implementierung und Überarbeitung der erarbeiteten Konzepte und Maßnahmen anhand mehrerer Zyklen und darin verorteter Teilstudien beschrieben. Das Unterrichtskonzept in Form eines extracurricularen Programmierkurses wurde 2016 zunächst mit einer Gruppe aus zehn Kindern im Rahmen eines Ferienkurses pilotiert. Danach wurde der Kurs in einem zweiten Zyklus mit insgesamt 38 Schülerinnen und Schülern der vierten Jahrgangsstufe, aufgeteilt in vier Gruppen, erprobt. Das Unterrichtskonzept wurde auf der Grundlage der gesammelten Erfahrungen sowie der Auswertung der Arbeitsergebnisse der Lernenden modifiziert. Das überarbeitete Unterrichtskonzept wurde im Folgejahr erneut mit weiteren 67 Schülerinnen der dritten und vierten Jahrgangsstufe in acht Gruppen erprobt und modifiziert. Sämtliche Erprobungen des zweiten und dritten Zyklus wurden von Personen mit einem einschlägig informatischen Hochschulstudium in Räumlichkeiten der Technischen Universität München durchgeführt. Zur Untersuchung, ob sich das Unterrichtskonzept auch bei der Durchführung von Lehrpersonen ohne einschlägige Vorbildung und unter praxisnäheren Bedingungen bewährt, wurde das Konzept von Mitte 2018 bis Ende 2019 an 20 bayerischen Grundschulen von jeweils zwei Lehrkräften erprobt. Um diesen vierten Zyklus zu ermöglichen, wurden die Lehrkräfte gemäß des entwickelten Fortbildungskonzepts geschult. Um zu untersuchen, wie das Konzept an den Schulen implementiert wird (Teilstudie 1) und welche Auswirkungen dies auf die Schülerinnen und Schüler hat (Teilstudie 2), wurden Interviews, die mit den Lehrkräften geführt wurden, sowie von ihnen erarbeitete Unterrichtsmaterialien ausgewertet. Die Interviews wurden zusätzlich im Hinblick auf die Gelingensbedingungen für das Programmieren in der Grundschule (Teilstudie 3) und das Fortbildungskonzept (Teilstudie 4) ausgewertet. Um die Auswirkungen des Fortbildungskonzepts auf die Lehrkräfte zu beleuchten,

wurde eine Fragebogenstudie mit vier Messzeitpunkten durchgeführt (Teilstudie 5). Die Ergebnisse wurden zudem mit Aussagen aus den Interviews ergänzt.

In Teil VI erfolgt zunächst die Synthese der verschiedenen Zyklen und Teilstudien in Form konsolidierter Leitlinien zur Umsetzung des Programmierens in der Grundschule. Diese beziehen sich sowohl auf die unterrichtliche Umsetzung als auch die Fortbildung von Lehrkräften. Zusätzlich werden mögliche Bezüge zum Lehrplan der bayerischen Grundschule aufgezeigt. Schließlich werden in den abschließenden Bemerkungen die Ergebnisse der Arbeit zusammengefasst sowie der Beitrag der Arbeit und die Qualität der Ergebnisse erörtert. Die Arbeit schließt mit einem Ausblick auf mögliche Anknüpfungspunkte für weitere Forschung.

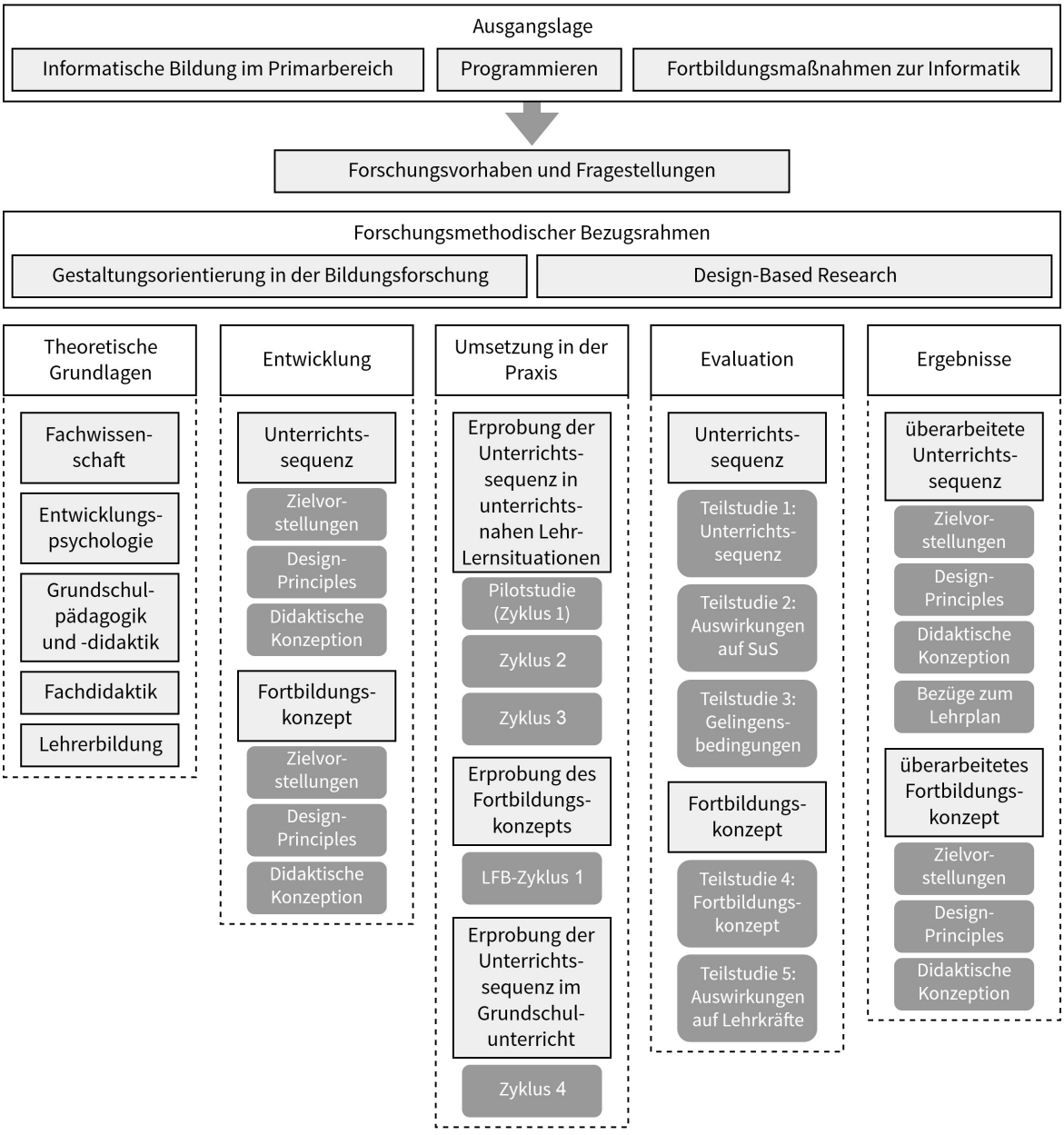



*Abbildung 1.1 Überblick über den Forschungsprozess und die entsprechenden Teile dieser Arbeit*

Teil I

# AUSGANGSLAGE UND FORSCHUNGSVORHABEN

# 2 Informatische Bildung im Primarbereich

Im Folgenden werden das Potenzial und die Chancen sowie der internationale und nationale Stand informatischer Bildung im Primarbereich dargestellt. Bei letzteren handelt es sich lediglich um eine Momentaufnahme, die zwar einen umfassenden Einblick in die internationalen und nationalen Entwicklungen gibt, jedoch keinen Anspruch auf Vollständigkeit erhebt. Das Kapitel schließt mit einer Zusammenfassung relevanter Erkenntnisse zu den Gelingensbedingungen von informatischer Bildung in der Grundschule.

## 2.1 Potenzial und Chancen

Obgleich Informatik inzwischen Pflichtfach an allen weiterführenden Schulen in Bayern ist und auch in anderen Bundesländern Einzug in die Curricula der Sekundarstufe findet (vgl. Schwarz, Hellmig und Friedrich 2022), gibt es verschiedene Gründe, warum Kinder bereits im Grundschulalter mit informatischen Inhalten in Kontakt kommen sollten.

*Teil der Allgemeinbildung*
Nach Klafki (vgl. 2007, S. 52 ff.) besteht die Aufgabe der allgemeinen Bildung darin, Lernende in ihrer Entwicklung zu mündigen Bürgerinnen und Bürgern so zu begleiten, damit sie ihre Lebenswelt und Zukunft verantwortungsvoll gestalten können. Bei der Frage, ob informatische Inhalte dabei berücksichtigt werden sollten, wird häufig auf die exponierte Rolle von Informatiksystemen verwiesen, die Claus und Schwill (2006, S. 314) als „spezifische Zusammenstellung von Hardware, Software und Netzverbindungen zur Lösung eines Anwendungsproblems“ definieren. Informatiksysteme sind nicht nur Teil der Lebenswirklichkeit von Kindern (vgl. Bitkom 2019; MPFS 2019), derart viele Bereiche des Lebens hängen von ihnen ab, dass jeder Mensch zwangsläufig ein grundlegendes Verständnis für die Informatik entwickeln sollte, um an der Gesellschaft teilzuhaben und sie mitzugestalten:

> „Wir gehen davon aus, dass die Digitalisierung in den heutigen Gesellschaften die Kultur, die Infrastruktur und entsprechend die weitere Technologieentwicklung wesentlich mitprägt und sprechen daher vom digitalen Wandel. Die Teilhabe an politischen, kulturellen und ökonomischen Prozessen innerhalb der Gesellschaft setzt Fähigkeiten im Umgang mit und zur Analyse, Reflexion und Gestaltung von digitalen Artefakten voraus. Erforderlich hierfür ist die Kenntnis der informatischen Grundlagen sowie der medienwissenschaftlichen und erziehungswissenschaftlichen Zugänge und Diskurse.“
>
> (Brinda u. a. 2019, S. 2)

Gleichzeitig zeigt sich, dass der Zugang zu digitalen Medien sowie der Umgang damit maßgeblich vom Elternhaus der Kinder abhängt. Dabei ist nicht das Einkommen der Eltern entscheidend, sondern deren digitale Lebenswelt und formaler Bildungsgrad:

> „Je selbstverständlicher Eltern im Internet sind und digitale Medien als festen Bestandteil in ihren Alltag integriert haben, desto mehr Selbstsicherheit zeigen ihre Kinder im Umgang mit digitalen Medien [...]. Kinder von Eltern mit geringer formaler Bildung haben im Kontext Spiele einen stärkeren Unterhaltungsfokus und nutzen das Internet deutlich seltener für Informationssuche und Lernzwecke als Kinder von Eltern mit formal höherer Bildung. Je geringer die formale Bildung der Eltern, desto weniger engagiert sind sie, ihre Kinder in die digitale Welt aktiv zu begleiten; sie sind vielmehr der Meinung, man bräuchte Kinder beim Erlernen des Umgangs mit digitalen Medien nicht anzuleiten, da sie dies von allein lernen würden." (DIVSI 2015, S. 16 f.)

Die Grundschule als *Schule für alle* kann hier einen Beitrag leisten, indem sie den Lernenden ermöglicht, vielfältige – auch konstruktive – Erfahrungen im Umgang mit Informatiksystemen zu sammeln. Damit Schülerinnen und Schüler ihre Lebenswelt auch in Zukunft mitgestalten können, müssen hierbei grundlegende universelle Konzepte der Informatik einbezogen werden, die langfristig gültig und von einem bestimmten Informatiksystem losgelöst sind (vgl. Schwill 1993, S. 2).

*Informatik als Denkwerkzeug*

Nach allgemeiner Auffassung gehört zu den basalen informatischen Fertigkeiten eine gewisse Grundfertigkeit im informatischen Denken – im Englischen *Computational Thinking* (z. B. Grover und Pea 2018; Lockwood und Mooney 2017; Yadav, Hong und Stephenson 2016). Der Begriff umfasst alle Denkprozesse, die daran beteiligt sind, ein Problem zu identifizieren und dessen Lösung so zu formulieren und aufzubereiten, dass ein Mensch oder Computer diese ausführen kann (Grover und Pea 2018, S. 21). Obwohl das Konzept bereits 1980 von Papert (1980, S. 155) als *Procedural Thinking* in ähnlicher Form beschrieben wurde, wird der Begriff *Computational Thinking* maßgeblich mit Wing verbunden, die ihn 2006 in einem Artikel einführte:

> *„Computational thinking is a fundamental skill for everyone, not just for computer scientists. [...] [It] involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science."*
>
> (Wing 2006, S. 33)

Berry (2015) konkretisiert dies weiter und benennt einzelne Kernaspekte des Problemlösens: *Logical Reasoning*, *Algorithms*, *Decomposition*, *Abstraction*, *Patterns/Generalization* und *Evaluation*. Sowohl Yadav, Hong und Stephenson (2016, S. 566) als auch Lamprou und Repenning (2018, S. 71) heben *Algorithms*, *Abstraction* und *Automation* als Schlüsselkonstrukte des *Computational Thinking* hervor. Erste Ergebnisse legen nahe, dass die Denkprozesse, die in der Informatik im Fokus stehen, hilfreich in anderen Fächern sind (vgl. Moreno-Leon, Roman-Gonzalez und Robles 2018). Döbeli Honegger (2017) sieht darüber hinaus auch den Nutzen im Alltag:

> „Informatik stellt Werkzeuge und Verfahren zur Verfügung, mit denen sich im Alltag Probleme strukturiert beschreiben, diskutieren und damit besser lösen lassen – auch ohne den Einsatz von Computern. [...] Bei der Beschäftigung mit Informatik lernt man diese Werkzeuge kennen und schult auch das entsprechende Denken."
>
> (Döbeli Honegger 2017, S. 94)

*Interesse an Informatik*

Ein erster Schritt, um informatisches Denken zu fördern, könnte darin bestehen, durch gezielte Angebote zunächst das Interesse der Kinder für die Informatik zu wecken und ihnen ihre eigenen Talente bewusst zu machen. Interessen spielen besonders in der frühen Kindheit eine wichtige Rolle:

> „[...] den Interessen kommt in der frühkindlichen Phase eine zentrale Funktion zu, da sie die Aufmerksamkeit der Kinder auf verschiedene Umweltanreize lenken. Welche Vorerfahrungen und Vorkenntnisse sie machen, hängt auch davon ab, welche Anreize sich den Kindern bieten, welchen Anreizen sie sich zuwenden sowie in welchem Umfang und mit welchen Unterstützungsangeboten sie dann ihren Interessen nachgehen können." (Miller und Velten 2015, S. 33)

In verschiedenen Fallstudien wurde bereits gezeigt, dass informatische Inhalte bei Kindern im Grundschulalter auf große Begeisterung stoßen (z. B. Wolking und Schmid 2017; Romeike und Reichert 2011). In einer amerikanischen Studie, an der über 1500 12- bis 18-Jährige teilnahmen, zeigte sich, dass Jungen und Mädchen im Alter von zwölf Jahren ein ähnliches Interesse am Lernen von Informatik besitzen. Dieses Interesse nimmt bei den Mädchen in den folgenden Jahren ab, während es bei den Jungen steigt. Sowohl Jungen als auch Mädchen zeigen einen Rückgang des ausgedrückten Interesses im Alter zwischen 15 und 18 Jahren (vgl. Google Inc. und Gallup Inc. 2017, S. 3 f.). In einer Studie des Stifterverbands (Suessenbach, Schröder und Winde 2022, S. 8) konnte bereits gezeigt werden, dass sich ein Pflichtfach Informatik in der Sekundarstufe I in einem ausgewogeneren Geschlechterverhältnis innerhalb der Oberstufenkurse Informatik widerspiegelte. Diese Ergebnisse legen nahe, dass die Grundschulzeit und Sekundarstufe I ein besonders vielversprechendes Zeitfenster sind, um das Interesse sowohl von Jungen als auch Mädchen an Informatik zu wecken und zu fördern.

*Vorurteile und Stereotype*

Zur Informatik gibt es zahlreiche stereotype Vorstellungen, die kulturell verbreitet und gefestigt sind. Darunter finden sich besonders viele genderspezifische Zuschreibungen, die meist negativ besetzt sind (vgl. Nelson 2014, S. 89). Laut Jaglo (2013, S. 275) wird der „typische Informatiker" von vornherein als männlich sowie als unattraktiv und unsozial beschrieben. Diese Rollenbilder festigen sich in der Pubertät (vgl. Margolis und Fisher 2002; Beyer u. a. 2003) – bis der Informatikunterricht in der Sekundarstufe einsetzt, sind demnach viele Rollenbilder bereits verankert. Kinder beginnen sehr früh, Einstellungen zu verschiedenen Themen zu entwickeln. Andre u. a. (1999) stellten fest, dass bereits Kinder im Kindergartenalter Berufe einem bestimmten Geschlecht zuordnen:

> *„Ratings of the sex-role stereotyping of jobs also followed the cultural stereotype, with boys and girls rating jobs that relate to math, life science, and physical science as more male dominated. [...] [This] also points to a need for career education and work with positive role models to begin at an earlier age than may be the norm in most schools.“*
>
> (Andre u. a. 1999, S. 741)

Ein möglichst früher Erwerb von informatischen Kompetenzen und das Vorhandensein weiblicher Rollenvorbilder könnte einerseits das Vorurteil zerstreuen, dass Informatik nichts für Mädchen sei, und andererseits allgemein ein realistischeres Bild der Informatik und der damit verbundenen Arbeitsweisen vermitteln.

*Informatisches Selbstkonzept*

Der Begriff Selbstkonzept bezeichnet nach Shavelson, Hubner und Stanton (1976, S. 411) die Vorstellungen, Einschätzungen und Bewertungen bezüglich verschiedener Aspekte der eigenen Person. Unter dem akademischen oder auch schulischen Selbstkonzept versteht man die generalisierte Einschätzung der eigenen Fähigkeiten in Bezug auf die Schulfächer (Möller 2005, S. 223). Laut Hellmich und Günther (2011, S. 34) werden die fähigkeitsbezogenen Selbstkonzepte hauptsächlich von zwei Aspekten beeinflusst – den Erfahrungen, die ein Lernender in Bezug auf die eigenen Leistungen macht, und den Rückmeldungen, die er über die eigenen Fähigkeiten bekommt. Das informatische Selbstkonzept wird außerdem geprägt von frühen Erfahrungen, die Kinder mit Informatiksystemen sammeln, sodass dessen Grundlagen meist schon vor dem ersten Informatikunterricht gelegt werden:

> „Da Jugendliche früh – im Allgemeinen vor dem ersten Informatikunterricht – und häufig mit Informatiksystemen in Kontakt kommen, deren grundlegenden Funktionsweisen und Prinzipien sie nicht kennen, haben sie meist nur Bedienfähigkeit, können nicht ‚neue oder schwierige Situationen bewältigen‘ und entwickeln so keine Selbstwirksamkeitsüberzeugung gegenüber der Informatik.“ (Müller 2015, S. 14)

Die Einordnung des Fachs Informatik als „Männerfach“ beeinflusst das informatische Selbstkonzept von Mädchen dahingehend, dass sie Informatik oft erst im Informatikunterricht für sich entdecken (Kuhl 2008, S. 120). Verschiedene Studien ergaben, dass das schulische Selbstkonzept in Wechselwirkung mit dem Lernerfolg steht und sich ein hohes schulisches Selbstkonzept positiv auf die Schulleistung auswirkt (vgl. Marsh und Martin 2011; Marsh und Craven 2006). Zudem weist Hartinger (2005, S. 10) darauf hin, dass es ohne positives Selbstkonzept in einem Bereich nicht möglich ist, längerfristiges Interesse daran zu entwickeln. Aus diesen Gründen sollte die Förderung eines positiven informatischen Selbstkonzepts bereits in der Grundschule gefördert werden. Dazu müssen Kinder die Chance bekommen, positive Erfahrungen mit der Informatik zu sammeln.

*Selbstwirksamkeit*

Unter Selbstwirksamkeit versteht man nach Schwarzer und Jerusalem (2002, S. 35) die subjektive Gewissheit einer Person, schwierige oder neue Anforderungen mithilfe der eigenen Fähigkeiten bewältigen zu können. Eine hohe Selbstwirksamkeit führt dazu, dass man sich Dinge zutraut und sich neuen Herausforderungen stellt – eine niedrige Selbstwirksamkeit kann hingehen dazu führen, dass man sie meidet. Ähnlich wie das Selbstkonzept wirkt sich also auch die Selbstwirksamkeit auf den Lernerfolg von Schülerinnen und Schülern aus. Laut Bandura (vgl. 1997, S. 79) kann sich Selbstwirksamkeit zum Beispiel durch positives Feedback, das Erfahren von Erfolgserlebnissen, Verhaltensmodellen und verbalen Zuspruch entwickeln. In mehreren Forschungsprojekten wurde außerdem gezeigt, dass die Selbstwirksamkeit von Kindern davon abhängt, wie sie aufwachsen und welche Unterstützungsmöglichkeiten ihnen offenstehen (z. B. Schneekloth und Pupeter 2010, S. 218 ff.). Dabei wird herausgestellt, dass Kinder, die in prekären Lebensverhältnissen oder unter mangelnder elterlicher Zuwendung aufwachsen, eine niedrige Selbstwirksamkeit haben. Studien, die sich auf das Fach Mathematik beziehen, fanden heraus, dass sich Jungen mehr zutrauen als Mädchen obwohl kein Unterschied in deren Fähigkeiten vorliegt (vgl. Schwippert u. a. 2020, S. 19 f.). Die Studie von Kind (2015) zeigte ähnliche Tendenzen bei Zweitklässlern in Bezug auf informatische Inhalte und konnte darüber hinaus aufzeigen, dass eine neunwöchige Intervention im Programmieren die Geschlechterunterschiede in der Selbstwirksamkeit verringerte:

> *„[...] teaching computer programming increased students' self-efficacy and attitude towards CS. While my sample size wasn't big enough to draw definitive, significant conclusions, the trend was for girls' self-efficacy and attitudes towards computers to be lower than boys' at the start of the project, but after the intervention girls' and boys' scores were similar."* (Kind 2015, S. 1)

Der Unterricht in der Grundschule hat das Potenzial, einer möglichen Benachteiligung sowohl durch den sozialen Status als auch durch das Geschlecht entgegenzuwirken und die Selbstwirksamkeitsentwicklung aller Kinder gleichermaßen zu unterstützen (Miller und Velten 2015, S. 18). Diese Chance sollte auch in Hinblick auf die Informatik genutzt werden.

## 2.2 Internationale Situation

In den vergangenen Jahren wird international zunehmend diskutiert, ob und inwieweit Inhalte der Informatik in die frühe Bildung integriert werden sollen und so ihren Weg in die Grundschule oder gar in den Kindergarten finden (z. B. Bell und Duncan 2018; Peyton Jones und Muuß-Meerholz 2014). Dieser Trend zeigt sich zum einen in verschiedenen Empfehlungen und darin, dass verschiedene Länder Aspekte der Informatik in die Curricula der Primarstufe aufgenommen haben, zum anderen gibt es immer mehr außerschulische Angebote, die sich überwiegend auf das Programmieren konzentrieren. Dabei ist zu beachten, dass die Dauer der Grundschule international variiert (siehe Tabelle 2.1).

*Tabelle 2.1 Dauer der Grundschule in den erwähnten Ländern*

| Dauer der Grundschule | Land |
|---|---|
| 4 Jahre | Litauen, Portugal, Türkei |
| 5 Jahre | Frankreich, Italien |
| 6 Jahre | China, Estland, Finnland, Großbritannien, Israel, Japan, Singapur, Südkorea |
| 8 Jahre | Neuseeland, Polen |
| 9 Jahre | Schweden, Slowakei, Tschechien |
| 4-6 Jahre (je nach Kanton/Bundesstaat) | Schweiz, USA |
| 6-7 Jahre (je nach Region) | Argentinien |
| 7-8 Jahre (je nach Bundesstaat) | Australien |

*Informatische Bildung in einem eigenen Fach*

In mehreren Ländern wird Informatik in der Grundschule als eigenes Fach unterrichtet, zum Beispiel in Australien (Falkner, Vivian und Falkner 2014), Israel (Storte u. a. 2019, S. 21), Neuseeland (Kellow 2018), Polen (Sysło und Kwiatkowska 2015), Großbritannien (Berry 2018), der Schweiz (D-EDK 2016) und der Slowakei (Kabátová, Kalas und Tomcsányiová 2016). Die konkreten Bezeichnungen des Fachs unterscheiden sich dabei erheblich. So wird das Fach teilweise eindeutig als Informatik bezeichnet[1], z. B. als *Computing* in Großbritannien oder *Informaticka vychova* in der Slowakei (auf Deutsch *grundlegende Informatik*). In beiden Ländern beinhalten die Fächer sowohl informatische Inhalte als auch Aspekte der Mediennutzung und Medienkompetenz. Während sich die informatischen Inhalte in der Slowakei auf Programmieren, Algorithmen und *Computational Thinking* beschränken, wird in Großbritannien zusätzlich das Thema Netzwerke behandelt. In der australischen Grundschule werden die Themen Daten, Informatiksysteme, Programmierung sowie algorithmisches Denken unterrichtet (vgl. Falkner, Vivian und Falkner 2014, S. 4 f.). Das Fach, welches zum größeren Lernbereich *Technologies* gehört, wird dort als *Digital Technologies* bezeichnet.

In Neuseeland wird Informatik und Medienkompetenz in der Grundschule eng verknüpft. Das neuseeländische Fach *Technology*, das sich zuvor primär auf das Nutzen von Werkzeugen und digitalen Technologien beschränkt hatte, wurde um den Bereich *Digital Technologies* erweitert und umfasst nun zusätzlich die Themen Algorithmen, Programmieren, Daten, Digitale Anwendungen, Informatiksysteme sowie Mensch und Computer (vgl. Kellow 2018). In Polen wurde ebenfalls ein neues Curriculum für die Grundschule entwickelt. Während das Fach *Computer Activities* früher hauptsächlich die Anwendung von Technologien in den Vordergrund stellte (vgl. Sysło und Kwiatkowska 2015, S. 145), wird der Fokus nun stärker auf das Programmieren und *Computational*

[1]Es ist darauf hinzuweisen, dass der Begriff *informatics* bzw. die daran angelehnte Übersetzung in einigen Ländern nicht dem deutschen Begriff *Informatik* entspricht, sondern synonym verwendet wird für *„information systems, information science, information theory, information engineering, information technology, information processing, or other theoretical or practical fields“* (siehe https://en.wikipedia.org/wiki/Informatics). Im Kontext der schulischen Bildung wurde der Begriff früher vor allem in Osteuropa oftmals als Synonym für den Umgang mit Computern und Software verwendet.

*Thinking* gelegt[2]. Auch in Israel wird derzeit schrittweise ein neues Curriculum für die Grundschule eingeführt, welches den Schwerpunkt auf *Computational Thinking* legt (Storte u. a. 2019, S. 21). Auch Tschechien befindet sich momentan in der Überarbeitung des bestehenden Faches *ICT* (*Information and Communications Technology*, auf Deutsch *Informations- und Kommunikationstechnik*), im Rahmen dessen die Themen Algorithmen, Programmierung und Grundlagen der Robotik verpflichtend in das Curriculum aufgenommen werden (vgl. Malisu und Bryndova 2020, S. 239). Die Themen sind bereits als optionale Bildungsbereiche im Curriculum vorgeschlagen, und können je nach Möglichkeiten der Schulen im Unterricht umgesetzt werden. Durch die Verankerung der Inhalte im verpflichtenden Teil des Lehrplans sollen die Schülerinnen und Schüler zum einen in der Entwicklung ihrer Problemlösefähigkeit unterstützt werden, zum anderen möchte man das Interesse an einem Informatikstudium steigern (ebd., S. 477).

In den USA wurde bereits für alle Altersstufen ein Informatikcurriculum – das *K-12 Computer Science Framework* (2016) – entwickelt, das die Inhaltsbereiche *Computing Systems*, *Networks and the Internet*, *Data and Analysis*, *Algorithms and Programming* und *Impacts of Computing* abdeckt. Auch die überwiegend in den USA tätige *Computer Science Teachers Association (CSTA)* veröffentlichte Standards für einen Informatikunterricht für Kinder bzw. Schülerinnen und Schüler von fünf bis sechzehn Jahren: die *CSTA K-12 Computer Science Standards* (2017). Das *K-12 Computer Science Framework* wird darin als wesentlicher Einfluss für den Entwicklungsprozess genannt, was sich auch darin äußert, dass die gleichen Inhaltsbereiche adressiert werden. Während jedoch das *K-12 Computer Science Framework „overarching, high-level guidance per grade bands"* bietet, beinhalten die *CSTA K-12 Computer Science Standards „detailed, measurable student performance expectations"* (CSTA 2017, S. 2). Das amerikanische Bildungssystem ist jedoch stark dezentralisiert, sodass die Standards bisher wenn überhaupt sehr unterschiedlich umgesetzt werden (Heintz, Mannila und Farnqvist 2016, S. 6).

*Integration informatischer Inhalte in ausgewählte Fächer*

Neben der Umsetzung in einem konkreten Fach werden informatische Inhalte in einigen Ländern in bereits vorhandene Fächer der Primarstufe integriert. Zum Beispiel entwarf Litauen ein sehr breit angelegtes Curriculum, das Inhaltsbereiche wie Algorithmen, Programmierung, Problemlösen, Daten und Informationen sowie Datenschutz und Virtuelle Kommunikation beinhaltet (vgl. Dagienė, Jevsikova und Stupurienė 2019, S. 89). Dieses wird ab dem Schuljahr 2020/21 in Mathematik und anderen Fächern integriert unterrichtet (European Commission/EACEA/Eurydice 2022, S. 24). Auch in Schweden wird Informatik integriert in mehrere Fächer unterrichtet (vgl. Heintz u. a. 2017, S. 121 ff.). In Mathematik wird in den Inhaltsbereichen Algebra und Problemlösen das Formulieren von eindeutigen Anweisungen, das Aufstellen von Algorithmen sowie das Programmieren in visuellen Programmierumgebungen behandelt. In dem Fach Technologie behandelt man darüber hinaus Themen, wie das EVA-Prinzip, Bestandteile eines Computers oder Netzwerke. Auch in den deutschsprachigen Kantonen der Schweiz unterrichten Lehrkräfte seit 2016 informatische

[2]https://www.zdnet.com/article/poland-wants-its-kids-to-code-but-time-is-against-them/

und medienbezogene Inhalte in der Primarstufe. Die Deutschschweizer Erziehungsdirektoren-Konferenz (D-EDK) veröffentlichte 2014 mit dem *Lehrplan 21* einen Lehrplan für die gesamte in der Schweiz obligatorische Schulzeit – vom Kindergarten bis Ende der Sekundarstufe I. 2015 verabschiedete die D-EDK den Teillehrplan *Medien und Informatik*, der im Kompetenzbereich Informatik die Themen Datenstrukturen, Algorithmen und Informatiksysteme beinhaltet (D-EDK 2016, S. 15 ff.). Dabei ist zu beachten, dass es sich beim Lehrplan 21 lediglich um eine Empfehlung handelt und die Kantone jeweils individuell entscheiden, ob und wann diese umgesetzt wird. Darüber hinaus wird *Medien und Informatik* nicht als eigenes Fach, sondern als Modul ausgewiesen, wodurch nicht festgelegt ist, wie die Umsetzung in der Praxis aussieht. Einige Kantone setzen das Modul bereits ab der 5. Jahrgangsstufe als eigenes Fach um, andere sehen es erst ab der 7. Jahrgangsstufe vor und integrieren die geforderten Inhalte davor in bestehende Fächer, z. B. Mathematik und Deutsch (Döbeli Honegger und Hielscher 2017, S. 100).

In den Grundschulen Südkoreas werden die Themen Problemlösen, Algorithmen und Programmieren unterrichtet. Die Inhalte wurden in den Kurs *Sil-Gwa* integriert, der in 17 Stunden pro Schuljahr praktische Fertigkeiten für das tägliche Leben vermitteln soll (Kwon und Schroderus 2017, S. 9). Eine ähnliche Umsetzung findet sich in der chinesischen Sonderverwaltungszone Hong Kong, in der *Technology Education (TE)* im Fach *General Studies (GS)* umgesetzt wird (vgl. CDC 2017b, S. 20). In dem inhaltlich breit angelegten Fach sollen Schülerinnen und Schüler der Grundschule Fachwissen sowie allgemeine Fähigkeiten und Einstellung zu sechs Schwerpunkten entwickeln – *Health and Living*, *People and Environment*, *Science and Technology in Everyday Life*, *Community and Citizenship*, *National Identity and Chinese Culture* and *Global Understanding and the Information Era* (vgl. CDC 2017a, S. 15). 2020 veröffentlichte das *Curriculum Development Council* eine Ergänzung zum bestehenden TE-Curriculum, in der *Computational Thinking* und *Coding* als Inhalte für die vierte bis sechste Jahrgangsstufe der Grundschule festgelegt werden (vgl. CDC 2020).

In Finnland wird ein besonderer Fokus auf das Programmieren gelegt (Kwon und Schroderus 2017, S. 6 f.). Dieses wird ab der ersten Jahrgangsstufe in Mathematik unterrichtet und darüber hinaus ab der dritten auch im Curriculum des Werkunterrichts aufgegriffen. Des Weiteren ist das Programmieren Teil der Querschnittsaufgabe *ICT Competence*, sodass die Lehrkräfte auch in anderen Fächern mit den Kindern programmieren können. Auch in Japan wird das Programmieren in verschiedenen Fächern integriert unterrichtet, um die Schülerinnen und Schüler im logischen Denken und *Computational Thinking* zu schulen. Programmiert wird laut Kanemune, Shirai und Tani (2017, S. 146) im Mathematikunterricht, im naturwissenschaftlichen Unterricht sowie in den Zeiträumen, die in der Stundentafel speziell für fächerübergreifende Inhalte reserviert sind.

*Informatische Bildung als Querschnittsaufgabe*

Während informatische Inhalte in einigen Ländern konkret einzelnen Fächern zugeordnet werden, finden sich in einigen Ländern auch freiere Umsetzungen wieder. In Italien, Frankreich und der Türkei wurde *Computational Thinking* und Programmieren laut Bocconi u. a. (2016, S. 28) als

Querschnittsaufgabe in der Grundschule eingeführt. Im Grundschulcurriculum in Estland findet man den Bereich Technologie und Innovation als Querschnittsaufgabe, die alle Lehrkräfte umsetzen müssen. Darüber hinaus gibt es verschiedene fakultative Inhalte, wie zum Beispiel Programmierung, Robotik oder 3D-Grafik, welche Schulen in ihr schulspezifisches Curriculum aufnehmen können[3]. Unterstützt werden sie dabei vom Projekt *ProgeTiger*, welches von der estnischen Regierung unterstützt sowie finanziert wird. Es wurde 2012 mit dem Ziel ins Leben gerufen, Programmieren und Robotik in alle Bildungsbereiche zu integrieren – vom Kindergarten bis über die Schulzeit hinaus (HITSA 2015).

In Portugal initiierte das Bildungsministerium 2015 ein mehrjähriges Pilotprojekt zum Programmieren in der Grundschule, an dem alle interessierten Schulen teilnehmen konnten (Balanskat und Engelhardt 2015, S. 69). In dem Projekt, an dem sich allein im ersten Jahr landesweit 625 Schulen beteiligten, stand im Mittelpunkt, wie das Programmieren in der dritten und vierten Jahrgangsstufe fächerübergreifend umgesetzt werden kann. Damit sollten Schülerinnen und Schüler zum einen die Möglichkeit bekommen, ihre Fähigkeiten im *Computational Thinking* zu entwickeln, zum anderen sollten Kernkonzepte anderer Lernbereiche – z. B. Lesen, Schreiben, Mathematik, Naturwissenschaften, Musik, Kunst – gefördert werden (Minitério da Educação e Ciência 2015). Andere Projekte untersuchen, wie das Programmieren und Robotik bereits in der Vorschule und in allen Jahrgangsstufen der Grundschule fächerübergreifend eingesetzt werden kann und wie Lehrkräfte entsprechend fortgebildet werden können (z. B. Pinto u. a. 2017; Monteiro u. a. 2019) .

In Argentinien rief 2013 die *Sadosky Foundation* das Projekt *Program.ar*[4] ins Leben, das in Zusammenarbeit mit öffentlichen Universitäten und mit staatlicher Unterstützung zunächst kostenfreie Fortbildungen für Lehrkräfte der Primar- und Sekundarstufe zum Programmieren anbot und mittlerweile auch Materialien und Handreichungen für das das Programmieren mit Kindern und Jugendlichen bereitstellt. Im Jahr 2015 veröffentlichte das argentinische *Federal Council of Education* einen Beschluss, der besagt, dass das Programmierenlernen von strategischer Bedeutung für das nationale Bildungssystem sei, und Inhalte der Programmierung und Robotik in die Liste der vorrangigen Lerninhalte aufgenommen werden (vgl. Monjelat und Lantz-Andersson 2020, S. 2178). Infolgedessen sind alle Provinzen Argentiniens verpflichtet, diese Inhalte in ihre curricularen Grundlagen aller Schularten aufzunehmen.

*Außerschulische Angebote*

Neben den formalen Schulsettings gibt es international sehr viele außerschulische Angebote für Kinder, die informatische Inhalte thematisieren. Diese freiwilligen Aktivitäten können Erfahrungen vermitteln, die nicht im Lehrplan verankert oder in regulären Klassenzimmern nicht möglich sind. Die meisten Angebote konzentrieren sich dabei vorrangig auf das Programmieren. So zum Beispiel

[3] https://digital-skills-jobs.europa.eu/en/inspiration/good-practices/progetiger
[4] http://program.ar/

die kostenfrei nutzbaren Programmierwebseiten *Code.org*[5], *Scratch*[6] oder *Blockly*[7] sowie Apps, wie *ScratchJr*[8] oder *codeSpark Academy*[9]. Die *Scratch*-Webseite bietet Kindern und Jugendlichen nicht nur die Möglichkeit zu programmieren, sondern auch, sich mit anderen zu vernetzen und auszutauschen. Die *Scratch Online Community* umfasst Hunderttausende Mitglieder, deren Programme man einsehen und weiterverwenden kann (Brennan 2013, S. 54). Das gemeinsame Arbeiten steht auch bei verschiedenen Clubs im Fokus, zum Beispiel dem *Clubhouse Network*[10] , das in den USA gegründet wurde. Diese vermitteln zwar auch informatische Inhalte, im Vordergrund steht jedoch, dass Kinder und Jugendliche lernen, sich mit neuen Technologien auszudrücken (Resnick, Rusk und Cooke 1998, S. 2). Auch bei den *CoderDojos*[11] können Kinder und Jugendliche in mittlerweile über 69 Ländern informatische Inhalte lernen – der Schwerpunkt liegt dabei vor allem auf dem Programmieren (vgl. Quigley 2017, S. 5). Die *CoderDojo Girls Initiative*[12] macht es sich darüber hinaus zum Ziel, den Anteil von Mädchen, die ein *CoderDojo* besuchen, zu erhöhen. In beiden Clubs arbeiten die Lernenden mit erwachsenen Mentorinnen und Mentoren zusammen, die sie ehrenamtlich mit Know-how unterstützen. In Großbritannien hat sich die Initiative *Code Club*[13] etabliert, deren Angebote nach dem Unterricht in Schulen stattfinden. Hier arbeiten Ehrenamtliche, von denen die meisten einen beruflichen Hintergrund in Informatik aufweisen, mit Lehrkräften zusammen, die in der Regel über keine informatische Vorbildung verfügen. Im Vordergrund steht dabei, dass die Kinder Erfolgserlebnisse sammeln und ihr Interesse am Programmieren sowie dem digitalen Gestalten entdecken (Smith, Sutcliffe und Sandvik 2014, S. 517). Nach eigenen Angaben werden mittlerweile in 160 weiteren Ländern wöchentliche *Code Clubs* angeboten. Ein ähnlicher Ansatz wird in Singapur mit dem staatlichen *Code for Fun Enrichment Programme* verfolgt, das mittels zehnstündiger Programmierkurse einer breiten Massen an Schülerinnen und Schülern der Grundschule einen Zugang zur Programmierung und Konzepten des *Computational Thinking* ermöglichen möchte (vgl. Seow u. a. 2019, S. 349). Interessierte Schulen können aus einer Reihe von Kursanbietern wählen, die vor Ort im Rahmen eines extracurricularen Clubs visuelle Programmiersprachen sowie Elemente der Robotik einführen. 30% der Finanzierung wird von der jeweiligen Schule aufgebracht, der Rest wird von der Regierung unter der Prämisse übernommen, dass eine Mindestanzahl von Kindern teilnimmt und die Lehrkräfte der Schule eine entsprechende Schulung zum Programm besuchen. Geplant ist, dass die Kurse zukünftig von den fortgebildeten Lehrkräften selbst durchgeführt werden können.

[5] https://code.org/
[6] https://scratch.mit.edu/
[7] https://blockly.games/
[8] https://www.scratchjr.org/
[9] https://codespark.com/
[10] https://theclubhousenetwork.org/
[11] https://coderdojo.com/
[12] https://coderdojo.com/girlsinitiative/
[13] https://codeclub.org/en/

## 2.3 Situation in Deutschland

Obwohl man sich in Deutschland noch nicht auf verbindliche Richtlinien für den Umgang mit informatischen Inhalten geeinigt hat, wird die Relevanz für die Grundschule auch hier immer deutlicher. Dies tritt beispielsweise in unterschiedlichen Grundlagenpapieren zur digitalen bzw. informatischen Bildung zutage und zeigt sich in konkreten curricularen Umsetzungen einzelner Länder sowie verschiedenen Forschungsprojekten, Initiativen oder außerschulischen Angeboten zur informatischen Bildung in der Grundschule.

*Normative Grundlagen*

Bereits 2016 wird die Vermittlung informatischer Inhalte in unterschiedlichen Grundlagenpapieren der Bundesregierung thematisiert. Das Bundesministerium für Bildung und Forschung (BMBF) weist in seinem Strategiepapier *Bildungsoffensive für die digitale Wissensgesellschaft* (2016, S. 15) darauf hin, dass eine digitale Kompetenzvermittlung in der Breite und von der frühkindlichen Bildung an nötig ist, um „in Zukunft als Nation von Gestaltern – und nicht allein als Anwender – die digitale Wirtschaft und Gesellschaft mitzuprägen". Zur digitalen Kompetenz zählt das Ministerium einerseits die Fähigkeit, Informationen im digitalen Raum zu suchen, zu bewerten und für andere zur Verfügung zu stellen, andererseits werden Inhalte angesprochen Inhalte angesprochen, die eindeutig der Informatik zuzuordnen sind:

> „[Digitale Kompetenz] umfasst auch ein technisches Grundverständnis, das über die Bedienung aktueller Geräte hinausgeht und Grundkenntnisse über ihre Funktionsweise und diejenige digitaler Medien, über die Software-Entwicklung und Algorithmik, über Netzwerktechnologien und IT-Sicherheit bzw. Datenschutz beinhalten muss. Dazu zählen nicht zuletzt Grundfertigkeiten im Programmieren (‚coding')."
>
> (BMBF 2016, S. 10)

Auch das Bundesministerium für Wirtschaft und Energie (BMWi 2016) zählt in seinem Positionspapier zur digitalen Bildung das Einführen von Grundkenntnissen in Informatik, Programmieren und Algorithmen als Pflichtbestandteil der Lehrpläne für die Grundschule zu den besonders wichtigen Zielen:

> „Grundkenntnisse in Informatik, im Programmieren und über Algorithmen sind Pflichtbestandteil der Lehrpläne in Primar- und Sekundarstufe. Einfache algorithmische Strukturen erkennen und selbst formulieren zu können, schult wichtige Schlüsselkompetenzen wie analytisches Denken, Kreativität und die Fähigkeit, Probleme eigenständig zu lösen. Reproduktives Lernen muss durch prozess- und ergebnisorientiertes Lernen erweitert werden. Deshalb sollte die Schule das sogenannte ‚computational thinking' vermitteln." (BMWi 2016, S. 13)

Im gleichen Jahr veröffentlicht die Kultusministerkonferenz ihr *Strategiepapier zur Bildung in der digitalen Welt* (KMK 2016), in dem der Kompetenzbereich *Problemlösen und Handeln* das Erkennen und Formulieren von Algorithmen beinhaltet. Darunter zählt die KMK das Kennen und Verstehen von Funktionsweisen und grundlegenden Prinzipien der digitalen Welt, das Erkennen und Formulieren von algorithmischen Strukturen in genutzten digitalen Tools sowie das Planen und Verwenden einer strukturierten, algorithmischen Sequenz zur Lösung eines Problems (ebd., S.18). Die Kompetenzen sollen laut KMK bei der Überarbeitung der Lehrpläne berücksichtigt und zukünftig integrativ in allen Fächern adressiert werden:

> „Die Länder beziehen in ihren Lehr- und Bildungsplänen sowie Rahmenplänen, beginnend mit der Primarschule, die Kompetenzen ein, die für eine aktive, selbstbestimmte Teilhabe in einer digitalen Welt erforderlich sind. Dies wird nicht über ein eigenes Curriculum für ein eigenes Fach umgesetzt, sondern wird integrativer Teil der Fachcurricula aller Fächer." (KMK 2016, S. 12)

Die Gesellschaft für Fachdidaktik (GFD) nimmt zu dem Strategiepapier der KMK Stellung und hält fest, dass aus ihrer Sicht Ergänzungen und Präzisierungen notwendig sind, z. B. die Verknüpfung der digitalen Kompetenzbereiche mit fachlichen Kompetenzen. Darüber hinaus wird festgestellt, dass digitale Bildung zwar im Fachunterricht gefördert werden kann, dazu jedoch Grundlagen nötig sind, die vorrangig in einem spezifischen Informatikunterricht vermittelt werden sollten (vgl. GFD 2018).

Die Gesellschaft für Informatik (GI) geht noch einen Schritt weiter und formuliert 2019 in ihren *Empfehlungen zur informatischen Bildung im Primarbereich* Kompetenzerwartungen zu fünf verschiedenen Inhaltsbereichen, die Schülerinnen und Schüler im Verlauf der Grundschulzeit entwickeln sollten (GI 2019): *Informationen und Daten*, *Algorithmen*, *Sprachen und Automaten*, *Informatiksysteme* sowie *Informatik, Mensch und Gesellschaft*. Die Inhaltsbereiche sind wiederum mit fünf Prozessbereichen verzahnt, die mögliche Formen der Auseinandersetzung beschreiben und damit gleichzeitig fachimmanente Denk- und Arbeitsweisen aufgreifen (siehe Abbildung 2.1). Die Autorinnen und Autoren machen darauf aufmerksam, dass es sich bei den Kompetenzerwartungen aus wissenschaftlicher Sicht nicht um empirisch fundierte Kompetenzstufen handelt, sondern vielmehr um Empfehlungen zur Strukturierung der Lehr-Lern-Prozesse (GI 2019, S. 8). Die Empfehlung legt sich nicht fest, in welcher Organisationsform die informatische Bildung im Primarbereich stattfinden sollte:

> „Informatische Bildung kann im Primarbereich in verschiedenen Organisationsformen stattfinden [...]. Informatik kann in der Grundschule als eigenständiges Fach oder als eigenständiger Lernbereich – verankert in einem bestehenden Fach (z. B. Sachunterricht) – umgesetzt werden. Darüber hinaus kann informatische Bildung der Kinder in allen anderen Fächern weiterentwickelt werden." (GI 2019, S. 4 f.)

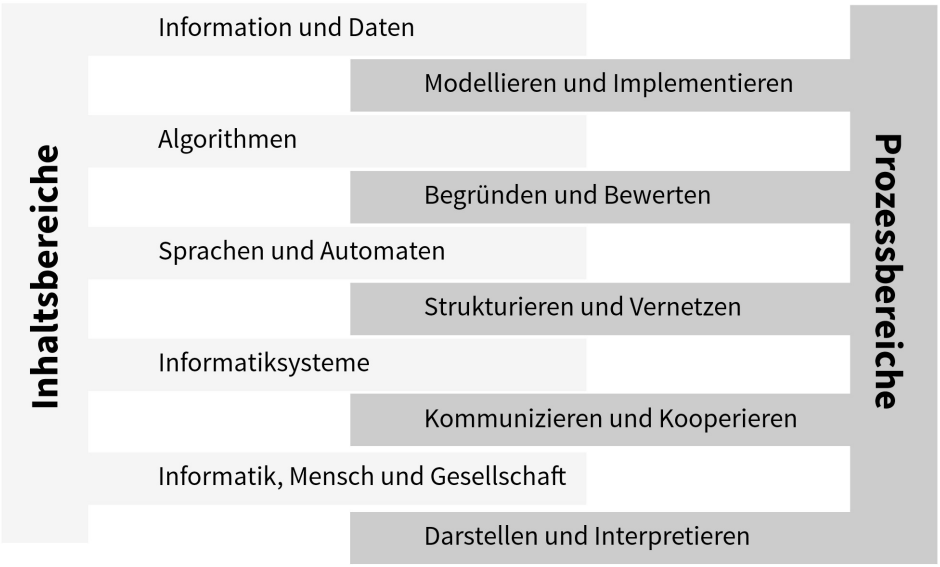



*Abbildung 2.1 Die Kompetenzbereiche des GI-Kompetenzstrukturmodells für den Primarbereich aus GI (2019, S. 7)*

Im Folgejahr veröffentlicht die GI gemeinsam mit dem Verband zur Förderung des MINT-Unterrichts (MNU) den *Gemeinsamen Referenzrahmen Informatik (GeRRI)*, der für unterschiedliche Niveaustufen beschreibt, zu was ein allgemeingebildeter Mensch in den Bereichen *Digitalisierung*, *Automatisierung* und *Informatiksysteme* fähig sein sollte (vgl. Röhner u. a. 2020). Die Niveaus A1 und A2 entsprechen – angelehnt an bestehende Referenzrahmen – einer elementaren informatischen Bildung.

Auf Initiative der Stiftung *Haus der kleinen Forscher (HdkF)* bildete sich parallel zur Arbeit der GI eine Arbeitsgruppe, die ebenfalls Zieldimensionen für eine frühe informatische Bildung und entsprechende Zielkompetenzen formulierte (HdkF 2018). Diese beziehen sich nicht nur auf die Grundschule, sondern auch auf den Kindergarten. Die Struktur der Zielkompetenzen orientiert sich nach Angaben der Autorinnen und Autoren am Kompetenzbegriff von Weinert (2001b, S. 27 f.), der Kompetenzen definiert als „die bei Individuen verfügbaren oder durch sie erlernten kognitiven Fähigkeiten und Fertigkeiten, um bestimmte Probleme zu lösen, sowie die damit verbundenen motivationalen, volitionalen und sozialen Bereitschaften und Fähigkeiten, um die Problemlösungen in variablen Situationen erfolgreich und verantwortungsvoll nutzen zu können“ (2001b, S. 27 f., siehe Abschnitt 11.1.1). Wie schon die Arbeitsgruppe der Gesellschaft für Informatik greifen sie dabei auf bereits bestehende Inhalts- und Prozessbereiche der Standards für die Sekundarstufen zurück (GI 2008; GI 2016a). Sie erweitern diese jedoch um den Prozessbereich *Interagieren und Explorieren* (vgl. Bergner u. a. 2018; Müller, Schulte und Magenheim 2019) und ergänzen zwei grundlegende Zieldimensionen: *Motivation, Interesse und Selbstwirksamkeit im Umgang mit Informatiksystemen* sowie *Übergreifende Basiskompetenzen* (siehe Abbildung 2.2). Straube u. a. (2018, S. 1) stellen ebenfalls fest, dass aktuelle gesellschaftliche Entwicklungen und Probleme sowie veränderte Lebensbedingungen von Kindern bzgl. der zunehmenden Digitalisierung, Automatisierung und Vernetzung aller Lebensbereiche nicht in der Primarstufe berücksichtigt werden. Sie

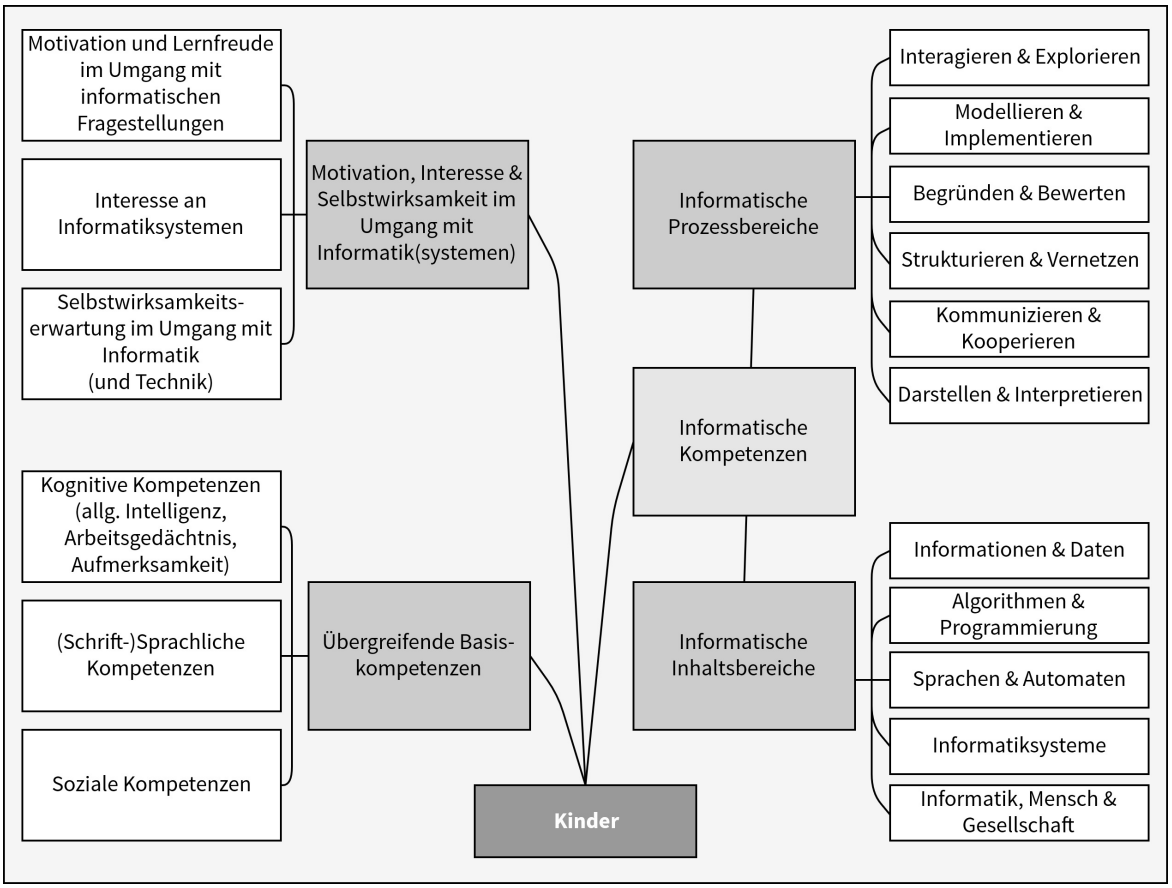



*Abbildung 2.2 Zieldimensionen informatischer Bildung auf Ebene der Kita- und Grundschulkinder aus Bergner u. a. (2018, S. 7)*

befürworten eine digitale Perspektive für den Sachunterricht, der als sechste Perspektive in den Perspektivrahmen Sachunterricht aufgenommen werden sollte, und die Digitalisierung bzw. Informatik, als der der Digitalisierung zugrundeliegende Disziplin, berücksichtigt (ebd., S. 8 f.). In einem später veröffentlichten Entwurf stellen sie drei perspektivbezogene Themenbereiche vor: *Robotik und Softwaregestaltung*, *Computer, Smartphone und co.* sowie *Alltag in einer Medienwelt* (vgl. Brämer u. a. 2020). In den Formulierungen zu den perspektivbezogenen Denk-, Arbeits- und Handlungsweisen zeigen sich neben Elementen, die sich auf das Anwenden und die Wirkung digitaler Medien beziehen, deutliche Bezüge zur Informatik. Diese gehen sowohl in die Richtung des *Computational Thinking* (z. B. „die Schülerinnen und Schüler können informatische Konzepte und Algorithmen für die Optimierung der eigenen Vorgehensweise beim Problemlösen nutzen“ (ebd., 4) als auch auf konkrete informatische Inhalte (z. B. „die Schülerinnen und Schüler können die Übertragung von Daten innerhalb und zwischen Informatiksystemen verstehen“ (ebd.). Die AG Medien und Digitalisierung der Gesellschaft für Didaktik des Sachunterrichts (GDSU) vertritt in ihrem Positionspapier *Sachunterricht und Digitalisierung* (2019) ebenfalls die Auffassung, dass Schülerinnen und Schüler im Sachunterricht ein „Grundverständnis des Algorithmisierens und Automatisierens [...] sowie der Mensch-Maschine-Interaktion als Verstehensgrundlage der Digitalisierung“ erlangen können und müssen, jedoch „nicht in einem zusätzlichen, die Phänomene der

Digitalisierung separierenden eigenen Themenblock oder gar Fach“ (ebd., S. 5). Sie führen weiter aus, dass informatische Inhalte, wie sie z. B. in den Empfehlungen der GI formuliert werden, im perspektivenvernetzenden Themenbereich *Medien*, der das Lernen mit und über Medien einschließt, aufgegriffen werden sollten (ebd., S. 6).

Im Jahr 2022 veröffentlichte die Ständige Wissenschaftliche Kommission der Kultusministerkonferenz (SWK) in ihrem *Gutachten zur Digitalisierung im Bildungssystem* Handlungsempfehlungen von der Kita bis zur Hochschule (SWK 2022). Darin empfiehlt sie, Informatikinhalte verpflichtend im Sachunterricht der Grundschulen zu verankern. Daneben werden drei Maßnahmen genannt, die schnellstmöglich umgesetzt werden sollten (ebd., S. 63): (1) Ausgewählte Aspekte der Informatik sollten bei der Überarbeitung der Lehr- und Bildungspläne für den Sachunterricht – in Anlehnung an die Bildungsstandards der GI – verankert werden. Parallel dazu sollten Angebote der Fort- und Weiterbildung für Grundschullehrkräfte ausgebaut werden. (2) In der ersten Phase der Lehrerbildung sollten Aspekte der Informatik und Informatikdidaktik als Pflichtmodul in die Ausbildung von Grundschullehrkräften im Sachunterricht integriert werden. (3) Breit angelegte Fortbildungsangebote und geeignetes *Creative Commons (CC)*-Material sollten für Grundschullehrkräfte bereitgestellt werden. Diese Angebote sollten den Lehrkräften die Möglichkeit bieten, sich die Materialien unter fachlicher und pädagogischer Begleitung gemeinschaftlich anzueignen und anzupassen.

*Länderspezifische Umsetzungen*

In Nordrhein-Westfalen wurde 2018 der *Medienkompetenzrahmen NRW* veröffentlich, der als verpflichtende Grundlage für die Medienkonzepte aller Schulen sowie die sukzessive Überarbeitung der Lehrpläne der Schulformen der Primar- und Sekundarstufe I dient (Medienberatung NRW 2020). Dieser beinhaltet den Kompetenzbereich *Problemlösen und Modellieren*, in dessen Beschreibung zum Beispiel das Erkennen von Algorithmen sowie das Modellieren und Programmieren gefordert werden (siehe Abbildung 2.3). In Bezug auf die Grundschule wird mitunter erklärt, dass die Schülerinnen und Schüler bis zum Ende der vierten Klasse in der Lage sein sollten, Algorithmen und Strukturen in verschiedenen Kontexten zu erkennen, zu verstehen und zu reflektieren sowie Algorithmen und Modellierungskonzepte in einfachen Programmierumgebungen zu planen und nutzen (ebd., 22). Der *Masterplan Grundschule*, der 2020 vom Ministerium für Schule und Bildung des Landes Nordrhein-Westfalen veröffentlicht wurde, greift den Medienkompetenzrahmen auf und erklärt, dass die Kompetenzerwartungen zwar in allen Fächern adressiert, jedoch vorrangig im Sachunterricht verankert werden sollen:

> „Ein Schwerpunkt der Weiterentwicklung des Lehrplans Sachunterricht wird die Einarbeitung der Kompetenzerwartungen sein, die sich aus dem Medienkompetenzrahmen NRW ergeben. Das Ziel, digitale Medien in den Lehr- und Lernprozess zu integrieren und die Kinder frühzeitig in altersangemessener Form an Grundprinzipien der Informatik heranzuführen, soll in der Grundschule im Unterricht aller Fächer erreicht werden. Aber auch wenn Medienbildung als interdisziplinäre Aufgabe aller Fächer verstanden wird, so soll sie doch schwerpunktmäßig im Sachunterricht verankert werden.“
>
> (MSB NRW 2020, S. 22)


*Abbildung 2.3 Kompetenzbereich Problemlösen und Modellieren des Medienkompetenzrahmens NRW aus Medienberatung NRW (2020, S. 11)*

In Sachsen trat im August 2019 ein überarbeiteter Lehrplan für die Grundschule in Kraft, in dem für das Fach Werken der Lernbereich *Begegnung mit Robotern und Automaten* eingeführt wird, der in Klassenstufe 4 im Umfang von sechs Unterrichtsstunden unterrichtet werden soll (vgl. SMK Sachsen 2019, S. 14 f.). Die Schülerinnen und Schüler sollen Einblick gewinnen in Einsatzbereiche von Robotern und Automaten sowie in einfache Programmierumgebungen zu deren Steuerung. Das Wissen über das EVA-Prinzip und Anweisungsfolgen sollen die Lernenden darüber hinaus auf konkrete Aufgabenstellungen übertragen (siehe Tabelle 2.2).

*Tabelle 2.2 Lernbereich ‚Begegnung mit Robotern und Automaten' aus dem sächsischen Lehrplan Grundschule für das Fach Werken (SMK Sachsen 2019, S. 14 f.)*

| Lernbereich 3: Begegnung mit Robotern und Automaten (6 Ustd.) | |
|---|---|
| Einblick gewinnen in Einsatzbereiche von Robotern und Automaten<br>- Eingabe-Verarbeitung-Ausgabe | Rasenmäherroboter, Staubsaugerroboter, Getränkeautomat, Fahrkartenautomat, E-V-A, Aktor, Sensor, Einsatzbereiche, Vor- und Nachteile, Einsatz traditioneller und digitaler Medien |
| Einblick gewinnen in eine einfache Programmierumgebung zur Steuerung<br>- Sachverhaltsanalyse<br>- Entwicklung eines Modells<br>- Programmierung eines einfachen Ablaufs<br>- Prüfen der Funktionalität | Symbolsprache, kurze Anweisungsfolgen, Erkennen des konstruktiven Problems, Erproben von Varianten |
| Übertragen des Wissens auf die Umsetzung einer konkreten Aufgabenstellung<br>- Nachvollziehen von E-V-A<br>- Bauen eines Modells<br>- Beurteilen der Umsetzung | Anleitung zum Bauen und Programmieren unterschiedlicher Modelle, Bauanleitung, Baukästen, Partner- und Gruppenarbeit, Benennen der Bauteile/Baugruppen, Fehlersuche, Optimierungsvorschläge, Bewertungskriterien gemeinsam festlegen |

Das saarländische Ministerium für Bildung und Kultur startete 2017 als erstes Bundesland das *Calliope mini*-Projekt. Im Rahmen dessen wurden zunächst Lehrerfortbildungen (LFBs) angeboten, die den Einsatz des *Calliope mini*[14] sowie Grundkonzepte der informatischen Bildung in der Primarstufe thematisierten (vgl. Reese und Wolf 2018, S. 108). Laut Webseite der *Calliope*-Initiative wurde der Einplatinencomputer im ganzen Saarland flächendeckend angeboten und ab der dritten Klasse im Regelunterricht eingesetzt. Pilotprojekte in der Primarstufe werden darüber hinaus in mittlerweile fast allen Bundesländern durchgeführt[15]. Der Einsatz des *Calliope mini* im Unterricht der Grundschule wurde in verschiedenen Studien wissenschaftlich untersucht (z. B. Murmann u. a. 2018; Baum u. a. 2019; Goecke und Stiller 2018).

Nenner und Bergner (2022) untersuchten, inwieweit informatische Inhalte bereits in den Curricula der deutschen Bundesländer verankert sind. Zu diesem Zweck analysierten sie insgesamt 53 Lehrpläne verschiedener Fächer der Bundesländer nach sechzehn Schlüsselbegriffen, die aus den *CSTA K-12 Computer Science Standards* (CSTA 2017) und den Kompetenzen für informatische Bildung im Primarbereich der GI (2019) generiert wurden. Im Anschluss wurde manuell geprüft, ob in den identifizierten Textstellen tatsächlich informatische Inhalte aufgeführt wurden. Die Ergebnisse der Dokumentenanalyse zeigen, dass in acht von sechzehn Bundesländern informatische Inhalte bereits in den Lehrplänen integriert sind (ebd., S. 8 f.). In vier Bundesländern werden sie in den Sachunterricht integriert (Berlin, Brandenburg, Hamburg, Nordrhein-Westfalen), in vier Bundesländern in Mathematik (Hamburg, Mecklenburg-Vorpommern, Nordrhein-Westfalen, Sachsen-Anhalt) und in zwei Bundesländern in Technik (Sachsen, Schleswig-Holstein).

*Forschungsprojekte*

Auf Seiten der Forschung werden verschiedene Anstrengungen unternommen, Kindern die Möglichkeit zu geben, grundlegende Kenntnisse im Bereich der informatischen Bildung zu erwerben und zu untersuchen, welche Unterrichtsmethoden und -inhalte sich für Grundschulen in Deutschland eignen. Im Projekt *IT2School – Gemeinsam IT entdecken* entwickelte die Universität Oldenburg in Zusammenarbeit mit der *Wissensfabrik*[16] Unterrichtsmodule und passende Materialien, die Einblicke in Themen wie Kommunikation, Daten, Programmiersprachen und das Zusammenspiel von Hard- und Software geben (vgl. Diethelm und Schaumburg 2016). Ausgewählte Module wurden vom niedersächsischen Kultusministerium aufgegriffen und im Projekt *Informatische Bildung und Technik in der Grundschule*[17] eingesetzt. In dem Projekt des niedersächsischen Kultusministeriums wurde die Vermittlung von informatischen Grundlagen an 31 Pilotschulen erprobt und vom Institut für innovative Bildung (ifib) evaluiert (Riefling, Fandrich und Diethelm 2020, S. 298).

[14] Der *Calliope mini* ist ein Einplatinencomputer, der nach dem Vorbild des *BBC micro:bit* entwickelt wurde. Er kann in verschiedenen blockbasierten Programmierumgebungen programmiert werden. Dafür schreiben die Schülerinnen und Schüler zunächst ein Programm, laden es herunter und übertragen es per USB-Kabel auf den *Calliope mini*. Er verfügt über einen Lage-, Bewegungs- und Helligkeitssensor sowie über Eingabe- und Ausgabeanschlüsse, zum Beispiel zwei Knöpfe (Eingabe) und ein Display aus 25 LEDs (Ausgabe).

[15] https://calliope.cc/schulen/pilotphase

[16] https://www.wissensfabrik.de/

[17] https://www.infgsnds.de

Die Forschungsgruppe Elementarinformatik (FELI) der Universität Bamberg entwickelte mit ihrer *Experimentierkiste Informatik* insgesamt sechs Module für den Einsatz im Kindergarten und der Grundschule, die Kinder spielerisch an informatische Themen, wie Pixel, Algorithmen, binäre Suche sowie Programmieren, heranführen (vgl. Gärtig-Daugs u. a. 2016). Die Module wurden in der Praxis erprobt sowie kontinuierlich angepasst. Dabei zeigte sich eine hohe Motivation der teilnehmenden Kinder sowie eine hohe Akzeptanz bei den pädagogischen Fachkräften und Lehrkräften (Schmid und Gärtig-Daugs 2018, S. 98, 101). Zuletzt entstanden weitere Materialien rund um das Thema *Künstliche Intelligenz*[18].

Die RWTH Aachen, die Universität Paderborn und die Bergische Universität Wuppertal führten gemeinsam das Projekt *Informatik an Grundschulen* durch, das vom Ministerium für Schule und Bildung des Landes Nordrhein-Westfalen gefördert wurde (Magenheim u. a. 2018a). Dabei wurden drei Module erarbeitet, die sich an Kinder der dritten und vierten Jahrgangsstufe richten und im Rahmen des Sachunterrichts erprobt wurden. Die Module, die in Zusammenarbeit mit Grundschullehrkräften entwickelt wurden, behandeln die Themen Informationen und Daten, Kryptologie und Programmierung. Die Evaluation des Projekts beschäftigt sich neben dem Lernfortschritt der Kinder mit deren Motivation sowie Interesse bezüglich Informatik.

Auch im Rahmen des Projekts *Informatik in der Grundschule (IGS)* der Universität Münster wurden verschiedene Unterrichtsbausteine entwickelt und gemeinsam mit Grundschullehrkräften erprobt, die sich unter anderem mit den Themen Kryptologie sowie Programmieren mit dem Bee-Bot und in Scratch auseinandersetzen. Der Forschungsschwerpunkt des Projekts ist die Untersuchung der Vorstellungen von Grundschullehrkräften zur Informatik und zum Informatikunterricht (vgl. Best 2020, S. 16).

Am Leibniz-Institut für Wissensmedien in Tübingen wurde ein Programmierkurs entwickelt und erprobt, der sich an besonders begabte und hochbegabte Kinder der dritten und vierten Klasse richtet. Im Kurs *Verstehen wie Computer denken* werden verschiedene analoge und digitale Spiele eingesetzt, die eigens für den Kurs entwickelt wurden (Tsarava, Moeller und Ninaus 2018). Die Arbeitsgruppe entwickelte außerdem ein Testinstrument, welches das Selbstkonzept sowie die Einstellung von Kindern in Bezug auf das Programmieren erfasst (Leifheit u. a. 2020).

*Initiativen*

Die *Roberta*-Initiative[19], die 2002 von der Fraunhofer-Gesellschaft gegründet wurde, hat es sich zur Aufgabe gemacht, bei Kindern und Jugendlichen ab acht Jahren, insbesondere auch bei Mädchen, Interesse für Informatik zu wecken. Dafür bildet sie zum Beispiel Lehrkräfte fort, veröffentlicht Lehr- und Lernmaterialien und stellt die Programmierplattform *Open Roberta Lab* zur Verfügung. Sie beschränkt sich dabei laut eigenen Angaben auf das Programmieren von Robotern, da diese einen spielerischen Zugang zur Technik ermöglichen und gleichzeitig eine Verbindung zwischen scheinbar abstrakten Lerninhalten und der Realität herstellen.

[18] https://www.uni-bamberg.de/feli/weitere-aktivitaeten/ki-und-schule/
[19] https://www.roberta-home.de/

Auch beim *Informatik-Biber*[20] steht im Mittelpunkt, Interesse an der Informatik zu wecken. Der jährlich stattfindende Wettbewerb für Schülerinnen und Schüler ab der dritten Jahrgangsstufe wird in verschiedenen Ländern angeboten. Die Teilnahme erfolgt online über die Webseite des Wettbewerbs und ist für alle Schulen und außerschulischen Lernorte möglich. Die Aufgaben des Bibers beziehen sich zwar immer auf informatische Inhalte oder Methoden, können jedoch auch ohne informatisches Vorwissen gelöst werden.

*Außerschulische Angebote in Deutschland*

Der Trend zu außerschulischen Angeboten, die sich mit informatischen Inhalten beschäftigen, zeigt sich auch in Deutschland. Es formierten sich zum Beispiel deutsche Vertretungen von internationalen Initiativen, wie den *CoderDojos*[21]. Zudem wurden Unternehmen gegründet, die Informatik-Kurse für Kinder und Jugendliche anbieten, z. B. die *Hacker School*[22] oder die *HABA Digitalwerkstätten*[23].

In vielen deutschen Städten bieten Schülerlabore Kurse sowohl für interessierte Kinder und Jugendliche als auch für ganze Klassen an und ergänzen damit den Schulunterricht. Einige Schülerlabore widmen sich ausschließlich der Vermittlung von informatischen Inhalten, zum Beispiel das *Schülerlabor Informatik InfoSphere*[24] der RWTH Aachen oder das *EduInf*[25] der TU Dresden. Auch online stehen zahlreiche Angebote in deutscher Sprache zur Verfügung, mit denen Kinder sich mit informatischen Inhalten – vorrangig dem Programmieren – auseinandersetzen können, z. B. auf den Webseiten *Code it!*[26], *Programmieren mit der Maus*[27] oder *Ronjas Roboter*[28]. Letztere ist auch als App verfügbar und bietet zusätzlich Tipps zur Lernbegleitung sowie Vertiefung der Inhalte an.

## 2.4 Gelingensbedingungen

In Bezug auf die Verbreitung informatischer Bildung im Primarbereich sehen Haselmeier u. a. (2016) Handlungsbedarf auf drei Ebenen: (1) die Qualifikation der Lehrkräfte und damit verbunden die Entwicklung ihres Informatikselbstkonzepts, (2) angemessenes Unterrichtsmaterial, das die Schülerinnen und Schüler herausfordert aber nicht überfordert, und (3) die Verankerung informatischer Allgemeinbildung in einem eigenständigen Fach. Die fehlende Lehrerbildung wird auch von Petko, Döbeli Honegger und Prasse (2018) und Humbert u. a. (2020, S. 91) als Herausforderung gesehen und eine entsprechende Weiterentwicklung gefordert. Letztere fordern die Integration informatischer Bildung in alle drei Phasen der Lehrerbildung.

[20] https://bwinf.de/biber/
[21] https://www.coderdojo-deutschland.de/
[22] https://hacker-school.de/
[23] https://www.digitalwerkstatt.de/
[24] https://schuelerlabor.informatik.rwth-aachen.de/
[25] https://tu-dresden.de/ing/informatik/smt/ddi/schulinformatik/eduinf-education_in_informatics
[26] https://code-it-studio.de/
[27] https://programmieren.wdrmaus.de/
[28] https://www.meine-forscherwelt.de/spiel/ronjas-roboter

Duncan, Bell und Atlas (2017) arbeiteten mit dreizehn Grundschullehrkräften zusammen, die verschiedene informatische Themen nach einer entsprechenden Fortbildung in ihrem Unterricht behandelt haben. Dabei zeigten sich verschiedene Fehlvorstellungen, sowohl auf Seiten der Schülerinnen und Schüler als auch auf Seiten der Lehrkräfte. Bei den Lehrkräften bezogen sich diese z. B. auf das Bild der Informatik – dass es darin nur um das Programmieren ginge – und auf die Verwendung der richtigen Fachbegriffe. Ein falsches Bild der Informatik beobachteten die Lehrkräfte auch bei ihren Schülerinnen und Schüler. Ein weiteres Problem bestand darin, ihnen zu vermitteln, dass eine Programmiersprache nicht „intelligent" ist und ein Programm nur das ausführt, was geschrieben wurden – nicht das, was gemeint war. Auch Magenheim u. a. (2018b) untersuchten die Erfahrungen von Grundschullehrkräften an rund 30 Schulen bei der Umsetzung informatischer Inhalte im Unterricht. Die Auswertung der Feedbackbögen zeigte, dass die Lehrkräfte nicht immer in der Lage waren, weiterführende Fragen der Schülerinnen und Schüler kompetent zu beantworten. Es zeichnete sich zwar der Wunsch ab, die Welt der Informatik zu verstehen, die Ausbildung eines *„bigger picture"* in Bezug auf die Informatik gelang jedoch nicht. Mehr als die Hälfte der befragten Lehrkräfte beklagte sich zudem über eine unzureichende technische Infrastruktur an den Schulen (ebd., S. 349).

Greifenstein, Graßl und Fraser (2021) befragten insgesamt 200 Grundschullehrkräfte, die bereits mit ihren Schülerinnen und Schülern programmiert hatten, welche Herausforderungen sie dabei erlebten. Mit 45.0% wurden organisatorische Herausforderungen der Schulen am häufigsten genannt. Diese betrafen z. B. die Finanzierung, den Zugang zu Ressourcen und die Internetverbindung. 41.5% nannten Herausforderungen im Zusammenhang mit den Schülerinnen und Schüler, z. B. mangelnde digitale Kompetenz oder eine hohe Varianz der individuellen Voraussetzungen. In Bezug auf die Lehrkräfte äußerten sich 36.5% der Befragten und nannten Herausforderungen in Bezug auf die individuelle Unterstützung der Schülerinnen und Schüler sowie fehlendes oder veraltetes Fachwissen. In Bezug auf das Programmieren selbst wurde von kognitiven und affektiven Herausforderungen berichtet (21.5%), z. B. dass die Schülerinnen und Schüler beim Verbessern ihrer Programme ungeduldig werden und Schwierigkeiten haben zu verstehen, dass jeder Schritt in einem Programm detailliert beschrieben werden muss. 17.0% nannten Herausforderungen im Zusammenhang mit der Zeit und Lehrpläne, z. B. dass die im Lehrplan vorgesehene Zeit nicht ausreicht, um das Programmieren zu verstehen. 4.0% äußerten Herausforderungen in Bezug auf das Elternhaus, z. B. dass Schülerinnen und Schüler zu Hause keinen Zugang zu Informatiksystemen haben. Vergleichbare Ergebnisse konnten auch von Sentance und Csizmadia (2017, S. 478 ff.) in Bezug auf das Unterrichten von Informatik identifiziert werden. In einer Befragung von 1126 britischen Lehrkräften aller Schulformen äußerten sich 339 Lehrkräfte zu Herausforderungen, mit denen sie konfrontiert waren. Die am häufigsten genannten Herausforderungen waren das Fachwissen der Lehrkräfte, mangelndes Verständnis der Inhalte seitens der Schülerinnen und Schüler, technische Probleme in den Schulen, der Umgang mit unterschiedlichen Leistungsniveaus sowie die Bereitschaft der Schülerinnen und Schüler, Probleme zu lösen.

Im Abschlussbericht des niedersächsischen Projekts *Informatische Bildung und Technik in der Grundschule* (siehe Abschnitt 2.3) benennen Breiter u. a. (2020, S. 76) die infrastrukturelle Situation an den beteiligten Schulen als Gelingensbedingung für die erfolgreiche Durchführung des Projekts und das Testen der darin entwickelten Unterrichtsmaterialien. In der Abschlussbefragung des Projekts (n=196) wurde die Verfügbarkeit und Zuverlässigkeit der Geräte und der Internetverbindung durchschnittlich mit der Schulnote befriedigend, der technische Support mit einer 3.43 und das WLAN mit einer 4.0 bewertet. Auch im Abschlussbericht der Explorationsstudie von Murmann u. a. (2018, S. 43 ff.) zum Einsatz des *Calliope mini* in der Grundschule werden verschiedene Herausforderungen beschrieben, die sich für Lehrkräfte während der Erprobung ergaben. Alle drei Schulen, die an der Studie teilnahmen, beschrieben verschiedene Probleme mit der Technik, z. B. dass die Laptops keine Verbindung zum Internet herstellen konnten, dass die Schülerinnen und Schüler sich nicht in das Schulnetz einloggen konnten oder dass der webbasierte Editor nicht richtig funktionierte. Eine weitere Herausforderung war die Unterrichtsdauer. Hier berichteten die Lehrkräfte, dass die Beantwortung von Schülerfragen oft mehr Zeit in Anspruch nahm als in anderen Unterrichtsstunden, und dass 60-minütige Unterrichtsstunden im Vergleich zu 90-minütigen tendenziell zu kurz waren. Als dritte Herausforderung wird die Lehrer-Schüler-Relation thematisiert. Hier wurde geäußert, dass es für eine einzelne Lehrkraft sehr hektisch werden kann und man sich eine weitere Hilfsperson oder das Arbeiten in Halbgruppen wünsche.

# 3 Programmieren

Die öffentliche und politische Debatte, ob und wann informatische Inhalte in die Curricula der Primarstufe aufgenommen werden sollten, beschränkt sich zumeist auf den Themenbereich der Programmierung. So äußert bspw. die ehemalige Bundeskanzlerin Angela Merkel bereits 2016 die Vermutung, dass – neben Lesen, Schreiben und Rechnen – „die Fähigkeit zum Programmieren eine der Basisfähigkeiten von jungen Menschen wird[1]“. Dorothee Bär, einstige Staatsministerin für Digitales, äußert in einem Interview aus 2018[2], dass das Programmieren in die Lehrpläne der Grundschule müsse und so wichtig sei, wie Lesen und Schreiben. Auch im Jahr 2021 ist das Thema erneut sichtbar in der Politik: die CSU fordert in einer Beschlussvorlage für die CSU-Landesgruppen-Klausur, dass es in ganz Deutschland flächendeckend das Fach Programmieren geben müsse[3]. Die Bundesregierung veröffentlicht im Jahr 2022 die *Digitalstrategie Deutschland* (BMDV 2022), die einen übergeordneten Rahmen der Digitalpolitik für die Legislaturperiode 2021–2025 vorgibt. Darin steht, dass im Rahmen des *MINT-Aktionsplans 2.0* mit gezielten Förderungen entlang der gesamten Bildungskette Zugänge zur MINT-Bildung geschaffen werden sollen. Explizit wird dabei die Vermittlung digitalisierungsbezogener Kompetenzen in der frühkindlichen und schulischen Bildung angesprochen. Im Kontext der Programmierung wird explizit auf die Initiativen *Girls' Day* und *YouCodeGirls* eingegangen, die Mädchen und junge Frauen frühzeitig auf ihrem Bildungsweg stärken sollen (ebd., S. 13 f.). Bildungsministerin Bettina Stark-Watzinger betont in einem Interview im Jahr 2023[4], dass digitale Bildung mehr sei als „Tablets, interaktive Whiteboards und WLAN in Schulen“. Sie fügt hinzu, dass junge Menschen verstehen müssen, „was ein Algorithmus ist und wie man programmiert“ und diese Themen in die Lehrpläne gehören.

Aus Sicht der Informatikdidaktik stellt das Lernen und Lehren der Programmierung jedoch eine große Herausforderung dar:

> *„Decades of experience have shown that learning to program is a difficult process for many people. Introductory programming courses typically have high rates of student dropout and failure. This creates significant challenges for educators, who naturally want their students to progress successfully, and sometimes face significant institutional pressure if they do not.“* (Robins 2019, S. 327)

[1] https://www.golem.de/news/bundeskanzlerin-merkel-programmieren-wird-basisfaehigkeit-von-jungen-menschen-1612-125047.html

[2] https://www.spiegel.de/politik/deutschland/dorothee-baer-programmieren-ist-so-wichtig-wie-lesen-und-schreiben-a-1196619.html

[3] https://www.handelsblatt.com/politik/deutschland/digitalisierung-csu-fordert-programmieren-als-schulfach-und-digitalere-kitas/26764316.html?ticket=ST-8616468-oADhlBiiJxzVQ5cZexLK-ap5

[4] https://www.bild.de/politik/inland/politik-inland/bildungsministerin-fordert-praemien-mehr-geld-fuer-top-lehrer-83173464.bild.html

## 3.1 Stellenwert der Programmierung

Warum gerade das Programmieren dennoch Einzug in den Grundschulunterricht halten sollte, lässt sich aus verschiedenen Perspektiven beantworten, die im Folgenden umrissen werden.

*Zentrales Element der Informatik*

Unter dem Programmieren versteht man laut Claus und Schwill (2006, S. 529) „zum einen den Vorgang der Programmerstellung und zum anderen das Teilgebiet der Informatik, das die Methoden und Denkweisen beim Entwickeln von Programmen umfasst." (vgl. Kapitel 9.3). Obwohl die Programmierung keinesfalls mit der Informatik gleichzusetzen ist (vgl. Denning, Tedre und Yongpradit 2017, S. 32), kann sie als zentrales Element der Informatik angesehen werden:

> *„The defining characteristic of the computer is its programmability and programming is the essence of computing/informatics. Indeed, computing is much more than programming, but programming – the process of expressing one's ideas and understanding of the concepts and processes of a domain in a form that allows for execution on a computing device without human interpretation – is essential to computing."*
>
> (Caspersen 2018, S. 109)

> „Zwar ist Programmierung nicht mit Informatik [...] gleichzusetzen, aber sie vermittelt selbst im einfachsten Assemblerprogramm ein Grundverständnis darüber, was ein Computer ist und worauf die Basis seiner Leistungsfähigkeit und Universalität beruht."
>
> (Mittermeir 2010, S. 54)

Hoppe und Luther (1996, S. 11) bezeichnen das Programmieren sogar als Primärerfahrung der Informatik – ebenso wie das elementare Rechnen als Primärerfahrung der Mathematik anzusehen sei. Schubert (1991, S. 27) warnt in diesem Kontext bereits 1991 davor, die Programmierung mit der Codierung gleichzusetzen und den Begriff auf das Schreiben eines Programms in einer Programmiersprache zu beschränken. Stattdessen solle im Unterricht das Problemlösen unter Anwendung von Prinzipien und Methoden der Informatik im Vordergrund stehen und die Programmiersprache als Mittel zum Zweck im Hintergrund bleiben. Auch Hubwieser (2007, S. 88), stellt das Programmieren als wesentlichen Bestandteil der Schulinformatik heraus – jedoch nur unter der Prämisse, dass es sich dabei um die Implementierung eines vorher entwickelten Modells handelt und die Syntax der verwendeten Programmiersprache nicht im Vordergrund steht.

*Problemlösen lernen*

Im Rahmen der Debatte um das Programmieren in der Schule stößt man immer wieder auf das Argument, dass es sich nicht lohnen würde, alle Schülerinnen und Schüler im Programmieren zu unterrichten, da diese Tätigkeit in Zukunft automatisiert und damit nicht mehr benötigt würde (Bell, Duncan und Rainer 2018, S. 13). Hierbei ist darauf hinzuweisen, dass es sich beim Unterrichten der Programmierung keinesfalls um eine vorgezogene Berufsbildung handeln soll (vgl. Schubert 1991, S. 28). Vielmehr verspricht man sich, die Kinder im *Computational Thinking* zu schulen und sie so zu mündigen Bürgerinnen und Bürgern der zunehmend digitalen Welt auszubilden:

> *„Rather, the goal is to equip children with the CT skills required to understand computing technology, use it effectively and be informed consumers and citizens in our increasingly digital world. Having a complete and deep knowledge of computer science and software engineering concepts is not necessary – rather, it is important to have enough understanding of these to be aware of how these might impact the world around us.“*
>
> (Bell, Duncan und Rainer 2018, S. 13)

Im informatikdidaktischen Diskurs wird das Erlernen der Programmierung als effektiver Weg angesehen, um Fähigkeiten des *Computational Thinking* auszubilden:

> *„As you learn to code, you also become a better thinker. For example, you learn how to break complex problems into simpler parts. You learn how to identify problems and debug them. You learn how to iteratively refine and improve designs over time. Computer scientist Jeannette Wing has popularized the term computational thinking to refer to these types of strategies. Once you learn these computational-thinking strategies, they can be useful in all types of problem-solving and design activities, not just in coding and computer science.“* (Resnick und Robinson 2017, S. 48)

Gleichzeitig wird jedoch darauf hingewiesen, dass dies kein automatischer Prozess ist. Um Aspekte des *Computational Thinking* durch das Programmieren zu fördern, müssen diese einerseits konkret adressiert werden, andererseits bedarf es adäquater didaktischer Ansätze:

> *„Although programming is an excellent tool for scaffolding the development of CT skills, it is unlikely to work if it is taught without a specific focus on these skills. It is very easy for pupils (and educators) to slip into using ‚copy and paste' coding exercises only, where pupils will be following instructions, rather than working through a problem-solving process.“* (Bell, Duncan und Rainer 2018, S. 17)

> *„Decades of research with children suggests that young learners who may be programming don't necessarily learn problem solving well, and many, in fact, struggle with algorithmic concepts especially if they are left to tinker in programming environments, or if the learning is not scaffolded and designed using the right problems and pedagogies.“*
>
> (Grover 2013[5])

Rich u. a. (2017a, S. 2) stellen weitere Vorzüge des Programmierenlernens heraus. So wurde bereits mehrfach der Zusammenhang zwischen dem Lernen des Programmierens und einer Verbesserung von mathematischen Fähigkeiten, z. B. geometrisches Verständnis, räumliches Vorstellungsvermögen oder logisches Denken, nachgewiesen. Darüber hinaus suggerieren weitere Forschungsergebnisse, dass das Programmierenlernen die Kreativität der Schülerinnen und Schüler fördert, sowie ihre Bereitschaft, sich mit herausfordernden Aufgaben auseinanderzusetzen.

[5] https://www.edsurge.com/news/2013-05-28-opinion-learning-to-code-isn-t-enough

*Verstehen und Gestalten der digitalen Welt*

Rushkoff macht in seinem Buch *Program or be Programmed* (2010) darauf aufmerksam, dass die Fähigkeit oder zumindest ein Grundverständnis des Programmierens nötig ist, um der digitalen Welt nicht ausgeliefert zu sein:

> *„Digital technology is programmed. This makes it biased toward those with the capacity to write the code. In a digital age, we must learn how to make software, or risk becoming the software. It is not too difficult or too late to learn the code behind the things we use – or at least to understand that there is code behind their interfaces. Otherwise, we are at the mercy of those who do the programming, the people paying them, or even the technology itself."* (Rushkoff 2010, S. 134)

Er kritisiert, dass die meisten *Computer Literacy Curricula* nur die Nutzung von Programmen statt der Programmierung vorsehen, und sieht die Gefahr, dass sich bei den Schülerinnen und Schülern eine nur auf die Nutzerperspektive beschränkte Sicht auf digitale Technologien manifestiert (ebd., S.136). Auch er betont, dass es ihm nicht darum ginge, möglichst viele Schülerinnen und Schüler für ein Informatikstudium zu gewinnen, sondern ihnen eine grundlegende Vorstellung über die Programmierung zu vermitteln. Sie sollten verstehen, welche Entscheidungen entlang des Programmierprozesses getroffen werden und wie diese Entscheidungen die Funktionsweise der Software und somit die Nutzerinnen und Nutzer beeinflussen (ebd., S.140).

Auch Resnick u. a. (2009; 2017) betonen mehrfach, dass sie im Hinblick auf das Programmieren im Unterricht das Potenzial sehen, den Schülerinnen und Schülern eine Perspektive für den Umgang mit digitalen Technologien zu eröffnen, die über deren bloße Nutzung hinausgeht. Sie stellen im Gegensatz zu Rushkoff jedoch das gestalterische Potenzial der Programmierung heraus:

> *„As we see it, digital fluency requires not just the ability to chat, browse, and interact but also the ability to design, create, and invent with new media [...]. To do so, you need to learn some type of programming. The ability to program provides important benefits. For example, it greatly expands the range of what you can create (and how you can express yourself) with the computer."* (Resnick u. a. 2009, S. 3)

Resnick und Robinson (2017) sehen in diesem gestalterischen Aspekt des Programmierens nicht nur eine Möglichkeit für Schülerinnen und Schüler, sich selbst auszudrücken, sondern auch aktiv an der Gesellschaft teilzunehmen. Zudem haben sie beim Programmieren in der Programmiersprache *Scratch* die Erfahrung gemacht, dass die Schülerinnen und Schüler Selbstvertrauen und Stolz bzgl. ihrer Fähigkeit, etwas zu erschaffen und sich mit neuen Technologien auszudrücken, entwickeln:

> *„In today's society, digital technologies are a symbol of possibility and progress. When children learn to use digital technologies to express themselves and share their ideas*

> *through coding, they begin to see themselves in new ways. They begin to see the possibility for contributing actively to society. They begin to see themselves as part of the future. As we've introduced Scratch to young people, I've been excited by what they've created — and what they've learned in the process. But what excites me most is the way that many Scratchers start to see themselves as creators, developing confidence and pride in their ability to create things and express themselves fluently with new technologies.“* (Resnick und Robinson 2017, S. 50 f.)

## 3.2 Auswirkungen des Programmierens im Kontext der Grundschule

Der Großteil der Forschungsarbeiten zu Auswirkungen des Programmierens im Kontext der Grundschule beleuchten und analysieren ausgewählte Teilaspekte. Barcelos u. a. (2018) analysieren in ihrer Metastudie verschiedene Untersuchungen, die den Zusammenhang zwischen *Computational Thinking* und mathematischen Fertigkeiten aufzeigen und beschreiben. Dabei identifizieren sie verschiedene Bereiche der Mathematik, für die ein Zusammenhang gefunden werden konnte, z. B. Euklidische Geometrie (in 20 Studien), Algebra (in 12 Studien) und Arithmetik (in 10 Studien) (ebd., S. 828). Die Metastudie beschränkt sich jedoch nicht auf die Schulform Grundschule. Taylor, Harlow und Forret (2010) untersuchen das Potenzial von Scratch zur Förderung des mathematischen Denkens. Die Studie untersucht 60 Kinder im Alter von neun bis zehn Jahren und zeigt, dass diese relativ anspruchsvolle mathematische Konstrukte erforschen und anwenden können, wenn sie in Scratch eingebettet sind. Die Autoren zweifeln jedoch an, dass mathematisches Fachwissen gelernt wird (ebd., S. 567). Gökçe und Yenmez (2022) untersuchen in einem Pretest–Posttest-Design 524 Schülerinnen und Schüler zwischen zehn und zwölf Jahren und zeigen, dass sich durch das Programmieren in Scratch ihre Fähigkeiten zum reflektierten Denken, Problemlösen und *Computational Thinking* verbessern. Diethelm u. a. (2020) untersuchen und vergleichen das informatische Selbstkonzept von Dritt- und Viertklässlern, die Informatikunterricht erhalten haben, mit dem von Schülerinnen und Schülern, die keinen Informatikunterricht erhalten haben. Die Ergebnisse zeigen, dass beide Gruppen ein eher positives informatisches Selbstkonzept haben, sich die Kinder mit Informatikunterricht jedoch in allen Dimensionen positiver einschätzen als die Kontrollgruppe. Der Informatikunterricht enthielt vier Module, die sich mit dem Aufbau des Internets, Verschlüsselung, Programmieren sowie Informationen und Daten beschäftigten. Duncan, Bell und Atlas (2017) analysieren Feedback-Formulare von dreizehn Grundschullehrkräften, welche das Programmieren und weitere informatische Themen nach einer entsprechenden Fortbildung in ihrem Unterricht behandelt haben. In den Antworten der Lehrkräfte wurde besonders betont, wie viel Spaß die Schülerinnen und Schüler am Unterricht hatten und wie sehr sie sich beteiligten und konzentrierten. Zudem erwähnten sie häufig die Themen Teamarbeit, Kooperation und Kommunikation und stellten fest, dass das Unterrichten der Inhalte dazu beitrug, soziale Kompetenzen zu vermitteln – etwas, das sie im Vorfeld nicht vermutet hätten. Mehrere Lehrkräfte beobachteten, dass Schüle-

rinnen und Schüler, die sich normalerweise nicht am Unterricht beteiligten, dies nun auf einmal taten. Magenheim u. a. (2018b) untersuchen in einem Pretest–Posttest-Design die Einstellungen, das Interesse und die Motivation von Grundschulkindern bzgl. Informatik. An der Studie nahmen 450 Schülerinnen und Schüler der zweiten bis vierten Jahrgangsstufe teil, die im Rahmen des Sachunterrichts an drei *Unplugged*-Modulen zur Informatik teilgenommen hatten, darunter eines zu den Grundlagen der Programmierung. Darin zeigte sich z. B., dass Schülerinnen und Schüler vor allem aus Interesse am Fach und aus themenbezogener Motivation am Informatikunterricht teilnehmen. Sie fanden die Module spannend, motivierend und hatten Spaß dabei. Die Schülerinnen und Schüler waren sowohl für Übungen ohne Computer als auch für Übungen mit Computern offen – es zeigte sich jedoch eine klare Tendenz zum Lernen mit Computern (ebd., S. 345).

Greifenstein, Graßl und Fraser (2021) befassen sich in ihrer Fragebogenstudie unter anderem mit dem ganzheitlichen Potenzial des Programmierens in der Grundschule. Sie befragten insgesamt 200 Grundschullehrkräfte, die bereits mit ihren Schülerinnen und Schülern programmiert hatten, welche Potenziale sie bzgl. des Programmierens in der Grundschule sehen. Sie identifizieren sieben Kategorien von Potenzialen, von denen für diese Arbeit in erster Linie die Kategorie *Skills Acquisition* relevant ist. Darunter fallen auf kognitiver Ebene die Förderung von logischem Denken und Problemlösen sowie eine Sprachförderung. Weiterhin werden affektive Aspekte wie Interesse, Motivation, Spaß und der Abbau von Vorurteilen gegenüber des Programmierens und Informatik im Allgemeinen genannt. Nach Angaben der Lehrkräfte werden darüber hinaus *Digital Literacy*, *Computational Literacy* und Kreativität sowie das Selbstvertrauen und die Selbstständigkeit der Schülerinnen und Schüler gefördert. Einen ebenfalls ganzheitlichen Ansatz verfolgen Murmann u. a. (2018) in ihrer Explorationsstudie zum Einsatz des *Calliope mini* in der Grundschule. Im Fokus der Studie steht die Erprobung des Microcontrollers und entsprechender Unterrichtsmaterialien an drei Grundschulen sowie die Frage, welche Kompetenzen die Schülerinnen und Schüler erworben haben und ob algorithmisches Denken initiiert werden konnte. Hierzu wurden über die Dauer eines Jahres verschiedene Daten gesammelt, die qualitativ ausgewertet wurden, z. B. Beobachtungsprotokolle, Audioaufnahmen von Gesprächen mit Lehrkräften und Kindern, videografierte Feedbackphasen und Bilder aus dem Unterricht (ebd., S. 33 ff.). In der Analyse, welche Kompetenzen bei den Schülerinnen und Schüler festgestellt bzw. gefördert werden konnten wurden folgende Kompetenzen als Hauptkategorien gebildet: (1) Zusammenarbeit, (2) Entwicklung und Umsetzung von Ideen, (3) Computer-Basics, (4) Programmierung eines Mikrocontrollers sowie Nachvollziehen informatischer Grundkonzepte, (5) Fachsprache – passives Verstehen und aktive Nutzung, (6) Bewusste Wahrnehmung von Informatiksystemen in der Lebenswelt, (7) Selbstwirksamkeit, (8) Lerntransfer, (9) Frustrationstoleranz und (10) Lernbereitschaft (ebd., S. 39). Es ist darauf hinzuweisen, dass unter den Kategorien auch Daten erfasst werden, die Aufschluss über die Frage geben, über welche Kompetenzen die Schülerinnen und Schüler nicht verfügten bzw. welche Kompetenzen sich nicht entwickeln konnten (ebd., S. 51). Bei der Frage, ob durch das Programmieren mit dem *Calliope mini* algorithmisches Denken initiiert werden konnte, beziehen die Forschenden sich auf die Definition des Algorithmisierens als wesentliche überfachliche

Kompetenz im Alltag von Hoffmann, Wendlandt und Wendlandt (2017) und schlagen vier Stufen des Algorithmisierens vor (ebd., S. 83):

- Stufe 1: Algorithmische Strukturen im Alltagsleben erkennen und entwickeln;
- Stufe 2: Algorithmische Strukturen für Automaten kennen;
- Stufe 3: Algorithmische Strukturen für Automaten nach Anleitungen ausführen;
- Stufe 4: Algorithmische Strukturen für Automaten selbst entwickeln;

Sie stellen fest, dass nur wenige Schülerinnen und Schüler am Ende der Unterrichtssequenz die Stufe 4 des algorithmischen Denkens erreicht hatten. Allerdings erreichten fast alle Schülerinnen und Schüler Aspekte der vier Stufen und hatten einen positiven Lerneffekt in Bezug auf das algorithmische Denken (ebd., S. 86).

# 4 Lehrerbildung im Bereich der informatischen Bildung

Lehrkräfte sind zentrale Akteure im Bildungssystem und laut Bildungs- und Erziehungsauftrag der Grundschule im bayerischen LehrplanPLUS „Experten für die Themen und Inhalte sowie für die Gestaltung von Unterrichtsprozessen“ (StMBKWK 2014, S. 24). Die Kultusministerkonferenz bezeichnet Grundschullehrkräfte in ihren *Empfehlungen zur Arbeit in der Grundschule* ebenfalls als „in gleicher Weise Fachleute für die Begleitung der Entwicklungsprozesse wie für das Lernen im Grundschulalter“ (2015, S. 21). Die Kerntätigkeit von Lehrkräften – das Unterrichten – wird von der KMK für die Grundschule wie folgt umschrieben:

> „Um jedes Kind in seiner Lernentwicklung zielgerichtet zu unterstützen, decken Lehrerinnen und Lehrer seine Potenziale auf, nehmen seine Kompetenzen als Ausgangspunkt nächster Lernschritte wahr und wertschätzen diese. Lehrkräfte initiieren, instruieren, begleiten und beraten, fordern und fördern, diagnostizieren und dokumentieren, reflektieren und bewerten den Lernprozess jedes Kindes. Zugleich finden sie effektive Wege, um mit der Unterschiedlichkeit der Lernenden entwicklungsfördernd umzugehen.“ (KMK 2015, S. 20)

## 4.1 Bedeutung der Lehrkraft im Allgemeinen

Nach Annahme der etablierten Angebots-Nutzungs-Modelle des schulischen Lernens (z. B. Seidel 2014; Helmke 2017) ist der Lernprozess als komplexes Zusammenspiel mehrerer Faktoren, z. B. sozio-kulturellen Rahmenbedingungen, individuellen Voraussetzungen der Schülerinnen und Schüler, anzusehen (siehe Abbildung 4.1). Unterricht ist darin als Lernangebot zu verstehen, welches wiederum von Lernenden als solches wahrgenommen und genutzt werden muss. Demnach kann selbst durch die beste Lehrkraft nie garantiert werden, dass es auf Seiten der Schülerinnen und Schüler zu den gewünschten Wirkungen kommt. Dennoch kann die Lehrkraft durch die Gestaltung der Unterrichtsangebote das schulische Lernen maßgeblich beeinflussen:

> „Das Unterrichts-Angebot [...] kann sehr unterschiedlich sein: Es kann viel oder wenig Lernzeit zur Verfügung stehen, der Unterricht und das Lernmaterial können mehr oder weniger gut strukturiert oder unterstützend sein, mehr oder weniger stark aktivieren. Je besser dieses Angebot, umso wahrscheinlicher ist es, dass Schülerinnen und Schüler lernen.“ (Kunter und Trautwein 2013, S. 17 f.)

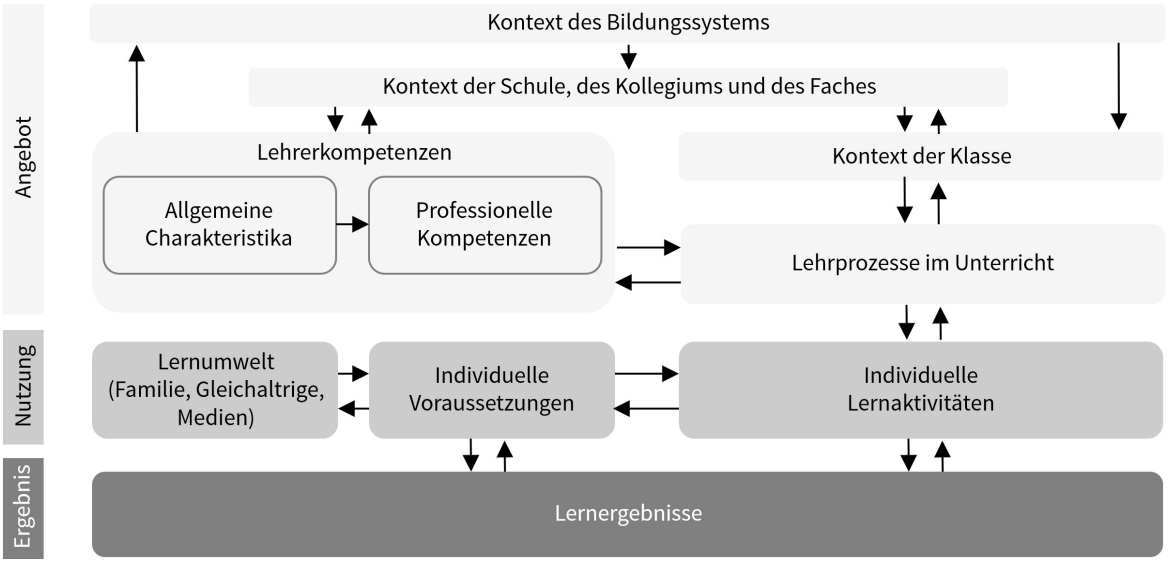



*Abbildung 4.1 Abstrahiertes Angebots-Nutzungs-Modell aus Seidel (2014, S. 858)*

*Professionelle Kompetenz von Lehrkräften*

Im wissenschaftlichen Diskurs über die Voraussetzungen für erfolgreiches Handeln im Unterricht wird meist auf die professionelle Kompetenz der Lehrkräfte verwiesen (z. B. Baumert und Kunter 2006; Baumert u. a. 2011). Da die Autorinnen und Autoren in der Regel den Kompetenzbegriff nach Weinert (2001b, S. 27 f. , siehe Abschnitt 11.1.1) verwenden, umfasst der Begriff der *professionellen Kompetenz* demnach die Fähigkeiten und Fertigkeiten, die zur Bewältigung spezieller beruflicher Aufgaben notwendig sind, sowie die Bereitschaft, dies zu tun. Die professionelle Kompetenz von Lehrkräften wird als mehrdimensionales Konstrukt angesehen, das einerseits aus kognitiven Aspekten, wie Wissen und Vorstellungen, und andererseits aus motivationalen und affektiven Aspekten, wie Zielen, Motiven oder Gefühlen, besteht. Baumert und Kunter (2006) entwickelten ein Modell zur Systematisierung dieser Aspekte, nach dem professionelle Handlungskompetenz aus dem Zusammenspiel der folgenden Faktoren entsteht:

> - „spezifischem, erfahrungsgesättigten deklarativen und prozeduralen Wissen (Kompetenzen im engeren Sinne: Wissen und Können);
> - professionellen Werten, Überzeugungen, subjektiven Theorien, normativen Präferenzen und Zielen;
> - motivationalen Orientierungen sowie
> - metakognitiven Fähigkeiten und Fähigkeiten professioneller Selbstregulation."
>
> (Baumert und Kunter 2006, S. 481)

Einer der Kernaspekte der professionellen Kompetenz bildet das Professionswissen (siehe Abbildung 4.2), welches sich im Modell von Baumert und Kunter in Anlehnung an Shulman (1987) in allgemeines pädagogisches Wissen (*General Pedagogical Knowledge*), Fachwissen (*Subject-matter Content Knowledge*) und fachdidaktisches Wissen (*Pedagogical Content Knowledge*) aufteilt (vgl. Baumert und Kunter 2006, S. 481 f.).

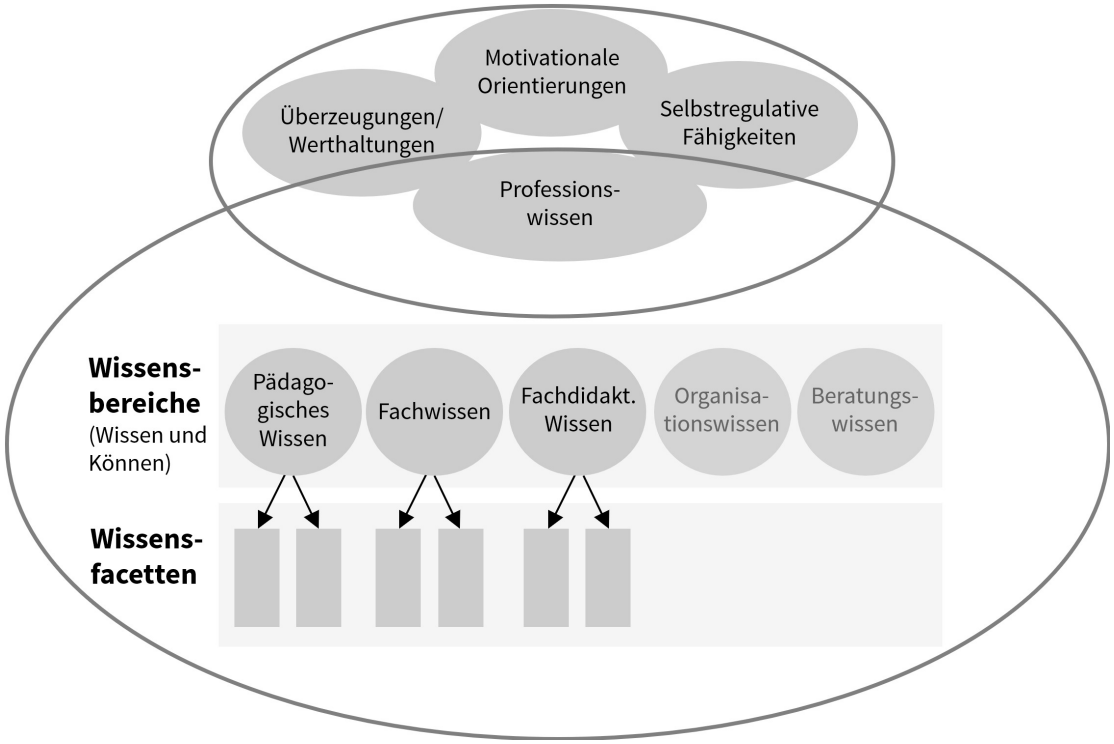



*Abbildung 4.2 Modell professioneller Handlungskompetenz aus Baumert und Kunter (2006, S. 482)*

Zu den Facetten des allgemeinen pädagogischen Wissens werden bspw. das Wissen um Leistungsbeurteilung, über Lernprozesse oder effektive Klassenführung gezählt (ebd., S. 484). Der Bereich des Fachwissens hat „sein Fundament in der akademischen Referenzdisziplin, stellt aber selbst einen Wissensbereich eigenen Rechts dar, der durch die Lehrplanarbeit definiert und in Rückkopplung mit der Unterrichtspraxis fortgeschrieben wird“ (ebd., S. 495). Die Wissensfacetten des Fachdidaktischen Wissens umschreibt Shulman wie folgt:

> *„Within the category of pedagogical content knowledge I include, for the most regularly taught topics in one's subject area, the most useful forms of representation of those ideas, the most powerful analogies, illustrations, examples, explanations, and demonstrations- in a word, the ways of representing and formulating the subject that make it comprehensible to others. [...] Pedagogical content knowledge also includes an understanding of what makes the learning of specific topics easy or difficult: the conceptions and preconceptions that students of different ages and backgrounds bring with them to the learning of those most frequently taught topics and lessons. If those preconceptions are misconceptions, which they so often are, teachers need knowledge of the strategies most likely to be fruitful in reorganizing the understanding of learners, because those learners are unlikely to appear before them as blank slates.“*
>
> (Shulman 1986, S. 9 f.)

*Bedeutung für die Unterrichtsqualität*

Dem bereits erwähnten Angebots-Nutzungs-Modell schulischen Lernens von Seidel (siehe Abbildung 4.1) ist zu entnehmen, dass die professionellen Kompetenzen der Lehrkräfte in Wechselwirkung mit den Lehrprozessen im Unterricht stehen. Lindner und Mayerhofer (2018, S. 59 ff.) befassen sich theoriebasiert mit der Relevanz der Lehrkraft im „kompetenzorientierten guten Unterricht" und zeigen auf, dass zahlreiche Merkmale guten Unterrichts direkt durch die Lehrpersonen beeinflusst werden können.

Das Forschungsprojekt *COACTIV* (Kunter u. a. 2011) nähert sich der Thematik empirisch und untersucht mit einem Schwerpunkt auf das Fach Mathematik die Genese, Struktur und Handlungsrelevanz professioneller Kompetenz von Lehrkräften der Sekundarstufe. Dabei konnte gezeigt werden, dass vor allem das fachdidaktische Wissen entscheidend für die Unterrichtsgestaltung ist und sich positiv auf dessen Qualität auswirkt: Lehrkräfte, die ein höheres Maß an fachdidaktischem Wissen aufwiesen, gestalteten ihren Unterricht kognitiv aktivierender und wurden von den Lernenden als stärker unterstützend wahrgenommen (vgl. Baumert und Kunter 2011, S. 184). Dies konnte für das Fachwissen, trotz einer hohen Korrelation mit dem fachdidaktischen Wissen, nicht gezeigt werden (ebd., S. 185). Baumert und Kunter machen in diesem Zusammenhang jedoch darauf aufmerksam, dass in verschiedenen qualitativen Fallstudien gezeigt wurde, dass das Fachwissen den Entwicklungsraum des fachdidaktischen Wissens und damit indirekt auch die Unterrichtsqualität definiert. Im Rahmen von *COACTIV* wurde außerdem der Enthusiasmus von Lehrkräften – aufgeteilt in Enthusiasmus für das Fach Mathematik und das Unterrichten im Allgemeinen – als ein Beispiel für unterrichtsbezogene Motivation untersucht (Kunter 2011). Dabei zeigte sich, dass ein hoher Enthusiasmus für das Unterrichten mit allen Dimensionen der Unterrichtsqualität assoziiert werden konnte: eine bessere Klassenführung, höhere kognitive Aktivierung und mehr Unterstützung für die Lernenden. Bzgl. der Begeisterung für das Fach konnte kein systematischer Zusammenhang mit Indikatoren der Unterrichtsqualität nachgewiesen werden (ebd., S. 268).

*Bedeutung für den Lernerfolg der Schülerinnen und Schüler*

Die Bedeutung der Lehrkraft für die schulische Entwicklung und den Lernerfolg von Schülerinnen und Schülern wird vor allem seit der Veröffentlichung der sogenannten *Hattie-Studie* (Hattie 2009) vielfach diskutiert und betont. Um die übergeordnete Frage zu beantworten, welche Merkmale für das Lernen in der Schule besonders relevant sind, wertet Hattie 815 englischsprachige Metaanalysen zum Einfluss unterschiedlicher Faktoren auf den kognitiven Lernerfolg von Schülerinnen und Schülern aus. Ein Ergebnis seiner quantitativen Synthese, die insgesamt über 52.000 Einzelstudien berücksichtigt, ist eine Rangliste, welche die Effekte von 138 einzelnen Faktoren ihrer Bedeutung nach ordnet. Die Faktoren werden wiederum sechs unterschiedlichen Faktorengruppen zugeordnet, für die eine durchschnittliche Effektstärke berechnet wird (siehe Tabelle 4.1): *Lernende*, *Familie*, *Schule*, *Lehrpersonen*, *Lehrpläne* und *Unterricht*. Die Faktorengruppe *Lehrpersonen* übt in Hatties Analyse mit einer Effektstärke von $d = 0.49$ den größten Einfluss auf die Leistungen der Schülerinnen und Schüler aus. Dieses Ergebnis, welches in Verbindung mit dem Zitat *„teachers*

*Tabelle 4.1 Durchschnittliche Effekte der wichtigsten Determinanten schulischen Lernens aus Hattie (2009, S. 18)*

| Contribution | No. | Studies | People | Effects | *d* | *SE* | *CLE* |
|---|---|---|---|---|---|---|---|
| Student | 139 | 11,101 | 7,513,406 | 38,282 | 0.40 | 0.044 | 29% |
| Home | 36 | 2,211 | 11,672,658 | 5,182 | 0.31 | 0.058 | 22% |
| School | 101 | 4,150 | 4,416,898 | 13,348 | 0.23 | 0.072 | 16% |
| Teacher | 31 | 2,225 | 402,325 | 5,559 | 0.49 | 0.049 | 35% |
| Curricula | 144 | 7,102 | 6,899,428 | 29,220 | 0.45 | 0.076 | 32% |
| Teaching | 365 | 25,860 | 52,128,719 | 55,143 | 0.42 | 0.071 | 30% |

*make the difference"* (Hattie 2009, S. 108) weite Verbreitung fand, wird häufig als zentrales Ergebnis der *Hattie-Studie* angesehen. Dennoch soll an dieser Stelle darauf verwiesen werden, dass Hattie in einer vorangegangenen Studie zu dem Schluss kommt, dass rund die Hälfte der Unterschiede in den Lernleistungen durch Merkmale der Schülerinnen und Schüler bedingt und nur etwa 30% auf die Lehrpersonen zurückzuführen sind (Hattie 2003, S. 1 f.). Zudem wurde durchaus Kritik an der Methodik der Studie geäußert, bspw. wurden hauptsächlich Studien aus den USA, Australien und Neuseeland berücksichtigt, diese stammten allesamt aus den 1980er und 1990er Jahren und beschränkten sich auf quantitativ messbare Faktoren (vgl. Terhart 2014, S. 14). Lotz und Lipowsky (2015) machen auf eine weitere Grenze der *Hattie-Studie* aufmerksam:

> „Die Hattie-Studie ist eine Meta-Metaanalyse, aus der man keine unmittelbaren Handlungsempfehlungen und Rezepte für die Gestaltung von Unterricht ableiten kann. Gleichwohl gibt die Studie Hinweise darauf gibt, welche Merkmale von Unterricht eine größere Beachtung verdienen und welche unwichtiger sind."
>
> (Lotz und Lipowsky 2015, S. 123)

Zu diesem Schluss kommen sie, nachdem sie sich mit zwei von Hattie berücksichtigten Unterrichtsmerkmalen – Lehrerfragen und Feedback – näher auseinandergesetzt haben. Für beide Merkmale zeigen sie, dass es für Rückschlüsse darauf, wie qualitätsvoller Unterricht konkret aussieht, notwendig ist, diese differenziert zu betrachten und weitere Literatur zu sichten (vgl. Lotz und Lipowsky 2015, S. 107 ff.).

Im *COACTIV*-Projekt wurde ebenfalls untersucht, ob sich die professionelle Kompetenz der Lehrkraft – vermittelt über die Gestaltung des Unterrichtsgeschehen – auf die Leistungen der Schülerinnen und Schüler im Mathematikunterricht auswirkt. Tatsächlich konnte nachgewiesen werden, dass die Vorteile in der Unterrichtsgestaltung von Lehrpersonen mit hohem fachdidaktischen Wissen mit einem höheren Lernzuwachs im Fach Mathematik einher gingen (vgl. Baumert und Kunter 2011, S. 183 f.). Vergleichbare Ergebnisse konnten im Rahmen des Programms *SINUS an Grundschulen* für die mathematischen Kompetenzen von Lernenden im Primarbereich gewonnen werden (vgl. Dalehefte u. a. 2014, S. 259). Weitere Evidenzen für den Einfluss professioneller Kompetenz von Lehrkräften auf die Leistungen von Schülerinnen und Schüler wurden außerdem bereits 2006 von Lipowsky zusammengetragen.

## 4.2 Kompetenzen zum Unterrichten des Fachs Informatik

Im Folgenden werden die informatikspezifischen Wissensbereiche der professionellen Kompetenz von Lehrkräften und mögliche zugehörige Wissensfacetten sowie nicht-kognitive Kompetenzbereiche beschrieben. Diese Unterteilung entspricht zum einen dem bereits beschriebenen Modell professioneller Handlungskompetenz von Baumert und Kunter (siehe Abbildung 4.2), zum anderen auch Kompetenzmodellen, die sich explizit auf das Unterricht von Informatik beziehen. Bspw. wurde im Verbundprojekt *Kompetenzen für das Unterrichten in Informatik (KUI)*[1] ein Kompetenzmodell entwickelt, das *Subject Content Competences*, *Pedagogical Content Knowledge* und *Non-Cognitive Competences* umfasst (Bender u. a. 2015b, siehe Abbildung 4.3).

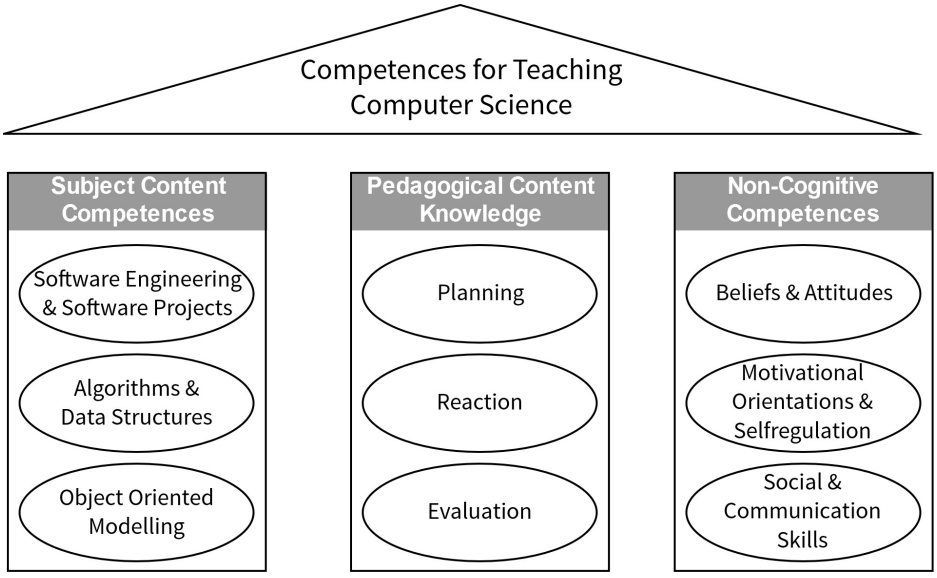



*Abbildung 4.3 Konzeptuelles Framework des KUI-Kompetenzmodells aus Bender u. a. (2015b)*

*Fachwissen*

Zum notwendigen Fachwissen für das Unterrichten des Fachs Informatik an weiterführenden Schulen zählen Bender u. a. (2015b, S. 4) die Wissensfacetten *Software Engineering & Software Projects*, *Algorithms & Data Structures* und *Object Oriented Modeling* (siehe Abbildung 4.3). Auch in den *CSTA Standards for Computer Science Teachers*[2] (2020) wird der Bereich *CS Knowledge & Skills* adressiert:

> *„Standard 1. CS Knowledge & Skills: Effective CS teachers demonstrate and continuously develop thorough knowledge of CS content. They demonstrate proficiency with the CS concepts of the grade bands they teach, and they integrate these concepts with CS practices, including computational thinking. They also understand the progression of content before and after the grade bands they teach.“* (CSTA 2020, S. 2)

[1] https://kw.uni-paderborn.de/fach-psychologie/arbeits-und-organisationspsychologie/forschung/forschungsprojekte/kompetenzen-fuer-das-unterrichten-in-informatik

[2] Die Standards wurden 2003 zunächst als *Standards for CS Educators* von der *International Society for Technology in Education (ISTE)* erstellt und veröffentlich, 2011 aktualisiert, 2019 in Zusammenarbeit mit der *ISTE* umgeschrieben und schließlich im Jahr 2020 von der *CSTA* veröffentlicht.

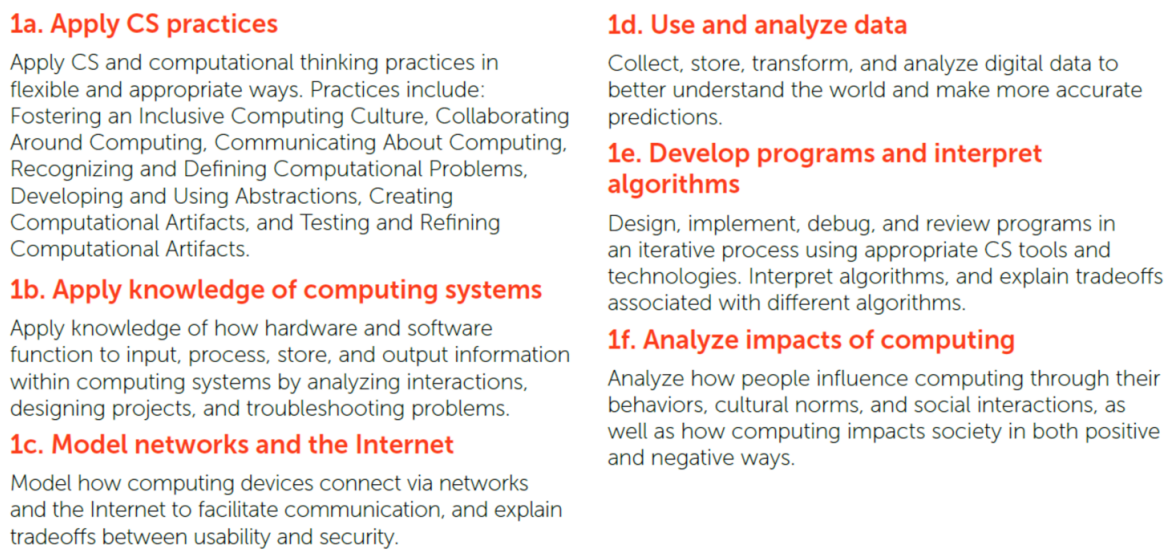



*Abbildung 4.4 Standard „CS knowledge & Skills“ der „CSTA Standards for Computer Science Teachers“ aus CSTA (2020, S. 2)*

Der Bereich wird aufgeteilt in die Unterpunkte *1a. Apply CS practices*, *1b. Apply knowledge of computing systems*, *1c. Model networks and the Internet*, *1d. Use and analysze data*, *1e. Develop programs and interpret algorithms* und *1f. Analyze impacts of computing* (siehe Abbildung 4.4). Innerhalb der *ländergemeinsamen inhaltlichen Anforderungen für die Fachwissenschaften und Fachdidaktiken in der Lehrerbildung* beschreibt auch die Kultusministerkonferenz (2019, S. 36 f.) ein informatikspezifisches Kompetenzprofil inkl. entsprechender fachwissenschaftlicher Studieninhalte für Lehrämter der Sekundarstufe I und das Lehramt an Gymnasien bzw. der Sekundarstufe II. Aufgezählt werden hier folgende Themen: (1) Formale Sprachen und Automaten, (2) Algorithmen und Datenstrukturen, (3) Datenmodellierung und Datenbanksysteme, (4) Programmierung und Softwaretechnik, (5) Rechnerstrukturen und Betriebssysteme sowie (6) Informatik, Mensch und Gesellschaft. Auch die Gesellschaft für Informatik (2016c, S. 13 ff.) beschreibt in ihren *Empfehlungen für Bachelor- und Masterprogramme im Studienfach Informatik an Hochschulen* fachwissenschaftliche Kompetenzen für Informatikerinnen und Informatiker. Bergner u. a. (vgl. 2018, S. 176 ff.) charakterisieren die informatische Fachkompetenz für Fach- und Lehrkräfte im Elementar- und Primarbereich. Sie orientieren sich dabei an den Prozess- und Inhaltsbereichen, die sie zuvor für die Ebene der Kinder formuliert haben (siehe Abschnitt 2.3):

> „Beim Erwerb von informatischen Fachkompetenzen soll jeweils ein enger Bezug zu den später für die Kinder zu organisierenden Lernszenarien hergestellt werden, so dass die gewonnenen Fachkenntnisse schnell in das eigene pädagogische Handeln integriert werden können.“ (Bergner u. a. 2018, S. 175)

*Fachdidaktisches Wissen*

Die Frage, welche Facetten das fachdidaktische Wissen in der Informatik genau umfasst, stand zwar bereits in verschiedenen Forschungsarbeiten im Zentrum (z. B. Guzdial 2010; Buchholz, Saeli und

Schulte 2013; Hubwieser u. a. 2013a; Hubbard 2018; Yadav und Berges 2019), konnte jedoch noch nicht abschließend geklärt werden. Hubwieser u. a. (2013b) entwickelten ein literaturbasiertes *PCK*[3]-Modell, das auf der Basis von Interviews mit erfahrenen Lehrkräften ergänzt wurde. Das Modell umfasst zwei Dimensionen: (1) *Fields of Pedagogical Operation* und (2) *Aspects of Teaching and Learning*. Die erste Dimension umfasst das Planen und Gestalten von Lehr-Lernsituationen, das Eingehen auf die Bedürfnisse der Lernenden während des Unterrichtsprozesses und die Evaluation von ebendiesem. Die zweite Dimension umfasst Unterrichtsaspekte, wie Lerninhalte, Lehrmethoden, Medien und andere fachspezifische Aspekte des Lehrens und Lernens (ebd., S. 98 f.). Auch in den bereits zitierten *CSTA Standards for Computer Science Teachers* (2020) werden zwei Felder des pädagogischen Handelns adressiert:

> *„Standard 4. Instructional Design: Effective CS teachers design learning experiences that engage students in problem solving and creative expression through CS, using pedagogical content knowledge (PCK). They plan to meet the varied learning, cultural, linguistic, and motivational needs of individual students in order to build student self-efficacy and capacity in CS."*
>
> *„Standard 5. Classroom Practice: Effective CS teachers are responsive classroom practitioners who implement evidence-based pedagogy to facilitate meaningful experiences and produce empowered learners of CS."* (CSTA 2020, S. 3)

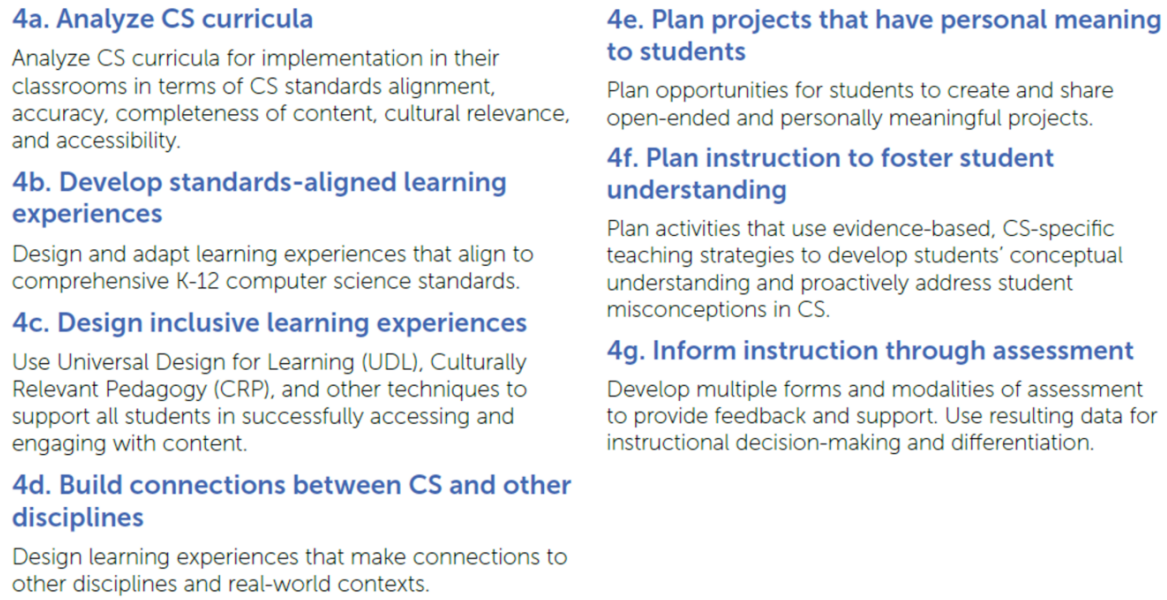
Der Bereich *Instructional Design* wird aufgeteilt in die Unterpunkte *4a. Analyze CS curricula*, *4b. Develop standards-aligned learning experiences*, *4c. Design inclusive learning experiences*, *4d. Build connections between CS and other disciplines*, *4e. Plan projects that have personal meaning to students*, *4f. Plan instruction to foster student understanding* und *4g. Inform instruction through assessment* (siehe Abbildung 4.5). Der Bereich *Classroom Practice* umfasst die Unterpunkte *5a. Use inquiry to facilitate student learning*, *5b. Cultivate a positive classroom climate*, *5c. Promote student self-efficacy*, *5d. Support student collaboration*, *5e. Encourage student communication* und *5f. Guide students' use of feedback* (siehe Abbildung 4.6).

Im informatikspezifischen Kompetenzprofil der Kultusministerkonferenz (2019, S. 35) heißt es bezogen auf die Kompetenzen der Studienabsolventinnen und -absolventen zur Planung und Durchführung von Informatikunterricht:

- „Sie können fachdidaktische Konzepte und empirische Befunde informatikbezogener Lehr-Lernforschung und Diagnosewerkzeuge nutzen, um individuelle Denkwege und Vorstellungen von Schülerinnen und Schülern je nach ihren persönlichen Voraussetzungen, Vorerfahrungen und Fähigkeiten zu analysieren, Schülerinnen und Schüler für das Lernen von Informatik zu motivieren sowie individuelle Lernfortschritte zu fördern und zu bewerten.

[3]Die Abkürzung *PCK* steht für den englischen Begriff *Pedagogical Content Knowledge*, der dem deutschen Begriff „fachdidaktisches Wissen" entspricht.

- Sie kennen Möglichkeiten zur Illustration von informatischen Prinzipien, welche die visuelle, auditive und haptische Wahrnehmung ansprechen und Regeln für leichte Sprache.
- Sie verfügen über ausreichende praktische Kompetenz für den Einsatz von schulrelevanter Hard- und Software, sie können insbesondere die Möglichkeiten, die sich durch den Einsatz von assistiven Technologien im Informatikunterricht eröffnen, einschätzen und bewerten." (KMK 2019, S. 35)



**4a. Analyze CS curricula**

Analyze CS curricula for implementation in their classrooms in terms of CS standards alignment, accuracy, completeness of content, cultural relevance, and accessibility.

**4b. Develop standards-aligned learning experiences**

Design and adapt learning experiences that align to comprehensive K-12 computer science standards.

**4c. Design inclusive learning experiences**

Use Universal Design for Learning (UDL), Culturally Relevant Pedagogy (CRP), and other techniques to support all students in successfully accessing and engaging with content.

**4d. Build connections between CS and other disciplines**

Design learning experiences that make connections to other disciplines and real-world contexts.

**4e. Plan projects that have personal meaning to students**

Plan opportunities for students to create and share open-ended and personally meaningful projects.

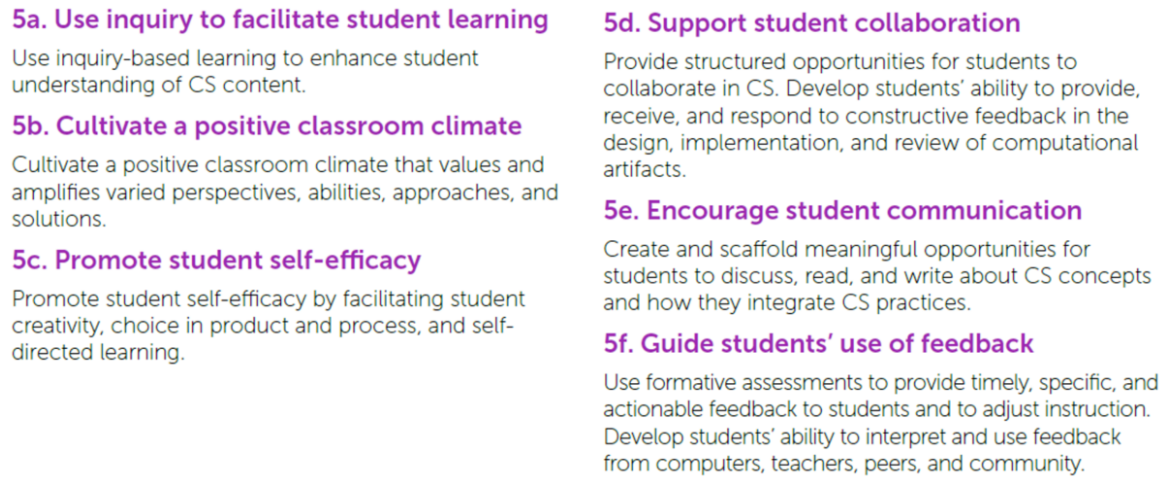
**4f. Plan instruction to foster student understanding**

Plan activities that use evidence-based, CS-specific teaching strategies to develop students' conceptual understanding and proactively address student misconceptions in CS.

**4g. Inform instruction through assessment**

Develop multiple forms and modalities of assessment to provide feedback and support. Use resulting data for instructional decision-making and differentiation.

*Abbildung 4.5 Standard „Instructional Design" der „CSTA Standards for Computer Science Teachers" aus CSTA (2020, S. 2)*



**5a. Use inquiry to facilitate student learning**

Use inquiry-based learning to enhance student understanding of CS content.

**5b. Cultivate a positive classroom climate**

Cultivate a positive classroom climate that values and amplifies varied perspectives, abilities, approaches, and solutions.

**5c. Promote student self-efficacy**

Promote student self-efficacy by facilitating student creativity, choice in product and process, and self-directed learning.

**5d. Support student collaboration**

Provide structured opportunities for students to collaborate in CS. Develop students' ability to provide, receive, and respond to constructive feedback in the design, implementation, and review of computational artifacts.

**5e. Encourage student communication**

Create and scaffold meaningful opportunities for students to discuss, read, and write about CS concepts and how they integrate CS practices.

**5f. Guide students' use of feedback**

Use formative assessments to provide timely, specific, and actionable feedback to students and to adjust instruction. Develop students' ability to interpret and use feedback from computers, teachers, peers, and community.

*Abbildung 4.6 Standard „Classroom Practice" der „CSTA Standards for Computer Science Teachers" aus CSTA (2020, S. 2)*

Hazzan, Lapidot und Ragonis (2011, S. 4) formulieren in ihrem *Guide to Teaching Computer Science* zwar nicht explizit fachdidaktische Kompetenzen, sie beschreiben jedoch Ziele für einen einführenden Kurs über die *Methods of Teaching Computer Science*, den sie als erste Etappe in der

beruflichen Entwicklung angehender Informatiklehrkräfte ansehen. Das Hauptziel besteht darin, die Studierenden auf ihre zukünftige Lehrtätigkeit vorzubereiten. Daraus leiten sie die folgenden Teilziele ab:

1. *„To enhance students' professional identity as computer science teachers*
2. *To heighten students' awareness to the uniqueness of computer science education*
3. *To expose students to difficulties encountered by learners when learning different topics from the computer science curriculum*
4. *To enable students to master pedagogical skills for teaching computer science, considering different kinds of learners.*
5. *To enable students to master pedagogical tools for teaching computer science, including the creation of a supportive and cooperative inquiry-based learning environment*
6. *To expose the students to a variety of computer science teaching methods*
7. *To expose students to the research conducted on computer science education and to its application in the teaching process"*

(Hazzan, Lapidot und Ragonis 2011, S. 4)

Saeli (2012) beschäftigt sich in ihrer Dissertation mit dem fachdidaktischen Wissen bzgl. des Lehrens der Programmierung an weiterführenden Schulen. Auf der Basis von Literatur und Gruppeninterviews mit Lehrkräften entwickelt sie ein programmierspezifisches *PCK*-Modell, das die folgenden Themen berücksichtigt: *control structures (with focus on loops)*, *decomposition of the problem*, *problem solving skills*, *parameters*, *algorithms*, *data structures* und *arrays* (ebd., S. 42 ff.). Zu jedem Thema werden unterschiedliche Angaben gemacht, z. B. zu den Gründen für das Unterrichten des Themas, zu erforderlichen Vorkenntnissen und Schwierigkeiten der Schülerinnen und Schüler sowie zu geeigneten Unterrichtsmethoden. Bergner u. a. (vgl. 2018, S. 202 ff.) formulieren fachdidaktische Kompetenzen für Fach- und Lehrkräfte bzgl. der Gestaltung von Lernsituationen der informatischen Bildung im Elementar- und Primarbereich (siehe Abbildung 4.7). Dabei orientieren sie sich einerseits an den bereits zitierten Ausführungen von Shulman sowie deren Umsetzungen in der Mathematikdidaktik, andererseits an Konzepten und Modellen der informatikdidaktischen Forschung (ebd., S. 192 ff.).

*Nicht-kognitive Merkmale*

Im KUI-Kompetenzmodell umfassen die *Non-Cognitive Competences* folgende Facetten: (1) *Beliefs & Attitudes*, (2) *Motivational Orientations & Selfregulation* und (3) *Social & Communication Skills* (siehe Abbildung 4.3). Das Projekt konzentrierte sich im Bereich der nicht-kognitiven Merkmale jedoch vor allem auf die ersten beiden Facetten. Die professionellen Überzeugungen und motivationalen Orientierungen von Lehrkräften spielen eine wichtige Rolle für die Kompetenzausübung, insbesondere bzgl. der Bereitschaft zum Handeln (Weinert 2001a, S. 49 f.). Lehrerüberzeugungen

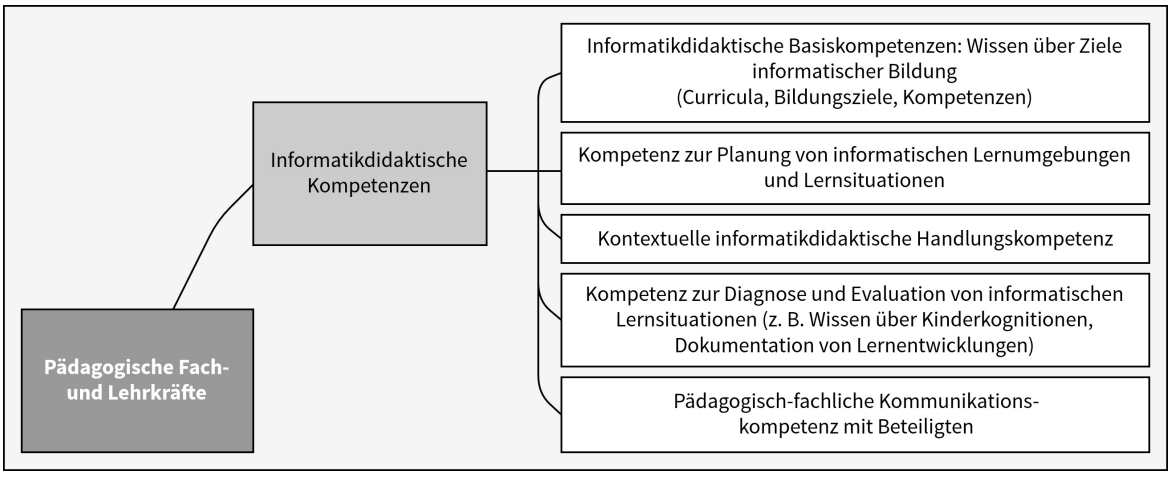



*Abbildung 4.7 Informatikdidaktische Kompetenzen auf Ebene der pädagogischen Fach- und Lehrkräfte im Elementar- und Primarbereich aus Bergner u. a. (2018, S. 168)*

„beinhalten Vorstellungen und Annahmen von Lehrkräften über schul- und unterrichtsbezogene Phänomene und Prozesse mit einer bewertenden Komponente“ (Kunter und Pohlmann 2009, S. 267). Im Kontext von Informatikunterricht beziehen sich die fachbezogenen Überzeugungen darauf, „wie in Informatik am besten gelehrt und gelernt wird“ (Bender, Schaper und Seifert 2018, S. 73). Motivationale Orientierungen umfassen „die individuellen Motive, Ziel- und Wertvorstellungen in Bezug auf die eigenen Fähigkeiten als Lehrkraft und tragen dazu bei, wie Lehrkräfte den alltäglichen Herausforderungen begegnen“ (ebd.). Zu den fachbezogenen motivationalen Charakteristika zählen Bender, Schaper und Seifert (ebd., S.79) den fachbezogenen Enthusiasmus und die fachbezogene Selbstwirksamkeitserwartung. Bender u. a. (2015a, S. 9 ff.) formulieren auf der Basis von Experteninterviews sowohl *teachers'beliefs* als auch *motivational orientations* für das Fach Informatik. Bezogen auf informatische Bildung im Elementar- und Primarbereich formulieren Bergner u. a. (vgl. 2018, S. 168 ff.) verschiedene Komponenten der Zieldimensionen „Motivation, Interesse und Selbstwirksamkeit (bezogen auf informatische Bildung)“ und „Einstellungen, Haltungen und Rollenverständnis (informatikbezogen)“.

## 4.3 Befunde und Programme zur Lehrerbildung

Um informatische Inhalte längerfristig in der Primarstufe zu verankern und adäquat zu unterrichten, muss zwangsläufig die entsprechende Lehrerbildung in den Blick genommen werden. Diese muss den Lehrkräften ein tieferes Verständnis der fachlichen Inhalte und Praktiken sowie der didaktischen Herangehensweisen ermöglichen:

> *„While the courses, curriculum content, lessons, and activities are the critical framework, it is the teacher practice, pedagogy, and classroom norms that bring purpose and engagement to student learning. For this reason, CS teacher professional development (PD) is critically important [...].“* (Goode, Margolis und Chapman 2014, S. 493)

*Notwendigkeit einer adäquaten Ausbildung*

Meerbaum-Salant, Armoni und Ben-Ari (2010, S. 75) fanden heraus, dass die blockbasierte Programmiersprache Scratch sich zwar zum Unterrichten von grundlegenden Programmierkonzepten eignet, sie weisen jedoch auf die Notwendigkeit hin, Lehrkräfte in der didaktischen Umsetzung anzuleiten. Yadav u. a. (2014) beschäftigten sich in einer Studie mit den Vorstellungen von Lehrkräften bezüglich *Computational Thinking*. Sie verglichen Lehrkräfte, die in diesem Gebiet spezifisch ausgebildet wurden, mit denen einer Kontrollgruppe. Dabei fanden sie heraus, dass die Lehrkräfte der Kontrollgruppe nicht nur unzulängliche, sondern auch falsche Vorstellungen von *Computational Thinking* und Zugängen zu dessen Vermittlung aufwiesen (ebd., S. 5 f.). Weitere Befunde zeigen, dass Lehrkräfte sich oft unsicher in Bezug auf ihr eigenes Fachwissen in Informatik fühlen und es für wichtig halten, entsprechend fortgebildet zu werden:

> *„Teachers want to feel well-prepared for the implementation of the learned content and/or pedagogic approaches. They often experience insecurities surrounding their own CS content knowledge and/or ability to give up ‚control' in the classroom to allow for more student-centered learning."* (Reding und Dorn 2017, S. 161)

> *„The analysis of teachers' qualitative responses indicates that teachers were concerned about the depth and breadth of their own Computing subject knowledge, and in particular that of computer science and programming. [...] Teachers express the worry ‚...that my own subject knowledge is not always secure'. They also report that they have spent hours of their own time trying to upskill in the subject [...]."*
>
> (Sentance und Csizmadia 2017, S. 479)

*Selbstwirksamkeitsempfinden als Ziel*

Erwachsene, vor allem Lehrkräfte, werden von Kindern als Verhaltensmodelle betrachtet und haben somit zwangsläufig Einfluss auf deren Selbstkonzept (vgl. Tausch und Tausch 1998, S. 19 ff.). Makris u. a. (2013) kommen in ihrer Studie zu dem Schluss, dass sich eine positive und selbstsichere Haltung der Lehrkräfte zur Programmierung auch auf die Schülerinnen und Schüler überträgt. Im Umkehrschluss gibt eine unsichere oder frustrierte Lehrperson diese negativen Haltungen ebenso an ihre Klasse weiter. Es ist demzufolge wichtig, dass sich Lehrkräfte sicher mit den Inhalten des Unterrichts fühlen.

Greaves (2017) befragte 42 englische Grundschullehrkräfte, wie sie auf die Einführung des neuen Fachs *Computing* vorbereitet wurden und mit welchen Hindernissen sie sich beim Unterrichten konfrontiert sahen. Nur fünf der Lehrkräfte gaben an, dass die Fortbildungsmaßnahmen fachdidaktische Themen und Praktiken thematisierten – vierzig hingegen lernten in den Fortbildungen, wie man Programmierwerkzeuge, wie z. B. Scratch oder Kodu, verwendet. Während fast die Hälfte der Lehrkräfte angab, sich bei der Anwendung der Programme sicher zu fühlen, gaben drei Viertel an, sich unsicher bzw. überhaupt nicht sicher dabei zu fühlen, die Programme im Unterricht einzusetzen:

> *„Despite positive attitudes held by teachers towards Computing, they still feel overwhelmingly ill-prepared and lacking in confidence when it comes to teaching the subject. It appears that insufficient training, poor subject knowledge, lack of pedagogical skills and a lack of time and support in schools all contribute to low confidence levels among teachers in teaching Computing.“* (Greaves 2017, S. 1429)

Bereits elf Jahre zuvor kommen Condie und Munro (2006, S. 63) in ihrer Metastudie bzgl. dem Einsatz von Informations- und Kommunikationstechnologien *(ICT)* in Schulen zu dem Ergebnis, dass es nicht ausreicht, neue Technologien lediglich bereitzustellen. Sie weisen zudem darauf hin, dass die fehlende Vermittlung von fachdidaktischen Inhalten sich negativ auf das Selbstwirksamkeitsempfinden der Lehrkräfte auswirkt. Sowohl Lehramtsstudierende als auch erfahrene Lehrkräfte äußerten in verschiedenen Studien die Sorge, dass sich ihr mangelndes Selbstwirksamkeitsempfinden im Umgang mit *ICT* negativ auf die Beziehung zur ihren Schülerinnen und Schülern auswirke, da sie weniger als Expertinnen bzw. Experten wahrgenommen würden (ebd., S. 18). Diese Befürchtung zeigt sich auch 2017 in einer Studie der *Royal Society*, die sich mit Stand des Fach *Computing* in Großbritannien befasst:

> *„There is an additional challenge in that teachers may perceive that their pupils are more confident with technology than they are. Through our discussion groups, some teachers felt that many of today's pupils have grown up with technology embedded in their lives which creates the image of a tech-savvy generation.“*
>
> (The Royal Society 2017, S. 55)

Die Studie macht in diesem Zusammenhang darauf aufmerksam, dass die Nutzung einer Technologie nicht zwangsläufig mit dem Verstehen der dahinter stehenden Technik und Informatik einhergeht (vgl. Schulmeister 2012, S. 44).

*Klassenleiterprinzip als Chance*

Während Informatik an Gymnasien in der Regel von Lehrkräften mit einer akademischen Ausbildung in Informatik unterrichtet wird, hat man es in der Primarstufe mit sogenannten Generalisten bzw. Generalistinnen zu tun, die bis auf wenige Ausnahmen alle Fächer unterrichten. In Bezug auf die Informatik ist das problematisch, da die meisten Grundschullehrkräfte und -lehramtsstudierende über kein oder sehr wenig Vorwissen bezüglich informatischer Inhalte verfügen und nur bedingt Unterrichtserfahrungen in der eigenen Schulzeit sammeln konnten (vgl. Döbeli Honegger und Hielscher 2017, S. 104). Die Vorstellungen von Grundschullehrkräften zur Informatik und zum Informatikunterricht scheinen zudem erheblich zu variieren (vgl. Best 2019, S. 63 f.).

Dass in der Grundschule viele Fächer von der gleichen Lehrkraft unterrichtet werden, scheint in Hinblick auf die informatische Bildung jedoch auch Vorteile mit sich zu bringen. Duncan, Bell und Atlas (2017) führten eine Fortbildungsmaßnahme mit 22 Grundschullehrkräften in Neuseeland durch, um sie auf die Umsetzung des neuen Lernbereichs *Digital Technologies* vorzubereiten. Für jede Unterrichtsstunde, in der sie informatische Inhalte behandelten, füllten die

Lehrkräfte einen Feedbackbogen aus, in dem sie diese kurz skizzierten und kommentierten. Die Analyse der Rückmeldungen zeigte, dass die Lehrkräfte in der Lage waren, andere Themen des Grundschullehrplans durch die Informatikmaterialien abzudecken, und dass die neuen Themen dadurch weniger Zeit in Anspruch nahmen als zuvor angenommen. Es wurde jedoch angemerkt, dass es nicht möglich war, die informatischen Inhalte vollständig in andere Fächer zu integrieren. Auch Sabitzer, Antonitsch und Pasterk (2014) weisen darauf hin, dass der Lehrplan der Grundschule – in diesem Fall Österreichs – in mehreren Fächern Themen enthält, die mit *Computational Thinking* zusammenhängen. Da die Lehrkräfte sich dessen nicht bewusst sind, solle man ihnen die Zusammenhänge zu informatischen Inhalten durch entsprechende didaktische Konzepte aufzeigen:

> *„Introducing informatics in primary schools is possible and reasonable when it is taught in a child-appropriate way. Although not anchored in the Austrian curriculum, many activities related to different informatics concepts are performed in the daily school life. Fostering these activities and relating them to informatics may be a way to prepare the children for computational thinking in its broadest sense. From the viewpoint of informatics didactics this calls for the development of a corresponding didactic concept to extend traditional primary education so as to include these new aspects [...].“*
>
> (Sabitzer, Antonitsch und Pasterk 2014, S. 111)

Lockwood und Mooney (2017, S. 16 f.) tragen verschiedene Studien zusammen, in denen *Computational Thinking* erfolgreich in andere Fächer, zum Beispiel Biologie, Mathematik oder Englisch, integriert wurde. Sie beziehen sich dabei jedoch nicht ausschließlich auf die Grundschule, sondern auf alle Schularten.

*Informatikbezogene Fortbildungskonzepte für Grundschullehrkräfte*

Webb u. a. (2017) untersuchen in ihrer Studie fünf Länder, die informatische Inhalte in ihre Curricula aufgenommen haben oder gerade neue Lehrpläne entwickeln: Großbritannien, Neuseeland, Australien, Israel und Polen. In allen Ländern stellt die Lehrerausbildung eine Herausforderung dar. Zum einen gibt es nicht viele Lehrkräfte mit entsprechenden fachlichen und fachdidaktischen Kenntnissen, zum anderen gibt es an den Universitäten im Vergleich zu anderen Fachrichtungen nur wenig Lehramtsstudierende des Fachs Informatik (ebd., S.463). Die Weiterbildung von Lehrkräften, die bereits im Lehrerberuf tätig sind, ist daher unerlässlich. International gibt es dazu unterschiedliche Ansätze, die sowohl zentral organisierte Fortbildungsangebote als auch Angebote von Universitäten und einschlägigen Initiativen umfassen.

Aus Überblicksartikeln lässt sich zwar entnehmen, dass es international unterschiedliche Fortbildungsangebote für Grundschullehrkräfte gibt (z. B. Balanskat und Engelhardt 2015, S. 59 ff.), diese werden jedoch nur selten genauer beschrieben. Reding und Dorn (2017) von der *University of Nebraska* stellen ein Fortbildungskonzept vor, in dem sich Lehrkräfte über mehrere Wochen hinweg mit informatischen Inhalten auseinandersetzen. In einer ersten Woche erhalten die Lehrkräfte in Präsenz theoretischen Input bzgl. der Informatik im Allgemeinen, bestehenden Curricula sowie

fachdidaktischer Konzepte. Sie werden zudem in die blockbasierten Programmiersprachen *Scratch* und *StarLogo Nova* eingeführt, mit denen sie in einer anschließenden Online-Übungsphase an vertiefenden Aufgaben arbeiten. Im Anschluss an die asynchrone Übungsphase treffen sich die Lehrkräfte erneut in Präsenz und erarbeiten über eine Woche hinweg konkrete Unterrichtsideen, die sie in einer darauffolgenden *Student Academy Week* mit Schülerinnen und Schülern erproben. El-Hamamsy u. a. (2020) entwickelten ein Fortbildungsprogramm für Grundschullehrkräfte der französischsprachigen Schweiz, welches vier ganztägige Fortbildungen in Präsenz umfasst, die sich über ein ganzes Schuljahr erstrecken. Die Lehrkräfte beschäftigen sich zunächst anhand verschiedener Übungen ohne Computer mit den Grundlagen der Algorithmik und lernen am zweiten Tag anhand verschiedener programmierbarer Roboter die Funktionsweisen und Komponenten von Maschinen, z. B. Sensoren und Aktoren, kennen. Am dritten Fortbildungstag stehen Algorithmik und Netzwerke sowie eine Einführung in die Programmierung in der App *ScratchJr* im Mittelpunkt. Am vierten und letzten Tag beschäftigen sich die Lehrkräfte mit fortgeschrittenen Konzepten der Algorithmik sowie Informationen und Datenstrukturen. Darüber hinaus wird thematisiert, wie Schülerinnen und Schüler im Rahmen des Informatikunterrichts kreativ arbeiten können. Kong und Lao (2019, S. 977) entwickelten ein Fortbildungsprogramm für Grundschullehrkräfte aus Hong Kong, welches zwei aufeinander aufbauende Kurse à 39 Stunden umfasst. In den dreizehn Sitzungen des ersten Kurses liegt der Schwerpunkt auf praktischen Übungen zu Konzepten und Praktiken des *Computational Thinking*, die im entsprechenden *Technology Education*-Curriculum verankert sind. Zudem sollen die Lehrkräfte die Möglichkeit erhalten, reale Probleme mittels Programmierung zu lösen, was durch das Entwickeln einer App am Ende des Kurses realisiert wird. In den 13 Sitzungen des zweiten Kurses werden fachdidaktische Themen behandelt, z. B. Binnendifferenzierung bei Programmieraktivitäten oder die Evaluation von Schülerprogrammen. In verschiedenen Gruppen entwickeln die Lehrkräfte außerdem Unterrichtsideen für einzelne Inhalte des Curriculums. Byrne, Fisher und Tangney (2015) beschreiben ihre eintägige Präsenzfortbildung zum Einsatz des Einplatinencomputer *Raspberry Pi*, die sich an irische Grundschullehrkräfte mit informatischer Vorbildung richtet. Sie folgen dabei dem *Bridge 21 Activity Model*[4], welches verschiedene Phasen vorgibt, in denen die Teilnehmenden sich dem Thema investigativ nähern, Projektideen zur Umsetzung im Unterricht entwickeln und im Anschluss präsentieren.

In Deutschland bildet die Stiftung *Haus der kleinen Forscher* seit 2017 Grundschullehrkräfte und pädagogische Fachkräfte im Bereich der informatischen Bildung fort (Günther und Nenner 2021, S. 1651 f.). In der eintägigen Fortbildung *Informatik entdecken – mit und ohne Computer* wird zum einen die Bedeutung von Informatik im Alltag thematisiert und den Teilnehmenden zum anderen Ideen für die informatische Bildung mit Kindern im Alter von drei bis zehn Jahren präsentiert, z. B. das Sortieren von Zahlen, Verschlüsseln von Nachrichten und algorithmische Vorgehensweisen zum Gewinnen eines Spiels. Das Fortbildungsangebot ist dabei so angelegt, dass nicht zwingend digitale Endgeräte für die Umsetzung in der Praxis vorhanden sein müssen – die Teilnehmenden

[4] https://tft.scss.tcd.ie/index.php/the-bridge21-model/

erhalten jedoch die Möglichkeit, verschiedene Programmierumgebungen und Robotiksysteme zu erproben. Innerhalb des Projekts PRIMA![5] wurde 2021 die Blended-Learning-Fortbildung *Informatische Bildung in der Grundschule* pilotiert, die sich mit der Umsetzung informatischer Bildung innerhalb verschiedener Fächer befasst. Thematisiert werden bspw. Themen wie Datenspeicherung, Verschlüsselung, Programmieren und Robotik. Die etwa zehnwöchige Fortbildung kombiniert Online-, Präsenz- und Praxiserprobungsmodule und möchte den Lehrkräften auf diesem Wege ermöglichen, sich den Themen niedrigschwellig und aktiv zu nähern. Laut Günther und Nenner (2021, S. 1652) konnten im Rahmen der Evaluation der Pilotierung mithilfe eines Pre-Post-Designs ein signifikanter Zuwachs in der Selbstwirksamkeit, dem Fachwissen und dem fachdidaktischen Wissen der Lehrkräfte festgestellt werden.

In Großbritannien hat sich die Initiative *Computing at School (CAS)*[6] etabliert, die Lehrkräfte befähigen möchte, den britischen Lehrplan mit Selbstvertrauen und Begeisterung zu unterrichten, indem sie regionale *Communities of Practice* aufbaut (vgl. Sentance und Humphreys 2015, S. 71 ff.). Der Begriff wurde von Wenger (1998) als praxisbezogene Gemeinschaft von Menschen beschrieben, die ähnlichen Aufgaben gegenüberstehen und voneinander lernen möchten. *CAS* richtete regionale Zentren ein, in denen sich Lehrkräfte treffen können, um Informatikmaterialien auszutauschen sowie Unterrichtsideen auszuprobieren und zu diskutieren. Zudem bildet die Initiative erfahrene Lehrkräfte zu sogenannten *Master Teachers* aus, die wiederum andere Lehrkräfte fortbilden (Smith u. a. 2015). Darüber hinaus betreibt *CAS* eine Online-Community, auf der sie Materialien zur Verfügung stellen, über Veranstaltungen informieren und Lehrkräfte sich austauschen können (Brown und Kölling 2013, S. 28). Ähnlich angelegt ist die Initiative *CoolThink*[7] aus Hong Kong, die Unterrichtsmaterialien zur Verfügung stellt und Lehrkräfte der Grundschule fortbildet. Die Initiative begann als Pilotprojekt, welches die Umsetzung des neuen Curriculums in 32 Grundschulen begleitete und evaluierte. Aus dem Projekt entwickelte sich eine Lernplattform, die mittlerweile von über 20.000 Schülerinnen und Schülern genutzt wird und auch als Austauschmöglichkeit zwischen Lehrkräften dient.

In Australien gibt es viele Grundschullehrkräfte in sehr abgelegenen Gebieten, für die es mit großem Aufwand verbunden ist, an Fortbildungsmaßnahmen teilzunehmen. Um dem zu entgegnen und gleichzeitig ein Angebot für viele Lehrkräfte zur Verfügung zu stellen, entwickelten Vivian, Falkner und Szabo (2014) einen *Massive Open Online Course (MOOC)*, der fachwissenschaftliche und fachdidaktische Inhalte der Informatik thematisiert. Zudem wurden soziale Medien eingebunden, um den Teilnehmenden einen direkten Austausch zu ermöglichen. In Deutschland entwickelte die Universität Bamberg in Zusammenarbeit mit der Ludwig-Maximilians-Universität München einen *MOOC* zum Verstehen der digitalen Welt[8], in dem sie in sieben Kapiteln *Computational Thinking* sowie grundlegende Konzepte der Informatik thematisieren.

[5] https://www.haus-der-kleinen-forscher.de/de/ueberuns/projekte/prima
[6] https://www.computingatschool.org.uk/
[7] https://www.coolthink.hk/en/
[8] https://t1p.de/ejv5

Auch in Finnland werden *MOOCs* eingesetzt, um Lehrkräfte in der Umsetzung informatischer Inhalte im Unterricht zu unterstützen (vgl. Kwon und Schroderus 2017, S. 7). Die Initiative *Koodiaapinen*[9] (auf Deutsch *Code Alphabet*) wurde von Lehrkräften und Forschenden der Informatikdidaktik gegründet, da es zwar viele Materialien für das Programmieren in der Grundschule gibt, diese jedoch größtenteils nur in Englischer Sprache zur Verfügung stehen und die Vielfalt besonders unerfahrene Lehrkräfte überfordern kann. Ihr kostenfreier *MOOC* zum Programmieren ist speziell für die Bedürfnisse finnischer Grundschullehrkräfte zugeschnitten und wurde allein in den ersten beiden Durchläufen von über 3000 Lehrkräften besucht. Zusätzlich werden über die Homepage der Initiative Materialien angeboten, welche die Lehrkräfte mit entsprechendem Urheberrechtshinweis im Unterricht einsetzen können. Um die Umsetzung des neuen Grundschullehrplans zu beschleunigen, führte die finnische Regierung außerdem ein Tutorensystem ein, bei dem jeder Schule eine im Programmieren erfahrene Lehrkraft als Tutor bzw. Tutorin zugewiesen wurde, um das Kollegium bei den Neuerungen zu unterstützen und zu beraten (ebd., S. 7).

*Informatik in der ersten Phase der Grundschullehrerbildung*

Spätestens, wenn informatische Inhalte im Curriculum der Grundschule gefordert werden, muss sich auch die erste Phase der Lehrkräfteausbildung – das Hochschulstudium – entsprechend anpassen. In der Schweiz, wo das Fach *Medien und Informatik* auch für die Primarstufe verpflichtend eingeführt wurde, sieht man sich dabei mit verschiedenen Herausforderungen konfrontiert. Zum einen fehlt bei den Dozierenden selbst das informatische Fachwissen und Unterrichtserfahrung bezüglich der Informatik, zum anderen stehen die Studierenden den Themen Medien und Informatik zurückhaltend oder gar ablehnend gegenüber (Döbeli Honegger und Hielscher 2017, S. 100). Darüber hinaus scheint der Großteil der Studierenden kein Verständnis der Informatik als eigenständige Disziplin zu besitzen – so nennt bspw. die Mehrheit der Studienanfängerinnen und -anfänger an der PH Schwyz als Antwort auf die Frage, was sie sich unter Informatik in der Primarstufe vorstellen, den Umgang mit *Office*-Programmen, Lernprogrammen und -spielen, das Bedienen des Computers, den Umgang mit dem Internet sowie Tastaturschreiben (Döbeli Honegger und Hielscher 2017, S. 103).

An der PH Schwyz wurde die Informatik-Lehrveranstaltung *Grundlagen der Informatik* eingeführt, die alle angehenden Grundschullehrkräfte im ersten Semester verpflichtend belegen. Döbeli Honegger und Hielscher (2017), die die Veranstaltung konzipierten, durchführten und evaluierten, schildern in Anbetracht der Begrenzung zeitlicher Ressourcen das Dilemma, den Fokus der Veranstaltung entweder auf die Motivation oder die informatischen Kompetenzen der Lehrkräfte zu setzen. Um das eher negative Bild der Informatik der Studierenden und deren Fehlvorstellungen zu korrigieren, empfehlen sie, derartige Veranstaltungen inhaltlich nicht zu überfrachten. Weiter sollte das Selbstvertrauen der Studierenden gestärkt und eine Vorstellung vermittelt werden, wie Informatik im Primarbereich aussehen kann:

[9] https://koodiaapinen.fi/en/

> „[Wir empfehlen] in der Aus- und Weiterbildung insbesondere motivationale Aspekte gegenüber der Vermittlung umfangreichen Fachwissens zu betonen. Aufgrund der fehlenden eigenen Erfahrungen braucht es konkrete Beispiele, um sowohl eine Vorstellung zu entwickeln wie Informatik in der Volksschule aussehen kann als auch das Selbstvertrauen der Lehrpersonen zu stärken, diese Inhalte selbst umsetzen zu können. Es gilt, das eher negative Bild der Informatik mit den vorherrschenden Fehlvorstellungen zu korrigieren. Bei der Veranstaltungsplanung muss das Vorwissen der Studierenden bzw. Lehrpersonen berücksichtigt werden. Eine inhaltliche Überfrachtung und damit verbundene Überforderung könnte langfristig der flächendeckenden Einführung von Informatik in der Schule schaden." (Döbeli Honegger und Hielscher 2017, S. 106)

An der PH der Fachhochschule Nordwestschweiz entschied man sich für einen Kurs über zwei Semester, den alle angehenden Grundschullehrkräfte obligatorisch besuchen. Im Kurs *Scalable Game Design* erwerben sie zum einen Kenntnisse in der Programmierung und der Informatik im Allgemeinen, zum anderen wird thematisiert, wie sie im Informatikunterricht nachhaltig kreative Prozesse initiieren können (Lamprou und Repenning 2018, S. 71). Die wöchentlichen Sitzungen beinhalten einen Theorieteil, in denen grundlegende informatische Themen, wie z. B. Daten, Algorithmen, Programmieren oder das Internet, behandelt werden, und einen Praxisteil. In diesem arbeiten die Studierenden an eigenen Projekten mit dem Programmiertool *AgentCubes*[10]. Das Ziel des Kurses ist *„for pre-service teachers to become computational thinkers themselves and to be able to help their students become computational thinkers."* (ebd., S. 71). Da jedoch über die Hälfte der 447 Studierenden, die an der Evaluation des ersten Seminardurchlaufs teilnahmen, die Frage, was *Computational Thinking (CT)* für sie bedeute, falsch beantwortete, wird der Kurs für den weiteren Einsatz modifiziert werden:

> *„Our findings suggest being more explicit in using and connecting CT with the practice of programming. One shift from the Introduction to Programming to Computer Science Didactics course will be the integration of CS with other disciplines such as STEM, art, music and languages. Seeing CT applied to other fields may help develop a better sense of purpose of CT."* (Lamprou und Repenning 2018, S. 74)

An der Bergischen Universität Wuppertal wird seit dem Sommersemester 2018 das vierstündige Seminar *Informatik in der Grundschule* angeboten, welches Lehramtsstudierende der Grundschule im Rahmen der Bildungswissenschaften wählen können (vgl. Haselmeier 2019, S. 93 f.). Die Veranstaltung, die sich an den Kompetenzempfehlungen der Gesellschaft für Informatik orientiert, vereint theoretische sowie praktische Arbeitsphasen. Die Studierenden entwickeln auf dieser Grundlage eigene Unterrichtsentwürfe, die am Ende des Semesters erprobt und diskutiert werden. Nach zwei Durchläufen mit zunächst zehn und schließlich zwanzig Studierenden zeigte sich,

[10] https://de.agentsheets.com/

dass der Lernprozess eine sehr enge Betreuung erfordert und sehr zeitaufwändig ist. Bei den Studierenden beider Durchgänge wurden große Unsicherheiten oder sogar falsche Vorstellungen gegenüber der Informatik festgestellt, die jedoch im Laufe des Semesters revidiert werden konnten:

> „Ängste und Unsicherheit, bzw. falsche oder fehlende Vorstellungen sind durchgängig und bislang ohne Ausnahme bei allen Studierenden in beiden Durchläufen zu beobachten. Es kann nur zum Teil davon gesprochen werden, dass neue Gegenstände vermittelt werden. Vielmehr geht es häufig zunächst darum falsche Vorstellungen zu revidieren, bevor mit der Entwicklung informatischer Kompetenz überhaupt begonnen werden kann." (Haselmeier 2019, S. 96)

An der Technische Universität Dresden wurden im Wintersemester 2020/21 im Rahmen einer Pflichtveranstaltung für Grundschullehramtsstudierende des Fachs Werken drei 90-minütige Workshop-Einheiten erprobt, in denen grundlegende informatische Inhalte und Kompetenzen sowie Anknüpfungspunkte zum sächsischen Lehrplan der Grundschule behandelt werden (Nenner, Damnik und Bergner 2021). Die Studierenden sollen auf fachlicher Ebene nach der Teilnahme in der Lage sein, den Aufbau und die Funktionsweisen von Informatiksystemen zu beschreiben, Algorithmen zu erkennen und nachzuvollziehen, und grundlegende Programmierkonstrukte in einfachen Programmierumgebungen anzuwenden. Auf fachdidaktischer Ebene sollen sie zum Unterrichtsziel passende Themen, Materialien und Programmierumgebungen auswählen und Unterrichtsideen zu verschiedenen informatischen Inhalten und Kompetenzen skizzieren können (ebd., S. 105). Innerhalb der Workshops werden die verschiedenen Fachinhalte vorgestellt und anhand passender Umsetzungsbeispiele für den Unterricht angewandt. In zwei Selbstlernphasen zwischen den Veranstaltungen sichten die Studierenden Materialien zur Informatik in der Grundschule und identifizieren Anknüpfungspunkte der Themen zum sächsischen Lehrplan der Grundschule. In der Evaluation des Angebots konnte mittels Fragebögen, welche zu Beginn und am Ende ausgefüllt wurden, ein Zuwachs an grundlegenden Fachkompetenzen im Bereich der informatischen Bildung und des spezifischen Wissens zur Vermittlung ebendieser festgestellt werden (ebd., S. 109). Außerdem zeigten sich die angehenden Lehrkräfte zuversichtlich, die Inhalte und Kompetenzen in ihrem späteren Berufsleben Schülerinnen und Schülern vermitteln zu können.

Neben spezifischen Angeboten für Grundschullehrkräfte gibt es Bestrebungen, Informatik in der allgemeinen Lehrerbildung zu verankern und somit angehende Lehrkräften aller Fächer und Schulformen zu erreichen (vgl. Dengel und Heuer 2018; Losch und Humbert 2019; Seegerer 2021). Da sich dieses Vorhaben jedoch erheblich vom Thema dieser Arbeit unterscheidet, wird es nicht weiter behandelt.

*Auswirkungen von Fortbildungsmaßnahmen*

Verschiedene Studien befassen sich mit den Auswirkungen von Fortbildungsmaßnahmen zur Informatik auf die teilnehmenden Grundschullehrkräfte. Zum Beispiel befragten Duncan, Bell und

Atlas (2017) 27 fortgebildete Lehrkräfte, wie selbstsicher sie sich beim Unterrichten informatischer Inhalte fühlten. In der Auswertung von Fragebögen zeigte sich, dass sich der Großteil der Lehrkräfte als *moderately confident* einschätzte – den Forschern zufolge waren die Lehrkräfte zuvor größtenteils unsicher, was die Vermittlung von informatischen Inhalten anging. Rich u. a. (2017b) untersuchten 27 Grundschullehrkräfte, die über einen Zeitraum von einem Jahr an einer wöchentlichen Fortbildung zu Informatik und Technik in der Grundschule teilnahmen. In einer abschließenden Umfrage wurde die Selbstwirksamkeitserwartung der Lehrkräfte in den Bereichen Informatik und Technik sowie hinsichtlich des Unterrichtens der neu gelernten Inhalte gemessen. Um sicherzustellen, dass die gemessenen Variablen durch die Fortbildung und nicht durch andere Einflüsse verändert wurden, wurde zum einen eine Kontrollgruppe mit 25 Lehrkräften untersucht, die nicht an der Fortbildung teilgenommen hatten. Zum anderen wurde zusätzlich die Selbstwirksamkeitserwartung in Bezug auf das Unterricht von Mathematik und Naturwissenschaften erhoben. In der Analyse zeigten sich signifikante Unterschiede zwischen den beiden Gruppen in der Selbstwirksamkeitserwartung in den Bereichen Informatik und Technik sowie hinsichtlich des Unterrichtens der beiden Fachbereiche. In Bezug auf Mathematik und Naturwissenschaften konnten keine Unterschiede zwischen der Experimental- und Kontrollgruppe gefunden werden.

Weitz u. a. (2017) untersuchten in einer Studie mit angehenden Erzieherinnen und Erziehern, wie sich Vorerfahrungen mit Informatiksystemen in der Kindheit, die Bereitstellung von Unterrichtsmaterialien und eine zweitägige Fortbildung auf die Einstellungen und das Selbstwirksamkeitsempfinden hinsichtlich der Vermittlung informatischer Inhalte auswirken. Der einzige Effekt, der gefunden werden konnte, war der positive Einfluss von Erfahrungen mit Informatiksystemen in der Kindheit auf das Selbstwirksamkeitsempfinden. Straube u. a. (2018) untersuchten mithilfe eines Online-Fragebogen die Lehrenden-Selbstwirksamkeitserwartung und das Interesse bzgl. informatischer Themenbereiche von 22 Grundschullehrkräften und 61 Studierenden der Grundschulpädagogik. Die Analyse zeigte, dass die Lehrkräfte gegenüber den Studierenden eine höhere Lehrenden-Selbstwirksamkeitserwartung aufwiesen. Das Interesse an informatischen Themenbereichen war in beiden Gruppen gleich gering. Yadav, Lishinski und Sands (2021) untersuchten 176 angehende Informatiklehrkräfte hinsichtlich ihrer Selbstwirksamkeitserwartung in Bezug auf das Unterrichten von Informatik. In der Analyse der Erhebung, die nach einer zweiwöchigen Fortbildungsmaßnahme stattfand, konnten mittels einer Clusteranalyse drei verschiedene Selbstwirksamkeitsprofile identifiziert werden. In einem nächsten Schritt wurde untersucht, inwieweit sich die Profile in weiteren erhobenen Merkmalen, z. B. der Lehrerfahrung, unterscheiden. Die Ergebnisse zeigen, dass das Studienfach der Teilnehmenden mit ihrer Selbstwirksamkeitserwartung zusammenhängt. Überraschenderweise wiesen Lehrkräfte mit einem MINT-Studienfach die geringste Selbstwirksamkeitserwartung auf. Standl und Schlomske-Bodenstein (2021) untersuchen die Entwicklung der Selbstwirksamkeitserwartung bzgl. des Lösens von Aufgaben zum algorithmischen Denken und Programmieren. Zu diesem Zweck maßen sie die Selbstwirksamkeitserwartung von acht Programmieranfängerinnen und -anfängern an drei Messzeitpunkten – jeweils bevor

und nachdem sie die Aufgaben bearbeiteten. Obwohl die Studierenden in der Lage waren, die Aufgaben zum algorithmischen Denken und Programmieren zu lösen, hatte sich ihre wahrgenommene Selbstwirksamkeit im Laufe der Zeit nicht signifikant verändert. Dies deutet laut Standl und Schlomske-Bodenstein darauf hin, dass Selbstwirksamkeit ein relativ stabiles Konstrukt ist und sich nicht leicht verändern lässt. Allerdings ist an dieser Stelle jedoch anzumerken, dass nur eine kleine Stichprobe untersucht wurde.

# 5 Forschungsvorhaben und Fragestellungen

In den bisherigen Ausführungen dieser Arbeit wurde umfassend argumentiert, warum Schülerinnen und Schüler sich bereits in der Grundschule mit informatischen Inhalten auseinandersetzen sollten. Darüber hinaus wurde dargestellt, warum sich das Programmieren in besonderem Maße für eine mögliche Auseinandersetzung im Unterricht eignet. Dabei wurde auch thematisiert, dass die Programmierung sowohl Lernende als auch Lehrende vor Herausforderungen stellt und es entsprechender didaktischer Ansätze bedarf, um das Thema adäquat im Unterricht zu behandeln. Die Ausführungen zur Bedeutung der Lehrkräfte für das schulische Lernen zeigten zudem, dass diese unbedingt bei Überlegungen zur Umsetzung des Programmierens in der Grundschule berücksichtigt werden müssen.

Im Rahmen dieser Arbeit soll untersucht werden, wie das Programmieren in der Grundschule implementiert werden kann. Dieses Forschungsanliegen wird auf drei verschiedenen Ebenen betrachtet – *Unterricht*, *Lehrkräfte* und *Rahmenbedingungen* – die jeweils durch eine übergeordnete und mehrere untergeordnete Forschungsfragen adressiert werden:

*RQ 1 Wie kann das Programmieren im Unterricht der Grundschule behandelt werden?*

(1-a) Welche Ziele sind in einem Unterrichtskonzept zum Programmieren in der Grundschule anzustreben?

(1-b) Welche Prinzipien zur Entwicklung eines Unterrichtskonzepts zum Programmieren in der Grundschule ergeben sich aus der Analyse begrifflicher, theoretischer und empirischer Grundlagen in den Bereichen der Entwicklungspsychologie, Grundschulpädagogik und -didaktik sowie Fachdidaktik und -wissenschaft der Informatik?

(1-c) Wie können diese Prinzipien in einem konkreten Unterrichtskonzept zum Programmieren in der Grundschule umgesetzt werden?

(1-d) Welche Auswirkungen hat das Unterrichtskonzept auf Schülerinnen und Schüler?

*RQ 2 Wie können Lehrkräfte der Grundschule ohne einschlägige Vorkenntnisse befähigt werden, das Programmieren im Unterricht zu behandeln?*

(2-a) Welche Prinzipien zur Entwicklung eines Fortbildungskonzepts ergeben sich aus dem Unterrichtskonzept sowie aus der Analyse theoretischer und empirischer Grundlagen im Bereich der Lehrerbildung und der Fachdidaktik der Informatik?

(2-b) Wie können diese Prinzipien in einem konkreten Fortbildungskonzept zum Programmieren in der Grundschule umgesetzt werden?

(2-c) Wie bewährt sich das entwickelte Fortbildungskonzept in der Praxis?

(2-d) Wie implementieren Lehrkräfte der Grundschule das Unterrichtskonzept in der Praxis, wandeln es ab oder erweitern es?

(2-e) Welche Auswirkungen hat das Fortbildungskonzept auf Lehrkräfte?

*RQ 3 Welche Rahmenbedingungen wären für das Programmieren in der Grundschule wünschenswert?*

(3-a) Mit welchen Herausforderungen sind die Lehrkräfte während der Erprobung des Programmierens in der Grundschule konfrontiert?

(3-b) Welche Gelingensbedingungen für das Programmieren in der Grundschule ergeben sich aus der Erprobung in der Praxis?

Ein Ziel dieser Arbeit ist daher die adäquate Aufbereitung des Themenbereichs der Programmierung sowohl für den Unterricht als auch die Lehrkräftefortbildung im Kontext der Grundschule sowie die Entwicklung damit einhergehender praxistauglicher Materialien und Leitfäden. Darüber hinaus soll das entwickelte Unterrichts- und Fortbildungskonzept empirisch erprobt und evaluiert werden, um Rückschlüsse auf dessen Wirksamkeit sowie die notwendigen Rahmenbedingungen zu ermöglichen. Um dem Anspruch der Arbeit auf theoretischer wie auch praktischer Ebene gerecht zu werden, wurde der Forschungsansatz der *Design-Based Research* gewählt.

Teil II

# FORSCHUNGSMETHODISCHER BEZUGSRAHMEN UND STUDIENDESIGN

# 6 Gestaltungsorientierung in der Bildungsforschung

Die empirische Lehr-Lernforschung wird immer wieder in verschiedenen Punkten kritisiert, z. B. wird ihr Praxisferne durch die Reduktion komplexer Situationen auf wenige kontrollierbare Variablen vorgeworfen, mangelnde theoretische Fundierung sowie eine fehlende Öffnung für innovative Lösungen zu Verbesserung der Bildungspraxis (Tulodziecki, Grafe und Herzig 2013, S. 9). In der Auseinandersetzung mit diesen Vorwürfen entwickelten sich seit den 1990er Jahren verschiedene Ansätze, die darauf abzielen, das Potenzial der Verknüpfung von praxisbezogenen Innovationen und Forschung zu nutzen.

## 6.1 Ausgangssituation

Das Verhältnis von Theorie und Praxis ist ein wiederkehrendes und vieldiskutiertes Thema in den Bildungswissenschaften und wird dort als sogenanntes *Theorie-Praxis-Problem* diskutiert. Kahlert (2007, S. 20) beschreibt, dass man zwar auch den Erziehungswissenschaften und Fachdidaktiken zugestehe, Grundlagenforschung zu betreiben, gleichzeitig jedoch erwartet werde, „dass ihre Erträge sich nicht nur von denen nutzen lassen, die über Schule und Unterricht forschen, kommunizieren und urteilen, sondern auch von Lehrerinnen und Lehrern, die Schule und Unterricht gestalten". Zugleich macht er darauf aufmerksam, dass wissenschaftliche Erkenntnisse nicht automatisch zu Veränderungen im Schul- und Unterrichtsalltag führen (vgl. ebd., S. 23). Während Kahlert den genannten Disziplinen rät, „ihre Erkenntnisse so zu kommunizieren, dass Lehrerinnen und Lehrer sie nutzen können – und es für sinnvoll halten, sie zu nutzen" (2007, S. 28), vermuten Vanderlinde und van Braak (2010) einen anderen Grund für das schwierige Verhältnis von Theorie und Praxis:

> *„One argument suggests that this gap [between research and practice] reflects two sharply contrasting types of knowledge. On the one hand, we have research-based knowledge that is published in scientific journals. On the other hand, we have pedagogical knowledge which is used by classroom teachers in their day-to-day teaching (McIntyre 2005). Bates (2002) argues that tension exists between researchers and practitioners, as the practitioner asks for new solutions to operational problems while the researcher seeks new knowledge."* (Vanderlinde und van Braak 2010, S. 301)

Grundsätzlich können innerhalb der Bildungswissenschaften in dieser Hinsicht verschiedene methodische Ansätze identifiziert werden, aus denen sich wiederum unterschiedliche Implikationen

*Tabelle 6.1 Zugänge zur Bildungsforschung aus Klauer und Leutner (2007, S. 14). Die Disziplin der gestaltungsorientierten Bildungsforschung wurde beim präskriptiven Zugang von der Autorin ergänzt.*

| | | Art des Zugangs | | |
|---|---|---|---|---|
| | | deskriptiv | präskriptiv | normativ |
| Art der Frage | Was-Frage | Inhaltsanalyse der Instruktion | Curriculum | Übergeordnete Ziele |
| | Wie-Frage | Interaktions- und Wirkungsanalyse der Instruktion | Gestaltung des Lehr-Lern-Prozesses | berufsethische Standards des Lehrens |
| Disziplinen | | Empirische Unterrichtsforschung | Instruktionsforschung, gestaltungsorientierte Bildungsforschung | Erziehungs-philosophie |

für die Gestaltung der Unterrichtspraxis ergeben (vgl. Klauer und Leutner 2007, S. 10 ff.). Der normative Ansatz bezieht sich auf Bildungstheorien sowie deren Interpretationen und betont die Auseinandersetzung mit Begriffen, Normen und Werten. Auf Grundlage der analytischen Durchdringung bildungstheoretischer Aussagen und Modellen werden Überlegungen für die Gestaltung der Bildungspraxis abgeleitet. Der deskriptive Ansatz hingegen orientiert sich am Forschungsparadigma der Naturwissenschaften, in dem Hypothesen geprüft werden. Durch eine möglichst objektive Erhebung von Daten mittels geeigneter Instrumente und das Ausschalten möglichst vieler Fehlerquellen werden die Bedingungen erforscht, die das Lehren und Lernen beeinflussen. Auf diesem Weg können grundlegende Ergebnisse erzielt werden, die zur Gestaltung von Lehr-Lernprozessen herangezogen werden können. Im präskriptiven Ansatz steht die Erzeugung von Handlungswissen im Vordergrund, bspw. durch die Entwicklung von Modellen, Gestaltungsaussagen und Empfehlungen für die Unterrichtspraxis. Da eine verantwortbare präskriptive Forschung und Entwicklung sowohl deskriptive Forschung als auch normative Festlegungen voraussetzen, stellt der präskriptive Ansatz gewissermaßen eine Verbindung der beiden Ansätze dar. Alle beschriebenen Ansätze beschäftigen sich mit den Kernfragen der Lehr-Lernforschung – die Frage nach den Lehrinhalten (Was-Frage) und angemessen Lehrverfahren (Wie-Frage) (siehe Tabelle 6.1).

Im präskriptiven Ansatz verorten Klauer und Leutner die Instruktionsforschung, die untersucht „wie Instruktionsziele möglichst optimal erreicht werden können“ (2007, S. 5). Den Begriff verwenden die Autoren synonym mit dem der Instruktionspsychologie, welche sie in Bezug auf Leutner (2001) als Disziplin definieren, die sich auf Basis von Lerntheorien mit Instruktionsmodellen und der Instruktionspraxis befasst. Zu den Aufgabengebieten gehören sowohl Fragen des Curriculums und der Lehrzieldefinition, als auch die Planung des Lehr-Lernprozesses:

> „Die zentralen Aufgabengebiete der Instruktionspsychologie sind [...] das Curriculum einerseits und die konkrete Gestaltung des Lehr-Lern-Prozesses andererseits. Die Curriculumkonstruktion [...] erfordert zunächst ein Verfahren, wie aus übergeordneten Zielvorgaben im einzelnen hergeleitet werden kann, welche Teilziele und Inhalte ver-

> mittelt werden sollen, um das übergeordnete Ziel zu erreichen. Vielfach erfordert es empirische Lehrzielforschung, um festzustellen, welchen Anforderungen die Lernenden später gerecht werden müssen. Darüber hinaus müssen die Ziele so analysiert werden, dass daraus Hinweise für die Umsetzung in Lehr-Lern-Prozesse [...] erfolgen. Bei dieser Umsetzung sind zahlreiche Randbedingungen zu berücksichtigen, insbesondere solche der Lernenden und des einzusetzenden Lehrmediums."
>
> (Klauer und Leutner 2007, S. 17 f.)

Andere Autoren sind der Meinung, dass es einer spezifisch gestaltungsorientierten Forschung bedarf, die sich den Gestaltungsfragen des pädagogischen Felds widmet (vgl. Preußler, Kerres und Schiefner-Rohs 2014, S. 256 f.). In diesem Kontext wird auf die Lücke zwischen Ergebnissen traditioneller Forschungsansätze und dem Nutzen für die Bildungspraxis hingewiesen, die bereits zu Beginn des Kapitels als *Theorie-Praxis-Problem* der Lehr-Lernforschung beschrieben wurde:

> „Das heißt keineswegs, dass die [traditionellen Ansätze der bildungswissenschaftlichen Forschung] überflüssig oder gar wertlos sind, sie erweisen sich aber offenbar als nicht ausreichend, um in Schule, Hochschule und Weiterbildung nachhaltige Veränderungen beim Lernen und Lehren anzustoßen, und sie erweisen sich als nahezu unfähig, Menschen in der Praxis Konzepte und Instrumente an die Hand zu geben, mit denen konkrete Lehr-Lernprobleme in spezifischen Situationen gelöst werden können."
>
> (Reinmann 2005, S. 58)

Neben der geringen Anwendbarkeit von Forschungsergebnissen werden zusätzlich eine unzureichende Relevanz vieler Forschungsfragen für die Bildungspraxis sowie fehlende Zugriffsmöglichkeiten für Praktikerinnen und Praktiker kritisiert, welche die Kluft zwischen Forschung und Praxis weiter verstärkt (vgl. Euler 2014b, S. 16).

## 6.2 Entwicklung neuer Forschungsansätze

Um die Lücke zwischen Forschung und Praxis zu schließen, entwickelten sich seit den 1990er Jahren vermehrt alternative gestaltungsorientierte Forschungsansätze, die sich auf die positive Veränderung der Unterrichts- und Schulpraxis konzentrierten:

> „Als Ansätze, die sowohl dem Streben nach Praxisrelevanz als auch nach wissenschaftlicher Fundierung und nach Verbesserung von Bildungsprozessen verpflichtet sind, lassen sich – unter anderem – die Aktionsforschung (als eine Ausprägung von Handlungs- und Praxisforschung nach Altrichter/ Posch 2007), der Design-based Research-Ansatz (nach DBRC 2003), die integrative Forschungsstrategie (nach Stark 2004), die didaktische Entwicklungsforschung (nach Einsiedler 2010) sowie die entwicklungsorientierte Bildungsforschung (nach Reinmann/ Sesink 2011) nennen."
>
> (Tulodziecki, Grafe und Herzig 2013, S. 205 f.)

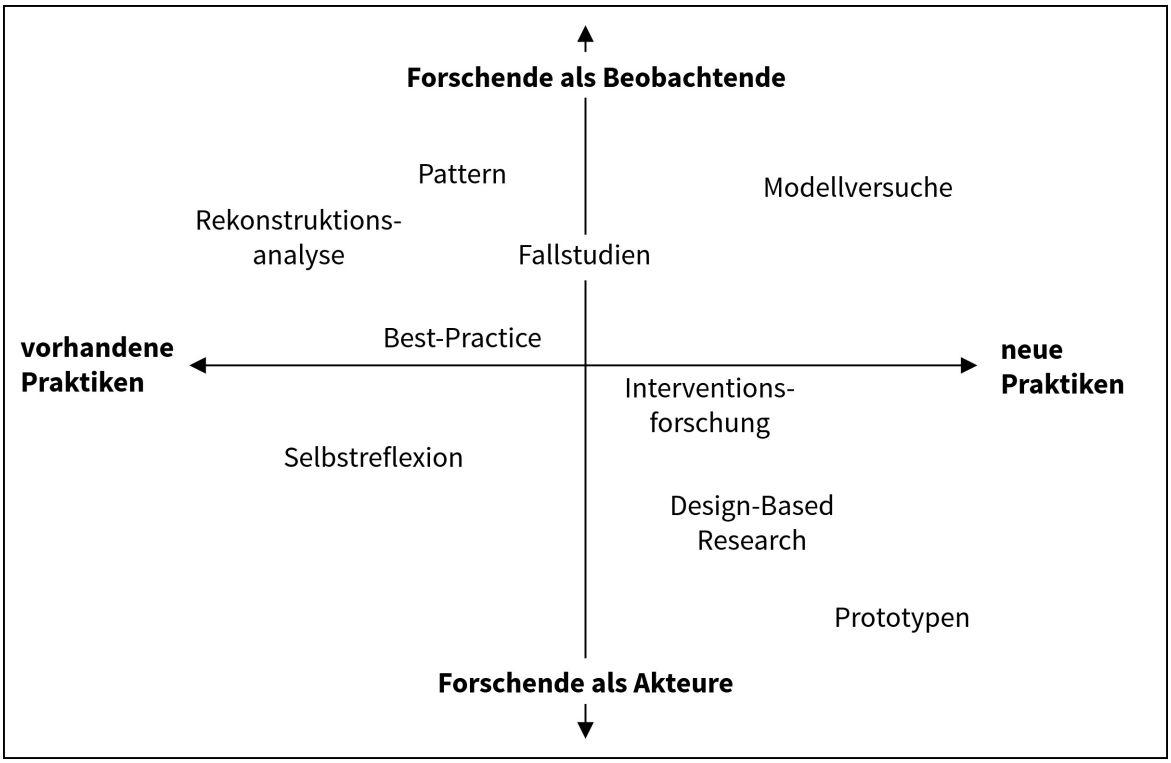



*Abbildung 6.1 Systematisierung einer gestaltungsorientierter Bildungsforschung aus Preußler, Kerres und Schiefner-Rohs (2014, S. 262). Es wird darauf hingewiesen, dass die einzelnen Forschungsansätze im konkreten Einzelfall auch anders verortet werden können.*

Mit Ausnahme der *Design-Based Research*, die als Forschungsansatz für die vorliegende Arbeit dient und im Folgekapitel ausführlich beschrieben wird, wird nicht näher auf die genannten Ansätze eingegangen. Schemme (2017, S. 16 f.) betont jedoch, dass der gestaltungsorientierten Forschung im Allgemeinen ein erweiterter Gestaltungsbegriff zugrunde liegt:

> „Dieser [Gestaltungsbegriff] umfasst Gegenständliches wie Immaterielles, also nicht nur die Formgestaltung von materiellen Dingen des täglichen Gebrauchs, sondern auch soziale Beziehungen, Prozesse der Kommunikation und Informationen sowie Strukturen." (Schemme 2017, S. 16)

Preußler, Kerres und Schiefner-Rohs (2014, S. 261 ff.) zeigen verschiedene Ausrichtungen der gestaltungsorientierten Forschung auf. Sie differenzieren einerseits in Bezug auf die Rolle, die Forschende im Forschungsprozess einnehmen, und andererseits in Bezug darauf, ob die Forschung an bestehenden Praktiken ansetzt oder neue Praktiken generiert (siehe Abbildung 6.1). Bei der Auswertung vorhandener Praxis steht im Fokus, erfolgreiche Vorgehensweisen in der Praxis zu identifizieren – das entsprechende Pendant bildet die Entwicklung und Erprobung neuer Handlungspraktiken. McKenney und Reeves (2019, S. 23 f.) schlagen ebenfalls eine Systematisierung für gestaltungsorientierte Forschung vor und unterscheiden zwischen drei Orientierungen hinsichtlich der Erkenntnisse, die im Rahmen der Forschung in Bezug auf die Intervention abgeleitet werden – *research for, on, and through interventions* (siehe Abbildung 6.2). Die Orientierungen unterscheiden sich hauptsächlich darin, ob der Implementierungsprozess berücksichtigt wird, und welche Rolle

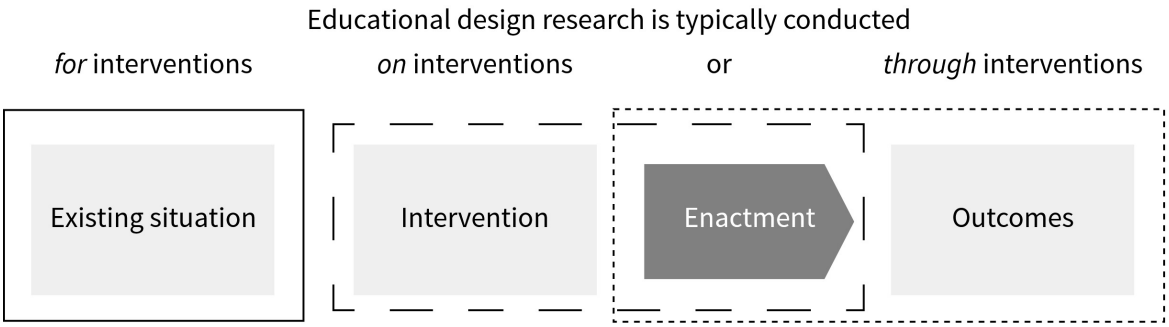



*Abbildung 6.2 Grundorientierungen in der gestaltungsorientierten Forschung bzgl. der angestrebten Erkenntnisse aus McKenney und Reeves (2019, S. 23).*

die entwickelte Intervention bei der empirischen Untersuchung spielt. *Research for interventions* konzentriert sich typischerweise auf das Ausgangsproblem, den jeweiligen Kontext und/oder die Zielgruppen und trägt so einerseits zum theoretischen Verständnis bei, andererseits kann sie wichtige Erkenntnisse für die Gestaltung einer Intervention ermöglichen. Daneben gibt es eine Orientierung, die primär darauf abzielt, Erkenntnisse über Eigenschaften und Funktionen einer oder mehrerer Interventionen zu gewinnen – *research on interventions*. Dabei steht die Gestaltung der Intervention im Vordergrund und wie diese zum Erreichen bestimmter Ziele geeignet ist. Informationen darüber, wie die Intervention funktioniert, mit wem, unter welchen Bedingungen oder mit welchen Ergebnissen, können genutzt werden, um diese in einem nächsten Schritt weiterzuentwickeln. *Research through interventions* konzentriert sich als dritte Orientierung auf die Auswirkungen, die eine Intervention hervorruft. Die Intervention kann in diesem Fall eher als ein Mittel gesehen werden, durch das tiefere Einblicke in bestimmte Phänomene des Lehrens und Lernens in einem authentischen Umfeld gewonnen werden können. Im Vergleich mit der Systematik von Preußler, Kerres und Schiefner-Rohs fällt auf, dass beide Modelle unterscheiden, ob vorhandene Praktiken untersucht oder neue Praktiken entwickelt werden. Während Preußler, Kerres und Schiefner-Rohs ganze Forschungsansätze in ihrem Vier-Felder-Schema verorten, weisen McKenney und Reeves (2019, S. 25) darauf hin, dass umfassende gestaltungsorientierte Forschungsprojekte jede der beschriebenen Orientierungen zu einem unterschiedlichen Zeitpunkt widerspiegeln. Je nachdem, welche Forschungsfragen beantwortet werden sollen und welche Informationen für die nachfolgende Entwicklung erforderlich sind, werden Maßnahmen vollzogen, die in ihrem Modell unterschiedlich eingeordnet werden.

## 6.3 Argumentationsformen der gestaltungsorientierten Forschung

Selbst Befürworter der gestaltungsorientierten Bildungsforschung sehen eine Schwachstelle in ihrer fehlenden *argumentative grammar* – der Logik, die eine Methode leitet und Forschungsfragen, Daten, Analysen sowie Folgerungen verbindet (z. B. Kelly 2004, S. 118 f.). Es wird z. B. bemängelt, dass zu viele Studien in narrative Beschreibungen abgleiten, anstatt genau darzulegen, woran sie ihre Folgerungen festmachen (Shavelson u. a. 2003, S. 27). Stattdessen fordern unterschiedliche Autoren, dass es einer Argumentationsform bedarf, die zu begründeten Aussagen führt:

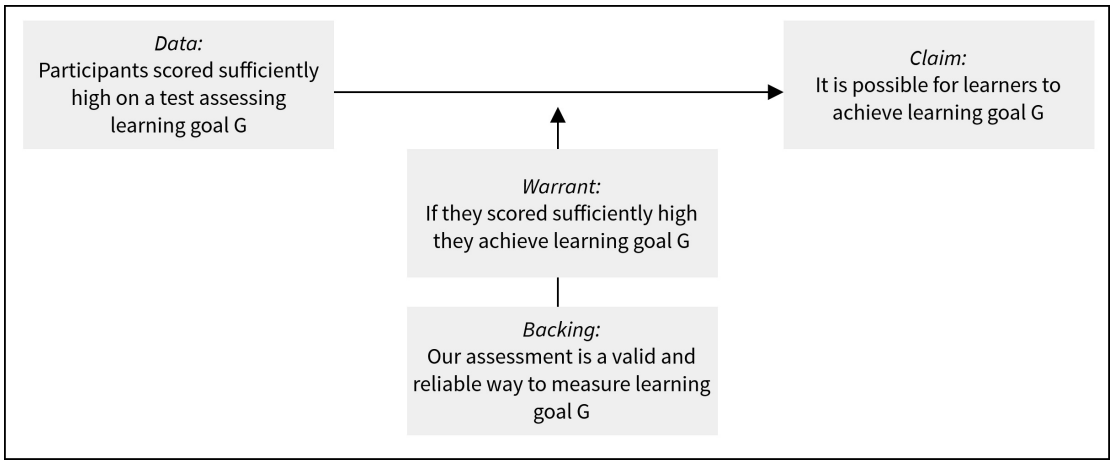



*Abbildung 6.3 Beispiel für einen Existenzbeweis in der Lehr-Lernforschung aus Bakker (2018, S. 101)*

> *„[A major current limitation is] the lack of an explicit, agreed-upon argumentative grammar. [...] this is a severe weakness that must be addressed if design research is to become a mature methodology with explicitly codified standards that can be used to judge the quality of proposals for and reports of particular classroom and professional development design studies.“* (Cobb, Jackson und Dunlap 2015, S. 496)

Bakker (2018, S. 100 ff.) schlägt unterschiedliche Argumentationsformen für die gestaltungsorientierte Forschung vor:

*Proof of Principle that something is possible*

In der Lehr-Lernforschung zeigt ein Existenzbeweis oder *Proof of Principle/Concept*, dass etwas möglich ist. Um den Rezipienten ein Bild zu vermitteln, *was* möglich ist, wird in der Regel mindestens ein detailliertes Beispiel aufgezeigt, wie ein Unterrichtsprinzip o. ä. entwickelt und umgesetzt wurde. Ein typisches Beispiel dieser Argumentation aus der gestaltungsorientierten Forschung ist, dass es durch eine Intervention für Schülerinnen und Schülern eines bestimmten Alters möglich war, bestimmte Lernziele zu erreichen (siehe Abbildung 6.3).

*Small Changes per Iteration*

Eine Argumentationsform, die sich an quasi-experimentellen Ansätzen orientiert, wäre eine Struktur, in der in mehreren Erprobungen jeweils nur wenige Variablen einer Intervention verändert werden (siehe Abbildung 6.4). Dreh- und Angelpunkt dieser Argumentation ist die Frage, ob die Schülerinnen und Schüler, mit denen die Intervention jeweils erprobt wurde, wirklich vergleichbare Gruppen darstellen.

*Experience of the Design Community*

Ein wesentliches Ergebnis gestaltungsorientierter Forschung ist das Generieren von Gestaltungsprinzipien oder konkreten Unterrichtskonzepten für die Schulpraxis. Wenn diese von anderen Lehrenden innerhalb einer Community angewandt werden und sich auch in anderen Kontexten bewähren, zeigt dies ihre breite Anwendbarkeit (siehe Abbildung 6.5).

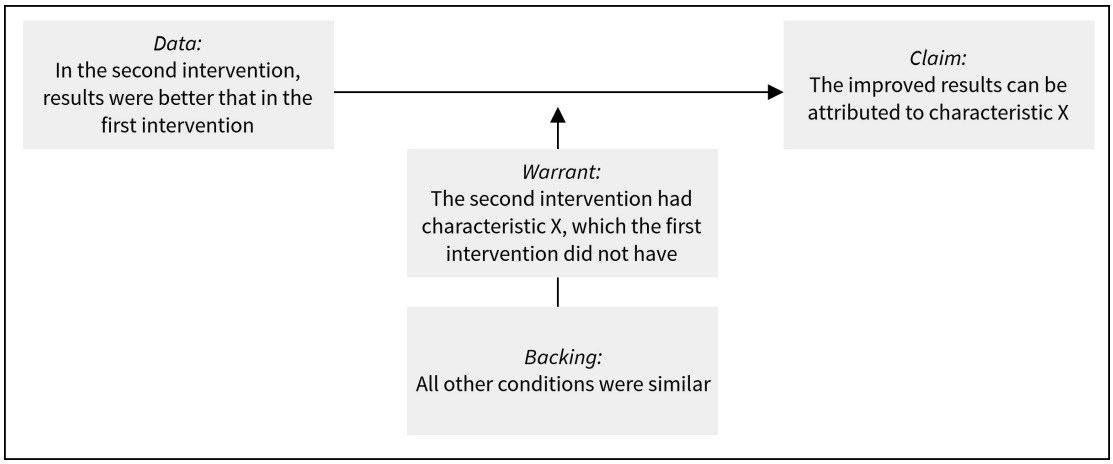



*Abbildung 6.4 Beispiel für die Argumentationsform eines Quasi-Experiments aus Bakker (2018, S. 102)*

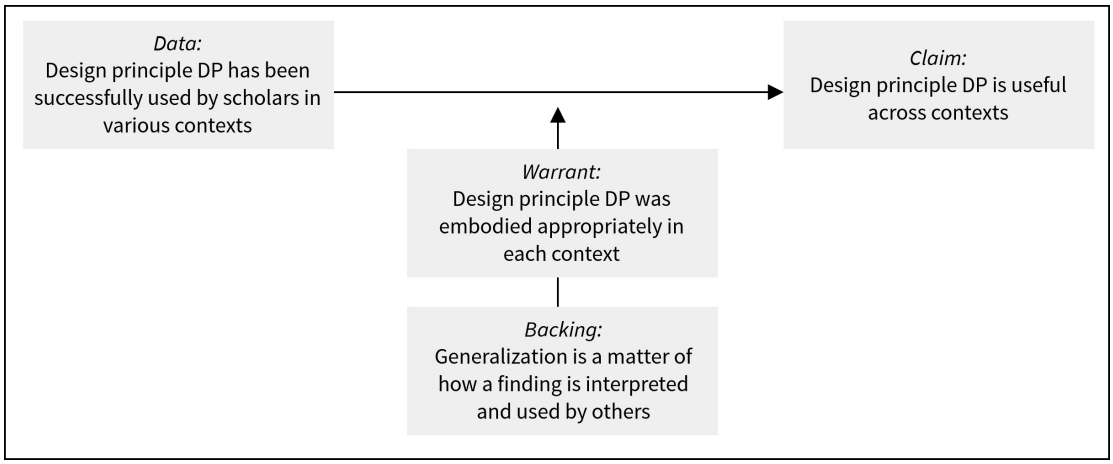



*Abbildung 6.5 Gestaltungsprinzipien werden von anderen Lehrenden innerhalb der Community angewandt aus Bakker (2018, S. 103)*

*Answering the How-Question*

Bei gestaltungsorientierten Forschungsprojekten geht es oft nicht nur darum, einen Existenzbeweis anzuführen, dass etwas möglich ist, sondern auch, wie etwas möglich ist, z. B. mit welchen Konzepten oder Methoden. Zur Beantwortung sollten verschiedene Fragen mitbedacht werden:

- *„What is students' prior knowledge, and what is an appropriate learning goal for this particular group of students?*
- *What is a teaching-learning strategy (design) that would help students to achieve this goal (or help teachers to support students in achieving that goal)?*
- *How (well) was this strategy (design) implemented?*
- *What were the effects/results of implementing this strategy (design)?*
- *What would an improved design look like?“* *(Bakker 2018, S. 104 f.)*

Zu beachten ist, dass Bakker (2018, S. 109) explizit darauf hinweist, dass seine vorgeschlagenen Argumentationsformen nicht als strikte Handlungsanweisungen zu verstehen sind. Vielmehr sollen sie Forschenden in der Planung gestaltungsorientierter Forschungsanliegen als Orientierungspunkte dienen.

# 7 Design-Based Research

Der Ansatz der *Design-Based Research (DBR)* entwickelte sich zu Beginn des 21. Jahrhunderts als eine Methodologie der gestaltungsorientierten Bildungsforschung, der eine Brücke zwischen Design und Forschung schlägt (Anderson und Shattuck 2012, S. 16). Der Begriff des Design wird von Edelson (2002, S. 108) allgemein als eine Abfolge von Entscheidungen beschrieben, die getroffen werden, um Ziele und vorherrschende Rahmenbedingungen in Einklang zu bringen. Reinmann (2014, S. 67 f.) verdeutlicht, dass Design im DBR-Ansatz ein Teil des Forschungsprozesses darstellt, jedoch nicht Gegenstand der Forschung ist, wie es bspw. im *Instructional Design* der Fall ist.

Während der DBR-Ansatz zunächst vorrangig in den USA eingesetzt wurde (vgl. Anderson und Shattuck 2012, S. 20), findet er mittlerweile auch in der deutschsprachigen Lehr-Lernforschung immer häufiger Beachtung. So setzten allein in den letzten Jahren mehrere Dissertationsprojekte aus verschiedenen Fachdidaktiken den Ansatz um (z. B. Hiller 2017; Hörmann 2018; Rohrbach-Lochner 2019; Sajons 2020; Weskamp 2019). Auch in der Informatikdidaktik wurde der Ansatz bereits angewandt, z. B. in den Dissertationsprojekten von Best (2020), Przybylla (2018) und Schulz (2018) sowie in weniger umfangreichen Forschungsarbeiten (z. B. Knobelsdorf, Otto und Sprenger 2017; Kastl und Romeike 2015).

## 7.1 Entstehung und Zielsetzung des Forschungsansatzes

DBR versteht sich als Forschungsansatz, der es im besonderen Maße ermöglichen kann, nachhaltige Innovationen für den Unterrichtsalltag hervorzubringen (vgl. Reinmann 2005, S. 66). Durch die theoriegeleitete Entwicklung von Konzepten für die Bildungspraxis und der empirischen Untersuchung ebendieser, werden in der DBR einerseits wirksame Interventionen entwickelt, andererseits verspricht man sich weitere praxisrelevante Erkenntnisse. Im Fokus steht dabei die Frage, wie, wann und warum sich ein neu entwickelter Ansatz in der Praxis bewährt hat (Cobb u. a. 2003; DBRC 2003). DBR zielt demnach nicht darauf ab, eine bestehende Intervention auf ihre Wirksamkeit hin zu überprüfen – vielmehr beschäftigt sie sich damit, wie ein bestimmtes Ziel durch eine Intervention erreicht werden könnte, die es erst zu entwickeln gilt (Edelson 2002, S. 106).

Der DBR-Ansatz geht zurück auf die Arbeiten von Brown (1992) und Collins (1992). Collins entwickelte einen Forschungsansatz, der Forschung und Schulpraxis kombinierte, und so die Implementierung sowie Evaluation von technologischen Innovationen im Klassenzimmer ermöglichte. Sein Ansatz der *Design Science of Education* sollte es darüber hinaus ermöglichen, systematisch Erkenntnisse zu generieren, die wiederum für zukünftige Innovationen genutzt werden können (Collins 1992, S. 7). Den Kern seines Ansatzes bilden sogenannte *Design Experiments*, die auch bei

Brown im Mittelpunkt standen. Brown, die sich zunächst auf Lehr-Lernforschung mittels Laborstudien beschränkte, kam zu der Erkenntnis, dass sich jene alltagsfernen Untersuchungssettings nur bedingt zur Erforschung des Lehrens und Lernens eigneten (Brown 1992, S. 165). Sie entwickelte das Konzept der *Design Experiments*, mit denen sowohl praxistaugliche Intervention also auch Theorien entwickelt werden sollten, die deren Wirkungsweisen erklärten (Brown 1992, S. 143). Um der Komplexität des Unterrichtsgeschehens gerecht zu werden, folgte sie in ihren Untersuchungen, die sie vorrangig in realen Unterrichtssettings durchführte, vorrangig einem *Mixed-Methods*-Ansatz (Brown 1992, S. 156).

Ausgehend vom *Design Experiment* bildeten sich weitere Ansätze und Begriffe heraus, die teilweise synonym zu verstehen sind bzw. einer ähnlichen Zielsetzung folgen. International entwickelten sich parallel in verschiedenen Ländern Forschungsansätze, die zwar unterschiedlich benannt wurden, jedoch gleiche Vorgehensweisen beschreiben und deren Bezeichnungen heute weitestgehend synonym verwendet werden, z. B. *Design Research* (Bakker 2018), *Development Research* (van den Akker 1999), *Educational Design Research* (Plomp 2007) oder *Design-Based Research* (Anderson und Shattuck 2012). In der deutschsprachigen Forschungslandschaft entwickelten sich unter anderem die didaktische Entwicklungsforschung (Einsiedler 2011) sowie die entwicklungsorientierte Bildungsforschung (Reinmann und Sesink 2011).

Einsiedler beschreibt die didaktische Entwicklungsforschung als „einen angewandten Bereich der Forschung, der durch theoretisch-erklärende Aussagen fundiert und gleichzeitig auf zielerreichendes Gestalten gerichtet ist“ (2011, S. 47). Als deren vorrangige Aufgaben sieht er die Erarbeitung konkreter Maßnahmen zur Unterrichtsverbesserung, z. B. die Entwicklung von Lehrplan- und Unterrichtssequenzierungen sowie Unterrichtsmodellen oder die Lehrwerksforschung (ebd., S. 58 f.). Zwar formuliert er verschiedene Standards, die wiederum Rückschlüsse auf die Durchführung didaktischer Entwicklungsforschung erlauben, z. B. Theoriefundierung, Erprobung in Lernumwelten, Wirkungs- und Aktivierungsevaluation (ebd., S. 63 ff.), legt die Vorgehensweise jedoch nicht explizit fest, wie dies z. B. bei der *Design-Based Research* der Fall ist.

Im Vorwort ihres Readers zur entwicklungsorientierten Bildungsforschung[1] schreibt Reinmann 2015, dass sie den Begriff gewählt habe, da sie zum einen die Tradition der entwicklungsorientierten Forschung in der deutschen Pädagogik hervorheben wollte, zum anderen wolle sie einen Fokus auf den Akt der Entwicklung setzen. In der Folgeversion des Readers von 2018[2], verwendet sie weitgehend durchgängig den Begriff *Design-Based Research*, da sich dieser ihrer Beobachtung nach in der deutsch- und englischsprachigen Literatur durchgesetzt habe.

Allen Ansätzen gemein ist der Anspruch, theoretisch fundierte sowie innovative Problemlösungen für praxisrelevante Fragestellungen zu entwickeln, diese schrittweise zu erproben und gleichzeitig zur Weiterentwicklung von Theorien beizutragen (Tulodziecki, Grafe und Herzig 2013, S. 227 f.). Der Argumentation von Reinmann folgend, wird im Weiteren ausschließlich der Begriff *Design-Based Research* verwendet.

[1] https://gabi-reinmann.de/?p=4823
[2] https://gabi-reinmann.de/?p=6281

## 7.2 Zentrale Merkmale des DBR-Ansatzes

Neben der zugrundeliegenden Zielsetzung lassen sich weitere Merkmale identifizieren, die für den DBR-Ansatz typisch sind. Die folgenden Charakteristika wurden von DBR-Forschenden als typisch für den Ansatz identifiziert (vgl. Anderson und Shattuck 2012; Reinmann 2017; Tulodziecki, Grafe und Herzig 2013; van den Akker 1999; Wang und Hannafin 2005).

*Problemstellungen der Unterrichtspraxis*
Ausgangspunkt der DBR sind relevante und authentische Probleme aus der Bildungspraxis, für deren Lösung neuartige Gestaltungslösungen entwickelt und erprobt werden (Euler 2014b, S. 17). Eine möglichst genau Spezifikation des Ausgangsproblems und der angestrebten Zielsetzungen sind daher grundlegend für den Forschungsprozess (Euler 2014b, S. 23 f.).

*Iterative Entwicklung einer Intervention*
Die Entwicklung einer Intervention stellt einen wesentlichen Punkt des Forschungsprozesses dar. Neben einer ausführlichen Analyse des jeweiliges Kontextes sind die theoretischen Grundlagen der spezifischen Problemstellung handlungsleitend für den Gestaltungsprozess (McKenney und Reeves 2019, S. 13). In mehreren iterativen Zyklen bestehend aus Phasen der Gestaltung, Durchführung, Analyse und Re-Design wird die Intervention schrittweise verbessert (Reinmann 2005, S. 62).

*Kooperation zwischen Wissenschaft und Praxis*
In sämtlichen Phasen der DBR sollten Praktiker in den Forschungs- und Entwicklungsprozess miteinbezogen werden. Dadurch wird einerseits eine hohe Alltagstauglichkeit sichergestellt, andererseits kann sich die Zusammenarbeit positiv auf die nachhaltige Implementierung der Intervention auswirken (Euler 2014b, S. 18). Im Gegensatz zu Ansätzen der Aktionsforschung (z. B. Elliott 1991, S. 69), ist es in der DBR nicht vorgesehen, dass Lehrkräfte selbst Untersuchungen durchführen. Reinmann und Sesink (2011, S. 12) betonen, dass die Bewahrung einer theoretischen Distanz wichtig ist, um ungenutzte Potenziale und Möglichkeiten zu erkennen, die im weiteren Verlauf des Forschungsprozesses berücksichtigt werden sollen.

*Methodenmix*
Aufgrund der Komplexität einer realen Unterrichtssituation ist es nahezu unmöglich, sämtliche Variablen zu erheben, die diese beeinflussen. Im Rahmen der Evaluation sollten demnach die Untersuchungs- und Auswertungsmethoden ausgewählt werden, die je nach Entwicklungsstand der Intervention neue Erkenntnisse bzgl. deren Überarbeitung oder der Theorieentwicklung zulassen (Anderson und Shattuck 2012, S. 17). Reinmann (2005, S. 62) hält fest, dass nicht die Nutzung bestimmter Methoden kennzeichnend für den DBR-Ansatz ist, sondern deren interventionsorientierter, zyklischer Einsatz.

## 7.3 Vorgehensweise im Forschungsprozess

Der Forschungsprozess der DBR besteht aus verschiedenen Phasen, die in der Literatur bereits in verschiedenen Prozessmodellen beschrieben wurden. Euler (2014b, S. 19 ff.) schlägt bspw. ein Modell aus sechs zyklisch angeordneten Phasen inklusive jeweils angestrebter Teilziele vor, in dem zunächst das Ausgangsproblem präzisiert wird und danach auf Basis entsprechender Literatur und Erfahrungsberichte ein *Theoretical Frame of Reference* entwickelt wird (siehe Abbildung 7.1). Auf dessen Basis wird ein erstes didaktisches Design entwickelt, welches im Anschluss erprobt und formativ evaluiert wird. Aus den Ergebnissen und Erfahrungen werden dann *Design Principles (DP)* abgeleitet. Bei Bedarf wird das Design erneut angepasst, erprobt und evaluiert – bis die Evaluation einen zufriedenstellenden Grad an *stability and robustness* erreicht hat. Die entwickelte Intervention wird nun summativ evaluiert, um zu beurteilen, ob sie die gewünschten Effekte erzielt. Die Ergebnisse können wiederum Anlass zur Auseinandersetzung mit einer neuen Problemstellung geben. In anderen Prozessmodellen zur *Design-Based Research* ist die formative und summative Evaluation der Intervention nicht in verschiedenen Phasen vorgesehen. Es gibt zumeist eine Phase der Evaluation, die Maßnahmen beider Evaluationsformen einschließt und nach Bedarf mehrfach durchlaufen wird.

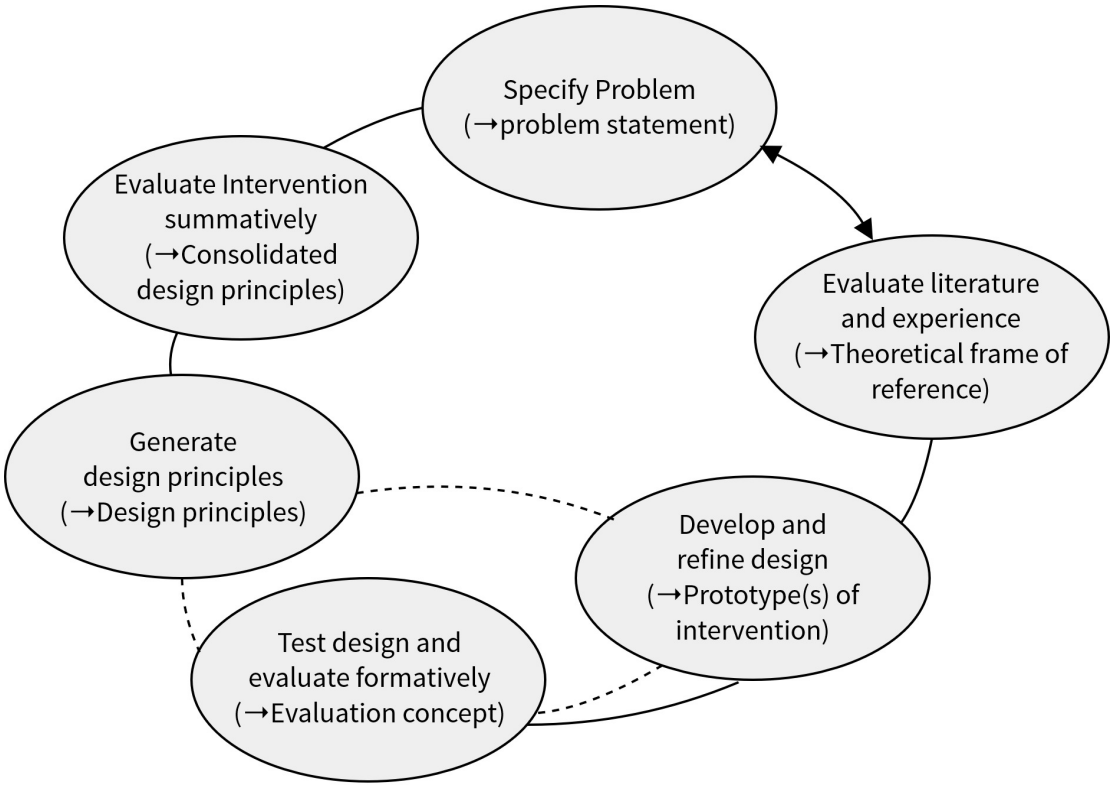



*Abbildung 7.1 Forschungs- und Entwicklungszyklen im Kontext der Design-Based Research aus Euler (2014b, S. 20)*

Reeves (2000, S. 9) schlägt bereits im Jahr 2000 ein Vier-Phasen-Modell zur *Design-Based Research* vor, in dem zunächst ein Problem sowohl von Forschenden als auch Praktikerinnen und Praktikern analysiert wird, dazu Lösungen anhand eines *Theoretical Framework* entwickelt werden, diese in

der Praxis erprobt und evaluiert werden und zuletzt *Design Principles* abgeleitet werden (siehe Abbildung 7.2). In Zusammenarbeit mit McKenney differenziert er das Modell in den Folgejahren weiter aus (vgl. McKenney und Reeves 2019, S. 83). Das resultierende Prozessmodell besteht aus drei Kernphasen: *Analysis/Exploration*, *Design/Construction* und *Evaluation/Reflection* (siehe Abbildung 7.3). Deutlich wird außerdem eine doppelte Zielsetzung des Prozesses – einerseits die immer ausgereifter werdende didaktische Intervention, andererseits theoretisches Verständnis bzgl. des Ausgangsproblems. Als zusätzliche Ebene kommt außerdem die Interaktion mit der Praxis hinzu, die im Laufe des Prozesses verstärkt praktiziert wird. Die Phasen des Modells sind zwar linear angeordnet, die jeweiligen Pfeile verdeutlichen jedoch, dass durchaus Wechselwirkungen zwischen den Prozessen möglich sind.

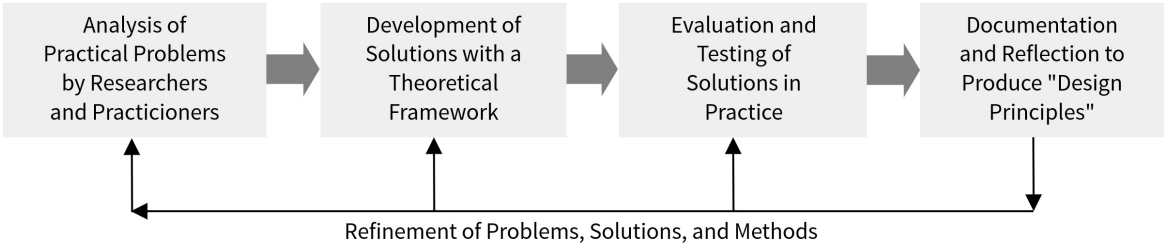



*Abbildung 7.2 Design-Based Research-Modell aus Reeves (2000, S. 9)*

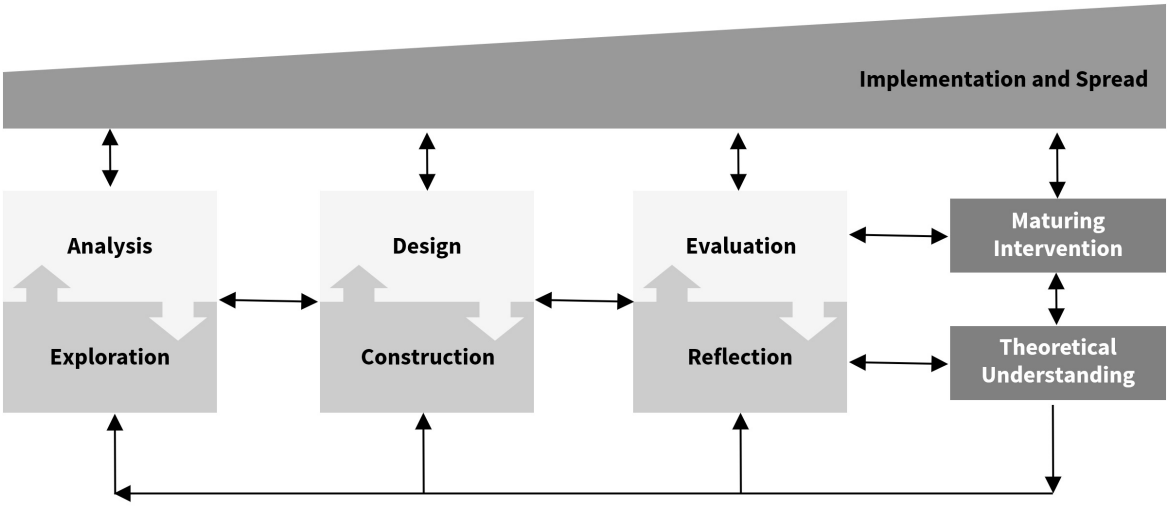



*Abbildung 7.3 Generisches Modell zur Durchführung von Design-Based Research aus McKenney und Reeves (2019, S. 83)*

Dem sehr detailreichen Modell von Wademan (2005) ist ebenfalls eine doppelte Zielsetzung in einem Beitrag zu Theorie und Praxis zu entnehmen (siehe Abbildung 7.4). Im Gegensatz zu den Modellen von Euler und Reeves ist im Rahmen der Design-Entwicklung jedoch nicht von einem *Theoretical Framework* die Rede, sondern von *Tentative Design Principles* - vorläufigen Gestaltungsaussagen. Da sich in der einschlägigen Literatur keine einheitliche Definition des *Theoretical Frameworks* finden lässt, wird im weiteren Verlauf der Arbeit die Formulierung von Wademan verwendet. Eine ähnliche Herangehensweise findet sich bei Jahn (2014):

„Am Ende der Analysephase wird ein erster theoretischer Rahmen anhand von Gestaltungsrichtlinien (Design Principles) für die Entwicklung eines Prototyps für die Intervention formuliert. [...] Durch die Sichtung des aktuellen Standes der Forschung bzw. von *Good Practice* können Vorstellungen entwickelt werden, wie das zu entwickelnde Design idealerweise entworfen und umgesetzt werden sollte [...]." (Jahn 2014, S. 9)

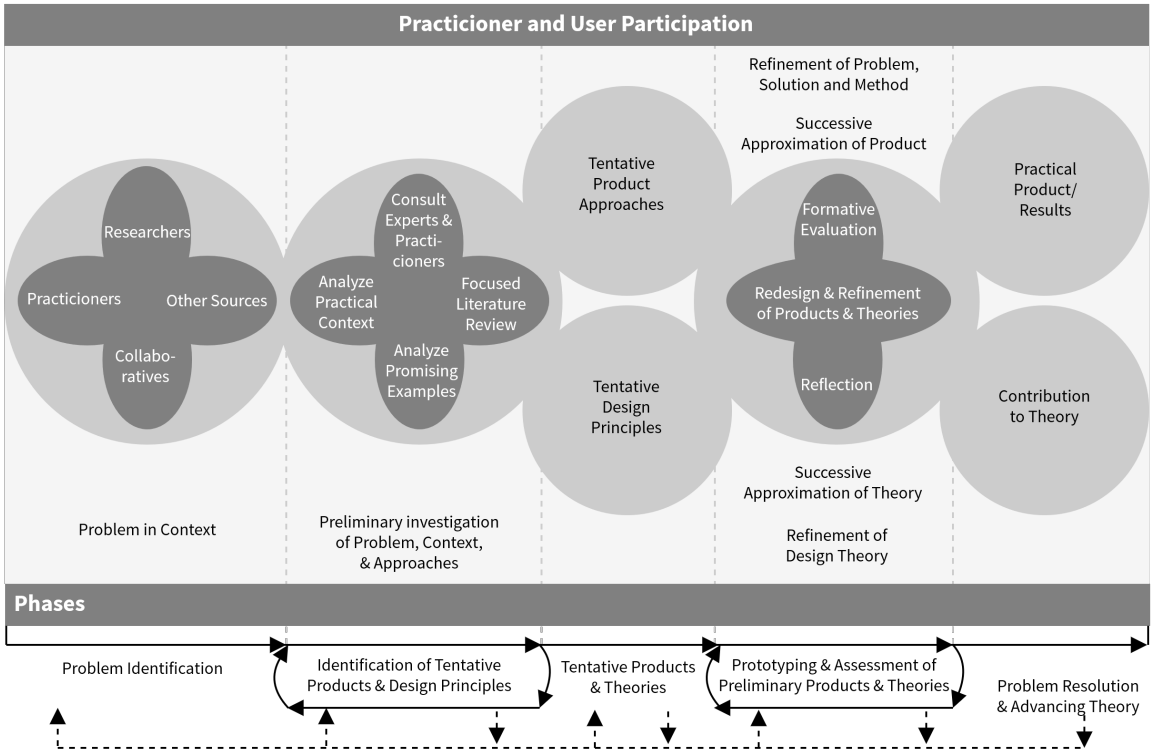



*Abbildung 7.4 Generisches Modell der Design-Based Research aus Wademan (2005, S. 228)*

Die in der Literatur beschriebenen Prozessmodelle unterscheiden sich z. T. zwar in den verwendeten Begrifflichkeiten sowie der Anzahl der Phasen, beinhalten jedoch alle sehr ähnliche Elemente. Idealtypisch werden im Rahmen der DBR die folgenden Schritte durchlaufen.

### 7.3.1 Analyse der Problemstellung und umfassende Recherche

Grundlegend für den Forschungsprozess der DBR ist die Spezifikation eines Problems, das im weiteren Verlauf adressiert werden soll, und erste damit einhergehende Forschungsfragen (Euler 2014b, S. 23 f.). Daran anschließend erfolgt die fachliche Klärung der Problemstellung sowie die Sichtung relevanter Theorien, verwandter Arbeiten und *Good Practice*-Beispielen. Neben einer einschlägigen Literaturrecherche können hier auch andere methodische Ansätze, wie z. B. Fokusgruppen, Experteninterviews oder die Dokumentenanalyse von Unterrichtsmaterialien, Curricula etc. herangezogen werden (McKenney und Reeves 2019, S. 106 f.).

### 7.3.2 Vorläufige Design-Principles und Entwicklung eines Prototyps

Ausgehend von den gewonnenen Erkenntnisse werden vorläufige Gestaltungsaussagen, sogenannte *Design Principles*, formuliert, an der sich die Entwicklung eines Prototypen für die Intervention orientiert. Kerres erläutert den Begriff der Gestaltungsaussagen wie folgt:

> „Gestaltungsaussagen in der Bildungsforschung beziehen sich etwa auf die Frage, wie bei der Planung eines Lernangebots vorzugehen ist oder welche didaktisch-methodischen Varianten sich für welche Lerninhalte oder Zielgruppen am besten eignen." (Kerres 2018, S. 78)

Diese Aussagen sind zu diesem Zeitpunkt zumeist noch sehr allgemein gehalten. Dennoch sollte es möglich sein, daraus konkrete Handlungsempfehlungen abzuleiten. Jahn (2014) illustriert anhand mehrerer Beispiele, wie didaktische Konzepte in gestalterische Handlungsempfehlungen übersetzt werden:

> „Gemeint sind gestalterische didaktische Konzepte wie etwa ‚Ideendiversität', ‚Erfahrungslernen', ‚direkte Instruktion' oder ‚ideale Sprechsituation'. Daraus lassen sich konkretere Handlungsempfehlungen ableiten, wie z. B. ‚gestalte Phasen des Unterrichts so, dass nach jeder Aktionsphase eine Phase der Reflexion anschließt' oder ‚sorge dafür, dass alle Meinungen gehört werden'." (Jahn 2014, S. 9)

Auf Basis der bisherigen Erkenntnisse und *Design Principles* werden nun potenzielle Lösungen für das Ausgangsproblem in Form eines Prototyps entworfen. McKenney und Reeves (2019, S. 135 ff.) beschreiben verschiedene Zwischenschritte und Herangehensweisen, die bei der Gestaltung dieses ersten Entwurfs einer Intervention durchlaufen bzw. genutzt werden können, unter anderem das Generieren unterschiedlicher Ideen mittels Brainstorming-Techniken, das Abwägen der entwickelten Ideen oder das Erstellen mehrerer Grobentwürfe (*Skeleton Designs*). Es wird empfohlen, den Prototypen sowohl mit erfahrenen Praktikerinnen und Praktikern als auch einschlägigen Expertinnen und Experten regelmäßig zu besprechen und auf diese Weise multiperspektivisches Feedback einzuholen (Jahn 2014, S. 10).

### 7.3.3 Erprobung und Evaluation in der Praxis

Um herauszufinden, ob sich die entwickelte Intervention in der praktischen Umsetzung bewährt, wird sie mehrfach in der Praxis erprobt und evaluiert. Je nach Forschungskontext können sich die Schwerpunkte der Evaluation unterscheiden, die jeweiligen Maßnahmen sollten jedoch Einblick darin verschaffen, ob bzw. warum die Intervention ihre Ziele erreicht hat (vgl. Collins, Joseph und Bielaczyc 2004, S. 35 f.). Von besonderem Interesse sind darüber hinaus Elemente der Intervention, die zu Problemen geführt haben oder nicht umgesetzt werden konnten. Sie können

z. B. auf unvorhergesehene Effekte oder Wechselwirkungen hinweisen, die im nächsten Zyklus berücksichtigt werden können.

In Bezug auf den jeweiligen Fokus der Evaluationsmaßnahmen unterscheiden McKenney und Reeves (vgl. 2019, S. 170 ff.) zwischen Alpha-, Beta- und Gamma-Tests einer Intervention. Während Alpha-Test sich auf die interne Konsistenz und Durchführbarkeit des Designs konzentrieren, legen Beta-Tests den Fokus auf dessen Optimierung und nachhaltige Implementierung. In Gamma-Tests stehen die Auswirkungen und Effektivität der jeweiligen Intervention im Vordergrund. Cobb u. a. (2003, S. 9) unterscheiden verschiedene Settings, in denen die Erprobungen stattfinden können:

- *One-on-One Design Experiments*
  Forschende erproben eine Intervention mit einer kleinen Anzahl von Schülerinnen und Schülern.
- *Classroom Experiments*
  Forschende arbeiten mit einer Lehrkraft zusammen, die die Verantwortung für den Unterricht übernimmt und durchaus auch Teil des Forschungsteams sein kann.
- *Preservice/In-Service Teacher Development Experiments*
  Forschende kollaborieren mit Lehrkräften bzw. zukünftigen Lehrkräften, um einen Beitrag zur Lehrerausbildung zu leisten und die Entwicklung einer Fach-*Community* zu unterstützen.
- *School and School District Restructuring Experiments*
  Forschende arbeiten mit Lehrkräften, Schulleitungen oder anderen Interessenvertreter zusammen, um institutionellen Wandel zu unterstützen.

Je nachdem, von wem und in welchem Kontext die Intervention erprobt wird – z. B. nur von Forschenden oder auch von Lehrkräften oder nur an einer Schule im Gegensatz zu einer Reihe von mehreren sehr unterschiedlichen Schulen – liegt der Erprobung eine andere *argumentative grammar* zugrunde (siehe Abschnitt 6.3).

### 7.3.4 Auswertung der erhobenen Daten und Re-Design

Die in der Erprobung generierten Daten werden in einem nächsten Schritt ausgewertet. Im Fokus steht dabei die Beleuchtung der im Lehr-Lerngeschehen abgelaufenen Prozesse sowie spezifische Forschungsfragen, die zu Beginn der jeweiligen Erprobung formuliert wurden. Die Ergebnisse dieser formativen Evaluation werden genutzt, um den Prototypen der Intervention zu verbessern und die zugrundeliegenden *Design Principles* zu schärfen. Die auf Basis der Evaluationsergebnisse modifizierte Intervention, das sogenannte *Re-Design*, wird erneut in der Praxis erprobt und evaluiert. Dieser Zyklus aus Erprobung, Evaluation und Modifikation wird durchlaufen, bis die zugrundeliegende Problemstellung des Forschungsvorhaben zufriedenstellend gelöst wurde (Plomp 2007, S. 19). Nieveen (2007, S. 94) führt weiter aus, das zur Beurteilung der Intervention

*Tabelle 7.1 Kriterien für qualitativ hochwertige Interventionen aus Nieveen (2007, S. 94)*

| Criterion | |
|---|---|
| Relevance (also referred to as content validity) | There is a need for the intervention and its design is based on state-of-the-art (scientific) knowledge. |
| Consistency (also referred to as construct validity) | The intervention is "logically" designed. |
| Practicality | *Expected*<br>The Intervention is expected to be usable in the settings for which it has been designed and developed.<br>*Actual*<br>The intervention is usable in the setting for which it has been designed developed. |
| Effectiveness | *Expected*<br>Using the intervention is expected to result in desired outcomes.<br>*Actual*<br>Using the intervention results in desired outcomes. |

verschiedene Qualitätskriterien herangezogen werden sollten, die es am Ende eines DBR-Prozesses zu erfüllen gilt (siehe Tabelle 7.1).

Sie weist zudem darauf hin, dass dabei zwischen erwarteter und tatsächlicher Praktikabilität und Effektivität unterschieden werden muss. Aussagen zum tatsächlichen Erreichen der Kriterien können nur getätigt werden, wenn die Intervention mit der Zielgruppe erprobt wurde bzw. die Zielanwenderinnen und -anwender sie eingesetzt haben. Evaluationsmaßnahmen, wie z. B. Experteneinschätzungen oder Gruppendiskussionen auf Grundlage von Materialien lassen nur Folgerungen zur erwarteten Praktikabilität bzw. Effektivität zu (Nieveen 2007, S. 94).

Die modifizierte Intervention, die am Ende dieses Zyklus steht, kann als wesentlicher praktischer Output des DBR-Prozesses angesehen werden. Dazu können neben Unterrichtsmaterialien oder Stundenentwürfen z. B. auch Lehrerhandreichungen, Strategien zur Binnendifferenzierungen, Lernziele oder Hinweise zur konkreten Unterrichtsdurchführung gehören (McKenney und Reeves 2019, S. 44)

### 7.3.5 Zusammenschau der Ergebnisse und Theoriebildung

Der letzte Schritt innerhalb des DBR-Prozesses besteht aus der Interpretation sämtlicher Ergebnisse der vorangegangenen Phasen. Damit sollen Theorien entwickelt werden, die einerseits nützlich für die Praxis sind, andererseits aber auch die wissenschaftlichen Erkenntnisse über den jeweiligen Lehr-Lernprozess erweitern (Reinmann 2005, S. 62). Da DBR-Projekte sehr vielfältig sind und je nach Disziplin andere Zielsetzungen verfolgen können, findet sich eine hohe Varianz im jeweiligen theoretischen Output. Die entwickelten Theorien können in Bezug auf ihren Zweck z. B. beschreibender oder erklärender Natur sein. Sie können außerdem zur Vorhersage eines Phänomens beitragen oder vorgeben, wie man bestimmte Phänomene verändern oder beeinflussen kann (McKenney

und Reeves 2019, S. 35 ff.). Edelson (2002, S. 113 ff.) beschreibt drei Ebenen der Generalisierung, auf denen sich diese Theorien bewegen können:

- *Domain Theories*
  Diese Theorien befassen sich mit dem Verhalten von Lernenden und Lehrenden sowie mit Lernumgebungen und deren Einfluss auf das Unterrichtsgeschehen. Edelson führt weiter aus:

  > *„Even though a domain theory in design research is developed through a design process, it is a theory about the world, not a theory about design per se. As such, it is descriptive, not prescriptive."* (Edelson 2002, S. 113)

- *Design Frameworks*
  *Design Framework* beschreiben Eigenschaften, die eine Intervention aufweisen sollte, um bestimmte Ziele in einem bestimmten Kontext zu erreichen und haben demnach präskriptiven Charakter. Bestehend aus einer Sammlung unterschiedlicher Gestaltungsaussagen (*Design Principles*), sind sie praxisnah formuliert und können ggf. auch auf ähnliche Lehr-Lernsituationen übertragen werden.
- *Design Methodologies*
  Im Zentrum der *Design Methodologies* steht das Vorgehen im Design-Prozess, genauer der Zusammenarbeit von Forschenden und Praktikern. Sie beschreiben präskriptiv die gemeinsamen Gestaltungs- und Interaktionsprozesse sowie das dafür erforderliche Fachwissen der Akteure.

Obwohl die *Design Principles* in der DBR eine zentrale Rolle spielen, ist der einschlägigen Literatur nicht eindeutig zu entnehmen, wie diese Prinzipien auszusehen haben bzw. wie man sie im Laufe des DBR-Prozesses weiterentwickelt (Sloane 2014, S. 126). Nach van den Akker (1999, S. 9) sind *Design Principles* in der Regel heuristische Aussagen, für die er folgendes Format vorschlägt:

> *„If you want to design intervention X [for the purpose/function Y in context Z], then you are best advised to give that intervention the characteristics A, B, and C [substantive emphasis], and to do that via procedures K, L, and M [procedural emphasis], because of arguments P, Q, and R."* (van den Akker 1999, S. 9)

Seine Unterscheidung zwischen *substantive* und *procedural emphasis* macht deutlich, dass sich die Gestaltungsleitlinien sowohl auf Merkmale der Intervention beziehen können, als auch darauf, wie die Merkmal im Unterricht zu realisieren sind (van den Akker 1999, S. 5). Euler (2014a, S. 101 f.) vergleicht *Design Principles* aus mehreren DBR-Forschungsprojekten und stellt fest, dass diese sich nicht nur in ihrer inhaltlichen Ausrichtung unterscheiden, sondern auch in ihrem Abstraktionsgrad. Sie bewegen sich von etablierten allgemeindidaktischen Prinzipien (z. B. *real world relevance*) bis hin zu spezifischen Gestaltungregeln, die sich auf eine konkrete Situation beziehen (z. B. *teacher introduces an authentic workplace task*). In Anlehnung an Van Akkeren unterscheidet er

zwischen Leitprinzipien (*substantive emphasis*) und Umsetzungsprinzipien (*procedural emphasis*). Zur Darstellung der *Design Principles* schlägt er eine Grundstruktur vor, die zum einen diese Abstraktionsgrade, zum anderen den jeweiligen Kontext der Intervention sowie die angestrebten Lernergebnisse berücksichtigt (siehe Tabelle 7.2).

*Tabelle 7.2 Grundstruktur Design Principles aus Euler (2014a, S. 107). Die Auslassungszeichen wurden aus dem Original übernommen und sind als Platzhalter für mögliche Lernziele, Leit- und Umsetzungsprinzipien zu sehen.*

| | |
|---|---|
| Kontexte:<br>- Organisationale und soziale Rahmenbedingungen<br>- Individuelle Lernvoraussetzungen | |
| Angestrebte Lernergebnisse (Lernziele):<br>- ...<br>- ... | |
| Intervention (syn.: Lernumgebung, Konzept) | |
| Leitprinzipien - *substantive emphasis*<br>(z. B. Leitideen, lehr-lerntheoretische Annahmen, Auslegung didaktischer Theorien auf den Anwendungskontext)<br>- ...<br>- ... | Begründung<br>Ausweisung von verwendeten theoretischen Referenzen, empirischen Befunden, Plausibilitäts-annahmen, etc.<br>Ggf. Erläuterung der Auswahlentscheidungen. |
| Umsetzungsprinzipien - *procedural emphasis*<br>(z. B. Hinweise über Ausprägung wesentlicher Lehr-Lernaktivitäten, Erfahrungen aus der Praxis)<br>- ...<br>- ... | Begründung<br>Hinweis auf die verwendeten Methoden bei der Erkenntnisgewinnung<br>(z. B. Beobachtungen, Dokumentenauswertung, qualitative oder quantitative Befragung) |

## 7.4 Wissenschaftlichkeit von Design-Based Research

Vergleicht man das Vorgehen der DBR mit anderen Forschungsansätzen innerhalb der Lehr-Lernforschung, z. B. Laborstudien oder sonstigen experimentellen Settings, wird deutlich, dass sich sowohl Zielsetzung als auch die Überprüfbarkeit der Ergebnisse unterscheiden. Während es in experimentellen Settings vorrangig darum geht, statistisch verallgemeinerbare Aussagen zu treffen, konzentriert sich der DBR-Ansatz auf das Generieren von neuen, für die Praxis nützlichen Theorien (Edelson 2002, S. 117 f.). Da die Interventionen im Rahmen der DBR in der Regel in authentischen Unterrichtssettings erprobt werden, wäre es ohnehin kaum möglich, einzelne Variablen zu kontrollieren, was den Nachweis einer statistischen Signifikanz erschweren würde (z. B. Collins, Joseph und Bielaczyc 2004, S. 18 ff.). Dies wirkt sich auch auf die Beurteilung der Qualität der Forschungsergebnisse aus:

> „Bewertungskriterien für DBR sind weniger die klassischen Gütekriterien wie Objektivität, Reliabilität und Validität (obschon diese beim Forschungsprozess selbst beachtet werden), sondern Neuheit, Nützlichkeit und nachhaltige Innovationen. Während sich die Experimentalforschung über statistische Methoden legitimiert, stellt der DBR-Ansatz die Erklärungskraft und interne Konsistenz ihrer Theorien und deren enge

> Verbindung mit praktischer und kontextualisierter Erfahrung in den Vordergrund."
>
> (Reinmann 2005, S. 63)

Außer Frage steht, dass ein Forschungsansatz bei allem Nutzen für die Praxis wissenschaftlichen Prinzipien genügen muss. Shavelson, Phillips und Towne kamen 2003 zu dem Schluss, dass der DBR-Ansatz den *Guiding Principles of Scientific Research* genügt, die das amerikanische *National Research Council* im Jahr 2002 für Forschung im Bildungsbereich formuliert hat. Um die Grenzen des DBR-Ansatzes und mögliche Stolperfallen besser einschätzen und im Forschungsprozess berücksichtigen zu können, ist es dennoch notwendig, kritische Schwachstellen offenzulegen und sich damit zu beschäftigen, wie man ihnen begegnen kann.

### 7.4.1 Kritische Betrachtung

Ein Kritikpunkt am DBR-Ansatz ergibt sich aus den hohen Anforderungen, die an die Forschenden gestellt werden (vgl. Jahn 2014, S. 13 f.). So sind bspw. zur Entwicklung und Analyse der Intervention vielfältige Kenntnisse notwendig, die sich zumeist über mehrere Fachgebiete erstrecken. Mangelnde Fähigkeiten oder Kenntnisse der Forschenden wirken sich direkt auf die Intervention und deren Umsetzung aus, was sich wiederum limitierend auf die Forschungsergebnisse auswirken kann. Diese Gefahr ist besonders groß, wenn der Designer einer Intervention diese auch selbst in der Praxis erprobt. Bezieht man weitere Akteure, z. B. Lehrkräfte, mit ein, läuft man jedoch Gefahr, dass die Intervention nicht wie gewünscht umgesetzt wird, was die Ergebnisse ebenfalls verzerren kann.

Collins, Joseph und Bielaczyc (2004, S. 18 f.) geben weiter zu bedenken, dass der Erfolg bzw. Misserfolg einer Intervention in der Unterrichtspraxis von vielen unkontrollierbaren Variablen beeinflusst wird und es somit sehr schwer ist, eindeutige Ursache-Wirkungs-Zusammenhänge zu beschreiben. Um das Unterrichtsgeschehen im Detail nachvollziehen zu können, erheben viele Forschende, die dem DBR-Ansatz folgen, sehr große Mengen an unterschiedlicher Daten. Dies stellt die Forschenden sowohl in der Auswertung als auch in der Zusammenführung verschiedenen Datenquellen und Datenarten vor eine Herausforderung. Forschende laufen zudem Gefahr, bewusst oder unbewusst jene Daten auszuwählen, die ihre Thesen unterstützen (Brown 1992, S. 162).

Weiterhin stellt sich die Frage, inwieweit *Design Principles*, die im Rahmen eines DBR-Projektes entstanden sind, generalisierbar sind. Reinmann und Sesink (2011, S. 15) konstatieren, dass ein Entwicklungsprojekt letztlich nichts anderes als eine Fallstudie sei, die immer einzigartig sei und daher nicht reproduziert werden könne. In der *Case Study Research* finden sich tatsächlich Parallelen zur *Design-Based Research*, so werden darin ebenfalls Phänomene – sogenannte *cases* – in ihrem realen Kontext mittels einer Fülle an Daten untersucht (vgl. Schögel und Tomczak 2009, S. 90). An dieser Stelle ist jedoch darauf zu verweisen, dass die *Case Study Research* durchaus den Anspruch erhebt, Erkenntnisse zu erzielen, die über die einzelnen Fälle hinausgehen (vgl. Yin 2014, S. 40). Plomp (2007, S. 21) zieht ebenfalls Parallelen zur *Case Study Research* und schlägt vor, sich deren *Replication Logic* zu bedienen. Diese wird im Rahmen von *Multiple-Case Studies* herangezogen:

> *„The replication logic is analogous to that used in multiple experiments (see Hersen & Barlow, 1976). For example, upon uncovering a significant finding from a single experiment, an ensuing and pressing priority would be to replicate this finding by conducting a second, third, and even more experiments. [...] Only with such replications would the original finding be considered robust. The logic underlying the use of multiple-case studies is the same. Each case must be carefully selected so that it either (a) predicts similar results [...] or (b) produces contrary results but for predictable reasons [...].“*
>
> (Yin 2014, S. 63)

Er schlägt vor, dass *Design Principles* mehrfach in verschiedenen Kontexten getestet werden sollten. Nur, wenn sich dabei ähnliche Ergebnisse zeigen, ist davon auszugehen, dass die resultierenden Theorien generalisiert werden können. Gleichzeitig weist er jedoch darauf hin, dass *Design Principles* lediglich als heuristische Aussagen zu verstehen sind:

> *„But a warning should be phrased here. Where design principles may have been supported by a number of replications, and a new context may be similar to the ones from which design principles have emerged, yet each context has unique characteristics that justifies that the design principles should be used as ‚heuristic' statements: they provide guidance and direction, but do not give ‚certainties'.“* (Plomp 2007, S. 22)

Die *Replication Logic* wird auch von McKenney und Reeves (2019, S. 39 ff.) bzgl. der Reichweite von Theorien aufgegriffen. Sie unterscheiden zwischen verschiedenen *Levels of Educational Theories*, die mittels DBR je nach Forschungsszenario erreicht werden können. Sie weisen jedoch darauf hin, dass es sich bei den Abstufungen nicht um klar voneinander trennbare Kategorien handelt, sondern eher um überlappende Bereiche auf einem Kontinuum.

- *Local Theories*
  Es handelt sich um Theorien, die entstehen, wenn eine Intervention mehrfach in vergleichbaren Settings erprobt wurde, z. B. mit wenigen Schulklassen mit ähnlichen Voraussetzungen. Es lassen sich damit zwar Aussagen über einen spezifischen Kontext treffen, darüber hinaus haben sie jedoch begrenzte Aussagekraft.
- *Middle-range Theories*
  Theorien mittlerer Reichweite können entwickelt werden, wenn die Umsetzung und Auswirkungen einer Intervention in verschiedenen Settings untersucht wurde, z. B. in mehreren Iterationen in verschiedenen Klassenräumen oder Schulen.
- *High-Level Theories*
  Sie entstehen, wenn mehrere Theorien mittlerer Reichweite zusammengefasst werden oder können ggf. entstehen, wenn ähnliche Ausprägungen einer Intervention in vielen verschiedenen Settings erprobt wurden.

Der unterschiedliche Grad an Generalisierbarkeit von Ergebnissen hängt wiederum mit der immanenten *Argumentative Grammar* der Anlage des jeweiligen *Design-Based Research*-Projekts zusammen (siehe Abschnitt 6.3). So gehen *Local Theories* nicht über einen *Proof of Principle* hinaus, da die zugrundeliegende Intervention nur in vergleichbaren Settings durchgeführt wurde. Basis der *Middle-range Theories* ist die Argumentationsform *Small Changes per Iteration*, die von Plomp auch als *Replication Logic* bezeichnet wird. Den *High-Level Theories* liegt die Argumentationsform *Experience of the Design Community* zugrunde, da sich durch die Erprobung in verschiedenen Kontexten und durch verschiedene Lehrende innerhalb einer Community eine breite Anwendbarkeit zeigen lässt.

Unabhängig von der Reichweite der *Design Principles* bleibt die Frage, wie diese in der Praxis genutzt werden können. Dabei ist festzuhalten, dass es sich dabei eher um Impulse als genaue Handlungsanweisungen handelt, die Lehrkräfte für ihre jeweilige Situation interpretieren müssen (vgl. Euler 2014a, S. 99).

### 7.4.2 Methodische Standards

Da sich DBR durch eine große methodische Offenheit auszeichnet und die unterschiedlichsten methodischen Zugänge möglich sind, ist es schwer, spezifische methodische Standards zu definieren. Es lassen sich jedoch unterschiedliche Grundsätze identifizieren, die bei der Durchführung eines DBR-Projektes bedacht werden sollten.

McKenney, Nieveen und van den Akker (2006, S. 83 f.) postulieren in ihren *Design Study Guidelines*, dass die Argumentationslinie des Studiendesigns transparent und eng verknüpft mit vorheriger Forschung, Theorie und Forschungsfragen sein sollte. Zudem sollten Datenquellen, Erhebungssettings, sowie Instrumente im Sinne einer Triangulation ausgewählt bzw. angewandt werden, um den Forschungsgegenstand aus mehreren Perspektiven betrachten zu können. Auch bei der Auswertung der Daten sollte man variieren und verschiedene Perspektiven einnehmen. Sinnvoll können dabei sowohl deduktive als auch induktive Analysen sein sowie wiederholte Zwischenanalysen. Diese sogenannten *Sequential Analyses* finden im Anschluss an eine Phase oder einen Zyklus statt und fassen die Ergebnisse für die nachfolgenden Arbeiten zusammen. Während des ganzen Prozesses der DBR sollten Praktiker und andere Forschende als *Critical Friends* einbezogen werden, um potentielle Schwächen oder vorzeitige Schlussfolgerungen zu vermeiden. Da die Generalisierbarkeit von Ergebnissen innerhalb der DBR umstritten ist, sollten sämtliche Phasen des Forschungsprozesses systematisch und umfassend dokumentiert, analysiert und reflektiert werden. Umfassende Beschreibungen der Phasen und des jeweiligen Kontextes sollen es den späteren Rezipienten ermöglichen, einen Einblick in die Geschehnisse zu gewinnen und auf Grundlage der Ergebnisse Rückschlüsse auf andere Situationen zu ziehen.

Auch andere Vertreterinnen und Vertreter der DBR betonen die Bedeutsamkeit einer systematischen Dokumentation und Reflexion in allen Phasen des Forschungsprozesses (z. B. Plomp 2007;

Tulodziecki, Grafe und Herzig 2013). Collins, Joseph und Bielaczyc (2004, S. 38 f.) schlagen verschiedene Aspekte vor, die für jeden Design-Zyklus festgehalten werden sollten:

- *Goals and Elements of the Design*
  Die wesentlichen Ziele und Elemente einer Intervention, z. B. Materialien, Lehr-Lernaktivitäten oder bestimmte didaktische Prinzipien, sollten für jede Erprobung dokumentiert werden. Darüber hinaus sollte beschrieben werden, wie die Elemente zusammenwirken sollen, um die Ziele zu erreichen.

- *Settings where implemented*
  Für jede Iteration sollten die jeweiligen Rahmenbedingungen dokumentiert werden, die für den Erfolg einer Intervention relevant sein könnten. Dazu gehören z. B. der Ort der Eprobung, benötigte Ressourcen und Informationen zu den Schülerinnen und Schülern. Falls die Intervention nicht von den Forschenden selbst durchgeführt wurde, sollte dokumentiert werden, ob bzw. welche Art von Fortbildung die Lehrenden erhalten haben.

- *Description of each Phase*
  Da sich die Intervention vermutlich in jeder Iteration verändert, müssen die jeweiligen Anpassungen nachvollziehbar festgehalten werden. Zudem sollten die Gründe für die Änderungen sowie deren Auswirkungen angegeben werden.

- *Outcomes found*
  Die erzielten Ergebnisse sollten für jede Erprobung dargelegt werden. Dabei können sowohl Ergebnisse der Schülerinnen und Schüler als auch Zwischenergebnisse der Evaluation berücksichtigt werden.

- *Lessons learned*
  Es sollte möglich sein, die Entwicklung der Intervention über den ganzen Forschungsprozess verfolgen zu können. Dafür ist es wichtig, die Grenzen und Schwächen sowie die Erfolge der jeweiligen Version sowohl bei der Umsetzung als auch bei den Lernergebnissen zu beschreiben.

# 8 Design der Untersuchung

Ziel dieser Arbeit ist es einerseits, Konzepte und Interventionen zu entwickeln, wie das Programmieren in der Grundschule implementiert werden kann, andererseits müssen die Lernwirksamkeit und die notwendigen Rahmenbedingungen der Interventionen empirisch erforschbar sein. Um dieser doppelten Zielsetzung gerecht zu werden, folgt die Arbeit dem Ansatz der *Design-Based Research* (siehe Abbildung 8.1).

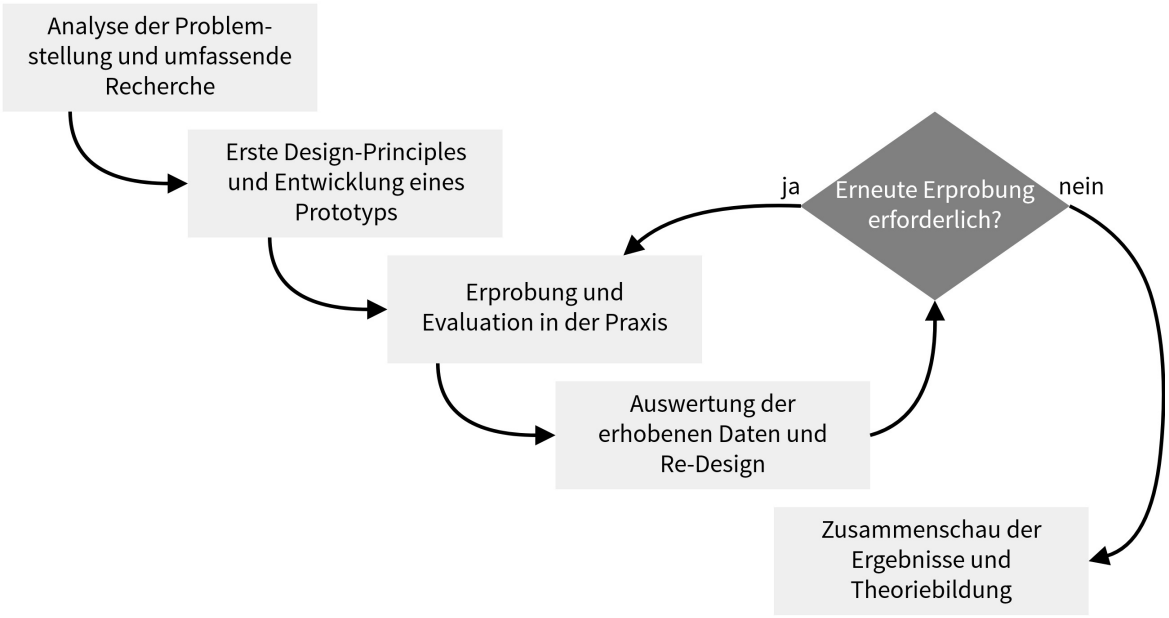



*Abbildung 8.1 Umsetzung des Design-Based Research-Ansatzes in der vorliegenden Arbeit*

Nach der Analyse der Problemstellung und einer umfassenden interdisziplinären Literaturrecherche werden vorläufige *Design Principles* und darauf aufbauend eine prototypische Unterrichtssequenz sowie ein Fortbildungskonzept zum Programmieren in der Grundschule entwickelt. Diese werden innerhalb mehrerer Zyklen in der Praxis erprobt, evaluiert und überarbeitet, bis die Problemstellung zufriedenstellend gelöst ist. In einem letzten Schritt werden sämtliche Ergebnisse der vorangegangenen Phasen interpretiert und daraus Theorien entwickelt, die nützlich für die Praxis sind und gleichzeitig den einschlägigen wissenschaftlichen Diskurs bereichern.

## 8.1 Recherche

Die Planung von Unterricht bzw. didaktischen Interventionen im Allgemeinen ist ein komplexer Prozess, der von vielen Faktoren abhängt. Um Lehrkräften bei der Planung, Vorbereitung und Auswertung von didaktischen Situationen zu helfen, wurden didaktische Modelle entwickelt,

welche meist sehr allgemein Zieldimensionen, Kategorien und Bereiche von Unterricht in einen strukturellen Zusammenhang bringen (Riedl und Schelten 2013, S. 252, vgl. Kapitel 12). Euler (2014a, S. 105 f.) konstatiert, dass sich diese zwar in ihren Ausdifferenzierungen unterscheiden, in der Regel jedoch folgende Aspekte als bedeutsam ausweisen:

> • „Aussagen über die individuellen Lernvoraussetzungen und weitere Rahmenbedingungen;
> • Vorstellungen über die angestrebten Lernergebnisse (Lernziele);
> • Annahmen darüber, welche *Lern*aktivitäten für die Lernenden geeignet sind, um die angestrebten Lernergebnisse zu erreichen;
> • Vorstellungen darüber, welche *Lehr*aktivitäten die geeigneten Lernaktivitäten unterstützen können." (Euler 2014a, S. 105 f.)

Er verbindet diese Aspekte in einem didaktischen Bezugsrahmen, der einerseits die Kernzusammenhänge einer didaktischen Situation modelliert, andererseits aufzeigt, inwiefern *Design Principles* darin verortet werden können (siehe Abbildung 8.2).

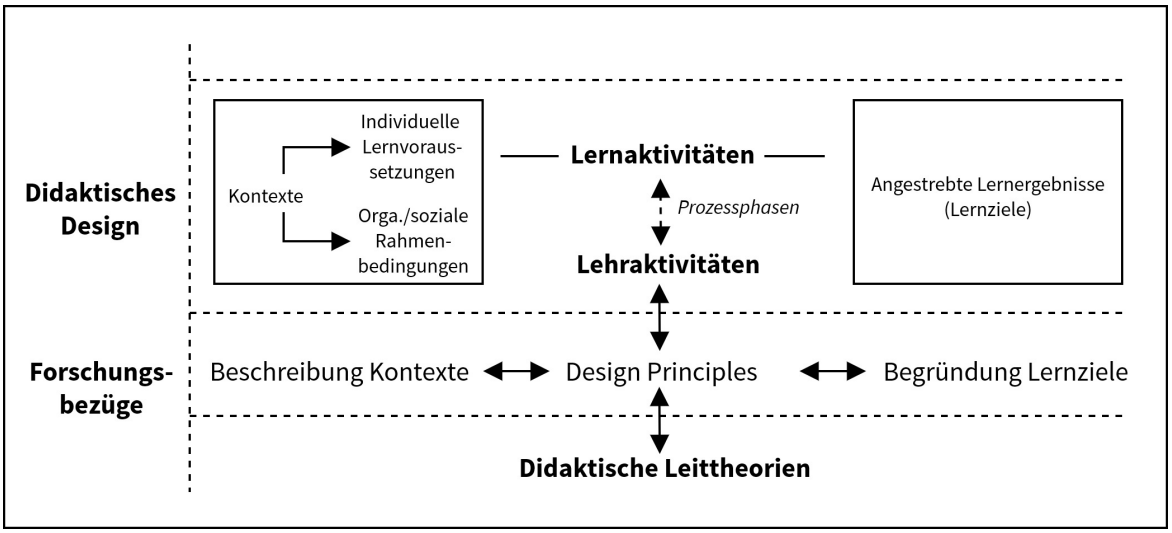



*Abbildung 8.2 Didaktischer Bezugsrahmen für die Entwicklung von Design Principles aus Euler (2014a, S. 106)*

In der Darstellung der Ausgangslage und der theoretischen Grundlagen dieser Arbeit werden Theorien und Befunde für die Entwicklung der Gestaltungsprinzipien und didaktischen Interventionen zum Programmieren in der Grundschule dargestellt, die in Eulers Abbildung als *Forschungsbezüge* aufgeführt werden. Tabelle 8.1 zeigt, wie sich die ausgewählten Themenbereiche auf die darunter geführten Aspekte verteilen.

Eulers Struktur wird aus mehreren Gründen nicht für die Aufteilung der Kapitel übernommen. Zum einen gibt es mehrere Überschneidungen der Kapitel, zum anderen bauen einzelne Kapitel aufeinander auf und bilden die Grundlage für Folgekapitel.

*Tabelle 8.1 Verortung der Inhalte des thematischen Bezugsrahmens dieser Arbeit im didaktischen Bezugsrahmen aus Euler (2014a, S. 106)*

| Aspekte des Forschungsbezugs | (Unter-)Kapitel dieser Arbeit |
|---|---|
| Beschreibung Kontexte | Informatische Bildung im Primarbereich (2)<br>Internationale Situation (2.2)<br>Situation in Deutschland (2.3)<br>Gelingensbedingungen (2.4)<br>Fortbildungsmaßnahmen im Bereich der informatischen Bildung (4)<br>Entwicklungspsychologische Grundlagen (10)<br>Hintergrund zur Lehrerbildung (13)<br>Strukturelle Merkmale (13.1) |
| Begründung Lernziele | Informatische Bildung im Primarbereich (2)<br>Potenzial und Chancen (2.1)<br>Internationale Situation (2.2)<br>Situation in Deutschland (2.3)<br>Programmieren (3)<br>Fortbildungsmaßnahmen im Bereich der informatischen Bildung (4)<br>Fachwissenschaftliche Grundlagen (9)<br>Fachdidaktische Grundlagen (12)<br>Hintergrund zur Lehrerbildung (13)<br>Inhaltliche Merkmale (13.3) |
| Didaktische Leittheorien | Grundschulpädagogische und -didaktische Grundlagen (11)<br>Fachdidaktische Grundlagen (12)<br>Hintergrund zur Lehrerbildung (13)<br>Aktivitäten (13.2)<br>Inhaltliche Merkmale (13.3) |

## 8.2 Entwicklung der prototypischen Interventionen

Im Anschluss werden die unterschiedlichen Themenbereiche des thematischen Bezugsrahmens zusammengeführt und in Leitlinien und Konzepte für die Umsetzung des Programmierens in der Grundschule überführt. Dabei geht es nicht um eine einfache lineare Umsetzung von einzelnen theoretischen Ansätzen in die pädagogische Praxis, sondern um die Darstellung wechselseitiger Beziehungen zwischen den ausgewählten theoretischen Ansätzen sowie der gewählten Zielvorstellungen und Lernvoraussetzungen (vgl. Grafe 2008, S. 136).

Zuerst wird die Entwicklung einer Unterrichtssequenz für die Grundschule beschrieben und darauf aufbauend die Entwicklung eines Fortbildungskonzepts für Lehrkräfte der Grundschule thematisiert. Dafür werden zunächst jeweils die angestrebten Zielvorstellungen erläutert, die aus den vorangegangenen Ausführungen abgeleitet werden. Gemäß der in Kapitel 7.3.2 beschriebenen Vorgehensweise werden in einem nächsten Schritt vorläufige *Design Principles* – sowohl für die Unterrichtssequenz, als auch das Fortbildungskonzept – erarbeitet sowie theoretisch fundiert. Die *Design Principles* für die Unterrichtssequenz beziehen sich auf die Lern- und Lehraktivitäten, die für das Fortbildungskonzept auf die Rahmenbedingungen und die Lernaktivitäten. Um die

didaktische Konzeption der beiden Interventionen vorzubereiten, werden die *Design Principles* in einem mehrstufigen Operationalisierungsprozess konkretisiert. Auf Basis der entwickelten Umsetzungsprinzipien und Musterbeispiele wird im Anschluss die prototypische Unterrichtssequenz bzw. das prototypische Fortbildungskonzept konstruiert. Deren didaktische Konzeption umfasst in Anlehnung an das Vorgehen von Grafe (2008) den Entwurf einer unterrichtlichen Handlungslinie sowie ein Artikulationsschema mit Beschreibungen von Lernaktivitäten und Lehrhandlungen. Die daraus resultierenden Ergebnisse werden sowohl mit erfahrenen Informatiklehrkräften als auch mit Experten im Bereich der Informatikdidaktikforschung und Lehrerausbildung besprochen, um ein multiperspektivisches Feedback zu erhalten. Diese Alpha-Tests sollen in erster Linie die interne Konsistenz der Interventionen sicherstellen (vgl. Kapitel 7.3.3)

## 8.3 Erprobung der Interventionen in der Praxis

Um die Durchführbarkeit der Unterrichtssequenz sicherzustellen, wird zunächst eine Pilotstudie (02/2016 – 05/2016) als weiterer Alpha-Test beschrieben, in der die Intervention in Form eines *One-on-One Design Experiments* als freiwilliger Ferienkurs am Schülerforschungszentrum Berchtesgadener Land (SFZ-BGL) erprobt wurde (siehe Zyklus 1 in Abbildung 8.3). Weiterhin werden Erfahrungen und Konsequenzen daraus vorgestellt. Für den Entwurf des Fortbildungskonzepts wird auf die Pilotstudie verzichtet, da zum Zeitpunkt des Entwurfs nicht genügend interessierte Grundschullehrkräfte gefunden werden konnten.

Im Anschluss werden Beta- und Gamma-Tests beschrieben, deren Fokus auf der Optimierung, den Auswirkungen und der Effektivität der Interventionen liegt. Diese finden im Rahmen mehrerer Forschungs- und Entwicklungszyklen innerhalb des DBR-Prozesses statt, die in Abbildung 8.3 in Abhängigkeit zum zeitlichen Verlauf der Studie dargestellt sind.

In Zyklus 2 (06/2016 – 05/2017) wird die Unterrichtssequenz auf Basis der Pilotierung weiterentwickelt und in Form von zwei Klassenkursen erprobt. Zusätzlich wird die Unterrichtssequenz als Poster auf einer internationalen Konferenz zur Informatik in der Schule (Geldreich, Funke und Hubwieser 2016) präsentiert und Feedback dazu eingeholt. In Zyklus 3 (06/2017 – 04/2018) werden anhand der Ergebnisse der vorangegangenen Erprobungen weitere Anpassungen an der Unterrichtssequenz vorgenommen. Die überarbeitete Intervention wird innerhalb vier Klassenkurse erneut eingesetzt und parallel dazu zusätzliche Varianten einzelner Aufgaben in zwei Ferienkursen am Schülerforschungszentrum Berchtesgadener Land erprobt. Die Unterrichtssequenz wird darüber hinaus in einem Workshop für Lehramtsanwärterinnen und -anwärter für die Grundschule am SFZ-BGL sowie in einem Workshop auf einer nationalen Konferenz für Informatik in der Schule (Geldreich, Funke und Hubwieser 2017) vorgestellt und Feedback dazu eingeholt. Sämtliche Klassenkurse finden als dreitägige *Classroom Experiments* in den Räumlichkeiten der Technischen Universität München (TUM) statt. Um eine umfassende Dokumentation der Kurse zu ermöglichen, werden die Klassen für die Kursdauer in zwei Gruppen aufgeteilt und getrennt unterrichtet. Dabei

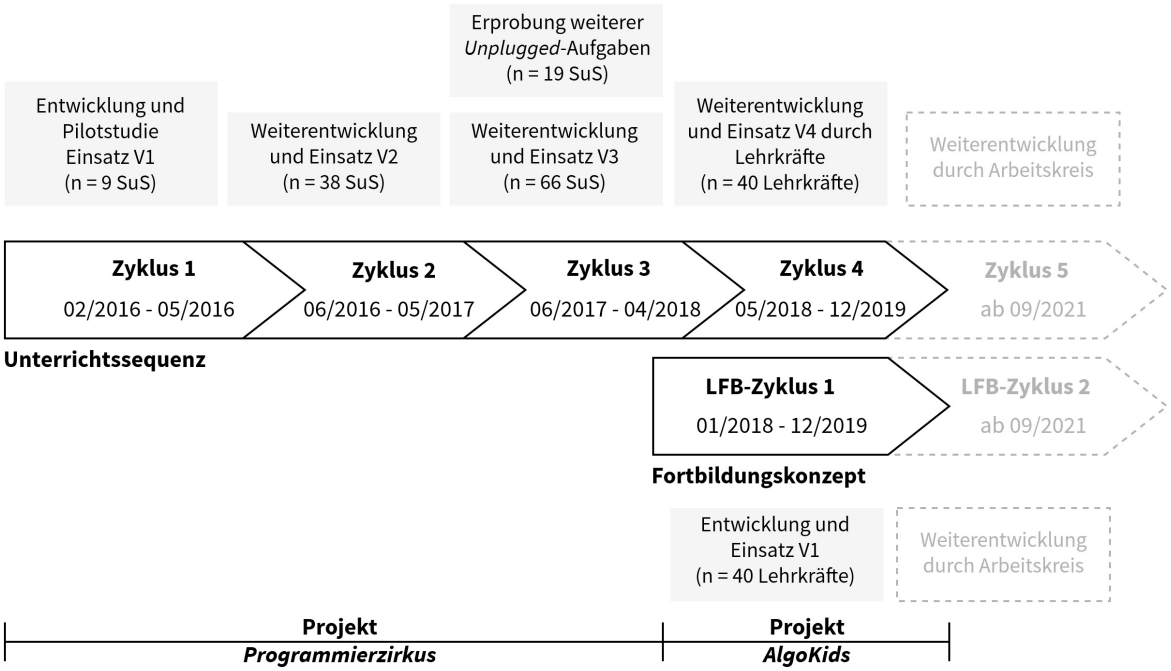



*Abbildung 8.3 Übersicht der Forschungszyklen (ausgegrauter Bereich wird in dieser Arbeit nicht behandelt).*

wird jeweils eine Gruppe von der Autorin unterrichtet und eine Gruppe von einer weiteren Person – hier kommen sowohl Studierende des Lehramts Informatik an Gymnasien als auch Forschende der Didaktik der Informatik zum Einsatz. Nach weiteren Anpassungen wird die Unterrichtssequenz in Zyklus 4 (05/2018 – 12/2019) von 40 Grundschullehrkräften mit ihren Schülerinnen und Schülern erprobt. Um dies zu ermöglichen, wird in LFB-Zyklus 1 ein entsprechendes Fortbildungskonzept entwickelt und als *In-Service Teacher Development Experiment* mit den Lehrkräften erprobt. Die Erfahrungen in Zyklus 4 und LFB-Zyklus 1 münden in einer – für diese Arbeit – vorerst letzten Version der Unterrichtssequenz und des Fortbildungskonzepts. Beides wird im Anschluss im Auftrag des Bayerischen Staatsministerium für Unterricht und Kultus (StMBKWK) durch einen Arbeitskreis am Staatsinstitut für Schulqualität und Bildungsforschung (ISB)[1] für einen bayernweiten Einsatz aufbereitet.

## 8.4 Einordnung in die Projekte *Programmierzirkus* und *AlgoKids*

Die vorliegende Forschungsarbeit ist maßgeblich im Rahmen zweier Projekte – *Programmierzirkus* und *AlgoKids* entstanden. Ersteres wurde von 2016 bis 2018 aus Eigenmitteln der Professur für Didaktik der Informatik der TUM (TUM-DDI) finanziert und hatte zum Ziel, die Möglichkeiten und Grenzen der Programmierung mit Grundschulkindern auszuloten. Dafür wurde eine Unterrichtssequenz für die dritte und vierte Jahrgangsstufe entwickelt, die in Form eines dreitägigen Programmierkurses mit 132 Schülerinnen und Schülern erprobt und währenddessen stetig weiterentwickelt wurde.

[1] Das Staatsinstitut für Schulqualität und Bildungsforschung (ISB) ist eine zentrale Einrichtung, die im Auftrag des Bayerischen Staatsministerium für Unterricht und Kultus Impulse zur qualitative Weiterentwicklung des bayerischen Schulwesens gibt. Ein Ziel ist dabei, Erkenntnisse der Forschung für die Schule nutzbar zu machen (http://www.isb.bayern.de/).

Die Entwicklung und Erprobung der Unterrichtssequenz stellt einen wesentlichen Teil dieser Arbeit dar. Aufgrund der vielversprechenden Ergebnisse der Erprobungen des *Programmierzirkus* wurde das Staatsministerium für Unterricht und Kultus auf die Arbeit der TUM-DDI aufmerksam. In der Folge wurde das Projekt *AlgoKids – Algorithmen für Kinder* als Kooperationsprojekt zwischen der Professur für Didaktik der Informatik und dem Staatsministerium gegründet und als offizieller Pilotversuch durchgeführt. Darin sollten jeweils zwei Lehrkräfte aus 20 bayerischen Projektschulen das Programmieren in ihrem Unterricht umsetzen. Die TUM-DDI war 2018 bis 2019 für die Konzeption, Durchführung und wissenschaftliche Auswertung des Projekts verantwortlich, während das Staatsministerium die Finanzierung des wissenschaftlichen Personals und die Reisekosten, die im Rahmen des Projekts anfielen, übernahm. In dem Projekt sollte einerseits untersucht werden, wie man die von der TUM-DDI entwickelte und erprobte Unterrichtssequenz im regulären Unterricht der bayerischen Grundschulen implementieren könnte. Andererseits stand im Fokus, wie man Grundschullehrkräfte ohne informatische Vorbildung so fortbilden kann, dass sie die Themen Algorithmen und Programmierung ausreichend kompetent unterrichten können. Im Rahmen des Projekts sollte dazu ein Fortbildungskonzept entwickelt, mit 40 Lehrkräften erprobt und evaluiert werden – was einen weiteren wesentlichen Teil dieser Arbeit ausmacht. Darüber hinaus ermöglichte die Erprobung der Unterrichtssequenz durch die fortgebildeten Lehrkräfte eine weitere Überarbeitung bzw. Anreicherung der Unterrichtssequenz.

Da beide Projekte im Freistaat Bayern durchgeführt wurden, mussten die Bestimmungen des Staatsministeriums für Unterricht und Kultus für die Forschung an Schulen beachtet werden[2]. Demnach wird die Durchführung von Erhebungen an öffentlichen Schulen in Bayern nur genehmigt, wenn (1) die Erhebungen nur an Schulen durchgeführt werden können, (2) die daraus resultierende Belastung der Schulen in einem vertretbaren Rahmen gehalten wird und (3) die zu erwartenden Ergebnisse von erheblichem pädagogischen wie auch wissenschaftlichen Interesse sind. Ein Genehmigungsverfahren erfordert zum einen eine sehr umfangreiche Antragstellung und zum anderen einen hohen Zeitaufwand, da die Anträge zeitintensiv geprüft werden und ggf. überarbeitet und neu eingereicht werden müssen.

## 8.5 Datenerhebung

Zur Untersuchung der Forschungsfragen werden verschiedene Methoden und Instrumente benötigt, die im Folgenden vorgestellt werden. Sie bilden den Ausgangspunkt für die durchgeführten Studien, die in Teil IV dieser Arbeit vorgestellt werden. Im Rahmen des Projekts *Programmierzirkus* wurden von der Autorin weitere Daten erhoben, deren Auswertung jedoch nicht im Fokus dieser Arbeit steht. Ergebnisse, die bereits veröffentlicht wurden und für die Überarbeitung der Unterrichtssequenz relevant sind, werden jedoch in den entsprechenden Kapiteln zusammenfassend dargestellt. Auf eine detaillierte Beschreibung der Datenerhebung wird an dieser Stelle verzichtet.

[2] https://www.km.bayern.de/ministerium/statistiken-und-forschung/forschung-an-schulen.html

### 8.5.1 Fragebögen

Für die Evaluation des Projekts *AlgoKids – Algorithmen für Kinder* wurden von der Autorin vier Fragebögen für Lehrkräfte (FB-LK) entwickelt (siehe Tabelle 8.2, Anhang A.3). Diese Erhebungen richteten sich in Form von *Paper-Pencil*-Fragebögen jeweils gleichzeitig an alle teilnehmenden Lehrkräfte[3]. Sie verteilten sich auf insgesamt vier Erhebungszeitpunkte: zu Beginn und am Ende der ersten Fortbildungsveranstaltung (FB-LK1 und FB-LK2), zu Beginn der zweiten Fortbildung (FB-LK3) und am Ende des Projekts (FB-LK4). Um die Ergebnisse der Fragebögen zu verknüpfen, erzeugten die Teilnehmerinnen und Teilnehmer einen Einweg-Hash-Wert aus ihren Personendaten, der zwar die Verfolgung einer Person durch alle Erhebungen, aber keine Rückschlüsse auf ihre Identität erlaubt. Eine grafische Übersicht der Erhebungszeitpunkte kann in Kapitel A.1 eingesehen werden.

*Tabelle 8.2 Übersicht über die Inhalte der Fragebögen*

| Fragebogen (Messzeitpunkt) | Inhalte |
|---|---|
| FB-LK1 (siehe Abschnitt A.3.1)<br>(Projektstart bzw. Beginn der ersten Fortbildungsveranstaltung) | demografische Angaben; Situation an der Schule; Erwartungen an das Projekt; Vorerfahrungen im Bereich der Programmierung; Einschätzung der Selbstwirksamkeit im Umgang mit Computern; |
| FB-LK2 (siehe Abschnitt A.3.2)<br>(Ende der ersten Fortbildungsveranstaltung) | Zufriedenheit mit der Fortbildungs-veranstaltung; persönliche Reflexion der Fortbildungsinhalte; Selbsteinschätzung bzgl. der Umsetzung der Fortbildungsinhalte; Selbsteinschätzung der Programmierfähigkeit; |
| FB-LK3 (siehe Abschnitt A.3.3)<br>(Beginn der zweiten Fortbildungsveranstaltung) | bisherige Umsetzung des Programmierens im Unterricht; Selbsteinschätzung bzgl. der Umsetzung der Fortbildungsinhalte; Selbsteinschätzung der Programmierfähigkeit; |
| FB-LK4 (siehe Abschnitt A.3.4)<br>(Projektende bzw. Ende der dritten Fortbildungsveranstaltung) | Umsetzung des Programmierens im Unterricht; persönliche Reflexion der Fortbildungsreihe; Einschätzung der Selbstwirksamkeit im Umgang mit Computern; Selbsteinschätzung bzgl. der Umsetzung der Fortbildungsinhalte; Selbsteinschätzung der Programmierfähigkeit; |

Der Fragebogen FB-LK1 erfasst allgemeine Angaben zur Situation der Lehrkräfte, ihre Erwartungen an das Projekt, einschlägige Vorerfahrungen sowie ihre eingeschätzte Selbstwirksamkeit im Umgang mit Computern. Der Fragebogen wurde theoriegeleitet entwickelt und durch projektbezogene Items ergänzt. Die Items zur technischen Ausstattung wurden dem Fragebogen des Länderindikators entnommen (Lorenz u. a. 2017, S. 57). Für die Operationalisierung der Sicherheit im Umgang mit Computern und Computeranwendungen wurde die entsprechende Skala von Richter, Naumann

[3] Die schriftlichen Befragungen wurden vom Bayerischen Staatsministerium für Unterricht und Kultus am 12.11.2018 unter dem Aktenzeichen IV.8-BO7106/113/9 genehmigt.

und Horz (2010) aus der revidierten Fassung des Inventars zur Computerbildung (INCOBI-R) genutzt. Darüber hinaus wurden verschiedene Skalen zum Arbeitsbezogenen Verhalten und Erleben aus dem Instrument AVEM (Schaarschmidt 2006) eingesetzt: Perfektionsstreben, Resignationstendenz (bei Misserfolg), offensive Problembewältigung und Innere Ruhe/Ausgeglichenheit. Diese werden jedoch nicht in die Auswertungen dieser Arbeit einbezogen.

Am Ende der ersten Fortbildungsveranstaltung wurden die Lehrkräfte im Fragebogen FB-LK2 befragt, wie zufrieden sie mit der Fortbildung waren und wie sie deren Nutzen, Dauer und Schwierigkeit einschätzten. Zusätzlich wurde erhoben, inwieweit sie sich zutrauen, das Themengebiet Algorithmen und Programmierung in ihrem Unterricht umzusetzen und wie sie ihre eigene Programmierfähigkeit einschätzen. Die Items für die allgemeine Evaluation der Fortbildung wurden der Arbeit von Vigerske (2017) entnommen, die Skala zur Einschätzung der Programmierfähigkeit (*CP-SES: Computer Programming Self-Efficacy Scale*) wurde von Tsai, Wang und Hsu (2017) entwickelt und vorab ins Deutsche übersetzt. In letzterer wurden zwei projektspezifische Items ergänzt. Die Items zur Selbstwirksamkeit in Bezug auf die Umsetzung der Fortbildungsinhalte im Unterricht wurden in Anlehnung an Jerusalem u. a. (2009) konstruiert.

In Fragebogen FB-LK3 wurden die Lehrkräfte zu Beginn der zweiten Fortbildung gefragt, ob sie die Fortbildungsinhalte bereits im Unterricht umsetzen konnten. Zudem wurden sie erneut befragt, inwieweit sie sich zutrauen, das Themengebiet Algorithmen und Programmierung in ihrem Unterricht umzusetzen und wie sie ihre eigene Programmierfähigkeit einschätzen.

Am Ende der dritten Fortbildung wurde in Fragebogen FB-LK4 ein letztes Mal erhoben, inwiefern sich die Lehrkräfte zutrauen, das Themengebiet Algorithmen und Programmierung im Unterricht umzusetzen und wie sie ihre eigene Programmierfähigkeit und Selbstwirksamkeit im Umgang mit Computern einschätzen. Außerdem wurde abschließend erhoben, wie die Lehrkräfte die Fortbildungsinhalte in ihrem Unterricht umsetzten, welchen Nutzen sie darin sehen und wodurch die Umsetzung erschwert wurde. Für entsprechende Skalen wurde erneut auf ein Instrument von Vigerske (2017) zurückgegriffen.

Die Skala zur Sicherheit im Umgang mit Computern und Computeranwendungen weist laut Richter, Naumann und Horz (2010) für acht Items einen Wert von Cronbachs Alpha von 0.88 auf, was für eine gute interne Konsistenz spricht (George und Mallery 2016, S. 240). Für die Skalen zur Evaluation der Fortbildung gibt Vigerske (2017) für Cronbachs Alpha Werte zwischen 0.72 und 0.92 bei maximal fünf Items an – die Werte sprechen demnach alle für eine mindestens akzeptable interne Konsistenz (siehe Anhang A.4). Für die Skala zur Selbsteinschätzung der Programmierfähigkeit von Tsai, Wang und Hsu (2017) wird für sechzehn Items ein Wert von Cronbachs Alpha von 0.96 angegeben, die Werte der fünf Subskalen variieren zwischen 0.84 und 0.96 – was auf eine gute bzw. sehr gute interne Konsistenz hinweist. Da die Skala ins Deutsche Übersetzt wurde und die Subskala *Logical Thinking* um zwei Items ergänzt wurde, wurde Cronbachs Alpha für die drei Messzeitpunkte erneut bestimmt. Da die Subskala *Cooperation* mit drei Items zum ersten Messzeitpunkt nur einen inakzeptablen Wert

von 0.497 aufwies, wurde die Subskala mit den Items 7-9 für die Auswertungen nicht berücksichtigt. In der Subskala *Debug* konnte mit drei Items ein Wert von Cronbachs Alpha von 0.597 bestimmt werden. Daraufhin wurde Item 18 entfernt und die Reliabilität mithilfe des Spearman-Brown-Koeffizienten $\rho$ berechnet, der sich für die Berechnung der Reliabilität von Skalen mit nur zwei Items eignet (Eisinga, Grotenhuis und Pelzer 2013). Hier ergab sich ein Wert von 0.830, was für eine gute interne Konsistenz spricht. Für die restlichen drei Subskalen konnten akzeptable bzw. gute Werte ermittelt werden: *Logical Thinking*: $\alpha$ = 0.846 (6 Items); *Algorithm*: $\alpha$ = 0.742 (3 Items); *Control*: $\alpha$ = 0.818 (3 Items). Für den zweiten Messzeitpunkt konnten für alle Subskalen gute Werte von Cronbachs Alpha bzw. des Spearman-Brown-Koeffizienten gemessen werden: *Logical Thinking*: $\alpha$ = 0.811 (6 Items); *Algorithm*: $\alpha$ = 0.805 (3 Items); *Control*: $\alpha$ = 0.818 (3 Items); *Debug*: $\rho$ = 0.890 (2 Items). Auch für den dritten Messzeitpunkt konnten akzeptable bzw. gute Werte bestimmt werden: *Logical Thinking*: $\alpha$ = 0.770 (6 Items); *Algorithm*: $\alpha$ = 0.844 (3 Items); *Control*: $\alpha$ = 0.747 (3 Items); *Debug*: $\rho$ = 0.890 (2 Items). Da die Skala Selbstwirksamkeit in Bezug auf die Umsetzung der Fortbildungsinhalte nur in Anlehnung an Jerusalem u. a. (2009) konstruiert wurde, wurde ebenfalls die interne Konsistenz für alle drei Messzeitpunkte berechnet. Hier ergab sich für den ersten Messzeitpunkt für neun Items ein Wert von $\alpha$ = 0.806 und für den zweiten $\alpha$ = 0.873 was für eine gute interne Konsistenz spricht. Zum dritten Messzeitpunkt betrug Cronbachs Alpha 0.792, was auf eine akzeptable interne Konsistenz hinweist.

### 8.5.2 Interviews

Um einen umfassenden Einblick in die Erfahrungen und Einschätzungen der Lehrkräfte bzgl. des Programmierens mit ihren Schülerinnen und Schülern zu erhalten, wurde mit ihnen ein exploratives Interview durchgeführt. Das explorative Interview ist keine – wie das klassische Interview – rein asymmetrische Kommunikationsform (Honer 2011, S. 47). Obwohl es nach wie vor eine Rollentrennung zwischen Frager und Befragten gibt, handelt es sich bei der Interviewsituation zunächst um ein quasi-normales Gespräch:

> „Dadurch also, dass auch die Interviewerin ‚etwas zum Besten' gibt, dass sie Fragen, Nachfragen, Be- und Anmerkungen, deutliche Zustimmung, kleine Geschichten, ja sogar gelegentlich einmal verhaltenen Widerspruch formuliert, dass sie ihr sachliches Engagement bekundet und sich als lern- und wissbegierig zeigt, stimuliert sie ihr Gegenüber so gut wie mit keiner anderen Interviewtechnik dazu, ‚aus sich herauszugehen', sozusagen ‚existenzielles' Interesse am Thema zu entwickeln [...]."
>
> (Honer 2011, S. 48)

Daran schließt sich eine zweite Phase der Interviewführung an, in der auf spezielle Interviewtechniken, z. B. des biographischen Interviews oder des Experteninterviews zurückgegriffen wird. Diese kann „entweder deutlich (durch zeitliche Distanz und oder explizite kommunikative ‚Markierungen') von der quasi-natürlichen Gesprächsführung abgesetzt oder in einer Art von fließendem Übergang mit der ersten Phase verwoben bzw. aus ihr heraus entwickelt werden" (ebd., S. 49).

Das Experteninterview eignet sich besonders dazu, einen Einblick in das Deutungswissen und Prozesswissen der Befragten zu erlangen. Nach Bogner (2014, S. 18 f.) umfasst das Deutungswissen „die subjektiven Relevanzen, Sichtweisen, Interpretationen, Deutungen, Sinnentwürfe und Erklärungsmuster der Befragten“ sowie normative Dispositionen, wie Zielsetzungen und Bewertungen. Das Prozesswissen umfasst „Einsicht in Handlungsabläufe, Interaktionen, organisationale Konstellationen, Ereignisse usw., in die die Befragten involviert sind oder waren“ (ebd.). Da beide Wissensformen für diese Arbeit relevant sind, wird die Interviewphase in Anlehnung an das Experteninterview gestaltet. Laut Honer (2011, S. 51) handelt es sich dabei um ein situationsflexibel gehandhabtes Leitfadeninterview, in dem die befragende Person ihren Fragenkatalog lediglich als Angebot für ein Gespräch versteht. In der abschließenden dritten Phase des explorativen Interviews werden Fragen, die in den ersten beiden Interviewphasen unbeantwortet blieben bzw. durch sie aufgekommen sind, exploriert (ebd. S. 52 f.).

Die Interviews, die im Rahmen dieser Arbeit ausgewertet werden, wurden ab Juli 2018 über das Schuljahr 2018/19 hinweg im Rahmen von Schulbesuchen und, bis auf einige Ausnahmen, gleichzeitig mit beiden am Projekt *AlgoKids* beteiligten Lehrkräften der jeweiligen Projektschule durchgeführt (siehe Abbildung 8.4; wann welches Interview geführt wurde, kann in Kapitel A.1 eingesehen werden). Im Sinne einer erzählungsgenerierenden Frage (vgl. Bogner 2014, S. 62) wurden die Lehrkräfte zunächst aufgefordert, über ihre Erfahrungen innerhalb des Projekts *AlgoKids* zu berichten. Im Laufe des Gesprächs wurden Kärtchen mit verschiedenen Piktogrammen als Impulsmaterial auf dem Interviewtisch verteilt, welche die Lehrkräfte als Gesprächsanlass nutzen konnten. Auf einen Interviewleitfaden im konventionellen Sinn wurde verzichtet, um keinen zu großen Bruch in der Gesprächsatmosphäre herbeizuführen. Geführt wurden sämtliche Interviews von der Autorin, die den Lehrkräften bereits als Kursleiterin der Fortbildungsveranstaltungen und Ansprechpartnerin zu allen Fragen bezüglich des Projekts bekannt war. Alle Lehrkräfte hatten im Erhebungszeitraum bereits eigene Erfahrungen in der unterrichtlichen Umsetzung der Fortbildungsinhalte gesammelt. Insgesamt wurden 19 Interviews mit insgesamt 32 Lehrkräften geführt, deren Dauer zwischen 22 und 113 Minuten lag (siehe Tabelle 8.3[4]). Im Durchschnitt dauerte ein Interview 55 Minuten. Die Interviews wurden aufgezeichnet und wörtlich transkribiert. Die Transkripte werden im weiteren Verlauf der Arbeit im Rahmen verschiedener Teilstudien zu unterschiedlichen Forschungsfragen ausgewertet. Eine detaillierte Beschreibung des explorativen Interviews kann in Kapitel A.2 eingesehen werden.

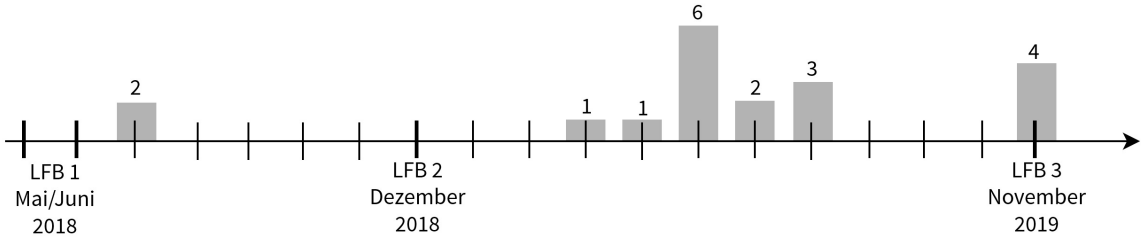



*Abbildung 8.4 Übersicht über die Anzahl und Verteilung der Interviews in Relation zu den Fortbildungsveranstaltungen*

[4]Der Interviewcode setzt sich aus dem Pseudonym und der Angabe $YYYYMMTT zusammen.

*Tabelle 8.3 Übersicht über quantitative Daten zu den Interviews*

| Nr. | Pseudonym | Anzahl Personen Interview (Anzahl Lehrkräfte im Projekt) | Interviewdauer (MM:SS) | Interviewcode |
|---|---|---|---|---|
| 1 | Schule1-1 | 2(2) | 28:56 | S1-1_20190515 |
| 2 | Schule2-1 | 2(2) | 53:04 | S2-1_20190508 |
| 3 | Schule3-1 | 1(2) | 60:49 | S3-1_20191124 |
| 4 | Schule3-2 | 1(2) | 22:36 | S3-1_20191124 |
| 5 | Schule4-1 | 2(2) | 85:05 | S4-1_20190409 |
| 6 | Schule5-1 | 2(2) | 82:30 | S5-1_20190515 |
| 7 | Schule6-1 | 2(2) | 60:23 | S6-1_20190527 |
| 8 | Schule7-1 | 2(2) | 41:10 | S7-1_20190524 |
| 9 | Schule8-1 | 1(2) | 33:47 | S8-1_20191130 |
| 10 | Schule8-2 | 1(2) | 24:01 | S8-2_20191130 |
| 11 | Schule9-1 | 1(2) | 40:50 | S9-1_20190718 |
| 12 | Schule10-1 | 2(2) | 56:04 | S10-1_20190607 |
| 13 | Schule11-1 | 2(2) | 113:38 | S11-1_20190705 |
| 14 | Schule12-1 | 2(2) | 69:56 | S12-1_20190627 |
| 15 | Schule13-1 | 2(2) | 38:18 | S13-1_20190716 |
| 16 | Schule14-1 | 2(2) | 36:24 | S14-1_20190312 |
| 17 | Schule15-1 | 1(1) | 50:53 | S15-1_20190529 |
| 18 | Schule16-1 | 2(2) | 68:34 | S16-1_20180724 |
| 19 | Schule17-1 | 2(2) | 92:35 | S17-1_20180723 |

### 8.5.3 Unterrichtsmaterialien

Um zu untersuchen, inwiefern die Lehrkräfte die Unterrichtssequenz bzw. die darin verwendeten Materialien verändern oder weiterentwickeln, wurden sie gebeten, der Autorin ihre selbst erstellten Unterrichtsmaterialien zur Verfügung zu stellen. Bereits in der ersten Fortbildungsveranstaltung wurden sie aufgefordert, ihre Materialien – sofern sie digital erstellt wurden – per Mail an die Autorin zu senden. Im Vorfeld der zweiten Fortbildungsveranstaltung wurden die Lehrkräfte per Mail gebeten, ggf. erstellte Materialien zu der Veranstaltung mitzubringen. Um die Materialien zu ordnen, wurde für jede Projektschule eine beschriftete Mappe ausgelegt, in der die Unterrichtsmaterialien abgelegt werden konnten. Diese wurden während der Veranstaltung fotografiert und konnten von den Lehrkräften wieder mitgenommen werden. Dieses Vorgehen wurde im Rahmen der dritten Fortbildungsveranstaltung wiederholt. Zusätzlich wurden an jede Mappe beschriftete USB-Sticks angebracht, auf denen digitale Unterrichtsmaterialien oder digitale Vorlagen für Unterrichtsmaterialien abgespeichert werden konnten. Zusätzlich konnten Materialien im Rahmen von Schulbesuchen an die Autorin übergeben werden bzw. wurden von ihr abfotografiert. Insgesamt liegen der Autorin 71 verschiedene Unterrichtsmaterialien im Bildformat vor, die sich teilweise über mehrere Seiten erstrecken – dabei handelt es sich um Arbeitsblätter, Arbeitsaufträge, Ablaufplanungen, Scratch-Programme und Fotos von Unterrichtssituationen. Die Gesamtmenge der Dateien beträgt 118.

## 8.6 Datenauswertung

Für die Analyse der gewonnen Daten werden verschiedene Methoden verwendet, die nachfolgend vorgestellt werden. Wie die Methoden in den einzelnen Teilstudien umgesetzt werden, wird in Teil IV dieser Arbeit erläutert.

### 8.6.1 Qualitative Inhaltsanalyse

Die im Rahmen dieser Arbeit erhobenen qualitativen Daten werden mithilfe der qualitativen Inhaltsanalyse ausgewertet, einer Analyseform zur Systematisierung von Daten mit dem Ziel einer möglichst regelgeleiteten Interpretation (Mayring 2010, S. 602). Zentral dafür sind Kategoriensysteme, deren Kategorien genau definiert sind und nach bestimmten Regeln Textstellen zugeordnet werden. Dabei ist ein erweiterter Textbegriff anzusetzen, der nicht nur primär schriftliche Erzeugnisse, sondern auch Bild-, Audio- und Videomaterial einschließt (Stamann, Janssen und Schreier 2016). Um die Reliabilität einer qualitativen Inhaltsanalyse zu gewährleisten, kann geprüft werden, ob das Kategoriensystem bei wiederholter Anwendung auf dasselbe Material zu gleichen Ergebnissen führt und sich somit als zuverlässig erweist. Die *Interrater*-Reliabilität wird überprüft, indem mindestens ein Ausschnitt des Materials von einem zweiten Codierer bzw. einer zweiten Codiererin recodiert wird (Mayring 2010, S. 604). Ein statistisches Maß zur Berechnung der *Interrater*-Reliabilität ist Cohens Kappa. Um die relative Stärke der Übereinstimmung zwischen zwei Codierern zu beschreiben, werden die Werte des Koeffizienten folgender Nomenklatur zugeordnet: <0.00 „*Poor*“, 0.00 - 0.20 „*Slight*“, 0.21 - 0.40 „*Fair*“, 0.41 - 0.60 „*Moderate*“, 0.61 - 0.80 „*Substantial*“ und 0.81 - 1.00 „*Almost Perfect*“ (Landis und Koch 1977, S. 165).

Es gibt verschiedene Formen der qualitativen Inhaltsanalyse, die je nach Ziel der Untersuchung und Datenmaterial Anwendung finden. In dieser Arbeit wird die inhaltlich strukturierende qualitative Inhaltsanalyse nach Kuckartz angewandt (2016, S. 97 ff.). Es handelt sich dabei um eine deduktiv-induktive Vorgehensweise, die folgende Schritte umfasst (ebd., S. 101 ff.):

*Phase 1: Initiierende Textarbeit, Markieren wichtiger Textstellen und Schreiben von Memos*
Sorgfältiges Lesen des Textes, Markieren von wichtig erscheinenden Passagen und Festhalten von Anmerkungen sowie Auswertungsideen in Form von Memos.

*Phase 2: Entwickeln von thematischen Hauptkategorien*
Die Hauptthemen können direkt am Material entwickelt oder aus einem theoretischen Bezugsrahmen, der Forschungsfrage oder dem Leitfaden der Studie abgeleitet werden.

*Phase 3: Erster Codierprozess: Codieren des gesamten Materials mit den Hauptkategorien*
Relevante Textstellen werden den Hauptkategorien zugewiesen. Dabei kann eine Textstelle mit mehreren Kategorien codiert werden. Die Größe der Codiereinheit sollte so gewählt werden, dass das codierte Segment außerhalb des Kontextes noch verständlich ist.

*Phase 4: Zusammenstellen aller mit der gleichen Kategorie codierten Textstellen und*

*Phase 5: Induktives Bestimmen von Subkategorien am Material*

Die Kategorien, die für die Studie von zentraler Bedeutung sind, werden in Subkategorien ausdifferenziert. Dafür werden alle mit der jeweiligen Kategorie codierten Textstellen zusammengestellt und Subkategorien am Material gebildet. Diese werden im Anschluss definiert und durch Zitate aus dem Material illustriert.

*Phase 6: Zweiter Codierprozess: Codieren des kompletten Materials mit den ausdifferenzierten Kategorien*

Die ausdifferenzierten Kategorien werden den mit den Hauptkategorien codierten Textstellen zugeordnet. Dies erfordert einen erneuten Durchgang durch das codierte Material.

*Phase 7: Einfache und komplexe Analysen, Visualisierungen*

Die eigentliche Analyse findet statt – dabei lassen sich verschiedene Auswertungsformen unterscheiden: (1) Kategorienbasierte Auswertung der Hauptkategorien, (2) Zusammenhänge der Subkategorien innerhalb einer Hauptkategorie, (3) Zusammenhänge zwischen Hauptkategorien, (4) Kreuztabellen (qualitativ und quantifizierend), (5) Konfigurationen von Kategorien und (6) Visualisierung von Zusammenhängen.

### 8.6.2 Deskriptive Statistik

Zur Auswertung der Fragebögen wird zunächst auf Kennwerte der deskriptiven Statistik zurückgegriffen: das arithmetische Mittel und die Standardabweichung. Das arithmetische Mittel $\overline{x}$ bzw. der Mittelwert gibt als Maß der zentralen Tendenz den Durchschnitt aller Messergebnisse wieder und berechnet sich durch die Summe aller Werte dividiert durch deren Anzahl $n$ (Rasch u. a. 2014a, S. 16):

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

Die Standardabweichung bzw. Streuung $s$[5] gibt als Dispersionsmaß an, wie stark die Messwerte um den Mittelwert streuen. Zur Berechnung wird in einem ersten Schritt die Varianz bestimmt, die sich aus der Summe der quadrierten Abweichungen aller Messwerte vom arithmetischen Mittel, dividiert durch die Anzahl aller Messwerte minus eins berechnet (ebd., S. 20):

$$s^2 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1}$$

Zur Bestimmung der Standardabweichung wird in einem zweiten Schritt die Wurzel aus der Varianz gezogen (ebd., S. 21):

$$s = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1}}$$

[5] Im Rahmen der Teilstudien werden die im Englischen üblichen Abkürzungen $M$ für den Mittelwert (*Mean*) und $SD$ für die Standardabweichung (*Standard Deviation*) verwendet.

Zur Berechnung der statistischen Kennwerte werden die Angaben innerhalb der Fragebögen Zahlenwerten zugeordnet, so entspricht z. B. die Angabe „trifft überhaupt nicht zu“ dem Wert eins. Anschließend erfolgt die Berechnung des Mittelwerts und der Standardabweichung für jedes einzelne Item sowie der summierten Items einer Skala. Für letzteres müssen invers formulierte Items im Vorfeld invertiert werden.

### 8.6.3 Inferenzstatistik

Innerhalb der Fragebögen werden verschiedene Skalen zu mehreren Messzeitpunkten erhoben. Um die errechneten Mittelwerte miteinander zu vergleichen, werden zwei Methoden der Inferenzstatistik eingesetzt: der $t$–Test für abhängige Stichproben und die einfaktorielle Varianzanalyse mit Messwiederholung.

Mithilfe des $t$-Tests lässt sich feststellen, ob sich die Mittelwerte zweier Gruppen systematisch voneinander unterscheiden. Es lässt sich also feststellen „ob zwei betrachtete Gruppen in einem untersuchten Merkmal wirklich einen Unterschied aufweisen oder nicht“ (Rasch u. a. 2014a, S. 43). Da im Untersuchungsdesign der vorliegenden Arbeit Messwerte derselben Personen zu unterschiedlichen Zeitpunkten verglichen werden, wird der t-Test für abhängige Stichproben verwendet. Für dessen Einsatz müssen verschiedene Voraussetzungen erfüllt sein (Bortz und Schuster 2010, S. 125): (1) Die Differenzwerte der Messwertpaare sollten normalverteilt sein und (2) die untersuchte Variable ist intervallskaliert.

Eine Erweiterung des $t$-Tests für abhängige Stichproben stellt die einfaktorielle Varianzanalyse mit Messwiederholung dar. Sie untersucht ebenfalls, ob sich die Ausprägung eines Merkmals zu verschiedenen Messzeitpunkten unterscheidet, ermöglicht jedoch den Vergleich beliebig vieler Messzeitpunkte (Rasch u. a. 2014b, S. 100). Für den Einsatz müssen folgende Voraussetzungen erfüllt sein (ebd., S. 107 f.): (1) die untersuchte Variable ist intervallskaliert, (2) die Residuen der abhängigen Variable sind innerhalb jedes Messzeitpunktes normalverteilt und (3) Sphärizität ist gegeben, d. h. die Varianzen zwischen den einzelnen Gruppen sind gleich. Wenn ein signifikanter Unterschied zwischen den verschiedenen Messzeitpunkten gefunden werden konnte, wird im Anschluss geprüft, zwischen welchen Zeitpunkten dieser besteht. Zu diesem Zweck wird der $Bonferroni$ Post-hoc Test angewandt, der mehrere $t$-Tests für abhängige Stichproben ohne eine Kumulierung des Alpha-Fehlers berechnet (Schwarz 2022).

Teil III

# THEORETISCHE GRUNDLAGEN

# 9 Fachwissenschaftliche Grundlagen

## 9.1 Algorithmen

Algorithmen sind einer der ältesten (abstrakten) Beschreibungstechniken für Abläufe. Bereits in der Antike kannte man informelle Beschreibungstechniken für eine Vielzahl von Rechenverfahren, z. B. das Ermitteln des größten gemeinsamen Teilers zweier natürlicher Zahlen (Hubwieser, Mühling und Aiglstorfer 2013, S. 15). Das Wort Algorithmus leitet sich von dem Namen des Universalgelehrten, Mathematiker und Astronom Mohammed Al-Khwarizmi ab: Sein Werk zur Arithmetik wurde Anfang des 12. Jahrhunderts ins Lateinische übersetzt und trug den Titel *Algorizmi dixit* (Strick 2009). Informell beschriebene Algorithmen lassen sich in Form von Gebrauchsanweisungen, Kochrezepten, Wegbeschreibungen etc. auch im Alltag finden. Es handelt sich dabei jedoch um nur unpräzise beschriebene Algorithmen (Broy 1998, S. 31). Stellt man einen Algorithmus in einer Programmiersprache dar – einer Sprache, die von einem Rechner automatisch interpretiert werden kann – spricht man von einem Programm.

### 9.1.1 Der Begriff „Algorithmus“

In der Literatur lassen sich viele Definitionen des Begriffs *Algorithmus* finden, die zum Teil unterschiedliche Aspekte hervorheben:

> „Ein Algorithmus ist ein Verfahren mit einer präzisen (d.h. in einer eindeutigen Sprache abgefaßten) endlichen Beschreibung unter Verwendung effektiver (im Sinne von tatsächlich ausführbarer) Verarbeitungsschritte.“ (Broy 1998, S. 31)

> „Ein Algorithmus ist ein Verfahren zur Lösung eines Problems. [...] Die Beschreibung eines Algorithmus besteht aus einzelnen Schritten und Vorschriften, die die Ausführung dieser Schritte kontrollieren. Jeder Schritt muss klar und eindeutig beschrieben sein und mit endlichem Aufwand in endlicher Zeit ausführbar sein.“
>
> (Güting und Dieker 2018, S. 33)

> „Der Begriff Algorithmus bezeichnet eine Vorschrift zur Lösung eines Problems, die für eine Realisierung in Form eines Programms auf einem Computer geeignet ist.“
>
> (Ehses 2012, S. 171)

Broy macht selbst darauf aufmerksam, dass seine Definition maßgeblich von der Auslegung der verwendeten Begriffe abhängt und somit nicht exakt ist. Auch Güting und Dieker weisen darauf hin, dass ihre Definition nicht festlegt, wann ein Schritt genügend klar beschrieben ist. Ehses umgeht

dieses Problem, indem er in seiner Definition lediglich darauf verweist, dass ein Algorithmus sich zur Darstellung in einer Programmiersprache eignet, die per Definition immer präzise genug ist (siehe Abschnitt 9.2).

### 9.1.2 Grundelemente von Algorithmen

Die Vielfalt an möglichen Algorithmen ist zwar nahezu grenzenlos, die darin vorkommenden Strukturen hingegen sind sehr überschaubar. Alle Algorithmen setzen sich aus wenigen strukturellen Bausteinen und deren Kombinationen zusammen (vgl. Hubwieser, Mühling und Aiglstorfer 2013, S. 12):

- Elementare Verarbeitungsschritte
  Unteilbare Verarbeitungsschritte (auch genannt Anweisungen) werden unbedingt ausgeführt:
  *Verarbeitungsschritt*

- Sequenzen
  Mehrere elementare Verarbeitungsschritte, die hintereinander ausgeführt werden, bilden eine Sequenz. Die einzelnen Anweisungen einer Sequenz werden mit einem festen Zeichen, wie einem Strichpunkt und/oder einem Zeilenumbruch, getrennt und der Reihe nach ausgeführt:
  *Verarbeitungsschritt 1*
  *Verarbeitungsschritt 2*
  *...*
  *Verarbeitungsschritt n*

- Bedingte Verarbeitungsschritte
  Es gibt Fälle, in denen manche Verarbeitungsschritte nur unter einer bestimmten Bedingung ausgeführt werden sollen. Zusätzlich kann man einen alternativen Verarbeitungsweg angeben, der ausgeführt wird, falls die Bedingung nicht erfüllt ist:
  `Falls` *Bedingung* `Dann`
  *Sequenz 1*
  `Sonst`
  *Sequenz 2*
  `Ende Falls`

- Wiederholung
  Oftmals müssen ein oder mehrere Verarbeitungsschritte wiederholt ausgeführt werden. Falls die Anzahl der Wiederholungen bereits vor dem ersten Durchlauf bekannt ist, wird die Wiederholung mit vorgegebener Wiederholungszahl (oder auch Wiederholung mit fester Anzahl) angewandt:
  `Wiederhole` *n* `Mal`
  *Sequenz*
  `Ende Wiederhole`

Ergibt sich die Anzahl der Wiederholungen erst im Laufe der einzelnen Wiederholungen, greift man auf die bedingte Wiederholung zurück. Die Verarbeitungsschritte werden wiederholt, bis eine Bedingung erfüllt ist:

```
Wiederhole solange Bedingung
Sequenz
Ende Wiederhole
```

- Variablen
  Algorithmen operieren auf Werten, die in Form von Variablen gespeichert werden (Schneider 2012, S. 174). Sie bezeichnen Speicherplätze, in denen Werte abgelegt werden können und die während der Programmausführung verschiedene Werte annehmen können (ebd., S. 174). Jede Variable wird durch einen Bezeichner identifiziert und kann genau einen Wert einer bestimmten Sorte enthalten, z. B. eine ganze Zahl, eine Gleitkommazahl oder einen Text (Hubwieser u. a. 2013a, S. 88).

### 9.1.3 Darstellungsformen für Algorithmen

Ein Algorithmus kann neben der Formulierung in einer Programmiersprache auf viele andere Arten dargestellt werden. So gibt es verschiedene grafische und textuelle Darstellungen, die sich am Kontrollfluss eines Algorithmus orientieren und dessen logische Struktur in den Vordergrund stellen. Zu den grafischen Darstellungen gehören bspw. der Programmablaufplan oder das Struktogramm. Beide Notationsformen stellen die Struktur eines Algorithmus mithilfe von festgelegten Symbolen bzw. Blöcken dar (siehe Abbildung 9.1 und 9.2[1]). Beide Darstellungsformen sind jedoch nicht gut dazu geeignet, komplexere Algorithmen übersichtlich darzustellen.

Eine Darstellungsform, die sich an der natürlichen Sprache orientiert, ist der sogenannten Pseudocode (siehe Tabelle 9.1). Im Gegensatz zu den grafischen Darstellungsformen gibt es keine Normierung, wie Pseudocode auszusehen hat. Je nach gewähltem Abstraktionsniveau kann er sich mehr an der natürlichen Sprache oder einer Programmiersprache orientieren. Natürlichsprachliche Beschreibungen haben den Vorteil, dass sie für gewöhnlich von allen Beteiligten beherrscht und verstanden werden. Allerdings bergen sie das Risiko, dass sie nicht ausreichend präzise und strukturiert genug sind (vgl. Broy und Malkis 2019, S. 22).

*Tabelle 9.1 Anweisung in Alltagssprache und in Pseudocode*

| Alltagssprache | Pseudocode |
|---|---|
| Zeichne ein Quadrat | Wiederhole 4 Mal<br>  Zeichne eine Linie<br>  Drehe Stift um 90°<br>Ende Wiederhole |

[1] Beide Abbildungen wurden von der Autorin in Anlehnung an die Programmiersprache Robot Karol (Freiberger 2018) erstellt.

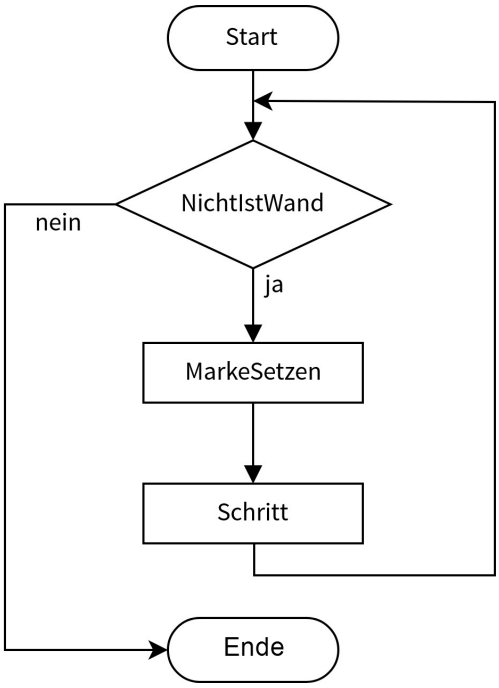



*Abbildung 9.1 Beispiel für einen Programmablaufplan*


*Abbildung 9.2 Beispiel für ein Struktogramm*

## 9.2 Programmiersprachen

Mithilfe von Programmiersprachen ist es möglich, Algorithmen so präzise darzustellen, dass Informatiksysteme sie direkt oder indirekt ausführen können. Letzteres bedeutet, dass der entsprechende Quellcode der Programmiersprache zunächst maschinell in ein direkt ausführbares Programm bzw. Maschinenbefehle übersetzt werden muss. Diese Darstellung eines Algorithmus durch Texte und/oder visuelle Elemente in Programmiersprachen bezeichnet man als Programme. Mittlerweile wurden mehr als 1000 Programmiersprachen entwickelt – welche davon zum Einsatz kommt, hängt maßgeblich vom jeweiligen Einsatzzweck und Kontext ab.

### 9.2.1 Syntax und Semantik

Alle Programmiersprachen zeichnen sich, wie natürliche Sprachen, durch eine bestimmte Syntax und Semantik aus (vgl. Broy 1998, S. 77 ff.). Die Syntax einer Programmiersprache bezieht sich auf deren äußere Form und legt fest, wie korrekt formulierte Programme aussehen. Auch wenn jede Programmiersprache eine eigene Syntax besitzt, gibt es durchaus syntaktische Ähnlichkeiten zwischen unterschiedlichen Sprachen. Im Gegensatz zur Syntax beschreibt die Semantik die Bedeutung der Programmtexte, z. B. durch die Festlegung des Ein-Ausgabeverhaltens eines Programms. Hubwieser, Mühling und Aiglstorfer (2013, S. 21) weisen darauf hin, dass Informatikerinnen und Informatiker in der Lage sein müssen, mit verschiedenen Programmiersprachen umzugehen oder sich diese zumindest schnell anzueignen. Dies ist möglich, da sich die semantischen Sprachkonstrukte verschiedener Programmiersprachen ähneln:

> „[Es gibt] nur einige wenige Sprachkonzepte, die man bei der Programmierung verwenden kann. Diese kommen in praktisch allen Programmiersprachen vor und funktionieren immer ähnlich (das ist die Semantik) – allerdings unterscheiden sie sich in der Art wir [sic] man sie aufschreiben muss (also der Syntax) zum Teil erheblich."
>
> (Hubwieser, Mühling und Aiglstorfer 2013, S. 24)

### 9.2.2 Visuelle Programmiersprachen

Die meisten Programmiersprachen beschränken sich zur Darstellung von Algorithmen auf die Verwendung von Text. Neben diesen textuellen Programmiersprachen gibt es visuelle Programmiersprachen, die zur Erstellung von Programmen vorwiegend grafische Notationen verwenden (vgl. Burnett 2001, S. 275). Das Ziel der visuellen Programmierung ist in erster Linie, die Verständlichkeit von Programmen zu erhöhen und Programmierung selbst zu erleichtern. Beispielsweise sollen damit auch Anwenderinnen und Anwender Applikationen erstellen können, die normalerweise von professionellem Personal programmiert werden müssten.

Schiffer (1999, S. 621) definiert den Begriff *visuelle Programmiersprache* als „eine visuelle Sprache zur vollständigen Beschreibung der Eigenschaften von Software." Der Begriff *visuell* bedeutet laut Duden "den Gesichtssinn betreffend, durch den Gesichtssinn vermittelt"[2] und bezieht sich somit auf die Wahrnehmung durch das Sehen. Schiffer führt weiter aus, dass zwar auch Text durch das Sehen erfasst wird und somit die Eigenschaft *visuell* besitzen kann, laut Kognitionspsychologie jedoch primär vom verbalen Wahrnehmungssystem – nicht vom visuellen – verarbeitet wird. In einer visuellen Sprache hingegen kann mindestens eine signifikante Information ausschließlich über das visuelle Wahrnehmungssystem gewonnen werden. Das heißt, visuelle Programmiersprachen können zwar auch Text enthalten, man kann sie jedoch nur vollumfänglich verstehen, wenn man sie sieht (siehe Abbildung 9.3).

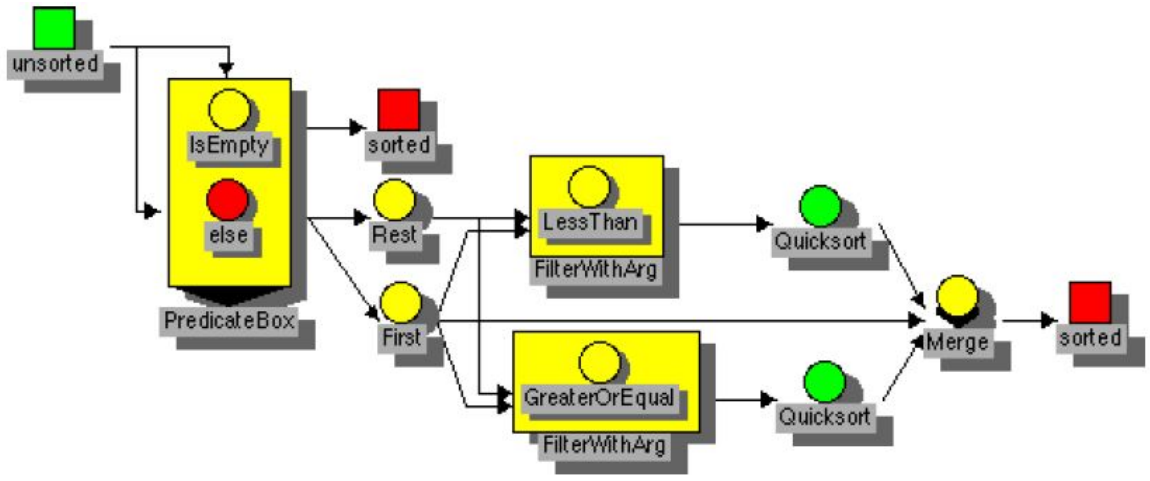



*Abbildung 9.3 Ein Programm der visuellen Programmiersprache VisaVis aus Schiffer (1998, S. 45)*

[2] https://www.duden.de/rechtschreibung/visuell

Eine Besonderheit von visuellen Programmiersprachen ist, dass sie fast immer auf ein bestimmtes visuelles Programmiersystem (VP-System) abgestimmt und damit eng verflochten sind (vgl. Schiffer 1998, S. 132). Die Programmelemente werden in dem VP-System interaktiv ausgewählt und automatisch in eine interne Repräsentation überführt. Die Interaktionsmechanismen des Systems bilden dabei in der Regel einen Teil der Sprache, was dazu führt, dass keine Trennung zwischen Sprache und Werkzeug möglich ist. Da die Regeln für gültige Kombinationen der Elemente im jeweiligen System hinterlegt sind, können dabei ausschließlich syntaktisch gültige Kombinationen entstehen.

Es wurden verschiedene Klassifikationen visueller Programmiersprachen vorgelegt, die den Fokus jeweils auf unterschiedliche Aspekte legen, z. B. den Anwendungsbereich oder Visualisierungsgrad einer Sprache (z. B. Burnett 2001; Boshernitsan und Downes 2004). Für die vorliegende Arbeit ist die Kategorie der steuerflussorientierten VP-Systeme relevant, die auf dem imperativen Programmierparadigma[3] beruhen. Der Steuerfluss kann darin auf verschiedene Arten festgelegt werden: mit Komponentennetzen, Transitionsnetzen und Anweisungssequenzen[4] (vgl. Schiffer 1998, S. 132 f.). Eine Anweisungssequenz bezeichnet eine Folge von Operationen, die linear ausgeführt werden, insofern die Ausführung nicht durch Unterprogrammaufrufe oder Kontrollstrukturen, wie Wiederholungen oder bedingte Anweisungen, modifiziert wurde. Sie entspricht somit dem Quelltext klassischer textueller Programmiersprachen, die Notation erfolgt jedoch in Form von Ablaufdiagrammen oder Block-Elementen (sieh Abbildung 9.4)

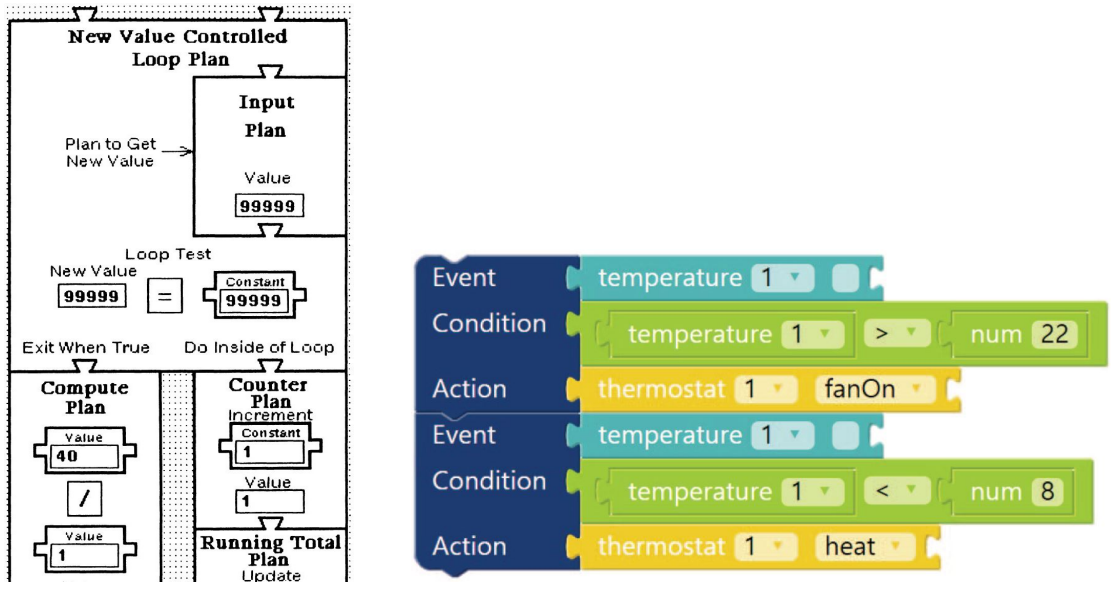



*Abbildung 9.4 Beispiele für Programme in der visuellen Programmiersprache (a) BridgeTalk (Bonar und Liffick 1987, S. 35) und (b) Smart Block (Bak, Chang und Choi 2018, S. 34)*

[3] Nach dem Paradigma der imperativen Programmierung werden Programme als Folgen von Anweisungen formuliert, deren Ausführungsreihenfolge sich aus der Reihung im Programmtext oder durch das Verwenden von Sprunganweisungen ergibt (Pomberger 1999, S. 517).

[4] Für den weiteren Verlauf der Arbeit ist ausschließlich der Steuerfluss mit Anweisungssequenzen relevant, weshalb nur darauf eingegangen wird.

## 9.3 Das Programmieren

Unter Programmieren versteht man sämtliche Vorgänge, die mit dem Aufstellen eines Algorithmus zur Lösung eines Problems verbunden sind, sowie die Beschreibung des Algorithmus in Code, also einer Programmiersprache. Dies hat zum Ziel, dass ein Informatiksystem die Schritte des Algorithmus ausführen kann. Zentral ist dabei, dass alle Anweisungen im Voraus gegeben und dann ausgeführt werden:

> *„A key point about programming is that all instructions are given in advance and then executed, which requires the programmer to anticipate how the program will be used after it has been handed over to the user.“* (Duncan, Bell und Atlas 2017, S. 68)

Der Begriff *Programmierung* wurde bereits in den 1960er Jahren verwendet, bezog sich im Wesentlichen aber ausschließlich auf die Tätigkeit, einen Computer zu codieren (Wirth 2008, S. 32), also das Überführen eines Algorithmus in eine Programmiersprache. Der Duden Informatik definiert *Programmierung* wie folgt und macht deutlich, dass der Begriff weitaus mehr umfasst:

> „Unter Programmieren versteht man zum einen den Vorgang der Programmerstellung und zum anderen das Teilgebiet der Informatik, das die Methoden und Denkweisen beim Entwickeln von Programmen umfasst.“ (Claus und Schwill 2006, S. 529)

In der öffentlichen Wahrnehmung hat sich der Begriff *Coding* als Synonym für das Programmieren verbreitet. Der Nachteil dieser Formulierung ist, dass sie suggeriert, es würde sich dabei lediglich um den Vorgang des Codeschreibens handeln (vgl. Duncan, Bell und Tanimoto 2014, S. 62).

### 9.3.1 Methoden der Programmierung

Zu den Methoden der Programmierung gehört zum Beispiel laut Broy (vgl. 1998, S. 85 f.) das Durchlaufen mehrere Teilaufgaben:

- Spezifikation
  Die Aufgabe wird genau festgelegt, dazu gehört das Erfassen aller relevanten Grundbegriffe, Strukturen und Zusammenhänge (Systemanalyse) sowie der Problemstellung (Anforderungsdefinition) und eventueller Nebenbedingungen.
- Planung der Vorgehensweise
  Auch wenn die Anforderungsdefinition nur beschreibt was implementiert werden soll und nicht wie, sollten erste Überlegungen zur Machbarkeit angestellt werden. Diese sollten sich auf die technische Machbarkeit, aber auch auf die vorhandenen Ressourcen beziehen. Gegebenenfalls wird direkt ein Arbeitsplan aufgestellt.

- Strukturierung
  Die Aufgabenstellung wird in Teilaufgaben unterteilt, z. B. das Festlegen der Rechenstrukturen sowie der Funktionen von Programmeinheiten oder die Auswahl von Datenstrukturen und Algorithmen.
- Dokumentation
  Die Programmeinheiten werden exakt dokumentiert, unter anderem durch eine Beschreibung ihrer Funktion und die Angabe der Verwendungsform der Parameter.

Wie und in welcher Reihenfolge diese Aufgaben durchgeführt werden, wird in verschiedenen Vorgehensmodellen spezifiziert, die den organisatorischen Ablauf von Projekten in generalisierter Form regeln (vgl. Goll 2011, S. 68 f.).

### 9.3.2 Modelle und Modellierung

Bei der Entwicklung von Programmen geht es vorrangig darum, Probleme der realen Welt zu lösen. Um diese Probleme mit Mitteln der Informatik bearbeiten zu können, müssen sie inklusive aller relevanten Zusammenhänge des Anwendungsgebietes mit Mitteln der Informatik modelliert werden (vgl. Broy und Malkis 2019, S. 12). Dabei zentral ist der Begriff des *Modells*, unter dem man laut Duden ein Objekt oder Gebilde versteht, welches die inneren Beziehungen und Funktionen von etwas abbildet bzw. veranschaulicht und vereinfacht[5]. Im Kontext der Informatik definieren Hubwieser, Mühling und Aiglstorfer den Begriff wie folgt:

> „Ein Modell ist eine abstrahierte Beschreibung eines realen oder geplanten Systems, das die für eine bestimmte Zielsetzung wesentlichen Eigenschaften des Systems erhält. Die Erstellung einer solchen Beschreibung heißt Modellbildung oder Modellierung [...]." (Hubwieser, Mühling und Aiglstorfer 2013, S. 4)

Das Objekt der Modellierung ist in dieser Definition das *System*, was die Autoren nach Wedekind u. a. (1998, S. 268) als „eine Menge von Elementen (Systembestandteilen), die durch bestimmte Ordnungsbeziehungen miteinander verbunden und durch klar definierte Grenzen von ihrer Umwelt geschieden sind" definieren. Broy und Malkis unterscheiden zwei Klassen von Modellen, die im Rahmen der Programmierung relevant sind (2019, S. 19) – das Domänenmodell und das System-, Software-, Programmmodell. Ersteres bildet einen Ausschnitt der realen oder gedachten Welt des jeweiligen Anwendungsgebietes ab und dient als Grundlage der Spezifikation und Lösung des gestellten Problems. System-, Software-, Programmmodelle hingegen modellieren die Struktur oder das Verhalten des jeweiligen Systems, das es zu entwickeln gilt. Modellierung findet demnach in verschiedenen Phasen der Programmierung statt. Für beide Klassen gibt es unterschiedliche Modellierungsmethoden und Notationsformen (z. B. Balzert und Liggesmeyer 2011; Goll 2011). Gemein ist allen, dass sie Komplexität reduzieren und so einen wichtigen Schritt der Problemlösung darstellen.

[5] https://www.duden.de/rechtschreibung/Modell

# 10 Entwicklungspsychologische Grundlagen

Der Begriff *Entwicklung* bezieht sich laut Trautner (1992) auf „die Beschreibung menschlichen Verhaltens und Erlebens [...] unter dem Aspekt ihrer Veränderung über die Zeit“ (S.16). Im Zentrum der Entwicklungspsychologie stehen demnach intraindividuelle Veränderungen des Verhaltens und Erlebens, darüber hinaus jedoch auch interindividuelle Unterschiede bezüglich dieser Veränderungen sowie die Analyse der Veränderungen in Bezug zum jeweiligen Umfeld (Lohaus und Vierhaus 2019, S. 5). Zur Aufgabe der Entwicklungspsychologie gehören laut Montada, Lindenberger und Schneider (2018, S. 38) die Erklärung und Beschreibung von Entwicklungsveränderungen zur Bestimmung und Prognose des aktuellen bzw. zukünftigen Entwicklungsstandes sowie das Aufzeigen von Möglichkeiten zur Beeinflussung des Entwicklungsverlaufs. Im Kontext der Grundschule gibt die Entwicklungspsychologie bspw. Aufschluss darüber, welche Fähigkeiten in einem bestimmten Alter erwartet werden können und wie die Entwicklung der Kinder gefördert werden kann. Die für die vorliegende Arbeit relevanten Theorien und Befunde werden im Folgenden aufgeteilt in die Bereiche *Kognition*, *Verarbeitung von Informationen* und *Lern- und Leistungsmotivation* beschrieben.

## 10.1 Kognition

Der Biologe Piaget fand in zahlreichen Studien heraus, dass sich die Denkweisen und intellektuellen Fähigkeiten von Kindern verschiedenen Alters unterscheiden und entwickelte daraus sein *Stufenmodell zur kognitiven Entwicklung* (2003, S. 63 ff.). Nach dieser Theorie befinden sich Kinder der Grundschule auf der konkret-operationalen Stufe (7. bis 11. Lebensjahr), die von Lohaus und Vierhaus (2019) wie folgt beschrieben wird:

> „In der konkret-operationalen Entwicklungsstufe findet eine Ablösung der Denkoperationen von den beobachteten Abläufen statt, aber die Denkoperationen sind noch immer auf konkrete Handlungen und Wahrnehmungen bezogen und die Abstraktionsfähigkeit ist dementsprechend noch immer gering. Die Fähigkeit zur Perspektivübernahme entwickelt sich, sodass zunehmend die Wünsche und Intentionen anderer Personen berücksichtigt werden können. Die Perspektivübernahmefähigkeit bleibt jedoch auf konkrete Personen bezogen und bezieht sich beispielsweise noch nicht auf abstrakte Perspektiven (wie beispielsweise die gesamtgesellschaftliche Perspektive). [...] Es entstehen weiterhin zunehmende Kompetenzen zur Planung von Handlungsabläufen und zur Koordinierung von Handlungen [...].“
>
> (Lohaus und Vierhaus 2019, S. 32 f.)

Die formal-operationale Entwicklungsstufe, welche sich durch die Fähigkeit zum abstrakten und systematischen Denken nach formal-logischen Regeln kennzeichnet, wird nach Piaget erst im Alter von 12 Jahren erreicht (Piaget 2003, S. 15). In Bezug auf die Altersangaben macht Hoppe-Graff (2014, S. 156) jedoch darauf aufmerksam, dass Piaget selbst mehrfach große individuelle Unterschiede im Entwicklungstempo von Kindern erwähnt. Weiter führt Hoppe-Graff aus, dass die Stufen zwar unterschiedlich schnell durchlaufen werden können, die Abfolge jedoch nicht variiert (ebd.). Auch Lohaus und Vierhaus (2019, S. 34) weisen darauf hin, dass Piaget die Kompetenzen von Kindern in vielen Inhaltsbereichen unterschätzt habe: „Vor allem durch die Verwendung vereinfachter Aufgaben oder noch sorgfältigerer Beobachtung des Entwicklungsgeschehens konnte vielfach gezeigt werden, dass zumindest die Altersangaben eher zu hoch angesetzt sind“ (ebd.). Neben seinem Stufenmodell betont Piagets Werk den Stellenwert aktiven Lernens, in dem Kinder die Freiheit haben, Sachverhalte zu erforschen und zu hinterfragen. Lehrkräfte sollten zudem visuelle oder haptische Hilfsmittel bereitstellen, um die Schülerinnen und Schüler in ihren Denkprozessen zu unterstützen. Er empfiehlt außerdem, im Unterricht vertraute Beispiele zu verwenden, um komplexe oder abstrakte Ideen einzuführen (vgl. Piaget 2003, S. 73 ff.). Zu ähnliche Schlussfolgerungen kommen Lohaus und Vierhaus (2019, S. 145) in Hinblick auf das schlussfolgernde Denken von Kindern, welches sich auf das induktive oder deduktive Ableiten neuen Wissens aus gegebenen Informationen bezieht:

> „[Es wäre] unangemessen zu behaupten, dass jüngere Kinder grundsätzlich weniger logisch denken als ältere Kinder oder Erwachsene. Grundlegende Fähigkeiten sowohl zum induktiven als auch zum deduktiven Denken bestehen bereits sehr früh. Die Leistungen, die Kinder diesbezüglich zeigen können, beruhen jedoch wie bereits erwähnt sehr stark auf dem dargebotenen Aufgabenmaterial und auf den Anforderungen an andere kognitive Fähigkeiten (z. B. Gedächtniskapazitäten). Stellt man jüngeren Kindern Aufgaben, die sich auf Konzepte beziehen, die ihnen bekannt sind und konkret dargeboten werden, sind sie grundsätzlich in der Lage, logische Schlussfolgerungen zu ziehen.“ (Lohaus und Vierhaus 2019, S. 145)

Der Psychologe Vygotsky (1978) erweitert diese subjektbezogenen Ansätze um den Aspekt der sozialen Interaktion und stellt dessen zentrale Bedeutung für den kognitiven Entwicklungsprozess heraus. Sein Konzept der *Zone der proximalen Entwicklung* beschreibt die Differenz zwischen dem aktuellen Entwicklungsstand eines Kindes, der durch die Fähigkeit des eigenständigen Problemlösens bestimmt wird, und dem potentiellen Entwicklungsstand, der mit der Hilfe und Unterstützung anderer, erfahreneren Personen erreicht werden kann. Schumacher und Denner fassen die pädagogischen Implikationen des Konzepts wie folgt zusammen:

> „[Der Prozess der kognitiven Entwicklung] setzt voraus, dass jedes Kind zum Lernen einen anderen Menschen braucht, der ihn hinsichtlich seines Denken und Handelns auf die jeweils nächste Stufe seiner Entwicklung führt. Diese Aufgabe vermag ein anderes Kind oder auch ein Erwachsener zu leisten, sofern das Zustandekommen von

> Intersubjektivität, z. B. in Form von wechselseitiger Aufmerksamkeit oder Referenz, gewährleistet ist." (Schumacher und Denner 2017, S. 199)

## 10.2 Verarbeitung von Informationen

Die Entwicklungspsychologen Bruner und Harttung (1973) beschäftigen sich ebenfalls mit der kognitiven Entwicklung von Kindern und legen einen Fokus dabei auf die Repräsentationsformen, in denen Informationen dargeboten werden können. Sie postulieren, dass Kinder nahezu jeden Lerngegenstand erfassen können, wenn dieser entsprechend aufbereitet wird:

> „Untersuchungen der intellektuellen Entwicklung von Kindern rücken ins Licht, daß das Kind auf jeder Entwicklungsstufe eine charakteristische Art und Weise hat, die Welt zu betrachten und für sich selbst zu erklären. Ein Kind bestimmten Alters in einem Lehrgegenstand zu unterrichten bedeutet, die Struktur dieses Gegenstandes in der Art und Weise darzustellen, wie das Kind Dinge betrachtet."
>
> (Bruner und Harttung 1973, S. 44)

Sie beschreiben die kognitive Entwicklung als einen spiralförmigen Prozess, in dem sich die Lernenden mit dem Lerngegenstand in verschiedenen Repräsentationsformen und Abstraktionsstufen auseinandersetzen (vgl. Bruner 1971, S. 27 ff.): auf konkrete Art und Weise durch eigenständig ausgeführte Handlungen (*enaktive Repräsentation*), durch graphische und bildliche Darstellungen (*ikonische Repräsentation*) und zuletzt durch Sprache und Zeichen mit besonderer Bedeutung (*symbolische Repräsentation*). Lernprozesse vollziehen sich dabei wesentlich bei den Übergängen zu anderen Repräsentationsformen (siehe Abbildung 10.1) – dabei sind nicht nur die Übergänge vom Konkreten zum Abstrakten, sondern auch der rückläufige Prozess der Konkretisierung miteingeschlossen (Zech 2002, S. 106 f.).

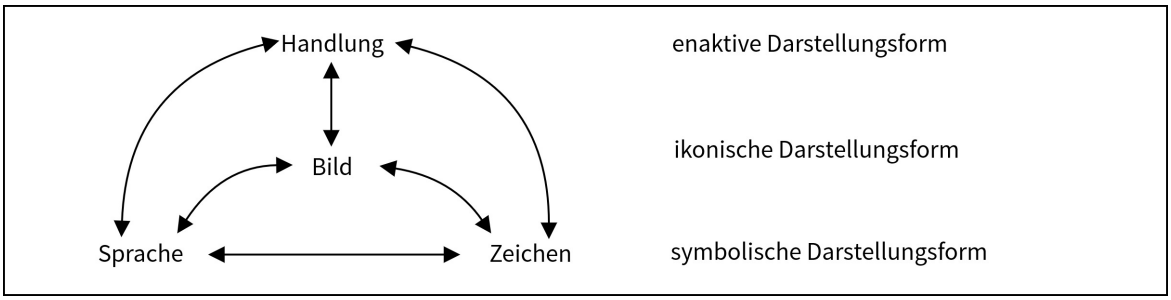



*Abbildung 10.1 Repräsentationsmodi nach Bruner aus Zech (2002, S. 106)*

Auch neuere Ansätze der kognitiven Entwicklung von Kindern nehmen zunehmend Bezug auf Theorien, die sich mit der Beschreibung der Informationsverarbeitung durch das menschliche kognitive System befassen (vgl. Lohaus und Vierhaus 2019, S. 34). Seitens der Informationsverarbeitungstheorien sind hier vorrangig Ansätze zu nennen, die sich auf die mentale Repräsentation von Informationen sowie die Kapazität des Arbeitsgedächtnisses beziehen. Mayers *Cognitive Theory of Multimedia Learning* bezieht sich bspw. vorrangig auf die optimierte Verknüpfung von Text-

und Bildrepräsentationen von Lerninhalten (vgl. Mayer 2001). Text umfasst in diesem Kontext sowohl geschriebenen als auch gesprochenen Text, und Bilder können in Form von Illustrationen, Fotos, Animationen oder Videos dargestellt sein. Mayers Theorie basiert wie bereits die Theorie der Dualen Kodierung von Paivio (1986) auf der Annahme, dass die Aufnahme von Informationen über voneinander getrennte Sinneskanäle (visuell und akustisch) erfolgt (siehe Abbildung 10.2). Die wahrgenommen Reize können im jeweiligen Kanal verarbeitet werden oder durch kognitive Prozesse in den jeweils anderen Kanal überführt werden. Aus dem sensorischen Gedächtnis werden selektierte Inhalte in das Arbeitsgedächtnis überführt, wo ein entsprechendes visuelles bzw. sprachliches mentales Modell konstruiert wird. Im letzten Schritt der Verarbeitung integriert der Lernende die beiden Modelle in bestehende Wissensstrukturen (Seufert und Brünken 2018, S. 576).

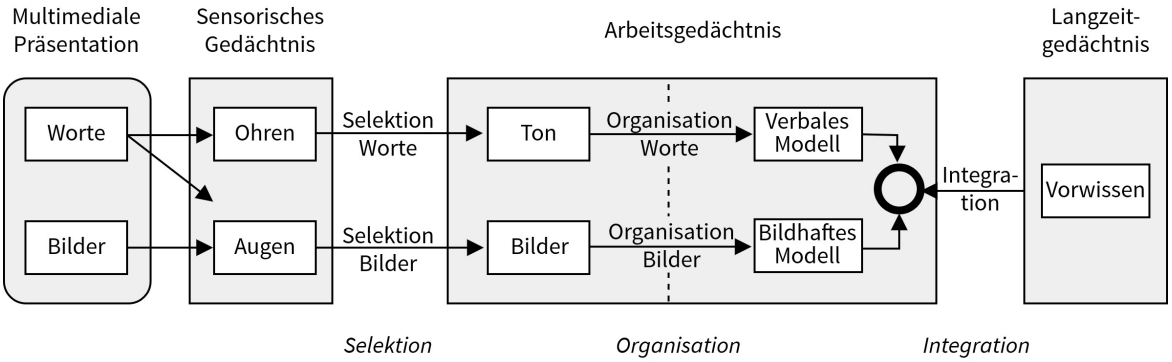



*Abbildung 10.2 Veranschaulichung des multimedialen Lernens aus Mayer (2001, S. 47)*

Laut Mayer (2001) werden Lernprozesse begünstigt, wenn Inhalte nicht nur in Textform vorliegen, sondern zusätzlich durch bildhafte Darstellungen ergänzt werden. Dieser sogenannte *Multimedia-Effekt* erklärt sich durch die bereits beschriebene duale Kodierung der Informationen im Arbeitsgedächtnis. Basierend auf seiner Theorie formuliert er verschiedene Prinzipien, die den Lernprozess begünstigen sollen: *Signaling Principle* (wichtige Stellen im Lernmaterialien hervorheben und Bezüge zwischen Bild und Text herstellen), *Coherence Principle* (nur Informationen berücksichtigen, die zum Lernen des Sachverhalts nötig sind), *Spatial Contiguity Principle* (Text und Bild möglichst in räumlicher und zeitlicher Nähe präsentieren), *Segmenting Principle* (Informationen in verständliche Abschnitte unterteilen), *Multimedia Principle* (Informationen in mehr als einer Repräsentationsform darlegen, aber Redundanz vermeiden) und *Personalization Principle* (Lernende direkt ansprechen).

Eine weitere Möglichkeit, den Lernprozess zu begünstigen, ist eine effizientere Nutzung der vorhandenen Kapazität des Arbeitsgedächtnisses. Ein wichtiger Ansatz hierbei ist die *Cognitive Load Theory* von Chandler und Sweller (1991), die auf der Annahme basiert, dass es unterschiedliche Quellen kognitiver Belastung gibt. Dabei unterscheiden sie drei Formen: *intrinsic*, *extraneous*, und *germane cognitive load*. *Intrinsic cognitive load* entsteht durch die inhärente Schwierigkeit des zu erlernenden Sachverhalts; *extraneous cognitive load* bezieht sich auf die kognitive Belastung, die durch instruktionale Begebenheiten und die Darstellung der Informationen verursacht wird; *germane cognitive load* entsteht durch die vertiefte Auseinandersetzung des Lernenden mit dem Lerninhalt und bezieht sich auf die Anstrengung, die dabei aufgebracht werden muss, den Inhalt zu

verstehen. Es wird angenommen, dass das Arbeitsgedächtnis aufgrund seiner begrenzten Kapazität nur eine bestimmte Menge an Informationen gleichzeitig verarbeiten kann und der Lernprozess daher unterbrochen wird, sobald diese Kapazität erschöpft ist – es kommt zu einem *cognitive overload* (vgl. De Jong 2010). Ein Ansatzpunkt um dies zu verhindern, ist die Verminderung des *extraneous cognitive load*:

> *„Many learning and problem-solving procedures encouraged by instructional formats result in students engaging in cognitive activities far removed from the ostensible goals of the task. The cognitive load generated by these irrelevant activities can impede skill acquisition."* (Chandler und Sweller 1991, S. 294)

Lernprozesse sollten so gestaltet werden, dass „neben dem unvermeidbaren *intrinsic cognitive load* der *extraneous cognitive load* gering genug ist, um freie Ressourcen für den *germane cognitive load* zu ermöglichen" (Seufert und Brünken 2018, S. 578). Sweller, van Merriënboer und Paas (1998, S. 262 f.) beschreiben verschiedene Effekte, die eine Verminderung des *extraneous cognitive load* zur Folge haben. Bspw. beschreibt der *Worked Example Effect*, wie sich kognitive Belastung durch den Einsatz von Schritt-für-Schritt-Demonstrationen zur Lösung eines Problems reduzieren lässt. Der *Completion Problem Effect* besagt darüber hinaus, dass sich die kognitive Belastung durch das Vorgeben von Teillösungen mindern lässt, die von den Lernenden lediglich vervollständigt werden müssen.

Die genannten Ansätze der Informationsverarbeitungstheorie, die auf eine Entlastung des Arbeitsgedächtnisses abzielen, sind aus entwicklungspsychologischer Sicht besonders deshalb relevant, da Kinder erst im Laufe ihrer Entwicklung fähig sind, ihre kognitiven Kapazitäten effizient auszunutzen (vgl. Lohaus und Vierhaus 2019, S. 38 ff.). Bspw. zeigt sich, dass Kinder mit steigendem Alter den Arbeitsspeicher des Kurzzeitgedächtnisses effizienter nutzen können, sowie Informationen zunehmend automatisiert und auch schneller verarbeiten können.

## 10.3 Lern- und Leistungsmotivation

Die Lern- und Leistungsmotivation stellt eine wichtige Voraussetzung für die Leistungsfähigkeit des Menschen dar und ist aus mehreren Gründen bei der Gestaltung von Lernprozessen in der Schule zu berücksichtigen (vgl. Lohrmann und Hartinger 2014, S. 276 f.). Zum einen ist es aus normativer Sicht erstrebenswert, in der Schule positive Emotionen zu evozieren, zum anderen zeigen empirische Befunde die Bedeutsamkeit von Motivation für den Lernprozess (vgl. Krapp und Hascher 2014b, S. 269 ff.).

Das *ARCS-Modell* von Keller (1987) unterscheidet vier Hauptbereiche, die die Lernmotivation wesentlich beeinflussen: *Attention* (Aufmerksamkeit erlangen), *Relevance* (Bedeutsamkeit des Lehrstoffes vermitteln), *Confidence* (Erfolgszuversicht) und *Satisfaction* (Zufriedenheit). Für jeden Bereich führt

*Tabelle 10.1 Ausgewählte Strategien zur Steigerung der Motivation im Unterricht aus Keller (1987, S. 4 f.)*

| *Components of the ARCS-Model* | *Strategies* |
|---|---|
| *Attention* | *- Show visual representations of any important object or set of ideas or relationships.*<br>*- Give examples of every instructionally important concept or principle.*<br>*- Vary the medium of instruction (platform delivery, film, video, print, etc.)*<br>*- Shift between student-instructor interaction and student-student interaction.*<br>*- Use humorous introductions.*<br>*- Build in problem solving activities at regular intervals.*<br>*- Give learners the opportunity to select topics, projects and assignments that appeal to their curiosity and need to explore.*<br>*- Use games, role plays, or simulations that require learners participation.* |
| *Relevance* | *- Use analogies familiar to the learner from past experiences.*<br>*- State explicitly how the instruction relates to future activities of the learner.*<br>*- Provide personal choices for organizing one's work.* |
| *Confidence* | *- Incorporate clearly stated, appealing learning goals into constructional materials.*<br>*- Organize materials on an increasing level of difficulty; that is, structure the learning material to provide a „conquerable" challenge.*<br>*- Help students set realistic goals.*<br>*- Allow students opportunity to become increasingly independent in learning and practicing a skill.* |
| *Satisfaction* | *- Allow a student to use a newly acquired skill in a realistic setting as soon as possible.*<br>*- Verbally reinforce a student's intrinsic pride in accomplishing a difficult task.*<br>*- Allow a student who masters a task to help others who have not yet done so.*<br>*- Provide informative, helpful feedback when it is immediately useful.*<br>*- Avoid surveillance (as opposed to positive attention)* |

Keller weitere Unterkategorien sowie Strategien auf, um die Motivation von Schülerinnen und Schülern gezielt zu fördern (siehe Tabelle 10.1). Zu seinem Modell merkt er an, dass es nicht für die Lösung individueller Probleme von Schülerinnen und Schülern konzipiert wurde, sondern sich auf das Unterrichtsgeschehen als Ganzes bezieht:

> *„There can be motivational challenges that differ from situation to situation. [...] However, the assumption is that the group as a whole will be responsive if an effective set of motivational strategies is employed. The ARCS Model, as presently constituted, is not designed as a behavioral change model; that is, it is not intended for use in solving individual personality problems or in teaching students how to be self-motivated."*
>
> (Keller 1987, S. 6)

Viele konkrete Vorschläge zur Förderung der Motivation im Unterricht beziehen sich außerdem auf die *Selbstbestimmungstheorie der Motivation* von Deci und Ryan (1993). Im Unterschied zum *ARCS-Modell* stehen hier „angeborene psychologische Bedürfnisse und grundlegende Fähigkeiten und Interessen des Individuums" im Zentrum (ebd., S. 223). Deci und Ryan proklamieren drei

grundlegende psychologische Bedürfnisse, die für den Aufbau und Erhalt intrinsischer Motivation grundlegend sind (vgl. Krapp und Hascher 2014a, S. 247 f.): die Bedürfnisse nach Empfinden von *Selbstbestimmung*, *Kompetenz* und *sozialer Eingebundenheit*. Lohrmann und Hartinger formulieren, wie diese Bedürfnisse zur Förderung der Motivation im Unterricht aufgegriffen werden können:

> „Günstig für das Empfinden von *Selbstbestimmung* sind z. B. Formen der Öffnung von Unterricht in inhaltlicher und/oder methodischer Hinsicht (vgl. Hartinger und Fölling-Albers 2002, S. 140 ff.). Sinnvoll sind auch der Verzicht auf unerbetene Hilfestellungen sowie Rückmeldungen, die eher informierend als kontrollierend wahrgenommen werden. Das *Kompetenzerleben* der Schüler(innen) kann durch realisierbare, kurzfristige Arbeitsziele sowie durch wertschätzende Rückmeldungen unterstützt werden. [...] *Soziale Eingebundenheit* lässt sich z. B. durch (sinnvoll konzipierte) Gruppenarbeiten, durch Austausch- und Unterstützungsmöglichkeiten in der Klasse sowie durch persönliche Anteilnahme von Seiten der Lehrperson fördern."
>
> (Lohrmann und Hartinger 2014, S. 277 f.)

# 11 Grundschulpädagogische und -didaktische Grundlagen

Die Grundschulpädagogik wird von Schorch (2007, S. 13) als eine Bereichsdisziplin der Schulpädagogik bezeichnet, die auf den Erfahrungsraum *Grundschule* begrenzt ist und die Entstehung, Funktion und Aufgaben der Schulform thematisiert. Während sich die Grundschulpädagogik vorrangig auf die Entwicklung einer „Theorie umfassender Grundbildung und einer Theorie der Persönlichkeitsentwicklung in der Kindheit" befasst (Einsiedler 2015, S. 57), stehen bei der Grundschuldidaktik Inhalte und Methoden des Unterrichts im Zentrum (Schorch 2007, S. 18). Schumacher und Denner schlagen ein interdisziplinäres Verständnis der Grundschulpädagogik und -didaktik vor, in dem diese als „sich wechselseitig stützende und ergänzende Funktionen und Aufgaben" zu sehen sind (2017, S. 44). Diesem Verständnis folgend werden die grundschulpädagogischen und -didaktischen Grundlagen der vorliegenden Arbeit gemeinsam dargestellt. Die Aufteilung orientiert sich dabei an den Zielhorizonten der Grundschulbildung, die von der KMK in ihren *Empfehlungen zur Arbeit in der Grundschule* genannt und beschrieben werden (2015, S. 9 ff.): *Lernprozesse und kompetenzorientiertes Lernen*, *Lernen und Leisten*, *Fördern und Fordern* sowie *Individualisierung und Differenzierung*.

## 11.1 Lernprozesse und kompetenzorientiertes Lernen

Der Kompetenzerwerb in der Grundschule zielt laut KMK (2015) darauf ab, „die motivationalen, volitionalen und sozialen Bereitschaften in konkreten Anwendungssituationen nutzen zu können und selbst zu individuellen Kompetenzen auszubilden. Ziel ist eine umfassende Persönlichkeitsbildung, die sich in der erfolgreichen und verantwortungsvollen Bewältigung aktueller Anforderungssituationen zeigt" (KMK 2015, S. 9). In Bezug auf die Lernprozesse in der Grundschule fordert die KMK eine Ausgewogenheit von Instruktion und Konstruktion sowie die Berücksichtigung des sozialen Charakters des Lernens:

> „Die Grundschule stellt eine pädagogische Balance her zwischen gesteuerten Bildungsprozessen und eigenaktiven Konstruktionsprozessen von Kindern. Für diese individuellen Prozesse stellt der Unterricht Lerngelegenheiten bereit, die individuell wie kommunikativ ausgerichtet sind. Somit ist Lernen immer auch ein sozialer Prozess. Von der Qualität der professionellen Unterstützung hängt ab, wie der Kompetenzerwerb der Kinder gelingt. [...] Im sozialen Miteinander der Gruppe lernen Kinder zunehmend selbstgesteuert sowie aktiv konstruierend und reflektieren ihren Lernprozess. Kooperative Lernprozesse werden in der Grundschule systematisch erweitert."
>
> (KMK 2015, S. 9)

### 11.1.1 Kompetenzbegriff

Das Konzept der *Kompetenz* hat in den letzten Jahren vor allem durch groß angelegte Schulleistungsstudien wie TIMSS, PISA oder IGLU an Aufmerksamkeit gewonnen. In deren Kontext werden die erfassten Schülerleistungen als Ausdruck spezifischer Kompetenzen betrachtet, bspw. Lesekompetenz, mathematische Kompetenz oder naturwissenschaftliche Kompetenz (Hartig und Klieme 2006, S. 128). Die in Deutschland gängige Definition von Kompetenz, auf die sich viele Bildungsstandards und Kompetenzmodelle beziehen ist jene von Weinert (2001b). Danach versteht man unter Kompetenzen „die bei Individuen verfügbaren oder durch sie erlernbaren kognitiven Fähigkeiten und Fertigkeiten, um bestimmte Probleme zu lösen, sowie die damit verbundenen motivationalen, volitionalen und sozialen Bereitschaften und Fähigkeiten, um die Problemlösungen in variablen Situationen erfolgreich und verantwortungsvoll nutzen zu können“ (ebd., S. 27 f.). Klieme u. a. (2003, S. 72) führen weiter aus, dass Kompetenzen nach diesem Verständnis Personen befähigen „bestimmte Arten von Problemen erfolgreich zu lösen, also konkrete Anforderungssituationen eines bestimmten Typs zu bewältigen“. Kompetenzen können folglich nur handelnd gezeigt werden. Die Bestimmung von Kompetenzen sollte daher nie nur normativ, sondern auch empirisch erfolgen (vgl. Schaper 2008, S. 98 f.).

Der Begriff *Kompetenz* wird umgangssprachlich und im wissenschaftlichen Kontext vielfältig verwendet. Schaper (2008, S. 92 f.) unterscheidet verschiedene Arten eines psychologisch fundierten Kompetenzverständnisses:

- *Kompetenz als Persönlichkeitsmerkmal:* Kompetenz wird vor allem als persönlichkeitsbezogene Eigenschaft verstanden, z. B. Intelligenz oder Kreativität;
- *Kompetenz als Eignungsmerkmal:* Kompetenzen als Beschreibungen relativ allgemeiner situationsübergreifender Leistungsvoraussetzungen;
- *Kompetenz als Wissens und Fähigkeitsvoraussetzung:* Kompetenz als Summe bestimmter tätigkeitsrelevanter Qualifikationen, z. B. Kenntnissen, Fertigkeiten und Fähigkeiten;
- *Kompetenz als Fähigkeit zum situationsangemessenen Verhalten:* Kompetenz als situations- und anforderungsgerechtes Verhalten;
- *Kompetenz als Befähigung zur handelnden Bewältigung komplexer Anforderungssituationen:* Kompetenzen sind als kognitive, sozial-kommunikative und emotional-motivationale Voraussetzungen zur Bewältigung komplexer Aufgabenstellungen zu verstehen;
- *Kompetenz als Expertise:* Kompetenz als besondere Wissens- und Handlungsformen, die durch intensive Erfahrung und Übung erworben wurden;
- *Kompetenz als Selbstorganisationsdisposition:* Kompetenzen befähigen Menschen, neuartige Situationen lernend und problemlösend zu bewältigen;
- *Kompetenz als biographisches Konstrukt:* Kompetenzen als Fähigkeiten, die durch bestimmte Entwicklungsanforderungen erworben wurden;

### 11.1.2 Instruktion und Konstruktion

Während sich die Grundüberzeugungen der unterschiedlichen Lerntheorien – Behaviorismus, Kognitivismus und Konstruktivismus – viele Jahrzehnte konträr gegenüberstanden (vgl. Reinmann und Mandl 2006, S. 617 ff.), werden heutzutage vermehrt Ansätze vorgestellt, in denen Wechsel zwischen einer primär aktiven und rezeptiven Position der Lernenden sowie einer reaktiven und aktiven Position der Lehrpersonen stattfinden:

> „Konstruktion und Instruktion lassen sich nicht nach einem Alles-oder-nichts-Prinzip realisieren. Lernen erfordert zum einen immer Motivation, Interesse und Eigenaktivität seitens der Lernenden, und der Unterricht hat die Aufgabe, ihre Konstruktionsleistungen anzuregen und zu ermöglichen. Lernen erfordert zum anderen aber auch Orientierung, Anleitung und Hilfe. Ziel muss es folglich sein, eine Balance zwischen expliziter und konstruktiver Aktivität des Lernenden zu finden (Linn 1990)."
>
> (Reinmann und Mandl 2006, S. 639)

Einsiedler weist ebenfalls darauf hin, dass „auch Instruktion bzw. lehrerzentrierter Unterricht interne konstruktive Prozesse zu fördern hat und dass auch Konstruktion bzw. selbstgesteuertes Lernen instruktionaler Hilfen bedarf" (2014, S. 362). Auch aus Ergebnissen der empirischen Unterrichtsforschung im Bereich der Grundschule resultiert, dass eine Kombination aus Methoden beider Unterrichtsausprägungen angebracht ist (ebd.). Als Beispiele für eine Lernform des Grundschulunterrichts, die eine aktive Rolle der Schülerinnen und Schüler betont, ist das entdeckende Lernen zu nennen. Charakteristisch für das entdeckende Lernen ist, dass sich die Kinder „aktiv mit den Sachverhalten des Unterrichts beschäftigen, (mehr oder weniger) selbstständig Probleme lösen und so neue kognitive Strukturen aufbauen" (Hartinger und Lohrmann 2014, S. 385). Bezüglich der Umsetzung im Unterricht der Grundschule schreiben Rehle u. a., dass das Ausmaß der Entdeckungen variieren kann:

> „[...] ob die Schüler in weitgehend selbstbestimmten Lernsituationen die Lerninhalte und Lösungswege selbst wählen oder ob sie mit präzisen Angaben, Lernhilfen und Materialien Teilziele ‚gelenkt entdecken', ist abhängig von den Voraussetzungen der Kinder und den Zielsetzungen des Unterrichts." (Rehle u. a. 2017, S. 161)

Auch Hartinger und Lohrmann (2014, S. 386) weisen darauf hin, dass entdeckendes Lernen in verschiedenen Lehrmethoden realisiert werden kann, und dabei keinesfalls auf überwiegend selbstbestimmte Formen des Unterrichts beschränkt ist. Vielmehr entscheidet die Lehrkraft, inwieweit der Entdeckungsprozess durch Lernhilfen gelenkt werden soll:

> „Solche Lernhilfen können sich auf die *Strukturierung des Lernmaterials* beziehen, also z. B. auf die Auswahl von Phänomenen (bzw. Texten/Beispielen/Materialien/Situationen), die das Entdecken einer Erkenntnis (z. B. einer Regelhaftigkeit) eher provozieren als andere. Lernhilfen können sich auch auf die *Gesprächsführung* im Unterricht selbst beziehen. Ein Beispiel wären Impulse, welche die Aufmerksamkeit der Lernenden auf bestimmte Aspekte lenken." (Hartinger und Lohrmann 2014, S. 387)

Neber und Neuhaus (2018, S. 121 ff.) unterscheiden drei Versionen des entdeckenden Lernens – durch *Konfliktinduktion*, *Beispiele* sowie durch *Experimentieren und Design*:

- Konfliktinduktion
  Im Mittelpunkt des Unterrichts bzw. einzelner Phasen stehen kognitive Konflikte, die durch widersprüchliche, falsche oder verfremdete Behauptungen oder die Darbietung entsprechender Phänomene bewusst herbeigeführt werden. Die Lernenden reagieren zunächst mit Erklärungen, oder Vermutungen, die sich jedoch als falsch herausstellen. Diese Konflikterfahrung löst wiederum Fragen nach der fehlenden Information hervor, was zur Motivierung des Wissenserwerbs beiträgt. Entdeckendes Lernen durch Konfliktinduktion eignet sich besonders für Unterrichtseinstiege, lässt sich jedoch auch in späteres Unterrichtsphasen, z. B. durch konfliktinduzierende Lernaufgaben realisieren.
- Beispiele
  Die Auseinandersetzung mit ausgewählten Beispielen ermöglicht abstrahierende, wissensgenerierende Prozesse über Begriffe oder Prozeduren. Zum Erwerb von Wissen über letzteres eignen sich *expositorische* Beispiele, die eine Fragestellung und deren Lösungsprozedur vorgeben. Diese *worked examples* können zudem Fragen und Impulse enthalten, die den Lernprozess zusätzlich unterstützen.
- Experimentieren und Design
  Die Schülerinnen und Schüler lösen in Lernaufgaben gestellte Probleme und generieren dabei fachspezifisches Wissen. Im entdeckenden Lernen durch Experimentieren, welches besonders im naturwissenschaftlichen Unterricht verbreitet ist, werden Beispiele in Anlehnung an den Prozess naturwissenschaftlicher Erkenntnisgewinnung selbst konstruiert und durch weiteres reflektieren verallgemeinert. Es konnte mehrfach nachgewiesen, dass besonders die Phase der Hypothesenbildung entscheidend für das resultierende Wissen ist und demnach nicht vernachlässigt werden sollte. Während beim entdeckenden Lernen durch Experimentieren gut definierte Probleme mit einer einzigen richtigen Lösung bzw. Lösungsmethode bearbeitet werden, wird beim entdeckenden Lernen durch Design mit schlecht definierten Problemen gearbeitet, die ein ganzes Spektrum an möglichen Lösungen und Lösungswegen ermöglichen.

Im Rahmen des entdeckenden Lernens stellt das Üben einen integralen Bestandteil des Lernprozesses dar und trägt einerseits zur Sicherung des Gelernten bei, andererseits birgt wiederum jede Übung das Potenzial, „etwas zu entdecken“ (Maras und Ametsbichler 2014, S. 346). Die Bedeutung des Übens für den Unterricht wird in der Literatur vielfach thematisiert. Hier wird das Üben zum einen im Sinne einer Wiederholung als nochmalige Ausführung einer inneren oder äußeren Handlung in unveränderter bzw. sinngemäßer Form beschrieben (*repetitives Üben*) (vgl. Riedl 2010, S. 58), zum anderen wird das *elaborierte Üben* erläutert, welches auf Transfer angelegt ist (vgl. Helmke 2017, S. 202). Letzteres zielt durch die Bezugnahme auf unterschiedliche Anwendungsbeispiele auf eine bessere Verknüpfung mit dem Vorwissen und eine flexible Anwendung des Wissens und Könnens auf neue Situation und Aufgaben ab.

### 11.1.3 Kindgemäßheit

Nach deutschem Recht gilt als Kind, wer das 14. Lebensjahr noch nicht erreicht hat – die Grundschulen in Deutschland werden demnach ausschließlich von Kindern besucht und sind als kindgemäße Bildungseinrichtungen zu verstehen. Der grundschulpädagogische Grundsatz der Kindgemäßheit ist ein normativer Begriff, der jedoch erheblich vom aktuellen wissenschaftlichen und gesellschaftlichen Diskurs zum Kindheitsbegriff abhängig ist und somit als Konstrukt im Wandel gesehen werden muss (Heinzel 2014, S. 155). Nichtsdestotrotz haben sich verschiedene Grundsätze für die Gestaltung des Unterrichts etabliert, die das Prinzip der Kindgemäßheit konkretisieren und auch im weiteren Verlauf der Arbeit berücksichtigt werden (vgl. Schorch 2007, S. 225).

Das Prinzip der Veranschaulich heißt laut Schröder, „den Unterrichtsstoff so darbieten, dass die Schüler ihn mithilfe ihrer Sinnesorgane und entsprechend ihrer Auffassungsfähigkeit umfassend und zutreffend erkennen können“ (Schröder 2000, S. 167). Wiater (2018) weist darauf hin, dass es sich dabei nicht um eine bloße Rezeption gehe und es nicht genüge den Schülerinnen und Schülern einen Lerngegenstand sinnlich wahrnehmbar zu machen:

> „Einen Gegenstand oder Sachverhalt veranschaulichen heißt, bei jemanden eine Anschauung von ihm zu ermöglichen. Ihn durch ein mediales Arrangement sinnlich wahrnehmbar zu machen, reicht als Veranschaulichung allein nicht aus. Wird er nämlich auf solche Weise präsentiert, erleichtert das zunächst nur seine Wahrnehmung durch die Sinnesorgane. Damit es wirklich zu einer Anschauung über den Sachverhalt beim Menschen kommt, bedarf es einer inneren, denkenden Verarbeitung des Wahrgenommenen, der Bildung einer Vorstellung von ihm.“ (Wiater 2018, S. 62)

Die Veranschaulichung im Unterricht zeichnet sich durch Sachbezogenheit aus, d. h. sie erfolgt dadurch, „dass man den Gegenstand, die Sache selbst in den Mittelpunkt des Unterrichts stellt, von ihr ausgeht und die notwendigen Erklärungen im ständigen Bezug zur Sache gibt“ (Schröder 2000, S. 169). Die *Sache* kann dabei auf verschiedenen Stufen der Konkretisierung bzw. Abstraktion eingebunden werden, z. B. indem man einen Gegenstand in seiner natürlichen Umgebung oder im Klassenzimmer betrachtet und untersucht, durch Präparate, Modelle oder bildhafte Darstellungen (ebd., S. 170 f.).

Schröder (2000, S. 174) weist darauf hin, dass das Prinzip der Veranschaulichung keinesfalls so interpretiert werden solle, die Kinder möglichst passiv zu halten und die Aufgabe der Lehrkraft darin zu sehen, im Unterricht möglichst viel zu demonstrieren und präsentieren. Sie weist in diesem Zuge auf das Unterrichtsprinzip der Aktivierung hin, dass sich darauf bezieht, „den Schüler anzuregen und ihm die Möglichkeit zu geben, im tätigen Umgang mit den Dingen Lernerfahrungen zu erwerben“ (Schröder 2000, S. 174). Die Umsetzung des Prinzips im Unterricht sieht Schröder in einer hohen Eigenaktivität der Lernenden und einer Zurücknahme der Steuerung durch die Lehrkraft:

> „In der unterrichtlichen Gestaltung führt dies dazu, dass man dem Schüler nicht nur die Möglichkeit gibt, tätig zu sein, sondern auch selbständig nach richtigen Lösungen zu suchen, auch dann, wenn der Lehrer weiß, dass der eingeschlagene Weg nicht zum Ziele führen wird. Offensichtliche Fehlversuche werden nicht vorzeitig korrigiert, um dem Schüler zu ermöglichen, am Sachverhalt Konsequenzen des einen oder anderen Weges zu erfahren." (Schröder 2000, S. 174)

### 11.1.4 Grundlegende Denk- und Arbeitsweisen anbahnen

Neben dem Erwerb von domänenspezifischen Kompetenzen ist eine zentrale Aufgabe der Grundschule der Erwerb fächerübergreifender Kompetenzen sowie das Anbahnen grundlegender Denk- und Arbeitsweisen (vgl. Schorch 2007, S. 134 ff.). Aufgrund der Relevanz für die vorliegende Arbeit wird im Folgenden auf die Förderung von Problemlösefähigkeit[1] und der Förderung des schlussfolgernden Denkens eingegangen.

Hellmich und Kiper beschreiben das Problemlösen als „Strategiekonstruktionen, Strategienutzungen sowie das Entdecken von Regelhaftigkeiten bei ähnlichen Problemstellungen" (2006, S. 95). Weiter definieren Merschmeyer-Brüwer und Schipper das Problemlösen als eine Kompetenz, „die über das Anwenden und das Reproduzieren hinausgeht, weil beim Problemlösen im Vergleich zum reinen Anwenden und Reproduzieren keine unmittelbare Lösungsroutine [...] zur Verfügung steht" (2014, S. 198). In der Grundschule sind Kinder also dann mit *Problemen* konfrontiert, wenn ihr bisheriges Verhaltensrepertoire nicht ausreicht, um eine Aufgabe zu lösen, und sie gefordert sind, Strategien zu entwickeln, um diese zu lösen. Hellmich und Kiper benennen folgende Prädiktoren für effizientes Problemlösen bei Kindern:

> „Problemlösen bei Kindern erfordert dabei häufig das Abstecken von Zielen, d.h. die Zielgerichtetheit beim Lösen von Aufgaben, (bereichsspezifische) Kenntnisse oder Vorwissen in Bezug auf Problemstellungen, den Einsatz von Mitteln zur Zielerreichung hinsichtlich der Lösung eines Problems sowie ‚organisiertes und kontrolliertes Vorgehen beim Einsatz von Mitteln' (Oerter/Dreher 2002, S. 471)."
>
> (Hellmich und Kiper 2006, S. 95)

Schlussfolgerndes Denken von Kindern gilt als wichtige Voraussetzung für das Lösen von Problemen und umfasst das Vergleichen von Sachverhalten, In-Beziehung-Setzen, Ziehen von Rückschlüssen und das Entdecken von Regelmäßigkeiten (Hellmich und Kiper 2006, S. 96). Grundsätzlich lassen sich drei Arten des schlussfolgernden Denkens unterscheiden – das analoge, induktive und

[1] Der deutsche Begriff des Problemlösens wird in der einschlägigen Literatur in der Regel als Übersetzung des englischen Begriffs *problem solving* übernommen. Es ist zu beachten, dass der Begriff *problem* im Englischen nicht unbedingt als ein tatsächliches Problem zu verstehen ist und eine negative Konnotation hat, sondern auch mit „Frage", „Aufgabe", „Schwierigkeit" sowie mit „Problem-, Aufgaben- und Fragestellung" übersetzt wird (siehe https://www.linguee.de/englisch-deutsch/uebersetzung/problem.html).

deduktive Schließen (vgl. ebd., S. 96 f.). Das analoge Schließen bezeichnet das Erkennen und Deuten von Ähnlichkeiten zwischen bereits Bekanntem und Unbekanntem. Darauf aufbauend meint induktives Schließen, dass auf Basis immer wiederkehrender Regelmäßigkeiten oder Wirkungszusammenhänge allgemeine Schlussfolgerungen – Regeln oder Gesetzmäßigkeiten – gezogen werden. Als Voraussetzung nennen Hellmich und Kiper hierbei, dass „Kinder Regelmäßigkeiten bei Problemstellungen entdecken und vor allen Dingen Gleiches und Verschiedenes als solches differenzieren und benennen können“ (ebd., S. 97). Das deduktive oder auch logische Schließen bezeichnet das Gegenstück zum induktiven Schließen: vom Allgemeinen wird auf das Besondere bzw. den Einzelfall geschlossen. Zur Förderung des schlussfolgernden Denkens im Grundschulalter wird folgendes empfohlen:

> „Voraussetzungen dafür, dass schlussfolgerndes Denken in diesem Alter gelingt, ist das Erkennen von Ähnlichkeiten, der Transfer von Wissen auf neue Sachverhalte sowie das Ableiten von bisher nicht bekannten Zusammenhängen. Kinder sollten besonders im Grundschulalter Angebote im Unterricht gemacht werden, bei denen sie – im Bereich des analogen Schließens – überlegen können, inwiefern das Lösen eines Problems dadurch erleichtert werden kann, indem Bezug genommen wird auf eine bereits bearbeitete und gelöste Problemstellung: Gibt es Ähnlichkeiten oder Gemeinsamkeiten bei zu bearbeitenden Aufgabenstellungen? Worin liegen die Unterschiede genau?“
>
> (Hellmich und Kiper 2006, S. 97 f.)

## 11.2 Lernen und Leisten

Der Grundschulunterricht soll laut KMK Situationen schaffen, „in denen sich Schülerinnen und Schüler als Könnende erleben und ihren Lernvoraussetzungen, Lernbedürfnissen und ihrer individuellen Lernentwicklung entsprechend ihre bestmögliche Leistung entfalten können“ (KMK 2015, S. 10). Die Leistungsanforderungen sollen dabei „individuell angemessen, herausfordernd und erfüllbar“ sein (ebd.). In Bezug auf das konkrete Unterrichtsgeschehen äußert sich die KMK wie folgt:

> „In einem solchen Unterricht sind Fragen erwünscht. Fehler werden produktiv genutzt, denn sie sind ein notwendiger Bestandteil verstehensorientierter Lernprozesse. Die Entwicklung persönlicher Lernstrategien und Arbeitshaltungen wird unterstützend begleitet. “ (KMK 2015, S. 10)

### 11.2.1 Pädagogischer Leistungsbegriff

Das Leistungsverständnis in der Grundschule ist ihrem Bildungs- und Erziehungsauftrag untergeordnet und zielt ab auf die Grundlegung der Möglichkeiten und Voraussetzungen für Leistungen (vgl. Rehle u. a. 2017, S. 173. f.). Laut Schorch (2007, S. 169) umfasst dies unter anderem eine Leistungserziehung „als Unterstützen, als ‚Hilfe zur Selbsthilfe‘“, was für ihn bedeutet:

- „Deutliche Trennung und für die Kinder erkennbare Unterscheidung von Lernsituationen (Fehler gehören dazu) und Situationen der Leistungserhebung (Fehlervermeidung) im Sinne pädagogischer ‚Fehlerkultur'.
- Erkennen und Ermuntern vorhandener persönlicher (auch noch so verborgener) Leistungsansätze und Stärken sowie Anerkennung dieser Bemühungen in der sozialen Gruppe,
- Wecken von Sachinteresse (auch für Spezialgebiete),
- Anregungen von Durchhaltefähigkeit und Vollendungswillen, Hilfestellung bei besonderen Lernschwierigkeiten und -behinderungen,
- insgesamt: Abstimmung der Leistungsanforderungen an ‚individuellen Bezugsnormen'." (Schorch 2007, S. 169)

Dieses Leistungsprinzip anzuwenden bedeutet einerseits, Lernsituationen so zu gestalten, dass jedes Kind erfolgreich dazulernen kann und diesen individuellen Lernfortschritt auch unter Einbezug von Gruppenleistungen anzuerkennen, und andererseits „den Blick auf den Anspruch der Aufgabe freigeben, auf das Ganze, das es zu erreichen gilt, nicht um zu entmutigen, sondern um die eigene Leistung anzuspornen und um Orientierungen fürs Weiterlernen zu geben" (Rehle u. a. 2017, S. 175). Rehle u. a. (ebd.) machen jedoch gleichzeitig darauf aufmerksam, dass um den Schülerinnen und Schülern Orientierung und eine Einordnung der eigenen Leistungen zu ermöglichen, objektive Kriterien miteinbezogen werden sollten. Diese ergeben sich „aus dem Anspruch der Sache, der erfüllt sein muss, bevor die Aufgabe gelöst ist oder bevor eine Leistung vollständig erbracht ist" (ebd.). Schumacher und Denner (2017, S. 264) verweisen in diesem Kontext darauf, dass in den Bildungsplänen festgelegt ist, welche Leistungen Schülerinnen und Schüler innerhalb einer bestimmten Zeitspanne erbringen sollten.

### 11.2.2 Leistungsbeurteilung

Grundsätzlich teilt sich die Leistungsmessung in der Schule auf in die Phasen der Leistungsfeststellung und Leistungsbewertung: „Zunächst sollen Schulleistungen in einem möglichst wertfreien Vorgang erfasst werden. Anschließend sollen die Schulleistungen bewertet werden, wobei verschiedene Bezugsmaßstäbe für die Bewertung in Betracht kommen" (Zumhasch 2014, S. 302). Die Leistungsfeststellung gibt Informationen „über den situativen Stand des Lernens bei einem Kind in Bezug auf die gestellte Aufgabe (in der jeweiligen Situation der Bearbeitung der Aufgabenstellung)" (Maras und Ametsbichler 2014, S. 350) und setzt voraus, dass der zu erfassende Aspekt der Schulleistung genau festgelegt und beschrieben wird:

> „In jedem Fall gründet professionelle Schulleistungsbeurteilung auf einer Präzisierung der fachspezifischen oder auch fächerübergreifenden Lernziele. Hierdurch wird

> zunächst inhaltlich der jeweils bedeutsame Kompetenzbereich einer schulischen Leistungsbeurteilung fixiert. Schulleistungen aber liegen nicht einfach offen zutage (vgl. Tent Stelzl 1993, 212), müssen vielmehr aus konkreten Tätigkeiten"(vgl. Jäger 2000, 41) bzw. aus Verhaltensweisen von Schülern im Unterricht erschlossen werden [...]. Daher müssen die leistungsrelevanten Inhalte, die Gegenstand einer Beurteilung sein sollen, in empirisch überprüfbare – d. h. einzelne, jeweils fest umrissene – Schülertätigkeiten spezifiziert werden. Diese fungieren als Indikatoren bzw. als Zeichen dafür, welche bzw. inwieweit Kompetenzen von Schüler im Unterricht erworben wurden."
>
> (Zumhasch 2014, S. 303)

Zur Feststellung von Schülerleistungen sind schriftliche, mündliche und praktische Verfahren zulässig sowie ihre individuelle Zusammenstellung, z. B. in Form von einem Rechtschreibausweis, Lesepass, Portfolio oder Lerntagebuch (vgl. Zumhasch 2014, S. 303). Jedes Verfahren umfasst wiederum spezifische Indikatoren, die einen bestimmten Lerninhalt wiederspiegeln. Jürgens und Sacher (vgl. 2008, S. 87) formulieren für die Auswahl der Verfahren und Indikatoren zwei Grundsätze – den Grundsatz der proportionalen Abbildung und der Variabilität. Diese besagen zum einen, dass die Indikatoren repräsentativ für den erteilten Unterricht sein sollen und zum anderen, dass die Verfahren zur Leistungsfeststellung „hinsichtlich der Prüfungs- und Aufgabenformen, des Umfangs und der Komplexität der Aufgaben" variiert werden sollen (ebd.). Letzteres wird gefordert, damit keine Schülerinnen und Schüler von einer einseitigen Modalität bevorzugt bzw. benachteiligt werden.

Sind die schulischen Leistungen der Schülerinnen und Schüler erhoben, erfolgt die Leistungsbewertung. Diese sollte in der Grundschule sowohl produkt- als auch prozessorientiert (vgl. Zumhasch 2014, S. 305) und – wie in allen Schularten – unter Festlegung einer Bezugsnorm erfolgen (vgl. Rehle u. a. 2017, S. 177). Hierbei werden drei Arten von Bezugsnormen unterschieden, die jeweils bestimmte, sich ergänzende Bewertungen ermöglichen (ebd.): die *individuelle Norm* (Messen der Leistungen am persönlichen Fortschritt der Schülerinnen und Schüler), die *kriteriale Norm* (Messen der Leistung an einem sachlichen Bezugssystem, z. B. Lehrplan) und die *soziale Norm* (Leistung wird verglichen mit der Leistung anderer und in Rangfolge gebracht). Da die Schülerinnen und Schüler in der Grundschule sehr unterschiedliche Lern- und Leistungsvoraussetzungen mitbringen, sollten laut Rehle u. a. (2017) verschiedene Leistungsmaßstäbe angewandt werden:

> „Die Bewertung der jeweiligen Leistung am individuellen Maßstab wird insofern jedem Kind gerecht, als es seine Anstrengung und Fortschritte beschreibt und würdigt, wie dies in den Wortgutachten der Zeugnisse angestrebt wird. Ergänzend wäre die kriteriale Norm, eine Einordnung der persönlichen Leistung in den Zusammenhang des Lernganzen nötig, um die Orientierung auf die nächsten Lernschritte und auf das Ziel freizugeben. "
>
> (Rehle u. a. 2017, S. 178)

### 11.2.3 Fehlerkultur

Während Fehler in der Leistungsbewertung in der Regel zu schlechteren Beurteilungen führen und somit negativ konnotiert sind, können sie in der Leistungsmessung wertvolle Einblicke in den jeweiligen Lernstand und die Denkweise der Schülerinnen und Schüler geben (vgl. Rehle u. a. 2017, S. 177). Die Leistungsbeurteilung kann bei entsprechender Gestaltung zur Lernförderung und – besonders in der Grundschule – zur Entwicklung einer stabilen Lernhaltung beitragen (vgl. Knauf 2009, S. 253 f.). Knauf (ebd.) beschreibt drei Ebenen, auf der sich Lernförderung im Rahmen der Leistungsbewertung realisieren lässt, und zu einer kindorientierten Lernkultur beitragen:

- Affektive Ebene
  Die Rückmeldung der Lehrkraft kann bspw. bei Enttäuschungen bzgl. der Lernfortschritte Trost spenden und Mut machen, Lernfortschritte anerkennen, Freude über die Fortschritte ausdrücken sowie neue Ziele für die Lernenden formulieren.

- Inhaltliche Ebene
  Die Lehrkraft kann bspw. Regeln in Erinnerung rufen, auf Hilfsmittel verweisen oder Hinweise geben, auf bestimmte Fehlerarten zu achten oder Analogien zu erkennen.

- Strategisch-Methodische Ebene
  Die Lehrkraft kann die Leistungsbewertung nutzen, um Hinweise zur Arbeitsweise der Schülerinnen und Schüler zu geben, z. B. das Konsultieren von Mitschülerinnen und Mitschülern bei Fragen oder das sinnvolle Einteilen der eigenen Zeit bei der Bearbeitung einer Aufgabe.

## 11.3 Fördern und Fordern

Fördernder Unterricht stellt laut KMK (2015, S. 10) Anforderungen, setzt realistische Ziele und legt diese den Schülerinnen und Schülern offen. Der Lernstand wird jedem Kind in regelmäßigen Lerngesprächen aufgezeigt, deren Basis „regelmäßige Lernbeobachtungen, die Dokumentation des Lern- und Entwicklungsprozesses sowie gezielte Erhebungen des Lernstands“ sind (ebd.). Die KMK betont, dass die Schülerinnen und Schüler durch fördernden Unterricht Zutrauen in die eigenen Fähigkeiten ausbilden sollen:

> „Aus der Erfahrung, etwas leisten zu können, dem individuellen Leistungsvermögen entsprechende Anforderungen bewältigt zu haben, erwachsen bei den Lernenden das Vertrauen und der Wille, sich weiteren Anforderungen zu stellen. Die Lehrkraft fördert das Zutrauen in die eigenen Kompetenzen, das entscheidend beim Lernen ist.“ (KMK 2015, S. 10)

### 11.3.1 Ausgangslage

Die Forderung, jede einzelne Schülerin und jeden Schüler optimal individuell zu fördern findet sich zwar in sämtlichen Texten, die den Bildungsauftrag der Schule als Ganzes sowie einzelner Schulformen formulieren, hat jedoch je nach Schulform einen unterschiedlichen Stellenwert (vgl. Sandfuchs 2014, S. 324). Der Förderauftrag der Grundschule, der besonders herausgestellt wird, ergibt sich in erster Linie durch die heterogene Schülerschaft der Grundschule:

> „Die Grundschule ist die Schulart mit der größten Heterogenität und muss mit der besonderen Belastung einer noch unausgelesenen Schülerschaft zurecht kommen: Von hochbegabten bis förderbedürftigen Schülern, von ‚Überfliegern' bis Langsam-Lernern, von sozioökonomisch Privilegierten bis sozial Benachteiligten."
>
> (Schorch 2007, S. 81)

> „Kinder unterscheiden sich bereits im Grundschulalter durch gesellschaftliche Individualisierungsprozesse, durch die differenzielle Übertragung des kulturellen und sozialen Kapitals ihres elterlichen Milieus und durch psychische und körperliche Entwicklungen" (Henze, Sandfuchs und Zumhasch 2014, S. 133)

Knauf macht darauf aufmerksam, dass die Fördermaßnahmen in der Grundschule allen Kindern zu Gute kommen sollen – „nicht nur den Kinder mit Lernproblemen, sondern auch denen mit besonderen Leistungsstärken oder Begabungen" (2009, S. 209).

### 11.3.2 Formatives Assessment

Grundlage für einen fördernden Unterricht ist die gezielte Erhebung des Lernstandes und dessen pädagogische Nutzung, was als Gegensatz zum summativen Assessment – welches primär dem Zweck der Rechenschaftslegung dient – als formatives Assessment oder auch *Assessment for Learning* bezeichnet wird (vgl. Schmidt 2020, S. 8). Black u. a. differenzieren die Begriffe *Assessment for Learning* und formatives Assessment weiter aus:

> *„Assessment for learning is any assessment for which the first priority in its design and practice is to serve the purpose of promoting pupils' learning. It thus differs from assessment designed primarily to serve the purposes of accountability, or of ranking, or of certifying competence. An assessment activity can help learning if it provides information to be used as feedback, by teachers, and by their pupils, in assessing themselves and each other, to modify the teaching and learning activities in which they are engaged. Such assessment becomes ‚formative assessment' when the evidence is actually used to adapt the teaching work to meet learning needs. "*
>
> (Black u. a. 2004, S. 10)

Als eine der Kernstrategien des formativen Assessments nennt Wiliam (2013, S. 16) das Offenlegen und Kommunizieren, was von einer bestimmten Unterrichtsaktivität gelernt werden soll.

### 11.3.3 Zielorientierung

Die Zielorientierung zählt laut Glötzl (2000, S. 127) zu den wichtigsten Prinzipien des Unterrichts und wird definiert als „Oberbegriff für die Orientierung des absichtsvollen unterrichtlichen Handelns des Lehrers an möglichst klar formulierten Zielen [...], für die Zielgerichtheit und Zielgemäßheit seiner Einzelmaßnahmen wie auch für die Orientierung der Schüler an Zielen, die sie gemeinsam mit dem Lehrer aufgestellt haben“ (ebd.). In ihren praktischen Hinweisen zur Zielorientierung in der Grundschule schlagen Maras und Ametsbichler (2014, S. 323) verschiedene Maßnahmen vor, bspw. das Festsetzen eines Schwerpunkts für die jeweilige Unterrichtseinheit, Formulieren einer klaren Zielangabe (schriftlich oder verbal), in Zusammenhang stellen von Teilergebnissen oder Bewusstmachen des Fortschreitens im Lernprozess.

## 11.4 Individualisierung und Differenzierung

Die Grundschule soll laut KMK (2015, S. 10) der Heterogenität der Schülerinnen und Schüler durch einen individualisierenden und differenzierten Unterricht begegnen, der sich an deren Ausgangslage orientiert. Lehrkräfte treffen „inhaltliche und didaktische Entscheidungen sowie Festlegungen hinsichtlich zielgerichteter Methoden, Sozialformen, Arbeitsweisen und Aufgabenformate“ auf Basis der individuellen Voraussetzungen und Bedürfnisse der Kinder (ebd.). Die Lernumgebungen sollen adaptiv gestaltet werden und so viel innere Differenzierung ermöglichen, „dass Schülerinnen und Schüler für sich jeweils passende Lernherausforderungen und Übungsmöglichkeiten vorfinden und unterschiedliche Lösungswege gehen können“ (ebd.). Zur Rolle der Lehrpersonen in dem beschriebenen Unterrichtsgeschehen heißt es in den Empfehlungen der KMK:

> „Die Lehrkräfte unterstützen Schülerinnen und Schüler dabei, Inhalte und Methoden, Sozialformen sowie den Arbeitsplatz und die Arbeitsmittel zunehmend selbst zu wählen und ihre Arbeitsergebnisse eigenständig zu kontrollieren. Selbstbestimmtes und kooperatives Lernen werden systematisch gefördert.“ (KMK 2015, S. 11)

### 11.4.1 Heterogenität in der Schulklasse

Wie bereits beschrieben hat es die Grundschule prinzipiell mit einer „unausgelesenen Schülerpopulation und damit mit der vollen Variabilität von Schülerleistungen, Lernvoraussetzungen und familialen Hintergrundmerkmalen zu tun, wie sie in der entsprechenden Grundgesamtheit vorkommt“ (Kluczniok, Große und Roßbach 2014, S. 194). Dies schließt Kinder mit Behinderungen ein, die laut Artikel 24 der UN-Behindertenrechtskonvention ein Recht auf ein inklusives Bildungssystem auf allen Ebenen haben (Beauftragter der Bundesregierung für die Belange von Menschen mit Behinderungen 2007).

Ein für den Unterricht besonders relevantes Merkmal stellt die Varianz der Leistungen und Leistungsmöglichkeiten von Kindern dar. Leistungsheterogenität in der Grundschule, welche bereits in einer Vielzahl von empirischen Studien nachgewiesen werden konnte, bezieht sich auf Unterschiede einerseits in den kognitiven Grundfähigkeiten der Kinder und andererseits in spezifischere Faktoren, z. B. die Rechtschreib- oder Leseleistungen (vgl. Decristan u. a. 2014, S. 182). Darüber hinaus erwies sich das Geschlecht in vielen Forschungsbereichen als zentrale Strukturkategorie, z. B. in Studien zu Interaktionen im Schulalltag, Unterrichtsprodukten, Schulleistungen, Interessen und dem Fähigkeitsselbstkonzept von Schülerinnen und Schülern (vgl. Heinzel und Prengel 2014, S. 201 f.).

### 11.4.2 Binnendifferenzierung

Ein Ansatz für den Umgang mit Heterogenität stellt die Binnendifferenzierung dar, die sämtliche Differenzierungsformen umfasst, „die *innerhalb* einer gemeinsam unterrichteten Klasse oder Lerngruppe vorgenommen werden, im Unterschied zu allen Formen sog. *äußerer* Differenzierung, in der Schülerpopulationen nach Gliederungs- oder Auswahlkriterien – z. B. den Gesichtspunkten unterschiedlichen Leistungsniveaus oder unterschiedlicher Interessen – in Gruppen aufgeteilt werden, die räumlich getrennt und von verschiedenen Personen bzw. zu unterschiedlichen Zeiten unterrichtet werden“ (Klafki 2007, S. 173). Hellmich und Kiper benennen verschiedene Umsetzungsformen der Differenzierung innerhalb des Unterrichts:

> „Diese Differenzierung kann unterschieden werden in soziale Differenzierung (nach den Sozialformen Einzelarbeit, Partnerarbeit, Arbeit in flexiblen Lerngruppen), in methodische Differenzierung (bezogen auf die Anzahl der Aufgaben, die Zeitvorgaben, die Formen des Lernens durch Handeln), in mediale Differenzierung (durch Einsatz je verschiedener Medien und Materialien oder deren eigene Herstellung) und in thematische Differenzierung (durch eine Differenzierung nach Neigung/Interessen, Leistung und Ergänzungen).“ (Hellmich und Kiper 2006, S. 20 f.)

### 11.4.3 Adaptivität

Ein weiterer Ansatz, der Heterogenität der Schülerschaft zu begegnen, findet sich im adaptiven Unterricht, welcher der pädagogischen Psychologie bzw. der Lehr-Lernforschung entwachsen ist (Inckemann 2014, S. 376). Hardy u. a. (2011, S. 820) beschreiben Adaptivität in diesem Kontext als „Ergebnis von Lernprozessen durch die optimale Nutzung von Lerngelegenheiten mit spezifischer methodisch-didaktischer Schwerpunktsetzung“ (siehe Abbildung 11.1). Adaptiven Unterricht beschreiben sie wie folgt:

> „Ein Unterricht, der adaptiv mit heterogenen Schülervoraussetzungen umgeht, bietet eine Angebotsstruktur, in der auf Schülerseite Lernprozesse so initiiert und aufrechterhalten werden, dass möglichst viele Schüler ihr Potenzial entfalten. Im Unterricht sind

methodisch-didaktische Schwerpunktsetzungen also als ein an generellen Lernprinzipien ausgerichtetes Angebot zu verstehen, welches in unterschiedlichem Maße an die Eingangsvoraussetzungen der Schüler angepasst sein kann."

(Hardy u. a. 2011, S. 821)

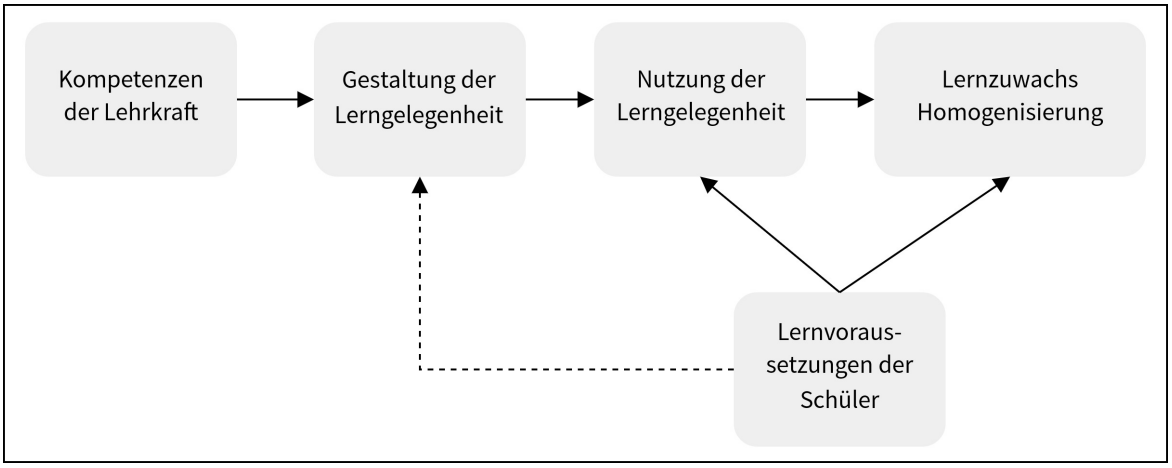



*Abbildung 11.1 Adaptivität als Ergebnis von Lernprozessen aus der Interaktion von methodisch-didaktischer Angebotsstruktur, Schülervoraussetzungen und individuell unterschiedlicher Nutzung aus Hardy u. a. (2011, S. 820)*

Zur Optimierung von Adaptivität beschreiben sie methodisch-didaktische Schwerpunktsetzungen auf unterschiedlichen Ebenen des Unterrichts (vgl. ebd., S. 823 ff.):

- Kognitive Strukturierung
  Durch die Abstimmung von Lernangeboten auf die kognitiven Voraussetzungen und das fachliche Kompetenzniveau der Schülerinnen und Schüler kann die Adaptivität auf Ebene der Lehrer-Schüler-Interaktion gesteigert werden. Dies kann bspw. durch den Einsatz kognitiv strukturierender, verständnisorientierter Fragen und Impulse im Unterrichtsgespräch und bei der Bearbeitung von Aufgaben erfolgen.

- Peer Learning
  Auf der Ebene der Schüler-Schüler-Interaktionen können Lerngelegenheiten durch Formen des kooperativen Lernens optimiert werden, in denen besonders leistungsschwächere Schülerinnen und Schüler in Hinblick auf Leistung und Motivation profitieren. Die Schülerinnen und Schüler können in ihren Aushandlungsprozessen durch gezielte Vorgaben der Lehrkraft unterstützt werden.

- Formatives Assessment
  In Bezug auf die individuelle Lernzielerreichung von Schülerinnen und Schüler kann durch Maßnahmen des formativen Assessments zunächst deren individuelle Ausgangslage bestimmt werden. Im Anschluss wird das Aufgabenangebot darauf abgestimmt, bspw. durch unterschiedliche Bearbeitungsmodi oder -zeiten, oder sogar neu konzipiert.

Das Konstrukt der Kognitiven Strukturierung beschreibt Hardy (2012, S. 52) als eine Facette des *Scaffolding* (auf Deutsch *Gerüst*). Der Begriff bezieht sich im pädagogisch-psychologischen Kontext auf die Unterstützung eines Lernprozesses durch verschiedene Maßnahmen wie Anleitungen, Denkanstöße oder andere Hilfestellungen (Schnotz 2006, S. 151 f.). Diese Maßnahmen sollen die Schülerinnen und Schüler auf dem Weg zum selbstständigen Handeln begleiten und werden mit zunehmendem Können zurückgenommen.

### 11.4.4 Hilfestellungen im Unterricht

Sowohl beim *Scaffolding* als auch bei der Kognitiven Strukturierung kann das *Contingent Shift Principle* nach Wood, Wood und Middleton (1978) zur Bestimmung der optimal auf Schülervoraussetzungen abgestimmte Maßnahmen einer Lehrkraft herangezogen werden:

> „Schafft ein Schüler eine Aufgabe nicht, sollte eine Kontrollerhöhung erfolgen, löst ein Schüler eine Aufgabe erfolgreich, sollte eine Kontrollverringerung erfolgen. Eine Definition von Kontrollerhöhung wird durch den Grad an expliziter Information und Instruktion gelegt. “ (Hardy 2012, S. 53)

Darüber hinaus beschreibt Wood (2003, S. 12 f.) in seinem Ansatz des *Contingent Support for Learning* verschiedene Ebenen der Hilfestellung, um Schülerinnen und Schüler bei Schwierigkeiten in ihrem Lernprozess zu unterstützen und mehr oder weniger Kontrolle auszuüben:

- Level 1: General Verbal Intervention
  Die Lehrkraft nimmt zwar Bezug auf die aktuelle Lernsituation, gibt dem Lernenden jedoch keine spezifischen weiteren Schritte vor, z. B. durch Äußerungen wie *„It's your turn now“* oder *„I'm not sure about that“*.
- Level 2: Specific Verbal Intervention
  Die Lehrkraft bezieht sich in ihrer Äußerung auf einen spezifischen Schritt zur Problemlösung oder stellt heraus, wo ein Fehler liegt (z. B. *„Why don't you find the next biggest blocks?“*, *„I don't think that one looks right, I think it's too small to go on top there.“*).
- Level 3: Specific Verbal Intervention plus Nonverbal Indicators
  Die Lehrkraft reichert ihre verbale Äußerung mit nonverbalen Elementen, z. B. Blicken oder dem Zeigen auf etwas, an und gibt dem Lernenden auf diesem Wege Hinweise, auf was sich die Äußerung bezieht.
- Level 4: Prepares for Next Action
  Die Lehrkraft gibt durch das Stellen von geschlossenen Fragen bestimmte Handlungsmöglichkeiten vor und übt somit verhältnismäßig viel Kontrolle über den Lernprozess aus, bspw. durch Fragen wie *„Is it A or is it B?“* oder *„Does that sound like s-or-t?“*.
- Level 5: Demonstrates Action
  Um die vollständige Kontrolle über die nächste Handlung zu übernehmen, kann die Lehrkraft selbst demonstrieren, was als nächstes getan werden muss oder das Vorgehen in Worten beschreiben.

# 12 Fachdidaktische Grundlagen

Neben allgemeindidaktischer Aspekte der Grundschulpädagogik spielen für die Unterrichtsgestaltung in der Primarstufe fachdidaktische Aspekte eine Rolle, die sich auf die einzelnen Unterrichtsfächer bzw. Lernbereiche der Grundschule beziehen (Schorch 2007, S. 18). Da das Schulfach Informatik verhältnismäßig jung ist (Magenheim u. a. 2018a, S. 182) und sich bisher vorrangig als Fach an den weiterführenden Schulen etabliert hat (Schwarz, Hellmig und Friedrich 2022), befassen sich die informatikdidaktische Auseinandersetzung und Forschung vorrangig mit dem Lehren und Lernen in der Sekundarstufe.

Unabhängig von der Schulstufe gibt es Strukturmodelle für unterrichtsbezogene Entscheidungen, die Lehrkräfte für die didaktische Analyse und Planung von Unterricht heranziehen können. Das *Berliner Modell* (vgl. Heimann, Otto und Schulz 1965, S. 23 ff.) basiert auf der Annahme, dass sämtliche didaktische Vorgänge im Unterricht einer formal konstanten Grundstruktur folgen (siehe Abbildung 12.1[1]). Es beschreibt zum einen vier Entscheidungsfelder, aus denen Unterricht aufgebaut ist: Intentionen, Inhalte, Methodik und Medien. Zum anderen werden zwei Bedingungsfelder benannt, die von der Lehrkraft bei ihren Entscheidungen bzgl. der Entscheidungsfelder zu berücksichtigen sind: anthropogene und sozialkulturelle Voraussetzungen[2].

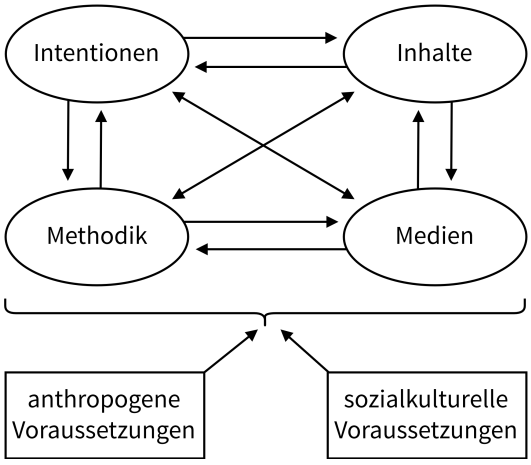



*Abbildung 12.1 Kategorienraster zur Strukturanalyse von Unterricht aus Jank und Meyer (2021, S. 263)*

Um sich Informatikunterricht in der Unterrichtsplanung (und Forschung) strukturierter zu nähern, entwickelten Diethelm u. a. (2011) das Modell der *Didaktischen Rekonstruktion für den Informatikunterricht*. Sie bauen auf dem Modell der *Didaktischen Rekonstruktion* von Kattmann u. a. (1997) auf und übernehmen die Teilaufgaben, die darin beschrieben werden: Fachliche Klärung, Erfassen

[1] In der Abbildung wurden die Begrifflichkeiten von Heimann, Otto und Schulz übernommen. Jank und Meyer verwenden anstelle von *Inhalte* den Begriff *Thematik* und anstelle von *Medien* den Begriff *Medienwahl*.
[2] Eine weitere Aufschlüsselung der sechs Strukturelemente findet sich z. B. in Riedl (2010, S. 106 f.)

von Schülerperspektiven und Didaktische Strukturierung. Um das Ziel zu betonen, Lernenden eine Sicht auf die Wirklichkeit aus dem Blickwinkel der Informatik zu ermöglichen, und aufgrund der jungen Tradition der Informatik als Schulfach sowie der Heterogenität der Lehrerbildung, erweitern sie das Modell für den Informatikunterricht um folgende Aspekte: Auswahl informatischer Phänomene, Klärung gesellschaftlicher Ansprüche ans Fach und Erfassung von Lehrerperspektiven (siehe Abbildung 12.2).

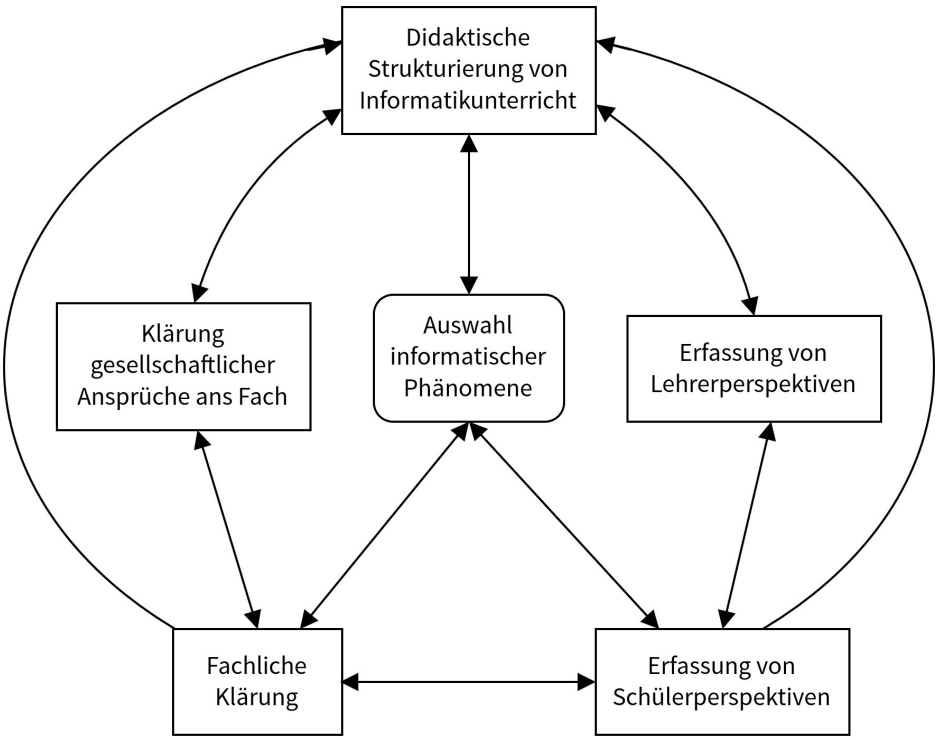



*Abbildung 12.2 Didaktische Rekonstruktion für die Informatik aus Diethelm u. a. (2011, S. 80)*

Im Folgenden werden die für die Arbeit relevanten fachdidaktischen Theorien und Befunde dargestellt. Der inhaltliche Fokus liegt dabei auf der Programmierung. In Anlehnung an die beiden zuvor beschriebenen Modelle wird zunächst näher auf die Perspektive der Lernenden eingegangen. Anschließend werden verschiedene grundsätzliche Zugänge zur Informatik für Kinder sowie konkrete Ansätze für die Gestaltung von Lehr-Lernsituationen zum Programmieren beschrieben.

## 12.1 Perspektive der Lernenden

Die Nutzung von Lerngelegenheiten im Unterricht hängt maßgeblich von den individuellen Voraussetzungen der Lernenden ab, z. B. den Vorstellungen zum Lerngegenstand oder motivationalen Aspekten (Seidel 2014, S. 859).

### 12.1.1 Alltagstheorien zum Programmieren

Bereits vor dem Schuleintritt entwickeln Kinder verschiedene subjektiven Theorien darüber, wie Dinge in ihrer Umgebung funktionieren. Diese Ideen entstehen aus ihren eigenen Erfahrungen,

sodass unterschiedliche Kinder unterschiedliche Vorstellungen zu bestimmten Themen besitzen (Worth 1999, S. 26). Da Kinder in ihrem Alltag regelmäßig mit Informatiksystemen in Kontakt kommen (vgl. Bitkom 2019; MPFS 2019), entwickeln sie somit zwangsläufig Vorstellungen darüber, wie diese funktionieren.

Sheehan (2003) forderte 36 Schülerinnen und Schüler der Grundschule aus zwei verschiedenen Altersgruppen (6-7 Jahre und 9-10 Jahre) auf, Personen zu zeichnen, die gerade einen Computer programmieren. Zusätzlich wurden mit einer Woche Abstand Befragungen durchgeführt, in denen die Kinder gebeten wurden, ihre Bilder zu kommentieren sowie Fragen über Computer und das Programmieren zu beantworten. Keine der beiden Gruppen zeigte ein tieferes Verständnis darüber, was ein Computerprogramm ist, obwohl sie verschiedene Programme aufzählen konnten, die sie in ihrem Alltag nutzen. Auf die Frage, wie Computerprogramme hergestellt werden, wurde häufig auf die Produktion verschiedener Medien verwiesen:

> *„Many answers made reference to the production of other media especially TV programmes. Producing art work, photos and sounds appeared frequently in the answers [...].“*
>
> (Sheehan 2003, S. 78)

Auch in anderen Studien wurde die Methodik des Zeichnens verwendet, um die Einstellungen und das Verständnis von Kindern bzgl. der Informatik und Funktionsweise von Computern zu erfassen (z. B. Brosnan 1999; Denham 1993; Hansen u. a. 2017). Hansen u. a. (2017) führten einen *Draw-A-Computer-Scientist*-Test mit 185 Schülerinnen und Schüler im Alter von 8-11 Jahren durch, bei dem die Kinder angewiesen wurden, *„to draw a computer scientist working and describe what was going on in the picture“* (ebd., S. 281). Der Großteil der Zeichnungen spiegelte gängige Stereotype wieder, z. B. wurden hauptsächlich männliche Personen gezeichnet. In Bezug darauf, was ein *computer scientist* tut, zeigten die Ergebnisse, dass diese in den Augen der Kinder hauptsächlich alleine und mit Computern arbeiten. Was sie genau tun, variierte in den Beschreibungen der Kinder:

> *„Based on the verbs used in the written descriptions of pictures, commonly reported actions of computer scientists included working (23%), coding (18%), making (16%), typing (%), doing (7%), looking (7%), fixing (7%), and testing (6%). When considering the object that the computer scientists were working on/coding/making, common responses included computers (26%), ideas (18%), games (7%), and science experiments (7%).“*
>
> (Hansen u. a. 2017, S. 282)

In einer interdisziplinären Literaturanalyse untersuchten Rücker und Pinkwart (2016) die Vorstellungen von Kindern darüber, wie Computer funktionieren, woraus sie bestehen und welche Fähigkeiten sie haben. Dabei konnten sie fünf Vorstellungen identifizieren:

> *„[..] computers can be seen as intelligent entities that are best understood in psychological rather than physical terms. They can also be seen as unlimited and potentially omniscient databases, which know the answers to virtually all questions, without having to actually compute anything. They can be seen as a complex network of wires and*

> *communication links in which the actual components might play a negligible role. They can be seen as containing a complex mechanical contraption. And they can be seen as programmable machines that need to be told what to do by their human masters.“*
>
> (Rücker und Pinkwart 2016, S. 282)

### 12.1.2 Notional Machine

Eng verknüpft mit den subjektiven Theorien zum Programmieren ist das Konzept der *Notional Machine*. Der Begriff wurde erstmals von Du Boulay (1986) eingeführt, der ihn beschreibt als *„the general properties of the machine that one is learning to control“* (ebd., S. 57). Sorva beschreibt die *Notional Machine* als *„not a physical computer but a model of program execution that can be taught to learners so that they can reason about program behavior and write programs that work.“* (Sorva 2020, S. 143). Um Den Prozess des Programmierenlernens und der Ausbildung der *Notional Machine* zu illustrieren, zieht Du Boulay (1986) verschiedene Analogien heran:

> *„A running program is a kind of mechanism and it takes quite a long time to learn the relation between a program on the page and the mechanism it describes. It's just as hard as trying to understand how a car engine works from a diagram in a text book. [... ] Even after becoming familiar with some of the pieces that make up a program, e. g. PRINT or IF, the novice may not see a program as a working mechanism in the same way as an experienced programmer. This ability to see a program as a whole, understand its main parts and their relation is a skill which grows only gradually. Two comparisons are often made. One is to foreign language learning with its halting progression from disconnected words and small phrases to fluent speech. The other is to the way an experienced chess player can ‚read' a chessboard, where a novice sees only a jumble of individual chessmen.“* (Du Boulay 1986, S. 59)

Im Bereich der Grundschule gibt es kaum Befunde bzgl. der *Notional Machine*. Lowe (2018, S. 12) zeigt jedoch, dass Zweitklässler bereits in der Lage sind zu verstehen, dass einzelne Programmierbefehle zu bestimmten Handlungen führen. Sorva (2020, S. 150 ff.) beschreibt verschiedene Herangehensweisen, um Schülerinnen und Schüler beim Verständnis grundlegender Programmierkonzepte zu unterstützen, bspw. der Einsatz von *code-reading activities* sowie Visualisierungen oder Rollenspiele, um abstrakte Konzepte greifbar zu machen.

### 12.1.3 Motivation beim Programmieren

Wie bereits beschrieben, spielt der Faktor Motivation eine wichtige Rolle im Lernprozess, und sollte bei der Gestaltung von Lernprozessen berücksichtigt werden (siehe Abschnitt 10.3). Martin, Hughes und Richards (2017, S. 301 ff.) beschreiben verschiedene *Learning Dimensions*, die helfen können, die persönliche Motivation der Lernenden beim Programmierenlernen zu steigern. Diese Dimensionen gehen auf die Ergebnisse von vier Feldstudien zurück, die mit Teilnehmerinnen und Teilnehmern aus der Vorschule bis hin zur Universität durchgeführt wurden. Hinsichtlich

der Gestaltung von Programmieraufgaben scheinen offene Aufgaben motivierender zu sein als geschlossene Aufgaben (vgl. Petre und Price 2004, S. 151). Außerdem sind die Lernenden motivierter, wenn die Aufgaben sich auf ihre Lebenswelt beziehen, wenn sie ihre Programme personalisieren können und wenn sie die Möglichkeit haben, ihre Ergebnisse mit anderen zu teilen. In Bezug auf die Gestaltung des Lernprozesses empfehlen sie, dass Lernende selbstgesteuert Arbeiten können und sie die Möglichkeit erhalten, die eingeführten Programmierkonzepte wiederholt zu verwenden. Um eine Überforderung der Lernenden zu vermeiden und Fehlvorstellungen frühzeitig abzufangen, sollten vor allem zu Beginn des Lernprozesses kurze, überschaubare Aufgaben eingesetzt werden, zu denen die Lernenden ein direktes Feedback erhalten. Zur Erklärung der Ergebnisse von Martin, Hughes und Richards (2017) können wiederum die Modelle zur Lern- und Leistungsmotivation herangezogen werden, die in Kapitel 10.3 beschrieben wurden.

### 12.1.4 Genderspezifische Unterschiede

(Die meisten) Mädchen und Jungen unterscheiden sich in verschiedenen Aspekten, die bei der Gestaltung von Lehr-Lernsituationen berücksichtigt werden sollten (siehe Abschnitt 11.4). Dies ist in Bezug auf die Informatik von besonderer Bedeutung, da genderspezifische Zuschreibungen weit verbreitet sind (siehe Abschnitt 2.1). Hubwieser, Hubwieser und Graswald (2016) untersuchten die Ergebnisse von Jungen und Mädchen beim Informatik-Biber 2014 und stellen fest, dass Jungen im Wettbewerb insgesamt erfolgreicher waren und dass die Leistungsunterschiede zwischen den Geschlechtern mit dem Alter zunahmen. Beim Vergleich der Ergebnisse einzelner Aufgaben arbeiteten sie heraus, dass die Mädchen zunehmend bessere Leistungen bei Aufgaben erbrachten, die drei Kriterien des *ARCS-Modell* von Keller (1987) erfüllten: Die Aufgaben müssen schön aussehen, um die Aufmerksamkeit der Mädchen zu erregen, sie müssen eine alltagsrelevante Situation darstellen und sie müssen vergleichsweise einfach zu lösen sein. Hinsichtlich des Programmierens in Scratch untersuchten verschiedene Arbeiten die Programmierergebnisse von Jungen und Mädchen auf Unterschiede (z. B. Adams und Webster 2012, Hsu 2014, Fields, Kafai und Giang 2017). Für die Unterrichtsgestaltung sind die Ergebnisse von Adams und Webster (2012) hervorzuheben, die zeigen, dass Mädchen beim Programmieren von Geschichten Dialoge bevorzugen, während Jungen *Action*-Elemente wie die Animation von Explosionen bevorzugen.

## 12.2 Kindgerechte Zugänge zur Informatik

Bergner u. a. (2018, S. 85 ff.) beschreiben mehrere grundsätzliche Zugänge zur Informatik für Kinder – Zugänge ohne Computer, softwarebeasierte Einstiege in die Programmierung, Programmierbares Spielzeug (Robotik) sowie außerschulische Lernorte und Communities. Die Unterscheidung wird im Folgenden aufgegriffen, um verschiedene Ansätze zur Unterrichtsgestaltung zu beschreiben. Diese sind nicht als trennscharf zu sehen und können durchaus allesamt in einer Unterrichtseinheit zur Anwendung kommen. Da der außerschulische Bereich in der vorliegenden Arbeit nicht im Fokus steht, wird er im Folgenden nicht weiter berücksichtigt.

### 12.2.1 Zugänge ohne Computer

Inhalte und Konzepte der Informatik können im Unterricht auch ohne den Einsatz von Informatiksystemen thematisiert werden. Mit dem Einsatz sogenannter *Unplugged*-Materialien, für deren Einsatz kein Computer nötig ist, werden den Lernenden handlungsorientiert und meist spielerisch Aspekte des *Computational Thinking* oder konkrete Fachinhalte verdeutlicht (vgl. HdkF 2018, S. 86 ff.). Das Konzept des *Unplugged Computing* wurde maßgeblich durch das *CS unplugged*-Projekt[3] geprägt, dessen Initiatoren sich den Erfolg des Ansatzes wie folgt erklären:

> *„Having activities away from computers is effective because children generally know the computer as a tool or toy, rather than the subject of study in itself. By stepping away from the computer they are able to think about issues that Computer Scientists face beyond simply programming. Topics such as algorithm complexity, data compression, graphics algorithms, interface design and models of computing can be tackled without having technical experience as a prerequisite. In many cases, children find the topics fascinating, but would otherwise have had to jump the hurdle of learning to program before they could engage in the deeper topics that the subject offers. "*
>
> (Bell u. a. 2009, S. 21)

Der *Unplugged*-Ansatz wird von Lehrkräften zwar immer wieder als lernförderlich beschrieben (z. B. Sentance und Csizmadia 2017, S. 485), seine Effektivität wird jedoch auch in mehreren Arbeiten kontrovers diskutiert (vgl. Bell und Vahrenhold 2014, S. 503 ff.). Bzgl. des effektiven Einsatzes von *Unplugged*-Materialien im Informatikunterricht schlagen Waite u. a. (2019) das Heranziehen eines Konzepts der *Legitimation Code Theory* vor. Diese bietet einen theoretischen Rahmen, um zu untersuchen, ob eine Lernerfahrung bzw. eine Erklärung effektiv ist. Grundlegend für die Theorie ist die Semantik der Erklärungen, genauer deren *Semantic Gravity* und *Semantic Density* (siehe Abbildung 12.3):

> *„Semantic gravity is about how contextualised language or examples are for the learner. It explores the context of meanings and how much meaning depends on the social context to make sense. So where meanings have greater dependence on the context (such as practical examples or personal experience) semantic gravity is stronger and where meanings are more abstract (such as theory), semantic gravity is weaker. [...] Semantic density is concerned with the use of technical and everyday knowledge. It explores the complexity of meanings rather than their context. Where meanings are relatively simple, such as describing something in everyday language, semantic density is weaker. Where meanings are more complex, such as using technical concepts, semantic density is stronger."* (Waite u. a. 2019, S. 2)

Die *Semantic Gravity* und *Semantic Density* kann sich im Laufe einer Lernerfahrung verändern und als *Semantic Profile* dargestellt werden (siehe Abbildung 12.4). In zahlreiche Untersuchungen in

[3] https://classic.csunplugged.org/

verschiedenen Fächern haben sich vor allem wellenförmige Profile (*Semantic Waves*) als besonders lernförderlich herausgestellt, während sogenannte *Flatlines* den Lernprozess zu behindern scheinen (ebd., S. 3). Curzon u. a. (2020, S. 8) unterstreichen in diesem Kontext die Wichtigkeit des *Unpacking* und *Repacking* (siehe Abbildung 12.4) und zeigen verschiedene Möglichkeiten zur Umsetzung im Unterricht auf, die sich in Fallstudien als wirksam erwiesen haben, z. B. das Verknüpfen einer Codezeile mit einer *Unplugged*-Aktivität (*Unpacking*) oder das Bereitstellen von *Follow On Pencil and Paper Exercises*, die sich direkt auf die *Unplugged*-Aktivität und das jeweilige Programmierkonstrukt beziehen (*Repacking*).

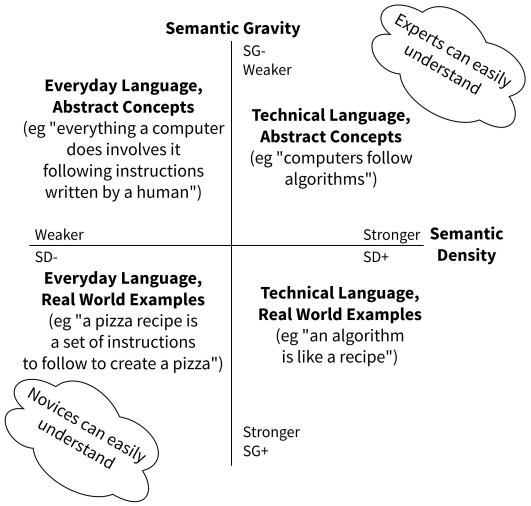



*Abbildung 12.3 Darstellung der ‚Semantic Density' und ‚Semantic Gravity' aus Waite u. a. (2019, S. 2)*

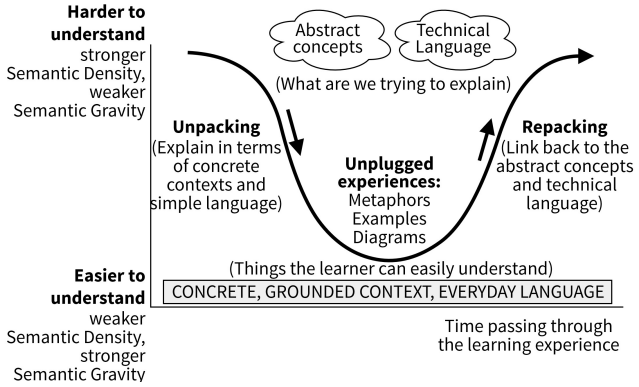



*Abbildung 12.4 Darstellung eines Unterrichtsgeschehen als ‚Semantic Profile' aus Curzon u. a. (2020, S. 2)*

Der Einsatz von *Unplugged*-Aktivitäten wird auch für die Gestaltung von Lehrerfortbildungen – besonders für Lehrkräfte ohne Vorerfahrungen – empfohlen:

> *„Since learning to teach programming can be daunting for teachers with no experience in this area, a CS Unplugged introduction can help to break down defenses and overcome teachers' fears. Using an ‚Unplugged' approach means that relatively deep ideas can be covered fairly quickly in a playful and empowering way, using familiar materials (paper, string, chalk and so on), rather than having to install and learn to use complex software. This provides quick success and a positive experience for teachers who might otherwise find it difficult to become engaged with the subject."*
>
> (Bell und Vahrenhold 2014, S. 502 f.)

### 12.2.2 Softwarebasierte Einstiege in die Programmierung

Auch wenn sich die Grundlagen der Programmierung ohne den Einsatz von Informatiksystemen thematisieren lassen, hat das Programmieren eines Informatiksystems auch in der Grundschule aus mehreren Gründen seine Berechtigung. Computer sowie programmierbare Roboter scheinen auf Kinder eine besondere Anziehungskraft zu haben und eignen sich daher besonders, um Neugierde

und Interesse an der Informatik zu wecken (z. B. Hermans und Aivaloglou 2017, S. 53 f.). Darüber hinaus eröffnen sich beim Programmieren eines Informatiksystems andere Gestaltungsmöglichkeiten für die Lernenden. Dies kommt Paperts Lerntheorie des Konstruktionismus entgegen, die das aktive Konstruieren in den Mittelpunkt stellt (vgl. Papert und Harel 1991). Mittlerweile gibt es zahlreiche Programmierumgebungen und Werkzeuge, die eigens gestaltet wurden, um Kindern einen softwarebasierten Einstieg in die Programmierung zu ermöglichen (vgl. Duncan, Bell und Tanimoto 2014, S. 65 ff.). Dabei haben sich vor allem blockbasierte Programmiersprachen – eine Form der visuellen Programmiersprachen (siehe Abschnitt 9.2) – hervorgetan, die Paperts Anforderungen der *low floors*, *high ceilings* und *wide walls* entsprechen (vgl. Weintrop und Grover 2020, S. 100 f.):

> *„Papert argued that programming languages should have a ‚low floor' (easy to get started) and a ‚high ceiling' (opportunities to create increasingly complex projects over time). In addition, languages need ‚wide walls' (supporting many different types of projects so people with many different interests and learning styles can all become engaged)."* (Resnick u. a. 2009, S. 63)

Ein Ansatz, Schülerinnen und Schülern die Grundlagen der Programmierung näher zu bringen und ihnen gleichzeitig kreatives Arbeiten zu ermöglichen, ist die Programmierumgebung Scratch[4], die am Massachusetts Institute of Technology (MIT) entwickelt wurde (vgl. Maloney u. a. 2004). Scratch bietet eine visuelle Programmiersprache, in der man den Programmcode aus Blöcken per *Drag and Drop* zusammenbaut, ohne Fehlermeldungen zu erhalten:

> *„Scratch scripts are built by snapping together blocks representing statements, expressions, and control structures. The shapes of the blocks suggest how they fit together, and the drag-and-drop system refuses to connect blocks in ways that would be meaningless. In Scratch, the visual grammar of block shapes and their combination rules play the role of syntax in a text-based language."* (Maloney u. a. 2010, S. 7)

Die Oberfläche von Scratch unterteilt sich in mehrere Bereiche (siehe Abbildung 12.5): In der Blockpalette (1) befinden sind sämtliche Programmblöcke, die sich nach rechts in den Programmierbereich (2) schieben und dort miteinander kombinieren lassen. Auf der rechten Seite befindet sich die Bühne (3), auf der die Figuren das ausführen, was man für sie programmiert hat. Insgesamt stehen den Lernenden 142 Blöcke zur Verfügung, die verschiedenen Blockkategorien zugeordnet sind: *Bewegung*, *Aussehen*, *Klang*, *Ereignisse*, *Steuerung*, *Fühlen*, *Operationen* und *Variablen*. In Kombination mit einer großen Auswahl an Figuren (*Sprites*) und Bühnenbildern können mittels der Blöcke viele verschiedene Arten von Programmen entwickelt werden, z. B. Geschichten, Spiele oder Animationen, die sich durch das Hochladen eigener Fotos, Tonaufnahmen sowie das Erstellen eigener Grafiken weiter personalisieren lassen (Resnick u. a. 2009, S. 64). Trotz der anfängerfreundlichen Gestaltung der Programmiersprache und -umgebung, können sämtliche algorithmischen Grundstrukturen (siehe Abschnitt 9.1) in Scratch realisiert werden (siehe Abbildung 12.6).

[4] https://scratch.mit.edu/projects/editor/

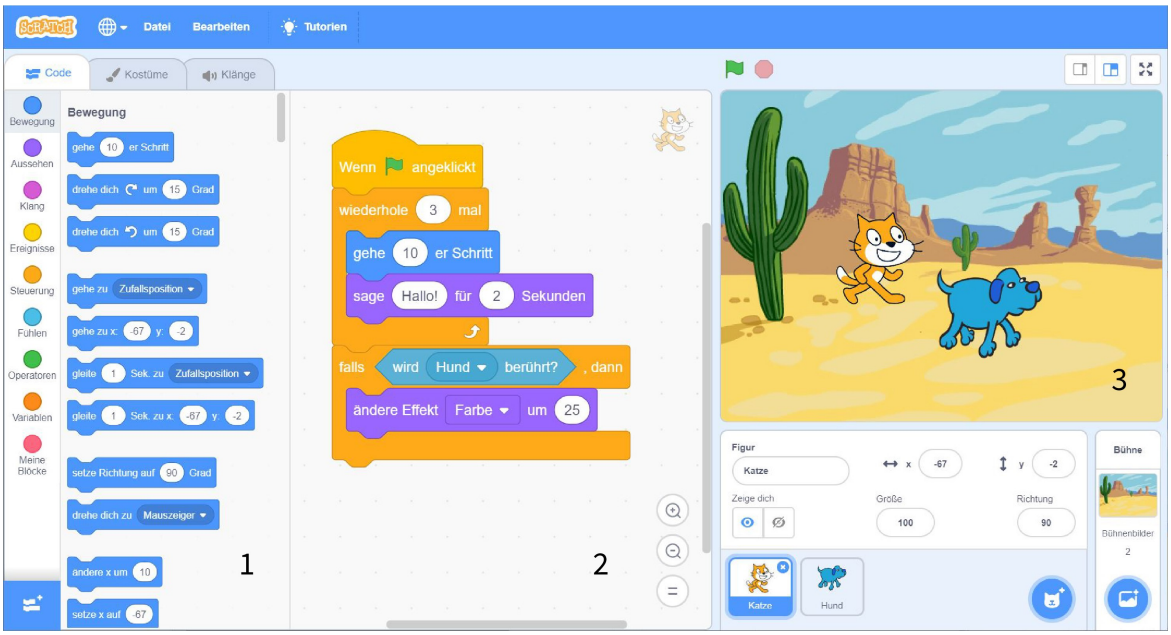



*Abbildung 12.5 Die Oberfläche von Scratch*

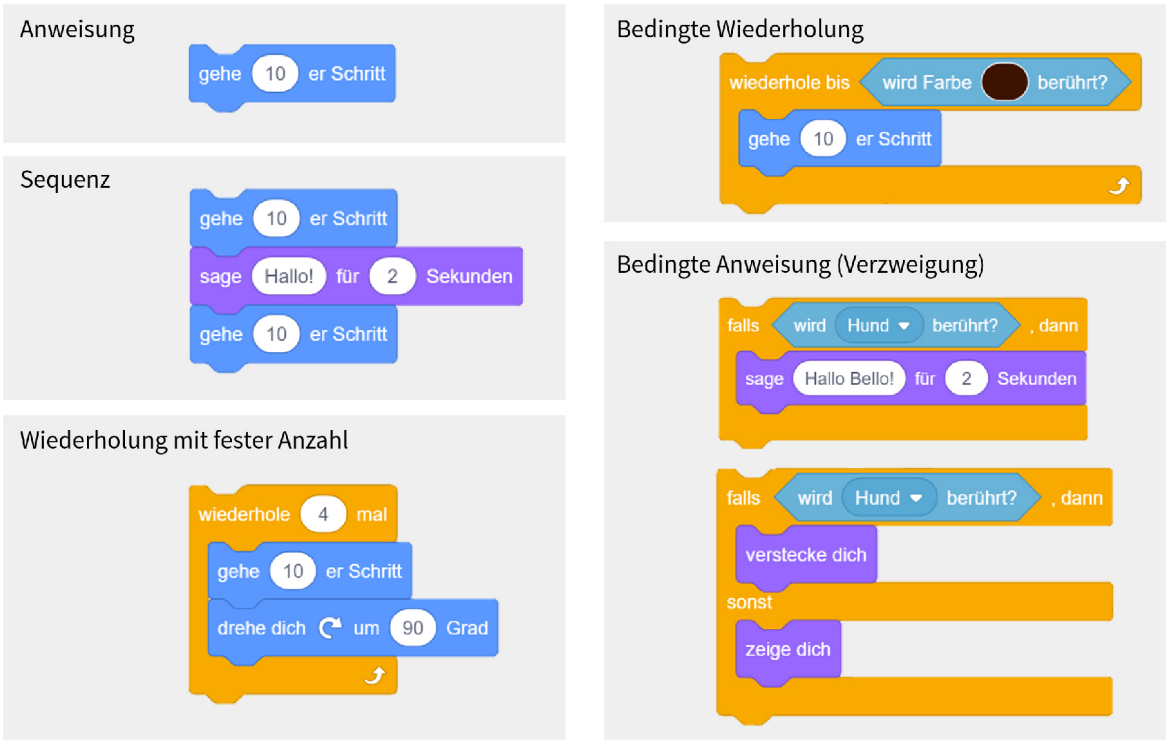



*Abbildung 12.6 Algorithmische Strukturelemente in Scratch*

Der Einsatz von Scratch im Unterricht und die Art und Weise, wie Lernende in Scratch programmieren wurde bereits in zahlreichen Studien untersucht. Hier finden sich bspw. Studien, welche die Programmierergebnisse von Lernenden in Hinblick auf verwendete Programmstrukturen (z. B. Adams und Webster 2012; Amanullah und Bell 2019; Moreno-Leon, Robles und Roman-Gonzalez 2020), Unterschiede in den Programmen von Mädchen und Jungen (z. B. Hsu 2014; Aivaloglou und Hermans 2019) oder lernhinderliche Programmiergewohnheiten (z. B. Hermans und Aivaloglou

2016; Frädrich u. a. 2020) analysieren. Darüber hinaus wurden bspw. konkrete Einsatzszenarien der Programmiersprache im Unterricht (z. B. Franklin u. a. 2015; Papadakis u. a. 2016; Iskrenovic-Momcilovic 2019) oder die Förderung von *Computational Thinking* durch das Programmieren in Scratch (z. B. Zhang und Nouri 2019; Fagerlund u. a. 2020) untersucht.

### 12.2.3 Programmierbares Spielzeug (Robotik)

Ein Ansatz, der ebenfalls auf Papert zurückgeht, ist die Einbindung von physischen Objekten im Informatikunterricht (vgl. Bergner u. a. 2018, S. 95). Dieser Ansatz des *Physical Computing* bezieht sich auf den Einsatz *„of both software and hardware to build interactive physical systems that sense and respond to the real world“* (Sentance und Childs 2020, S. 250). Die Hardware, die dabei eingesetzt wird, hat sich in den vergangenen Jahren zunehmend ausdifferenziert und umfasst unterschiedlichste Geräte (siehe Abbildung 12.7) – einen Überblick liefern Hodges u. a. (2020, S. 26). Für den Einsatz in der Grundschule werden sowohl spielzeugartige Geräte empfohlen, die zumeist batteriebetrieben sind und ohne einen Computer programmiert werden können (z. B. den *BeeBot*[5] oder *Ozobot*[6]) als auch am Computer programmierbare Bausätze, wie *LEGO WeDo*[7] (vgl. Bergner und Müller 2018, S. 272 ff.). Darüber hinaus werden vermehrt Microcontroller, wie der *Calliope mini*[8] oder *micro:bit*[9] im Unterricht der Grundschule erprobt und eingesetzt (z. B. Baum u. a. 2019; Sentance u. a. 2017b). Mit dem Einsatz von physischen Objekten beim Programmieren verspricht man sich unterschiedliche Nutzen:

> *„Nonphysical computing is screen-based, so the success (or not!) of the program is indicated by something happening on the screen. With a physical decive, instant feedback from the device shows whether the program code works as desired. This can be rewarding as well as accelerate learning.“* (Sentance und Childs 2020, S. 250)

> *„In the form of programmable and interactive robots, computers can become powerful learning tools. Robotics offers children the opportunity to engage with content from the domain of computer science, practice problem-solving skills, and work on fine-motor skills and eye–hand coordination.“* (Bers u. a. 2014, S. 146)

Der Einsatz von Elementen des *Physical Computing* in der Grundschule ist Gegenstand vieler Untersuchungen, die sich bspw. mit der Sichtweise von Lehrpersonen (z. B. Jaipal-Jamani und Angeli 2017; Sentance u. a. 2017a), Unterrichtsansätzen (z. B. Scaradozzi u. a. 2015; Camilleri 2017) oder Auswirkungen auf die Lernenden (z. B. Chalmers 2018; Theodoropoulos u. a. 2018) beschäftigen.

[5] https://www.b-bot.de/produkte/bee-bots/
[6] https://ozobot.com/
[7] https://education.lego.com/de-de/products/lego-education-wedo-2-0-set/45300#wedo-20-set
[8] https://calliope.cc/
[9] https://microbit.org/

*Abbildung 12.7 Calliope mini (links), BeeBot (Mitte) und LEGO WeDo (rechts); Quelle: eigene Aufnahmen*

## 12.3 Unterrichtsgestaltung

Die folgenden Aspekte beziehen sich auf die konkrete Gestaltung von Lehr-Lernsituationen zum Programmieren, beziehen sich jedoch nur vereinzelt explizit auf den Bereich der Grundschule. Die Strukturierung orientiert sich an den Entscheidungsfeldern des *Berliner Modells*.

### 12.3.1 Intentionen und Inhalte

*Algorithmische Grundstrukturen*

Laut Hubwieser (2007, S. 82 f.) sollten Lerninhalte im Informatikunterricht einen möglichst breiten Anwendungsbereich abdecken. Dies trifft zu, wenn sie nicht nur ein konkretes System betreffen, sondern z. B. relevant für alle Informatiksysteme sind oder auch außerhalb der Informatik angewandt werden können. Bell (2016, S. 6) unterstreicht, dass der unterrichtliche Fokus beim Programmieren auf den algorithmischen Strukturen der Sequenz, Wiederholung und bedingten Anweisung liegen sollte. Zum einen finden sie sich in den meisten Programmiersprachen wieder, zum anderen sind es im Prinzip die einzigen Kontrollstrukturen, die für die Programmierung eines Informatiksystems notwendig sind. Er führt weiter aus, dass man beim Unterrichten der Programmierung auch allgemeine Herangehensweisen zur Lösung eines Problems vermitteln sollte.

*Förderung von Problemlösefähigkeit*

Die Anbahnung von Problemlösefähigkeiten durch das Programmieren wird durch wachsendes Interesse am Konzept des *Computational Thinking* vermehrt diskutiert und untersucht (z. B. Zhang und Nouri 2019; Fagerlund u. a. 2020). Hazzan, Lapidot und Ragonis (2011, S. 64 f.) betonen in ihrem *Guide to Teaching Computer Science*, dass die Entwicklung von Problemlösefähigkeiten im Zentrum jedes Informatikunterrichts stehen sollte. Sie schlagen verschiedene Schritte vor, welche die Lernenden bei der Lösung eines Problems durchlaufen sollten, machen jedoch gleichzeitig darauf aufmerksam, dass diese nicht linear durchlaufen werden müssen und zum Teil ineinandergreifen:

> „1. ***Problem analysis***. *Understand and identify the problem, understand what the problem is about. If one does not understand the problem, one cannot proceed and solve it.*

2. ***Alternative consideration***. *Think about alternative ways to solve the problem.*
3. ***Choosing an approach***. *Choose an appropriate approach to solve the problem.*
4. ***Problem decomposition***. *Decompose the problem into subtasks.*
5. ***Algorithm development***. *Develop the algorithm in stages according to the recognized subtasks.*
6. ***Algorithm correctness***. *Check the algorithm correctness.*
7. ***Algorithm efficiency***. *Calculate the algorithm efficiency.*
8. ***Reflection***. *Reflect on and analyze the process you went through, conclude what can be improved in future problem-solving processes."*

(Hazzan, Lapidot und Ragonis 2011, S. 64 f.)

Darüber hinaus wurden weitere Vorgehensmodelle zum *Problem Solving* entwickelt. Winslow (1996, S. 19) beschreibt, dass das Lösen von Problemen in die folgenden Schritte aufgeteilt werden kann: *„1. understand the problem, 2. determine how to solve the problem: a. in some form and b. in computer compatible form [...], 3. translate the solution into computer language program, and 4. test and debug the program."* Die OECD (2013, S. 126) beschreibt in ihrem *Assessment and Analytical Framework* der PISA-Studie 2012, dass beim Problemlösen die folgenden Prozesse beteiligt sind: *„exploring and understanding, representing and formulating, planning and executing, monitoring and reflecting".*

### 12.3.2 Methodik

*Planungsprozesse beim Programmieren*

Je komplexer Programme sind, desto offensichtlicher ist es, dass man nicht direkt mit dem Schreiben des Codes beginnt, sondern einen Plan benötigt, wie das jeweilige Ziel erreicht werden kann. Bagge und Grover (2020) beschreiben vier Abstraktionsebenen, auf denen Planungsprozesse beim Programmieren stattfinden, die im Informatikunterricht berücksichtigt werden sollten und auf der Arbeit von Armoni (2013) und Waite u. a. (2018) basieren. Auf dem *Problem Level* beschreiben die Schülerinnen und Schüler die Idee, die sie mit dem Programm umsetzen wollen. Dazu gehört, wie die Eingabe und die gewünschte Ausgabe aussieht, und für wen das Programm bestimmt ist. Auf dem *Algorithm Level* wird dieser Plan detaillierter und die Schülerinnen und Schüler beschreiben den Algorithmus mittels der algorithmischen Strukturelemente. Der Algorithmus wird auf dem *Program Level* in einer Programmiersprache formuliert und auf dem *Program Execution Level* ausgeführt und bei Bedarf modifiziert. Das Planen von Programmen und das Durchdenken des Algorithmus wirkt sich laut Statter und Armoni (2020, S. 31) positiv auf die Lernergebnisse aus und sollte im Unterricht gefördert werden, obgleich Schülerinnen und Schüler in der Regel lieber direkt mit dem Schreiben des Codes beginnen.

*Verbessern von Programmen*

Beim Programmieren geht es nicht nur um die Erstellung eines Algorithmus und dessen Implementierung in einer Programmiersprache. Wenn ein Programm nicht das gewünschte Ergebnis hervorbringt, müssen Fehler im Programm erkannt und in einem nächsten Schritt korrigiert werden (Rich und Strickland 2020, S. 213). Das Beheben von Fehlern stellt besonders für Programmieranfängerinnen und -anfänger eine Herausforderung dar (vgl Michaeli 2020, S. 46 f.) und sollte deshalb im Unterricht thematisiert werden. Dadurch können sich Schülerinnen und Schüler zum einen konkrete Problemlösestrategien aneignen, zum anderen können sie ein Bewusstsein dafür entwickeln, dass es beim Programmieren ganz natürlich ist, Fehler zu machen (Wong und Jiang 2018, S. 329). Für das Suchen und Beheben von Fehlern in Programmen schlagen Rich und Strickland (2020, S. 213 ff.) verschiedene Strategien vor: Um einen Fehler zu lokalisieren, soll das Programm zunächst Zeile für Zeile gelesen und ausgeführt werden, lange Programme können in kleinere Skripte aufgeteilt werden, die wiederum einzeln getestet werden. Um Fehler zu beheben empfehlen sie, fehlerhafte Skripte mit ähnlichen funktionierenden zu vergleichen oder das Problem auf Ebene des Algorithmus zu betrachten.

*Programmieren als gemeinsame Tätigkeit*

Ein Ansatz im Informatikunterricht, der den Fokus auf Zusammenarbeit und Absprache legt, ist das *Pair Programming* (vgl. Braught, Wahls und Eby 2011, S. 1 f.). Hier schreiben zwei Lernende gemeinsam ein Programm und teilen sich dabei einen Computer. Dabei nehmen sie im Wechsel zwei verschiedene Rollen ein – *Driver* und *Navigator*. Der *Driver*, der die Maus und Tastatur bedient, erstellt das Programm nach Angaben des *Navigator*. So kann er sich ganz auf das Schreiben des Codes bzw. die Bedienung der Programmierumgebung konzentrieren. Die Aufgabe des *Navigator* ist es, im Auge zu behalten, wie man die Aufgabe oder das Problem lösen kann. Er liest dem *Driver* außerdem Arbeitsaufträge vor und sucht nach Fehlern im Programm. Studien haben ergeben, dass *Pair Programming* bei Schülerinnen und Schülern beliebt ist (Luxton-Reilly u. a. 2018, S. 70), und vor allem leistungsschwächere Lernende im Lernprozess unterstützen kann (Sands 2019, S. 47).

*Praktische Hinweise zur Umsetzung in der Grundschule*

Franklin u. a. (2015, S. 553 ff.) formulieren verschiedene Ratschläge für die Umsetzung des Programmierens im Grundschulunterricht, die ihren Erfahrungen in der Entwicklung, Pilotierung und Evaluation eines auf Scratch basierenden *Computational Thinking*-Curriculums entstammen:

- Programmierumgebung
  Das Herunter- und Hochladen von Dateien von einem gemeinsam genutzten Computer bzw. Server sollte vermieden werden. Falls die vorhandene Internetverbindung nicht stabil ist, sollte eine lokale Installation der Programmierumgebung der Nutzung im Browser vorgezogen werden.
- Unterrichtsmaterialien
  Arbeitsblätter sollten im Computerraum nur eingeschränkt verwendet werden. Um den unterschiedlichen Leistungsständen der Schülerinnen und Schüler im Schriftspracherwerb entgegen-

zukommen, sollten die Schreibanforderungen bei der Gestaltung von Arbeitsblättern reduziert werden, z. B. indem die Lernenden Antworten einkreisen, Bilder zeichnen oder lediglich Lücken ausfüllen.

- Im Klassenraum
  Für die Umsetzung des Programmierens wird empfohlen, dass alle Schülerinnen und Schüler mit einer Computermaus ausgestattet werden, da beim Einsatz von Laptops Schwierigkeiten mit dem Touchpad aufkamen. Um der unterschiedlichen Arbeitsgeschwindigkeit der Kinder zu begegnen, sollten Zusatzmaterialien bzw. zusätzliche Arbeitsaufträge zur Verfügung stehen. Um die Aufmerksamkeit der Schülerinnen und Schüler zu erlangen, sollten Signale eingeübt werden, die automatisch dazu führen, dass sie ihre Hände von der Tastatur entfernen, z. B. Klatschen oder Verschränken der Hände hinter dem Nacken.
- Feedback
  Die Lehrkraft sollte sich bei Programmieraufgaben nicht zu sehr auf eine einzige korrekte Lösung versteifen, sondern die übergeordneten Ziele im Blick behalten. Feedback sollte so formuliert werden, dass es die Schülerinnen und Schüler nicht entmutigt, z. B. durch die Formulierung *You are not finished yet* anstatt *The answer is incorrect*.

### 12.3.3 Medien/Materialien

*Programmieraufgaben*

Aufgaben sind ein zentraler Bestandteil des Informatikunterrichts und spielen sowohl für das Erarbeiten neuer Inhalte als auch für die Ermittlung des Leistungsstands eine wichtige Rolle. Besonders beim Programmierenlernen wird betont, dass sich Expertise nur durch praktische Aufgaben sowie intensives Üben einstellen kann (vgl. Hassinen und Mäyrä 2006). Wie der Prozess des Programmierenlernens gestaltet werden sollte, ist jedoch nicht abschließend geklärt.

Caspersen und Bennedsen (2007) ziehen Theorien der Kognitionswissenschaft und pädagogischen Psychologie zur Gestaltung eines einführenden Programmierkurses für Studierende heran – vorrangig die *Cognitive Load Theory*, *Theory of Cognitive Apprenticeship* und *Theory of Worked Examples* (siehe Abschnitt 10.2). Um die Lernenden beim Verstehen der Programmierkonzepte zu unterstützen, empfehlen sie, mit mehreren Beispielen zu arbeiten, die an Komplexität zunehmen und im Problemtyp variieren. Diese sollten im Wechsel mit passenden Aufgaben eingesetzt werden (ebd., S. 188 f.). Gray u. a. (2007, S. 101) weisen in diesem Kontext darauf hin, dass der Übergang von *Complete Worked Examples* zum eigenständigen Lösen von Aufgaben nur funktioniert, wenn diese einfach gehalten sind. Um die kognitive Belastung für die Lernenden weiter zu reduzieren, schlagen sie den Einsatz von *Fading Worked Examples* vor, die immer weniger Teile der Lösung beinhalten und die Lernenden schrittweise an das eigenständige Lösen von Aufgaben heranführen.

Auch Lee u. a. (2011) sowie Sentance und Waite (2017) beschreiben didaktische Ansätze, in denen Lernende auf dem Weg zum eigenständigen Programmieren begleitet werden können ohne sie zu überfordern. Lee u. a. (2011, S. 35) schlagen die *Use-Modify-Create Learning Progression* vor,

in der die Lernenden zunächst mit bestehenden Programmen arbeiten, diese modifizieren und schließlich eigene Programme erstellen. Darauf folgt ein iterativer Prozess des Testens, Analysierens und Verfeinerns der Programme. Sentance und Waite (2017) führen den Ansatz *PRIMM* zum Lernen textbasierter Programmiersprachen ein, der auf der Arbeit von Lee u. a. aufbaut und die folgenden Schritte umfasst: *Predict*, *Run*, *Investigate*, *Modify* und *Make*. In beiden Ansätzen arbeiten die Schülerinnen und Schüler zunächst mit bereits vorhandenen Programmen und machen sich diese nach und nach zu eigen, bis sie komplett eigene Programme erstellen. Brennan (2013) verweist darauf, dass es in Hinblick auf die längerfristige Motivation der Schülerinnen und Schüler wichtig ist, dass sie Gelegenheiten erhalten, ihre eigenen Ideen für Programme und Projekte im Unterricht umzusetzen.

*Aspekte der Inklusion*

Mit der Ratifizierung der UN-Behindertenrechtskonvention im Jahr 2009 sind Bund, Länder und Kommunen in Deutschland verpflichtet, das Recht von Menschen mit Behinderungen auf selbstbestimmte und gleichberechtigte Teilhabe an der Gesellschaft umzusetzen. In diesem Kontext wird gefordert, dass Kinder mit Behinderungen inklusiv – gemeinsam mit nicht behinderten Kindern – an allgemeinen Schulen unterrichtet werden (vgl. Beauftragter der Bundesregierung für die Belange von Menschen mit Behinderungen 2007, Art. 24). Das Thema Inklusion wurde für den Bereich der informatischen Bildung bislang vorrangig im Kontext der weiterführenden Schulen und der universitären Bildung behandelt, z. B. am Beispiel von Menschen mit Sehschädigung (Capovilla 2015, Akao und Fischer 2022) oder zur Klärung der Frage, welche Rolle das Thema im Lehramtsstudium spielt bzw. wie es umgesetzt werden kann (Akao und Fischer 2021, Ehlenz, Heinemann und Schroeder 2022). In der Auseinandersetzung mit der Lerntypentheorie weist Capovilla (2015, S. 35 ff.) auf eine Tendenz zur Visualisierung von Unterrichtsinhalten im Bildungsbereich hin, die für Menschen mit Sehschädigung problematisch ist. So erfordern z. B. das Beobachtungslernen und der Einsatz von Visualisierungstechniken sowie *Algorithmic Visualizations* eine methodischen Erweiterung, die den besonderen Lernvoraussetzungen Rechnung trägt. Für die informatische Bildung stellt Capovilla (2019, S. 39) den Ansatz des *Universal Design for Learning* (Rose und Meyer 2002) als geeignet heraus, der für die vier Handlungsfelder Lernziele, Methoden, Bewertung und Material eine Vielfalt der Handlungs- und Ausdrucksmodi, der Motivations- und Intentionsmodi sowie der Repräsentationsmodi vorschlägt.

# 13 Hintergrund zur Lehrerbildung

Analog zum Erfolg von Unterrichtssituationen wird auch die Wirksamkeit von Fortbildungen für Lehrpersonen von verschiedenen Faktoren beeinflusst:

> „Für die Entwicklung von Lehrpersonen im Rahmen von Fortbildungs- und Professionalisierungsmaßnahmen spielen individuelle Determinanten, kontextuelle Bedingungen sowie strukturelle und didaktische Merkmale der Fortbildungen und deren Interaktionen eine entscheidende Rolle. Diese Variablenbündel beeinflussen die Art und Weise, wie Lehrpersonen die Lernangebote wahrnehmen, nutzen und verarbeiten, was sie von den Fortbildungen ‚mitnehmen', wie gut und intensiv sie die Anregungen in ihrem Unterrichtsalltag umsetzen und wie stark sie selbst und ihre Schüler/innen davon profitieren." (Lipowsky 2010, S. 62)

In Lipowskys *Angebots-Nutzungs-Modell* zur Wirksamkeit von Fortbildungs- und Professionalisierungsmaßnahmen (Lipowsky 2010, S. 63) nimmt das Fortbildungsangebot eine zentrale Stellung ein (siehe Abbildung 13.1). Dieses wird bestimmt durch strukturelle Merkmale, inhaltliche Merkmale, die Aktivitäten innerhalb der Fortbildung sowie durch die Expertise der Referentinnen und Referenten und Moderierenden (ebd., S. 64).

Da die genannten Faktoren bei der Gestaltung einer Fortbildungsmaßnahme berücksichtigt werden sollten, werden sie im Folgenden näher beleuchtet. Dabei wird vor allem bei den inhaltlichen Merkmalen der Fokus auf die Programmierung bzw. Informatik im Allgemeinen gelegt. Auf die Expertise der mitwirkenden Personen wird nicht weiter eingegangen, da dieser Faktor in der vorliegenden Arbeit nicht beeinflusst werden kann.

## 13.1 Strukturelle Merkmale

Die folgenden Aspekte beziehen sich auf verschiedene Rahmenbedingungen von Fortbildungsmaßnahmen. Beschrieben wird, wie diese die Maßnahmen beeinflussen können.

### 13.1.1 Dauer

Grundlegend für den Erfolg von Fortbildungsmaßnahmen und nachhaltiges berufliches Lernen ist das Bereitstellen von expliziten Lernzeitfenster, die von anderen Tätigkeiten freigehalten werden (Schmidt-Lauff 2010, S. 221). Sogenannte *One-Shot*-Fortbildungen, die lediglich einen halben oder ganzen Tag umfassen, werden in diesem Kontext kritisiert. Einerseits wird angezweifelt, dass auf diesem Weg Routinen und Handlungsmuster, die sich über einen längeren Zeitraum ausgebildet

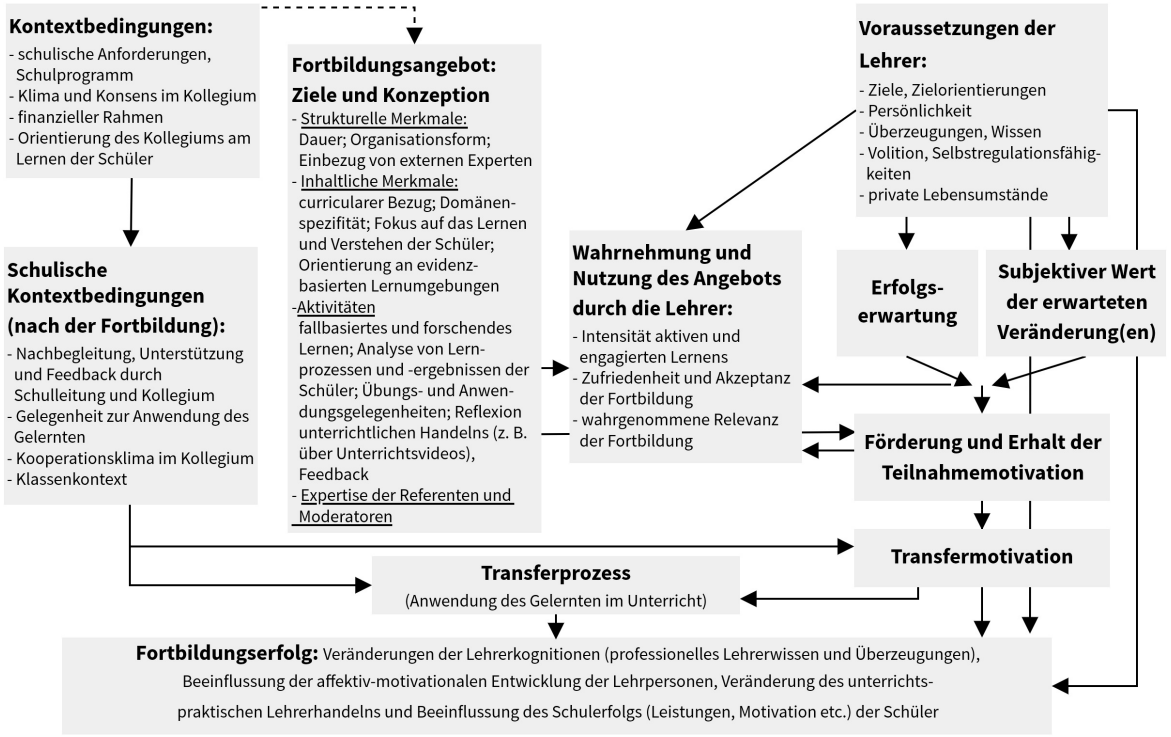



*Abbildung 13.1 Erweitertes Angebots-Nutzungs-Modell zur Erklärung der Wirksamkeit von Fortbildungs- und Professionalisierungsmaßnahmen für Lehrpersonen aus Lipowsky (2010, S. 63)*

und verfestigt haben, verändern lassen (z. B. Lipowsky und Rzejak 2012, S. 5), andererseits wünschen sich auch Lehrkräfte selbst längere Fortbildungsangebote (vgl. Jäger und Bodensohn 2007, S. 21 f.). Auch wenn nicht davon ausgegangen werden kann, dass längere Fortbildungen automatisch erfolgreicher sind (Timperley u. a. 2007, S. 75), können sie „in der Regel mehr Gelegenheiten für aktives Lernen und wiederholtes Erproben neuer Handlungsmuster bieten und somit Veränderungen im Lehrerwissen und -handeln erleichtern“ (Lipowsky und Rzejak 2012, S. 5). Penuel u. a. weisen zudem darauf hin, dass sich Fortbildungsmaßnahmen insgesamt über einen längeren Zeitraum erstrecken sollten, um eine Umsetzung der Inhalte in der Praxis zu ermöglichen:

> *„There is growing consensus that to make [substantive changes], teachers need professional development that is interactive with their teaching practice, allowing for multiple cycles of presentation and assimilation of, and reflection on, knowledge (Blumenfeld, Soloway, Marx, Guzdial, & Palincsar, 1991; Kubitskey, 2006). Professional development that is of longer duration and time span is more likely to contain the kinds of learning opportunities necessary for teachers to integrate new knowledge into practice (J. L. Brown, 2004).“* (Penuel u. a. 2007, S. 929)

### 13.1.2 Gruppenzusammensetzung

In Bezug auf die Zusammensetzung der Teilnehmergruppe sehen Garet u. a. (2001) mehrere Vorteile darin, dass mehrere Lehrkräfte einer Schule an einer Fortbildung teilnehmen:

> *„First, teachers who work together are more likely to have the opportunity to discuss concepts, skills, and problems that arise during their professional development experiences. Second, teachers who are from the same school, department, or grade are likely to share common curriculum materials, course offerings, and assessment requirements. By engaging in joint professional development, they may be able to integrate what they learn with other aspects of their instructional context. Third, teachers who share the same students can discuss students' needs across classes and grade levels. Finally, by focusing on a group of teachers from the same school, professional development may help sustain changes in practice over time, as some teachers leave the school's teaching force and other new teachers join the faculty."* (Garet u. a. 2001, S. 922)

Diese Vorteile zeigen sich auch in den Ergebnissen ihrer *Large-Scale*-Studie, in der sie die Effekte unterschiedlicher Charakteristika von Fortbildungen auf den Lernerfolg von Lehrkräften untersuchen (ebd., S. 936). Hier zeigen sich Zusammenhänge zwischen der Teilnahme von Lehrkräften derselben Schule und ihrem Lernerfolg sowie Veränderungen in der Unterrichtspraxis. Die Befunde von Kunnari, Ilomäki und Toom (2018) zeigen zudem, dass eine stärkere Zusammenarbeit zwischen Lehrkräften im Rahmen von Fortbildungsmaßnahmen zu signifikanten Veränderungen in deren Unterrichtspraxis und dem Lernerfolg der Schülerinnen und Schüler führt.

### 13.1.3 Fortlaufende Unterstützung

Verschiedene Forschungsergebnisse legen nahe, dass Coaching-Angebote und andauernde Unterstützungsmaßnahmen für Lehrkräfte das Implementieren neuer Lehrpläne, Werkzeuge oder Praktiken erheblich begünstigen können (z. B. Penuel, Gallagher und Moorthy 2011; Roth u. a. 2011). Ein Coaching-Angebot scheint darüber hinaus die Wahrscheinlichkeit zu erhöhen, dass Lehrkräfte neue Inhalte oder Methoden angemessen implementieren und das Gelernte weiterhin umsetzen (Knight 2010, S. 289).

## 13.2 Aktivitäten

Im Folgenden werden Aspekte beschrieben, die sich auf die didaktische Gestaltung von Fortbildungen beziehen. Dabei steht nicht im Vordergrund, mit welchen Inhalten sich die Lehrkräfte auseinandersetzen, sondern wie diese Auseinandersetzungen angelegt sind.

### 13.2.1 Aktive Teilnahme

Die aktive Auseinandersetzung mit dem Lerngegenstand wird in vielen Publikationen als Voraussetzung für erfolgreiche Fortbildungsmaßnahmen gesehen (z. B. Darling-Hammond u. a. 2017; Birman u. a. 2000; Penuel u. a. 2007). Beispiele für *Active Learning* sind nach Birman u. a. (2000, S. 32) *„opportunities to observe and be observed teaching; to plan classroom implementation, such as practicing in simulated conditions and developing lesson plans; to review student work; and to*

*present, lead, and write – for example, present a demonstration, lead a discussion, or write a report*“. Das Bereitstellen von Zeiträumen zur Planung der Implementierung von Fortbildungsinhalten erhöht darüber hinaus die Chance einer tatsächlichen Umsetzung im Unterricht der teilnehmenden Lehrkräfte (vgl. Penuel u. a. 2007, S. 952).

Laut Darling-Hammond u. a. (2017) unterscheiden sich Maßnahmen des *Active Learning* maßgeblich vom Vorlesungsformat, in dem die Teilnehmenden in erster Linie die Rolle der Rezipienten einnehmen. Im Idealfall stellen sie eine Verbindung zur Unterrichtspraxis der Lehrkräfte dar:

> „‚*Active learning‘ suggests moving away from traditional learning models that are generic and lecture based toward models that engage teachers directly in the practices they are learning and, preferably, are connected to teachers’ classrooms and students. Active learning, in sharp contrast to sit-and-listen lectures, engages educators using authentic artifacts, interactive activities, and other strategies to provide deeply embedded, highly contextualized professional learning.*“
>
> (Darling-Hammond u. a. 2017, S. 7)

### 13.2.2 Lernpfad der Schülerinnen und Schüler folgen

Eine Möglichkeit, Lehrkräften aktives Lernen zu ermöglichen, ist mit Materialien zu arbeiten, die sie im Anschluss in ihrem eigenen Unterricht anwenden können, und ihnen auf diesem Weg zu ermöglichen, dem Lernprozess der Schülerinnen und Schüler selbst zu folgen (Darling-Hammond u. a. 2017, S. 8). Diese Herangehensweise hat sich in verschiedenen Studien als wirksam erwiesen (z. B. Buczynski und Hansen 2010; Heller u. a. 2012) und ist vor allem in den Naturwissenschaften schon lange verbreitet. In diesen Disziplinen ist es üblich, dass Lehrkräfte die Möglichkeit bekommen, den *Conceptual Change* der Lernenden – die Veränderung der ursprünglichen Vorstellungen (vgl. Möller 2015, S. 244 f.) – selbst zu erleben. Auf diese Weise können sie dessen Bedeutung besser einschätzen und im Unterricht gezielt evozieren (vgl. Ho, Watkins und Kelly 2001, S. 145 f.). Dazu kommt, dass Lehrerinnen und Lehrer Fortbildungen als zufriedenstellend erleben, wenn sie sich auf den konkreten Unterrichtsalltag und auf das Curriculum beziehen (Lipowsky 2010, S. 52). Darüber hinaus hat es sich als wirksam erwiesen, Lehrkräften in Fortbildungen Beispiele für effektive Umsetzungen der jeweiligen Fortbildungsinhalte zugänglich zu machen und ihnen auf diesem Wege Anknüpfungspunkte für die eigene Unterrichtspraxis zu eröffnen. Darling-Hammond u. a. (2017, S. 11) nennen als Beispiele dafür „*video or written cases of teaching, demonstration lessons, unit or lesson plans, observations of peers, and curriculum materials including sample assessments and student work samples*“.

### 13.2.3 Erprobungs- und Reflexionsphasen

Neben dem Aneignen von neuem Wissen, sollte für die Lehrkräfte in Fortbildungen die Möglichkeit bestehen, die neuen Inhalte im Unterricht zu erproben und sich mit anderen Teilnehmenden über die Erprobung auszutauschen:

> *„[The professional development activities] need to be designed in a way that teachers' reflection about their practice is encouraged, the use of new methods is promoted, and discussions as well as the exchange of experiences and ideas are enhanced. Because always-working recipes do not exist in education, professional development activities should offer teachers rich-learning opportunities for questioning their current practices and make them evolve."* (PRIMAS 2013, S. 5 f.)

> „Der Austausch mit anderen Lehrkräften ermöglicht den Teilnehmenden, einen Einblick in deren Handlungspraxis und möglicherweise Ideen für ihr eigenes Handeln zu erlangen. Auch ist es den Teilnehmenden möglich, von den anderen Lehrkräften Rückmeldungen hinsichtlich des eigenen Handelns zu erhalten und dieses auf dieser Basis zu reflektieren und bei Bedarf zu modifizieren." (Aldorf 2016, S. 78)

Wie eng die Vorgaben für die unterrichtlichen Umsetzungen sein dürfen, ist laut Lipowsky und Rzejak (2012, S. 7) nicht eindeutig geklärt. Einerseits können sich Gestaltungsspielräume in den Vorgaben positiv auf die Motivation der Lehrkräfte auswirken, da dies dem Streben nach Autonomie entgegenkommt (vgl. Deci und Ryan 1993), andererseits besteht das Risiko, die Teilnehmenden durch einen hohen Grad an Selbstbestimmung zu überfordern.

## 13.3 Inhaltliche Merkmale

Die folgenden Aspekte beziehen sich auf die Auswahl und Ausgestaltung von Fortbildungsinhalten. Dabei wird auf Erkenntnisse und Befunde Bezug genommen, die sich sowohl konkret auf den Bereich der Informatik bzw. des Programmierens beziehen, als auch allgemein auf den Bereich der Lehrerbildung.

### 13.3.1 Auswahl der Fachinhalte

Die Kenntnis einer Programmiersprache ist zwar eine notwendige, aber bei weitem nicht ausreichende Voraussetzung für das Programmierenlernen:

> *„But teaching programming is much more than teaching a programming language. [...] Students also need knowledge about the programming process, i. e. how to develop programs, and they need to extend that knowledge into programming skills."* (Caspersen 2018, S. 110)

Bell bezieht sich auf Böhm und Jacopini (1966) und verweist darauf, dass der Fokus beim Programmierenlernen auf den Grundlagen und Kernkonzepten der Programmierung liegen sollte:

> *„The Böhm-Jacopini theorem underpins the idea that teaching programming should cover three basic constructs: sequence, selection and iteration. In principle, these are the only control structures needed to program any computing device. In addition to these control structures, a language needs input, output and storage (variables) to give*

> *sufficient scope to program anything that is computable. An important consequence of this observation is that ‚toy' languages like Scratch are actually just as capable as the most advanced languages, or at least, they capture the key logic needed to make things happen on a computational device [...].“* (Bell 2016, S. 6)

Es gilt außerdem sowohl für Schülerinnen und Schüler als auch für Lehrkräfte, dass sich Expertise im Programmieren nur durch Übung einstellen kann. Bell (2016, S. 5) weist darauf hin, dass Programmieren nicht durch das Lesen einer Handreichung oder dem Besuch eines kurzen Kurses erlernt werden kann, sondern dass Lehrkräfte kontinuierlich an kleineren Programmieraufgaben arbeiten sollten, die sie nicht überfordern.

### 13.3.2 Verschränkung von Fachwissenschaft und Fachdidaktik

Dem Forschungsstand bezüglich Lehrerfortbildungen aus verschiedenen Fächern ist zu entnehmen, „dass wirksame Fortbildungen darauf abzielen, das fachliche Verständnis der Lehrpersonen für den Unterrichtsinhalt zu vertiefen, das Lehrerwissen über typische Schülerkonzepte und -misskonzepte zu erweitern und Lehrpersonen zu befähigen, aus Antworten und Lösungswegen von Schülern Rückschlüsse auf deren Vorstellungen, Wissen und Fähigkeiten zu ziehen“ (Lipowsky und Rzejak 2012, S. 5). Während in der Informatik zwar Einigkeit in Hinblick auf die fachlichen Inhalte besteht, ist jedoch nicht abschließend geklärt, was zu den fachdidaktischen Kompetenzen von Lehrkräften gehört und wie sie diese entwickeln (vgl. Hubwieser u. a. 2013b; Yadav u. a. 2016, siehe Abschnitt 4.2).

Eine Variante, Fachwissenschaft und Fachdidaktik in Fortbildungen zu verknüpfen, ist der Ansatz des *situierten Lernens* (vgl. Rank u. a. 2012, S. 182). Dieser basiert auf der Annahme, dass vorhandenes Wissen in der Praxis ungenutzt bleibt, wenn sich die Lern- und spätere Anwendungssituation zu stark unterscheiden (z. B. Gerstenmaier und Mandl 1995, S. 875). Demzufolge sollten Lernsituationen möglichst authentisch und anwendungsorientiert sein. Konzepte des situierten Lernens wurden bisher vor allem in Disziplinen mit starken berufs- und anwendungsbezogenen Schwerpunkten entwickelt und erprobt, bspw. in der Medizinerausbildung und der Lehrerbildung (Hartinger, Fölling-Albers und Mörtl-Hafizovic 2005, S. 115). Um den Ansatz im Rahmen von Lehrerfortbildungen zu realisieren, könnten sich Lehrkräfte in authentischen Lernsituationen mit fachlichen Inhalten auseinandersetzen, bevor in einem nachgeschalteten Theorie-Input die didaktischen Hintergründe vorgestellt werden (Gebauer 2019, S. 163).

### 13.3.3 Anknüpfen an Bekanntes

Insbesondere wenn Informatik nicht Teil des Lehrplans der Grundschule ist, sollte man Lehrkräften laut Reding und Dorn (2017) genügend Zeit in den Fortbildungen einräumen, um die neuen Inhalte mit dem bestehenden Lehrplan zu verknüpfen oder konkrete Möglichkeiten der Integration in andere Fächern aufzeigen:

> *„Allowing teachers the time during PD experiences to map out the connections between their current content standards and CS standards allows them to identify optimal places in their curriculum in which to implement the CS lessons. Allowing for the teachers themselves to do this rather than being told is important for their buy-in. This connection made between their current curriculum and CS lessons provides a solid foundation for practical application as it demonstrates how the CS component can be smoothly integrated into their existing curriculum.“* (Reding und Dorn 2017, S. 161)

Grover und Yadav (2020, S. 84 ff.) tragen verschiedene Ansätze zusammen, das Programmieren in andere Fächer der Primar- und Sekundarstufe zu integrieren. Dabei werden Beispiele bereitgestellt für die Einbindung des Programmierens in *Mathematics*, *Science*, *Language Arts*, *Social Studies* und *Music*. Ein Vorteil des fächerübergreifenden Unterrichtens von informatischen Inhalten ist, dass diese den bestehenden Unterricht positiv verändern können, anstatt nur die ohnehin begrenzten Zeitressourcen zu reduzieren (Bell und Duncan 2018, S. 143). Reding und Dorn (2017) empfehlen darüber hinaus, nicht nur inhaltliche, sondern auch didaktische Anknüpfungspunkte zum Unterricht der Grundschule zu schaffen. Verbindungen zwischen der Grundschuldidaktik sowie bekannten fachdidaktischen Prinzipien bestehender Grundschulfächer sollten aufgezeigt oder mit den Lehrkräften erarbeitet werden:

> *„Not only should PD experiences incorporate new CS-speciffic learning, but they also should begin with making connections between teachers' current knowledge and CS principles/concepts. Highlighting connections between current classroom methodologies and the new pedagogic approaches also serves to improve comfort. “*
> (Reding und Dorn 2017, S. 161)

### 13.3.4 Angemessene Fachsprache

In den vergangenen Jahren wurde zunehmend festgestellt, dass Sprache im Lernprozess eine wichtige Rolle spielt. Sprachliches und fachliches Lernen sind eng miteinander verbunden und bedingen sich gegenseitig (vgl. Michalak, Lemke und Goeke 2015, S. 10). Sprache stellt nicht nur eine wichtige Voraussetzung für das fachliche Lernen dar, sie kann auch gezielt im Fachunterricht gefördert werden. Dies ist besonders in der Grundschule relevant, da hier unterschiedlichste Sprachniveaus aufeinandertreffen (ebd., S. 22 f.) und Kinder mit sprachlichen Schwierigkeiten Nachteile erfahren:

> „Kinder mit wenig Literalität gelten insbesondere aufgrund der Schwierigkeiten mit dem akademischen Sprachniveau, der sogenannten Bildungssprache als Verlierer im Bildungssystem. Die Schule setzt die Beherrschung der Bildungssprache in der Regel für das Lernen als selbstverständlich voraus und lehrt sie nicht.“
> (Rank und Wildemann 2015, S. 474)

Der Aspekt der Bildungssprache wurde vereinzelt auch im Kontext der Informatikdidaktik beleuchtet (z. B. Diethelm und Goschler 2015; Diethelm, Goschler und Lampe 2018; Batur und Strobl 2019). Laut Diethelm und Goschler (2015, S. 21) braucht die Informatik wie jedes andere Fach *„specific terms to communicate about topics of our science in class and also outside in everyday life“*. Die Fachbegriffe, die mit dem Programmieren und anderen informatischen Inhalten einhergehen, können laut Bell (2016, S. 3) vor allem fachunerfahrene Lehrkräfte einschüchtern. Hinzu kommt für den deutschen Sprachraum, dass viele Fachbegriffe der englischen Sprache entstammen:

> *„In the context of German schools, the fact that a lot of computer science terminology consists of English words makes the situation different from English speaking countries: [...] Having to learn and use a variety of words in another language that the one the subject is taught in could be an additional difficulty for pupils.“*
>
> (Diethelm, Goschler und Lampe 2018, S. 210 f.)

Um Lehrkräften den Gebrauch des Fachjargons zu erleichtern, wird vorgeschlagen, die Begrifflichkeiten erst nach dem Behandeln der Fachkonzepte einzuführen:

> *„We have found that it's effective to engage teachers new to the subject by first using the idea (such as showing an algorithm to calculate the checksum in a product barcode), and then labelling it (in this case distinguishing the ‚algorithm' from a program that implements the algorithm, and of course introducing the term "checksum") so that the mysterious terminology appears after the concept has been brought down to size.“*
>
> (Bell, Duncan und Rainer 2018, S. 3)

Diese Kombination aus impliziter Sprachaneignung und expliziter Kommunikation über Begriffe hat sich ebenfalls in Bezug auf die sprachliche Bildung von Schülerinnen und Schülern bereits als besonders wirksam erwiesen (vgl. Darsow, Paetsch und Felbrich 2012). Für den Sachunterricht wird bspw. empfohlen, bereits bei der Planung von Lehr-Lernsituationen sprachliche Aktivitäten für implizites Sprachlernen zu berücksichtigen und für das Thema relevante Begriffe implizit und explizit, z. B. durch zusätzliche Erläuterungen, einzuführen (ebd., S.225).

Teil IV

# ENTWICKLUNG VON LEITLINIEN UND KONZEPTEN ZUR UMSETZUNG DES PROGRAMMIERENS IN DER GRUNDSCHULE

# 14 Entwicklung einer Unterrichtssequenz

## 14.1 Zielvorstellungen (Unterrichtssequenz)

Als Ausgangspunkt für das Kapitel werden im Folgenden die Ziele erläutert, die mit der Unterrichtssequenz erreicht bzw. angestrebt werden sollen. Aufbauend auf den Ausführungen in Kapitel 2.1, Kapitel 2.2 und Kapitel 2.3 wird dabei auf informatische Kompetenzen und affektive Merkmale der Lernenden eingegangen. Da die Schülerinnen und Schüler über ein gewisses Maß an Sicherheit im Lesen und Schreiben verfügen sollten, wird die Zielgruppe auf Schülerinnen und Schüler der dritten und vierten Klasse festgelegt. Dies entspricht zudem den Ergebnissen der Fragebogenstudie von Seegerer (2021, S. 61), in der alle Schulleiterinnen und Schulleiter sowie Lehrkräfte der Grund- und Mittelschulen aus einem ländlichen und einem städtischen Schulbezirk in Bayern befragt wurden. Darin geben 38.4% der befragten Lehrkräfte an, dass informatische Grundlagen für das Verständnis der digitalen Welt bereits ab der dritten Klasse thematisiert werden sollten, 15.8% votieren für den Beginn in der vierten Jahrgangsstufe. Im Gegensatz dazu sprechen sich nur 10.0% für den Beginn in der ersten und 13.1% in der zweiten Klasse aus.

### 14.1.1 Informatische Kompetenzen

Ausgangspunkt für die Entwicklung der Unterrichtssequenz zum Programmieren ist die Klärung bzw. Formulierung der kognitiven Lernziele für die Schülerinnen und Schüler. Die übergeordneten Ziele der Unterrichtssequenz ergeben sich aus den von der Gesellschaft für Informatik (2019, S. 13) formulierten Kompetenzerwartungen für informatische Bildung im Primarbereich (siehe Tabelle 14.2), den Zieldimensionen für eine frühe informatische Bildung (siehe Tabelle 14.1) der Stiftung *Haus der kleinen Forscher* (Bergner u. a. 2018, S. 158 f.), in der beispielhafte Kompetenzerwartungen für das Modellieren und Implementieren von Algorithmen und Programmen[1] formuliert werden (siehe Abschnitt 2.3) sowie aus dem Bereich *Algorithms and Programs* der *CSTA K-12 Computer Science Standards* (2017, siehe Tabelle 14.3 und Kapitel 2.2). Die Unterrichtssequenz wird für die Jahrgangsstufen drei und vier entwickelt, was *Level 1B* der *CSTA K12 Computer Science Standards* entspricht. Da die Level innerhalb der Standards aufeinander aufbauen und sie für einen umfangreicheren Informatikunterricht entwickelt wurde, als dies im Rahmen dieser Arbeit möglich ist, werden nur die Standards von *Level 1A* berücksichtigt. Bei den Kompetenzerwartungen der GI werden sowohl die Kompetenzen, die bis zum Ende der zweiten Klasse erreichen werden sollen, als auch die Kompetenzen, die bis zum Ende der vierten Klasse erreicht werden sollen, berücksichtigt.

[1] Die Autorinnen und Autoren weisen darauf hin, dass dieser Bereich „– leicht vereinfachend und zugespitzt – als Programmieren zusammengefasst werden [kann]“ (HdkF 2018, S. 157).

*Tabelle 14.1 Beispielhafte Zielkompetenzen auf Ebene der Kinder (Bergner u. a. 2018, S. 158 f.)*

| Beispielhafte Kompetenzerwartungen im Bereich Modellieren und Implementieren von Algorithmen und Programmen |
|---|
| Umgang mit Handlungsvorschriften (vor allem Entwerfen, aber auch Lesen und Verstehen)<br>- Die Kinder benennen und formulieren Handlungsvorschriften zur Steuerung eines altersentsprechenden Informatiksystems.<br>- Die Kinder erklären gelesene Handlungsvorschriften und -abläufe für die Steuerung eines altersentsprechenden Informatiksystems.<br>- Die Kinder entwerfen eine Vorschrift zur Verschlüsselung von Nachrichten (Daten) mit altersentsprechenden Verfahren (z. B. Skytale). |
| Implementieren (nutzen einer formalen Notation)<br>- Die Kinder entwerfen Handlungsvorschriften/-abläufe mit vorgegebenen altersentsprechenden Bausteinen bzw. Befehlen. |
| Einfache Algorithmik (Handlungsvorschriften anwenden und untersuchen)<br>- Die Kinder wenden ein gegebenes Verfahren zur Lokalisierung einer fehlerhaften Stelle an.<br>- Die Kinder erläutern gegebene Algorithmen.<br>- Die Kinder führen Algorithmen schrittweise aus (simulieren). |

*Tabelle 14.2 Kompetenzerwartungen zum Inhaltsbereich Algorithmen (GI 2019, S. 13)*

| Ende Klasse 2 | Ende Klasse 4 |
|---|---|
| Die Schülerinnen und Schüler<br>- führen Algorithmen in ihrer Lebenswelt aus<br>- verwenden algorithmische Grundbausteine<br>- beschreiben Algorithmen alltagssprachlich | Die Schülerinnen und Schüler<br>- entwerfen, realisieren und testen Algorithmen mit den algorithmischen Grundbausteinen Anweisung, Sequenz, Wiederholung und Verzweigung<br>- stellen Algorithmen in verschiedenen formalen Darstellungsformen dar<br>- vergleichen Algorithmen unter Verwendung der Fachsprache<br>- programmieren ein Informatiksystem |

*Tabelle 14.3 Auszug aus den CSTA K12 Computer Science Standards (2017): Level 1A und 1B des Inhaltsbereichs „Algorithms and Programming“*

| *Subconcept* | *Level 1A (Ages 5-7)*<br>*By the end of Grade 2, students will be able to…* | *Level 1B (Ages 8-11)*<br>*By the end of Grade 5, students will be able to…* |
|---|---|---|
| *Algorithms* | *Model daily processes by creating and following algorithms (sets of step-by-step instructions) to complete tasks.* | *Compare and refine multiple algorithms for the same task and determine which is the most appropriate.* |

| | | |
|---|---|---|
| *Variables* | *Model the way programs store and manipulate data by using numbers or other symbols to represent information.* | *Create programs that use variables to store and modify data.* |
| *Control* | *Develop programs with sequences and simple loops, to express ideas or address a problem.* | *Create programs that include sequences, events, loops, and conditionals.* |
| *Modularity* | *Decompose (break down) the steps needed to solve a problem into a precise sequence of instructions.* | *Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.* |
| | | *Modify, remix, or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features.* |
| *Program Development* | *Develop plans that describe a program's sequence of events, goals, and expected outcomes.* | *Use an iterative process to plan the development of a program by including others' perspectives and considering user preferences.* |
| | *Give attribution when using the ideas and creations of others while developing programs.* | *Observe intellectual property rights and give appropriate attribution when creating or remixing programs.* |
| | *Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops.* | *Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended.* |
| | | *Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation, and review stages of program development.* |
| | *Using correct terminology, describe steps taken and choices made during the iterative process of program development.* | *Describe choices made during program development using code comments, presentations, and demonstrations.* |

Die Kompetenzsets werden in einem ersten Schritt miteinander abgeglichen und inhaltliche Entsprechungen kenntlich gemacht (siehe Tabelle 14.5). Hierbei ist darauf hinzuweisen, dass es durch unterschiedliche Schwerpunktsetzungen in den jeweiligen Kompetenzformulierungen häufig zu kleineren inhaltlichen Abweichungen kommt: z. B. „Die Schülerinnen und Schüler führen Algorithmen in ihrer Lebenswelt aus“ (GI 2019, S. 13) als Entsprechung zu „Die Kinder führen Algorithmen schrittweise aus (simulieren)“ (HdkF 2018, S. 159). In diesen Fällen werden die Formulierungen dennoch als vergleichbar eingeordnet. Die Kompetenzformulierungen, welche nicht in einer der drei Quellen zu finden sind, werden für das weitere Vorgehen zunächst nicht berücksichtigt. Diese Entscheidung wird jedoch nach der Erprobung der Unterrichtssequenz erneut geprüft. Um die Kompetenzen innerhalb der zu entwickelten Unterrichtssequenz zu verankern, werden dazu entsprechende Grobziele formuliert, die erreicht werden sollen (siehe Tabelle 14.5). Diese sind einerseits konkret formuliert und ermöglichen nur einen geringen Interpretationsspielraum, andererseits beschreiben sie nicht im Detail, welche Schülerhandlungen zur Erreichung vorgesehen sind (Riedl 2010, S. 32). Zur weiteren Legitimation werden die Grobziele in Tabelle 14.4 mit den von der KMK (2016, S. 18) formulierten Kompetenzen des Unterbereichs *5.5 Algorithmen erkennen und formulieren* abgeglichen (siehe Abschnitt 2.3). An dieser Stelle ist darauf hinzuweisen, dass es sich bei den Grobzielen nicht um Kompetenzen nach dem in der Wissenschaft gängigen Verständnis von Weinert (2001b, S. 27 f., siehe Abschnitt 11.1.1) handelt. Vielmehr sind sie als für die Tätigkeit des Programmierens relevante Fertigkeiten zu verstehen. Dies entspricht nach Schaper (2008, S. 91) dem eher klassischen Verständnis einer Kompetenz als „Summe von bestimmten tätigkeitsrelevanten Qualifikationen (z. B. Kenntnissen, Fertigkeiten und Fähigkeiten)“ (ebd., siehe Abschnitt 11.1.1).

*Tabelle 14.4 Abgleich der entwickelten Grobziele mit den Kompetenzen der Kultusministerkonferenz (KMK 2016)*

<table>
<tr><th>Grobziele</th><th colspan="2">Kompetenzen in der digitalen Welt (KMK 2016)</th></tr>
<tr><td>Die Schülerinnen und Schüler führen Algorithmen in ihrer Lebenswelt aus</td><td rowspan="3">Eine strukturierte, algorithmische Sequenz zur Lösung eines Problems planen und verwenden (KMK 3)</td><td rowspan="4">Funktionsweisen und grundlegende Prinzipien der digitalen Welt kennen und verstehen (KMK 1)</td></tr>
<tr><td>Die Schülerinnen und Schüler entwerfen, realisieren und testen Algorithmen mit den algorithmischen Grundbausteinen Anweisung, Sequenz, Wiederholung und Verzweigung</td></tr>
<tr><td>Die Schülerinnen und Schüler beschreiben Algorithmen in ihrer Alltagssprache</td></tr>
<tr><td>Die Schülerinnen und Schüler programmieren ein altersentsprechendes Informatiksystem</td><td>Algorithmische Strukturen in genutzten digitalen Tools erkennen und formulieren (KMK 2)</td></tr>
</table>

*Tabelle 14.5 Ableitung der Grobziele aus den Kompetenzen der Gesellschaft für Informatik (2019) und dem Haus der kleinen Forscher (2018) sowie den CSTA K12 Computer Science Standards (2017).*

| Kompetenzen für informatische Bildung im Primarbereich (GI 2019) | Informatische Kompetenzen auf Ebene der Kinder (HdkF 2018) | CSTA K12 Computer Science Standards (CSTA 2017) | **Grobziele** |
|---|---|---|---|
| Die Schülerinnen und Schüler führen Algorithmen in ihrer Lebenswelt aus (GI 1) | Die Kinder führen Algorithmen schrittweise aus (simulieren) (HdkF 7) | Model daily processes by creating and following algorithms (sets of step-by-step instructions) to complete tasks (CSTA 1) | **Die Schülerinnen und Schüler führen Algorithmen in ihrer Lebenswelt aus** |
| Die Schülerinnen und Schüler verwenden algorithmische Grundbausteine (GI 2) | Die Kinder entwerfen Handlungsvorschriften/ -abläufe mit vorgegebenen altersentsprechenden Bausteinen bzw. Befehlen (HdkF 4) | Develop programs with sequences and simple loops, to express ideas or address a problem (CSTA 3)<br><br>Decompose (break down) the steps needed to solve a problem into a precise sequence of instructions (CSTA 4)<br><br>Develop plans that describe a program's sequence of events, goals, and expected outcomes (CSTA 5) | **Die Schülerinnen und Schüler entwerfen, realisieren und testen Algorithmen mit den algorithmischen Grundbausteinen Anweisung, Sequenz, Wiederholung und Verzweigung** |
| Die Schülerinnen und Schüler entwerfen, realisieren und testen Algorithmen mit den algorithmischen Grundbausteinen Anweisung, Sequenz, Wiederholung und Verzweigung (GI 4) | | | |
| Die Schülerinnen und Schüler stellen Algorithmen in verschiedenen formalen Darstellungsformen dar (GI 5) | | | |
| Die Schülerinnen und Schüler beschreiben Algorithmen alltagssprachlich (GI 3) | Die Kinder erläutern gegebene Algorithmen (HdkF 6) | Model daily processes by creating and following algorithms (sets of step-by-step instructions) to complete tasks (CSTA 1) | **Die Schülerinnen und Schüler beschreiben Algorithmen in ihrer Alltagssprache** |
| | Die Kinder erklären gelesene Handlungsvorschriften und -abläufe für die Steuerung eines altersentsprechenden Informatiksystems (HdkF 2) | | |
| Die Schülerinnen und Schüler vergleichen Algorithmen unter Verwendung der Fachsprache (GI 6) | | | |
| Die Schülerinnen und Schüler programmieren ein Informatiksystem (GI 7) | Die Kinder benennen und formulieren Handlungsvorschriften zur Steuerung eines altersentsprechenden Informatiksystems (HdkF 1) | Develop programs with sequences and simple loops, to express ideas or address a problem (CSTA 3) | **Die Schülerinnen und Schüler programmieren ein altersentsprechendes Informatiksystem** |
| | Die Kinder wenden ein gegebenes Verfahren zur Lokalisierung einer fehlerhaften Stelle an (HdkF 5) | Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops (CSTA 7) | |
| | Die Kinder entwerfen eine Vorschrift zur Verschlüsselung von Nachrichten (Daten) mit altersentsprechenden Verfahren (z. B. Skytale) (HdkF 3) | | |
| | | Model the way programs store and manipulate data by using numbers or other symbols to represent information (CSTA 2) | |
| | | Give attribution when using the ideas and creations of others while developing programs (CSTA 6) | |
| | | Using correct terminology, describe steps taken and choices made during the iterative process of program development (CSTA 8) | |

### 14.1.2 Motivation, Interesse und Selbstwirksamkeit

In den Ausführungen zu Potenzialen und Chancen informatischer Bildung im Primarbereich wurde bereits die Bedeutung verschiedener affektiver Merkmale von Lernenden herausgestellt (siehe Abschnitt 2.1). Einzelne dieser Aspekte sind auch im Bereich *K5 non-cognitive skills* des Kompetenzmodells für Modellierung und Systemverständnis in der Informatik wiederzufinden, das innerhalb des Projekts *MoKoM*[2] entwickelt wurde (Linck u. a. 2013, 6, siehe Tabelle 14.6).

*Tabelle 14.6 Überblick der nicht-kognitiven Kompetenzen des MoKoM-Kompetenzmodells aus Linck u. a. (2013, S. 6)*

| K5 NON-COGNITIVE SKILLS |
|---|
| **K5.1 Attitudes** |
| K5.1.1 Perceive & Anticipate effects of Informatic Systems |
| K5.1.1.1 Demystify informatic systems |
| K5.1.1.2 Recognize meaningfulness of system application, comprehension and development |
| K5.1.1.3 Assess appropriateness and quality of informatics systems |
| K5.1.1.4 Recognize solvability of informatics systems |
| K5.1.1.5 Know & evaluate consequences of informatics systems for social realities |
| K5.1.2 Expectations for informatics literacy & professional practise |
| K5.1.2.1 Affinity and enthusiasm for informatics issues |
| K5.1.2.2 Willing to face demanding informatics challenges |
| **K5.2 Social-communicative skills** |
| **Recognize social-communicative skills as significant** |
| K5.2.1 Cooperation & Teamwork |
| K5.2.1.1 Communicative skills |
| K5.2.1.1.1 Discuss informatics topics |
| K5.2.1.1.2 Present informatics topics |
| K5.2.1.1.3 Able & willing to communicate knowledge |
| K5.2.1.1.4 Able & willing to criticize constructively |
| K5.2.1.2 Cooperation |
| K5.2.1.2.1 Make and fulfill agreements in a team |
| K5.2.1.2.2 Willing to work according to agreements |
| K5.2.1.2.3 Willing to pick up ideas of other team members |
| K5.2.2 Empathy: Ability to change perspectives and roles |
| **K5.3 Motivational and volitional skills** |
| K5.3.1 Open to new ideas and new requirements |
| K5.3.1.1 Willing to deploy new and unknown informatics approaches |
| K5.3.1.2 Avoid mechanistic informatics approaches |
| K5.3.2 Learning motivation |
| K5.3.2.1 Willing to improve informatics abilities and knowledge |
| K5.3.2.2 Willing to fulfill informatics tasks successfully |
| K5.3.3 Commitment & Engagement |
| K5.3.3.1 Feel obligated to fulfill informatics tasks |
| K5.3.3.2 Exert oneself to fulfill informatics tasks |

[2] MoKoM – Entwicklung von qualitativen und quantitativen Messverfahren zu Lehr-Lernprozessen für Modellierung und Systemverständnis in der Informatik (https://web.archive.org/web/20160124115232/http://ddi.uni-paderborn.de/forschung/mokom.html)

Motivation, Interesse und Selbstwirksamkeit finden sich zudem in den Zieldimensionen für frühe informatische Bildung der Stiftung *Haus der kleinen Forscher* (Bergner u. a. 2018, S. 135) wieder (siehe Abbildung 14.1). Die Formulierungen von Bergner u. a. werden für diese Arbeit auf den Gegenstand der Programmierung übertragen und als Ziele der Unterrichtssequenz verwendet:

- Motivation und Lernfreude im Umgang mit der Programmierung;
- Interesse an der Programmierung;
- Selbstwirksamkeitserwartung im Umgang mit der Programmierung;

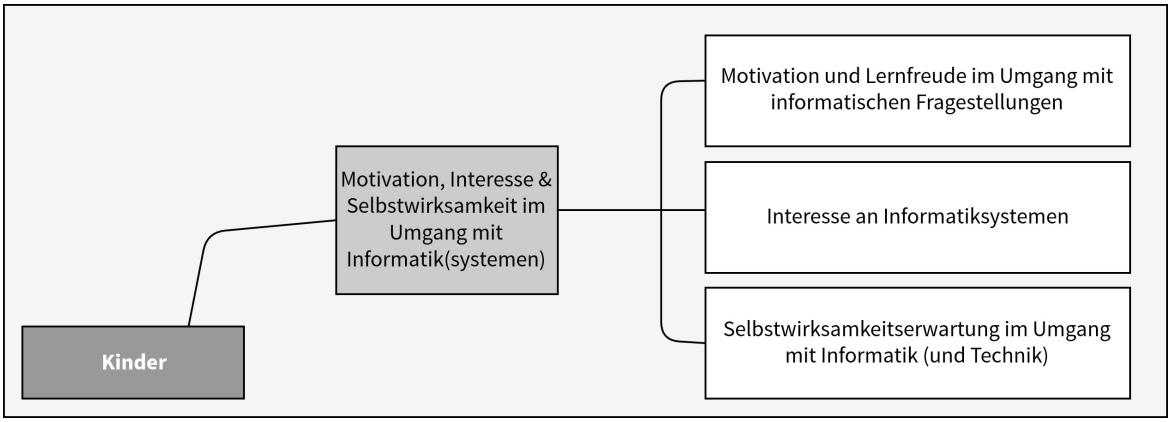



*Abbildung 14.1 Affektive Zieldimension informatischer Bildung auf Ebene der Kita- und Grundschulkinder aus Bergner u. a. (2018, S. 135)*

## 14.2 *Design Principles* (Unterrichtssequenz)

Auf Basis der theoretischen Grundlagen werden im folgenden Teilkapitel vorläufige *Design Principles* formuliert, an denen sich die Entwicklung der Unterrichtssequenz orientiert. An dieser Stelle sei zunächst auf die Begriffsklärung im Rahmen des forschungsmethodischen Bezugsrahmen verwiesen (siehe Abschnitt 7.3.2): *Design Principles* werden darin als Gestaltungsaussagen definiert, die sich auf die Frage beziehen „wie bei der Planung eines Lernangebots vorzugehen ist oder welche didaktisch-methodischen Varianten sich für welche Lerninhalte oder Zielgruppen am besten eignen“ (Kerres 2018, S. 78). Nach Euler können sich *Design Principles* sowohl auf die Gestaltung von Lernaktivitäten als auch auf die darauf bezogenen Lehrhandlungen beziehen (2014a, S. 106). Diese Dichotomie wird im Folgenden für die Strukturierung der *Design Principles* übernommen.

Die folgenden *Design Principles* sind relativ abstrakt formuliert und werden zunächst als Leitprinzipien mit Querverweisen zum thematischen Bezugsrahmen theoretisch begründet. Anschließend werden sie zur Vorbereitung der didaktischen Konzeption der Unterrichtssequenz in einem zweistufigen Operationalisierungsprozess konkretisiert (vgl. Reinmann 2014, S. 70 f.). Die Leitprinzipien werden zunächst zu Umsetzungsprinzipien ausgearbeitet, die sich bereits konkret auf die Ausgestaltung der Lehr-Lernaktivitäten beziehen. In einem nächsten Schritt werden dazu passende

Musterbeispiele im Sinne eines *Prototyping*[3] ausgearbeitet. Die Umsetzungsprinzipien und Musterbeispiele dienen im weiteren Verlauf des Forschungsprozesses als Basis für die Rücksprache und Diskussion mit *Critical Friends*.

### 14.2.1 *Design Principles* mit Bezug zu Lernaktivitäten

| |
|---|
| *DP 1 Die Lernenden begegnen den Lerninhalten in unterschiedlichen Repräsentationsformen.* |

*Theoretische Fundierung*

Die theoretische Herleitung dieses Prinzips fußt in erster Linie auf den entwicklungspsychologischen Grundlagen der Zielgruppe: Zum einen befinden sich Grundschulkinder auf der konkret-operationalen Stufe von Piagets Stufenmodell der kognitiven Entwicklung (siehe Abschnitt 10.1) und profitieren demnach von einer Kopplung ihrer Denkoperationen an konkrete Handlungen und Wahrnehmungen. Visuelle oder haptische Hilfsmittel werden in diesem Zusammenhang als hilfreich beschrieben. Zum anderen vollziehen sich Lernprozesse laut Bruner und Harttung wesentlich bei den Übergängen zu anderen Repräsentationsformen eines Lerngegenstands: enaktiv, ikonisch, symbolisch (siehe Abschnitt 10.2). Letzteres wird zudem gestützt von Mayers *Cognitive Theory of Multimedia Learning*, die im gleichen Unterkapitel erläutert wird. Darüber hinaus entspricht die Vielfalt der Handlungs-, Ausdrucks- und Repräsentationsmodi dem Ansatz des *Universal Design for Learning* (siehe Abschnitt 12.3.3), der es ermöglicht, unterschiedlichen Lerntypen bzw. der Heterogenität der Lernenden im Allgemeinen zu begegnen und somit Lehr-Lernsituationen inklusiver zu gestalten. Zur weiteren Fundierung des *Design Principles* kann im Bereich der Grundschuldidaktik auf das Prinzip der Veranschaulichung verwiesen werden (siehe Abschnitt 11.1) sowie auf die Einbettung von *Unplugged*-Materialien als kindgerechten Zugang zur Informatik (siehe Abschnitt 12.2). Visualisierungen und Rollenspiele werden zudem als förderlich zur Ausbildung der *Notional Machine* beschrieben (siehe Abschnitt 12.1.2).

*Umsetzungsprinzipien*

Innerhalb der Unterrichtssequenz werden verschiedene Unterrichtsansätze kombiniert. Zunächst sollen die Schülerinnen und Schüler dem Programmieren und den algorithmischen Grundstrukturen durch eigenständig ausgeführte Handlungen begegnen. In einem nächsten Schritt durch graphische und bildliche Darstellungen und zuletzt durch Sprache und Zeichen. Um den Schülerinnen und Schülern im tätigen Umgang mit der Programmierung Lernerfahrungen zu ermöglichen, wird eine kindgerechte blockbasierte Programmierumgebung auf einem Informatiksystem genutzt. Das Visualisieren von programmierten Handlungen durch das Ausführen von Handlungen im Klassenraum kann zur Überprüfung von Lösungen und als Hilfestellung eingesetzt werden.

[3] „In der Informatik und Technik meint *Prototyping* die Herstellung von Prototypen im Sinne vereinfachter, aber funktionsfähiger Modelle bzw. konkreter Beispiele eines ‚Produkts'[...]" (Reinmann 2014, S. 71), die frühzeitiges Feedback bezüglich deren Eignung ermöglichen

*Musterbeispiele*

- Beispiel DP1-1: Lehrer-Roboter
  Die Schülerinnen und Schüler programmieren die Lehrkraft mittels mündlicher Anweisungen. Die Lehrkraft spielt einen Roboter, der kleine Aufgaben im Klassenzimmer erfüllen soll, zum Beispiel das Fenster öffnen. Es wird nicht erklärt, wie man ihn steuert oder welche Anweisungen er kennt. Da er sich bei zu allgemeinen Anweisungen nicht rührt und nur präzise Befehle befolgt, wird jedoch schnell deutlich, dass die Anweisungen verständlich, genau und eindeutig formuliert werden müssen. Größere Vorgänge müssen dabei in Teilschritte zerlegt werden, die der Roboter nacheinander ausführt. Der Roboter führt die Anweisungen zudem wörtlich aus: z. B. bei der Anweisung „drehe dich nach rechts“ dreht sich der Roboter endlos nach rechts um die eigene Achse. Im Anschluss spielen Schülerinnen und Schüler den Roboter – der Rest der Klasse gibt mündliche Anweisungen und überprüft, ob der Roboter diese korrekt ausführt.

- Beispiel DP1-2: Vom Bild zur Sprache
  Es wird herausgearbeitet, wo die Schülerinnen und Schüler Algorithmen in ihrem Alltag begegnen, zum Beispiel in Form von Bastelanleitungen oder Rezepten. Es wird geübt, Abläufe in natürlicher Sprache zu beschreiben. Die Schülerinnen und Schüler verfassen zu einer bildlichen Anleitung sprachliche Anweisungen (siehe Abbildung 14.2). Dabei können sie die Anleitung auch durch zusätzlich Bilder ergänzen. Die sprachlichen Anweisungen werden befolgt, um zu prüfen, ob sie präzise genug formuliert wurden.

- Beispiel DP1-3: Programmieren im Parcours
  Um das Darstellen von Algorithmen zu vertiefen, programmieren die Schülerinnen und Schüler sich gegenseitig, um verschiedene Aufgaben in einem Parcours zu lösen (siehe Abbildung 14.3). Sobald eine Aufgabe bearbeitet wurde, laufen sie die Lösung in einem aufgebauten Parcours ab. Falls nötig, verbessern sie die Programme gemeinsam. Im Anschluss können die Schülerinnen und Schüler sich eigene Aufgabenstellungen überlegen.

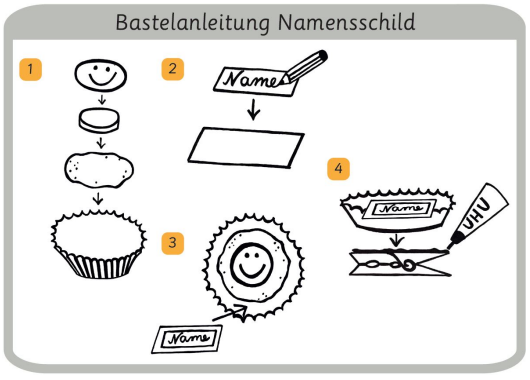



*Abbildung 14.2 Bastelanleitung für ein Namensschild*

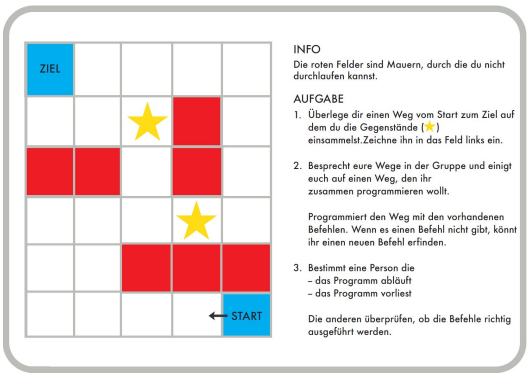



*Abbildung 14.3 Beispiel für eine Parcoursaufgabe*

*DP 2 Die Unterrichtssequenz führt vom Bekannten zum Unbekannten.*

*Theoretische Fundierung*

Das Prinzip wird ebenfalls durch Piaget gestützt, der empfiehlt, im Unterricht vertraute Beispiele zu verwenden, um komplexe oder abstrakte Ideen einzuführen (siehe Abschnitt 10.1). Auch in Hinblick auf das schlussfolgernde Denken wird konstatiert, dass grundsätzlich auch jüngere Kinder dazu in der Lage sind, wenn Aufgaben sich auf Konzepte beziehen, die ihnen bekannt sind und konkret dargeboten werden (ebd.). Die Herstellung von Alltagsbezug ist darüber hinaus nicht nur in der Grundschule gängige Praxis.

*Umsetzungsprinzipien*

In der Unterrichtssequenz wird an den Alltag und das Vorwissen der Schülerinnen und Schüler angeknüpft, indem vertraute Beispiele und Formate angeboten werden. Dies kann sich sowohl auf die Aufgabenform, als auch den Inhalt oder Kontext beziehen. Besonders zu Beginn der Sequenz werden Aufgaben eingesetzt, welche in ähnlicher Form aus anderen Schulfächern oder Alltagssituationen bekannt sind.

*Musterbeispiele*

- Beispiel DP2-1: Vorgangsbeschreibung
  Im Deutschunterricht der Grundschule lernt man die Vorgangsbeschreibung als Textform kennen, die möglichst wirklichkeitsnah und genau über einen Vorgang informiert, z. B. in Form von Bedienungsanweisungen, Rezepten, Bau-, Spiel- oder Bastelanleitungen. Sie dient dazu, dass die Adressatin bzw. der Adressat den Vorgang selbst ausführen kann – jeder einzelne Schritt muss also abgebildet werden und der Text muss nachvollziehbar und in einfacher Sprache verfasst werden. Diese Vorkenntnisse helfen den Schülerinnen und Schülern beim Bearbeiten der Aufgabenstellung aus Bsp. DP1-2.

- Beispiel DP2-2: Brainstorming
  Zu Beginn der Unterrichtssequenz wird in Form eines Brainstormings gesammelt, was die Schülerinnen und Schüler bereits über das Programmieren wissen. Um zu vermeiden, dass nur diejenigen etwas sagen können, die konkrete Vorkenntnisse mitbringen, wird folgende Frage gestellt: „An was denkt ihr, wenn ihr das Wort ‚programmieren' hört?".

- Beispiel DP2-3: *LEGO*-Analogie
  Das Prinzip der visuellen Programmiersprache Scratch – Programme aus Blöcken zusammenzubauen – kann anhand von Legosteinen verdeutlicht werden. Diese können auch unterschiedliche Formen und Farben haben und passen genau ineinander. In diesem Rahmen kann z. B. auch gezeigt werden, dass die Scratch-Blöcke eines Skripts miteinander verbunden sein müssen. Zur Verdeutlichung können z. B. übergroße Legosteine eingesetzt werden.

*DP 3 Die kognitive Belastung der Lernenden soll möglichst gering gehalten werden.*

*Theoretische Fundierung*

Das Prinzip basiert auf der *Cognitive Load Theory* von Chandler und Sweller und zielt auf eine Entlastung des Arbeitsgedächtnisses ab. Lernprozesse werden so gestaltet, dass unnötige kognitive Belastung im Bereich des *extraneous cognitive load* reduziert wird und somit Ressourcen für das Verständnis des Lerngegenstands frei werden (siehe Abschnitt 10.2). Im Rahmen der Gestaltung von Programmieraufgaben kann dies bspw. durch *Fading Worked Examples* oder im Rahmen der Unterrichtsgestaltung durch den didaktischen Ansatz *PRIMM* bzw. die *Use-Modify-Create Learning Progression* erreicht werden (siehe Abschnitt 12.3.3). Bezüglich einer Begrenzung der Lerninhalte wird empfohlen, sich auf die algorithmischen Strukturen der Sequenz, Wiederholung und bedingten Anweisung sowie allgemeine Herangehensweisen zur Lösung eines Problems zu beschränken (siehe Abschnitt 12.3.1).

*Umsetzungsprinzipien*

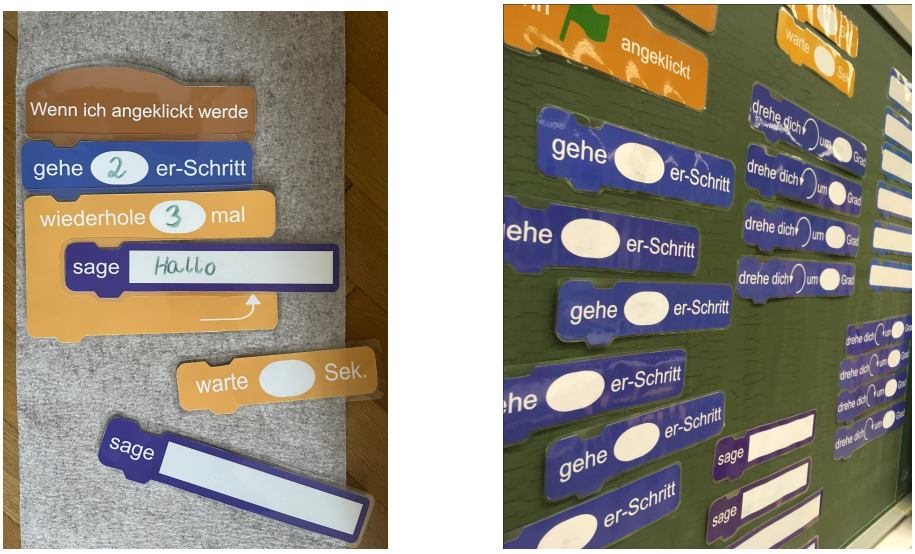
Um die kognitive Belastung der Schülerinnen und Schüler während der Arbeit in der blockbasierten Programmierumgebung zu reduzieren, werden die charakteristischen Blöcke bzw. die algorithmischen Strukturen bereits während der Arbeit mit den *Unplugged*-Materialien eingeführt. Auf diese Art und Weise sollen sich die Schülerinnen und Schüler besser in der Programmierumgebung zurechtfinden, die mit ihren zahlreichen Funktionen schnell überfordern kann. Die Programmierumgebung wird zudem kurz vorgestellt, bevor die Schülerinnen und Schüler diese eigenständig nutzen. Um die kognitive Belastung in Bezug auf die Programmieraufgaben zu reduzieren, wird zunächst mit vorgegebenen Programmen gearbeitet, die in einem nächsten Schritt angepasst oder erweitert werden, bis schließlich eigene Programmideen umgesetzt werden.

*Musterbeispiele*

- Beispiel DP3-1: Haptische Scratch-Blöcke
  Um die Schülerinnen und Schüler für das Programmieren in Scratch vorzuentlasten, werden die Befehle der Programmiersprache zunächst *unplugged* verwendet: Für die Bearbeitung der Aufgaben im Parcours (siehe Bsp. DP1-3) werden haptische Programmierblöcke angefertigt, die optisch den Programmierbefehlen in Scratch entsprechen (siehe Abbildung 14.4). Diese werden mit Magneten und Klettverschlüssen ausgestattet, sodass die Schülerinnen und Schüler mit ihnen sowohl an der Tafel als auch auf Filzbahnen programmieren können.
- Beispiel DP3-2: Orientierung in Scratch
  Um den Schülerinnen und Schülern den Einstieg in Scratch zu erleichtern, wird die Programmierumgebung gemeinsam betrachtet, z. B. über den Beamer. Hier können verschiedene Fragen gestellt werden, z. B. „Wer sieht etwas, das er bereits kennt?“, „Hat jemand eine Idee, wie man das Programm starten kann?“. Denkbar ist außerdem, den ersten Kontakt mit Scratch mittels eines Arbeitsblatts vorzustrukturieren, auf dem die verschiedenen Bereiche von Scratch benannt werden müssen.

- Beispiel DP3-3: Angeleitetes Programmieren

  In Scratch bearbeiten die Schülerinnen und Schüler einen Lernzirkel, in dem die Grundfunktionen von Scratch nacheinander thematisiert werden. Ausgehend von Fragen der Bedienung führen die Zirkelkarten (siehe Abbildung 14.5) über einfache Sequenzen hin zu Kontrollstrukturen wie Wiederholungen und bedingte Anweisungen. An jeder Station wird die entsprechende Funktion zunächst schrittweise erklärt und die Schülerinnen und Schüler wiederholen jeden Schritt an ihrem eigenen Rechner. Im Anschluss daran bearbeiten sie eine dazu passende Aufgabe, bei der sie die eingeführte Funktion bzw. Struktur in einem anderen Kontext oder leicht abgewandelt verwenden müssen.




*(a)* *(b)*

*Abbildung 14.4 Haptische Scratch-Befehle zum Programmieren (a) auf Filzbahnen und (b) an der Tafel*


*(a)* *(b)*

*Abbildung 14.5 Zirkelkarte mit (a) einem Programm, das nachprogrammiert wird und (b) einer passenden Aufgabe.*

*DP 4 Die Aufgaben sind motivierend und ermöglichen aktives sowie kooperatives Lernen.*

*Theoretische Fundierung*

Hinsichtlich der Förderung von Lernmotivation ist auf das *ARCS-Modell* von Keller und die *Selbstbestimmungstheorie der Motivation* von Deci und Ryan zu verweisen (siehe Abschnitt 10.3). Auf Seiten der informatischen Fachdidaktik wurde beschrieben, dass Computer sowie programmierbare Roboter eine besondere Anziehungskraft auf Kinder ausüben (siehe Abschnitt 12.2). Zudem wurden von Martin Dimensionen beschrieben, die zur Gestaltung von motivierenden Programmieraufgaben herangezogen werde können (siehe Abschnitt 12.1.3). Diese beschreiben z. B. die positiven Auswirkungen, die eine selbstbestimmte und aktive Rolle der Lernenden auf deren Motivation hat. Zusätzlich sollten hierbei auch genderspezifische Aspekte miteinbezogen werden (siehe Abschnitt 12.1.4). Der hohe Stellenwert aktiven Lernens wird bereits von Piaget betont. In der Grundschulpädagogik und -didaktik wird zwar auf die Notwendigkeit des Wechsels zwischen einer primär aktiven und rezeptiven Position der Lernenden hingewiesen, die Bedeutung der aktiven Auseinandersetzung mit den Sachverhalten des Unterrichts wird dennoch betont, bspw. im Unterrichtsprinzip der Aktivierung (siehe Abschnitt 11.1.3). Im Kontext des entdeckenden Lernens werden verschiedene Ansätze beschrieben, wie die aktive Auseinandersetzung mit den Lerninhalten von der Lehrkraft angeregt werden kann (siehe Abschnitt 11.1.2). Von weiterer Bedeutung für den kognitiven Entwicklungsprozess und somit das Lernen ist – laut Vygotskys Konzept der *Zone der proximalen Entwicklung* – die soziale Interaktion (siehe Abschnitt 10.1). Durch das Arbeiten in Gruppen kann zudem adaptiv auf heterogene Schülervoraussetzungen eingegangen werden sowie eine thematische Differenzierung im Sinne der Binnendifferenzierung ermöglicht werden (siehe Abschnitt 11.4). Das gemeinsame Arbeiten wird auch im Rahmen der informatikdidaktischen Literatur als lernförderlich beschrieben: das *Pair Programming* ist nicht nur beliebt bei den Schülerinnen und Schülern, vor allem leistungsschwächere Schüler können durch den Ansatz in ihrem Lernprozess unterstützt werden (siehe Abschnitt 12.3.2).

*Umsetzungsprinzipien*

Um die persönliche Motivation der Schülerinnen und Schüler zu wecken, wird ein durchgängiger Kontext für die Unterrichtssequenz gewählt, der sich durch die Aufgaben und Materialien zieht. Dieser Kontext soll an die persönlichen Erfahrungen der Schülerinnen und Schüler anknüpfen, eine lebendige und unterhaltsame Vermittlung der Inhalte und ein breites Spektrum an Aufgaben ermöglichen. Außerdem sollen weder Jungen noch Mädchen vorrangig damit assoziiert werden. Die Schülerinnen und Schüler nehmen von Beginn an eine aktive Rolle im Unterrichtsgeschehen ein. Die Lehrkraft gibt zwar immer wieder Instruktionen und Hilfestellungen, lange lehrerzentrierte Unterrichtsphasen sollen jedoch vermieden werden. Die Aufgaben werden so gestaltet, dass sie Erfolgserlebnisse seitens der Lernenden ermöglichen, personalisiert werden können und – wo möglich – an ihre Alltagserfahrungen anknüpfen. Auch offen gestaltete Programmieraufgaben werden eingesetzt. Während der Unterrichtssequenz gibt es immer wieder Gruppenarbeitsphasen – sowohl in den *Unplugged*-Phasen als auch während der Arbeit am Computer. Die Programmieraufgaben

können im Sinne des *Pair Programming* in Partnerarbeit bearbeitet werden. Falls ergebnisoffen programmiert wird, können sich Gruppen oder Paare je nach persönlicher Neigung bzw. Programmierziel zusammenfinden.

*Musterbeispiele*

- Beispiel DP4-1: Programmierzirkus
  Als Kontext für die Programmieraufgaben kann bspw. das Thema *Zirkus* gewählt werden. Zum einen knüpft das Thema an die Alltagserfahrungen der Schülerinnen und Schüler an, zum anderen ergeben sich daraus viele Programmieranlässe, die sich auf konkrete Handlungen beziehen, z. B. die Begrüßung des Zirkusdirektors, die Bewegung eines Tigers in der Manege, das Erzählen von Witzen eines Clowns.
- Beispiel DP4-2: Schnelle Erfolgserlebnisse
  Sowohl die *Unplugged*-Programmieraufgaben (siehe Bsp. DP1-3) als auch die Stationen im Scratch-Lernzirkel (siehe Bsp. DP3-3) werden zunehmend anspruchsvoller. Durch die relativ einfachen Aufgaben zu Beginn kommen die Schülerinnen und Schüler relativ schnell zu ersten Erfolgserlebnissen.
- Beispiel DP4-3: Personalisierte Programme
  Innerhalb der Programmieraufgaben haben die Schülerinnen und Schüler immer wieder die Möglichkeit, ihre Programme zu personalisieren. Bspw. können sie entscheiden, welchen Witz ihr Clown erzählen soll, oder welchen Zaubertrick ihr Magier aufführt. In einer offenen Aufgabenstellung haben sie außerdem die Gelegenheit, eigene Programmideen zu implementieren. Um die Schülerinnen und Schüler an dieser Stelle nicht zu überfordern können inhaltliche Vorgaben (z. B. „Jeder programmiert seine eigene Zirkusaufführung!“) oder strukturelle Vorgaben (z. B. „Jedes Programm muss mindestens eine Wiederholung enthalten!“) erfolgen.
- Beispiel DP4-4: Gemeinsam programmieren
  Die Programmieraufgaben, die *unplugged* im Parcours gelöst werden, werden in Gruppen bearbeitet. Um eine Aufgabenteilung zu forcieren, kann dies bereits im Aufgabentext gefordert werden (z. B. „Bestimmt ein Kind, dass (1) das Programm vorliest, (2) das Programm im Parcours abläuft, (3) alles auf Richtigkeit überprüft!“). Während des Programmierens in Scratch können die Schülerinnen und Schüler in Paaren zusammenarbeiten. Hier kann z. B. regelmäßig gewechselt werden, wer die Maus bedient.

*DP 5 Die Lernenden durchlaufen verschiedene Schritte des Problemlösens.*

*Theoretische Fundierung*

Schülerinnen und Schüler sind mit Problemen konfrontiert, wenn keine unmittelbare Lösungsroutine zur Verfügung steht. Die Anbahnung des Problemlösens als grundlegende Denk- und Arbeitsweisen wird als zentrale Aufgabe der Grundschule beschrieben (siehe Abschnitt 11.1.4). Als wichtige Voraussetzung für das Lösen von Problemen wird das schlussfolgernde Denken genannt, das vor

allem durch Angebote, welche analoges Schließen erfordern, angeregt werden kann. Im Rahmen von Übungsphasen kann eine flexible Anwendung des Wissens und Könnens durch *elaboriertes* Üben gefördert werden, welches auf Transfer angelegt ist (ebd.). Das Entwickeln einer Problemlösefähigkeit wird auch in der informatikdidaktischen Literatur und Forschung diskutiert und als Ziel des Programmierunterrichts bzw. des Informatikunterrichts beschrieben (siehe Abschnitte 2.1 und 3.1). Mehrere Autoren beschreiben verschiedene Schritte, welche die Lernenden bei der Lösung eines Problems durchlaufen sollten (siehe Abschnitt 12.3.1).

*Umsetzungsprinzipien*

Einige Aufgaben der Unterrichtssequenz werden so konzipiert, dass sie durch Schlussfolgern gelöst werden können. Zudem werden die Schülerinnen und Schüler mit Aufgaben konfrontiert, für die keine offensichtliche Lösungsroutine abgerufen werden kann. Im Rahmen der gemeinsamen Programmiertätigkeit im Plenum werden einzelne Schritte des Problemlösens mit visuellen Hilfsmitteln kenntlich gemacht. Diese können von den Schülerinnen und Schülern im Anschluss beim eigenständigen Programmieren herangezogen werden. Im Rahmen einer offenen Aufgabenstellung werden die Schülerinnen und Schüler an die Planung von Programmen herangeführt.

*Musterbeispiele*

- Beispiel DP5-1: Schlussfolgern ermöglichen
  Der Scratch-Lernzirkel ermöglicht Schlussfolgern, indem Lösungsansätze aus den *Worked Examples* zur Lösung der dazugehörigen Programmieraufgaben herangezogen werden können. Bspw. kann zunächst Schritt für Schritt erklärt werden, wie man in einem Programm prüft, ob eine Figur eine andere Figur berührt, und in der dazugehörigen Aufgabe kann ein Programm gefordert werden, das prüft, ob eine Figur den Mauszeiger berührt.
- Beispiel DP5-2: Problemlöseprozess anleiten
  Im Laufe der Unterrichtssequenz wird im Plenum thematisiert, wie man beim Problemlösen[4] in Scratch vorgeht, z. B. anhand von Symbolen (siehe Abbildung 14.6). Diese können im Klassenraum aufgehängt werden und als Orientierung für die Schülerinnen und Schüler dienen, wenn sie eine Aufgabe bearbeiten.
- Beispiel DP5-3: Planungsphasen
  Da die Komplexität der Programme der Schülerinnen und Schüler in der Regel überschaubar ist, können sie ihre Programmideen zunächst in natürlicher Sprache in Sätzen oder Stichpunkten notieren. Einfache Handlungen können sie außerdem in Form von Bildern oder kurzen Storyboards festhalten. Für den Planungsprozess kann die Lehrkraft eine Vorlage zur Verfügung stellen (siehe Abbildung 14.7).
- Beispiel DP5-4: Austausch ermöglichen
  Bei besonders schwierigen oder offenen Aufgaben, kann das Problemlösen in einem Unterrichtsgespräch thematisiert werden. Bei schwierigen Aufgaben können einzelne Schülerinnen und Schüler über den Beamer zeigen, wie sie die Aufgabe gelöst haben und beschreiben, wie sie

[4] Als Problem wird hier kein Fehler im Programm angesehen, sondern das Implementieren einer Handlung in Scratch.

dabei vorgegangen sind. Im Rahmen offener Aufgaben können verschiedene Ergebnisse von den Schülerinnen und Schülern vorgestellt werden. Dabei wird die Frage „welche besonders harten Nüsse geknackt wurden - und wie“ in den Fragenkanon aufgenommen.


*Abbildung 14.6 Visuelle Impulse „Schritte beim Problemlösen“*

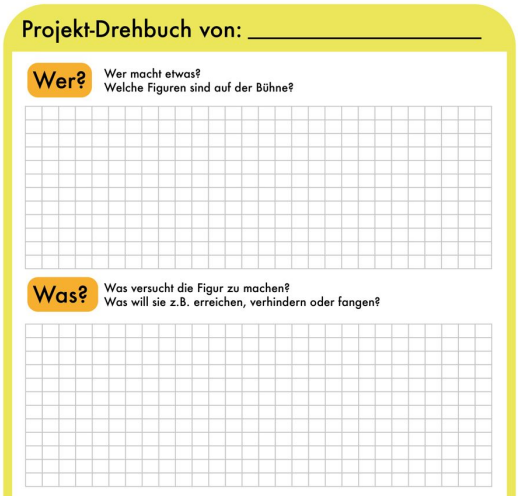



*Abbildung 14.7 Ausschnitt einer Vorlage für ein Projekt-Drehbuch*

### 14.2.2 *Design Principles* mit Bezug zu Lehrhandlungen

Da die Lehraktivitäten im Unterrichtsgeschehen vollzogen werden und in der Regel nicht mit eigens dafür entwickelten Unterrichtsmaterialien o. ä. verbunden sind, wird auf die Konkretisierung in Form von Musterbeispielen verzichtet. Stattdessen werden in der Beschreibung der Umsetzungsprinzipien beispielhafte Unterrichtsszenarien dargestellt. Die Autorin ist sich bewusst, dass Lehr-Lernaktivitäten von vielen Faktoren beeinflusst werden und insbesondere die Lehrer-Schüler-Interaktion schwer vorherzusagen und zu planen ist. Die beschriebenen Szenarien sind lediglich als Anregungen und keinesfalls als streng zu befolgende Handlungsanweisungen zu verstehen.

*DP 6 Die Lehrkraft fördert eine positive Fehlerkultur.*

*Theoretische Fundierung*

Fehler können wertvolle Einblicke in den jeweiligen Lernstand und die Denkweise der Schülerinnen und Schüler geben. Knauf beschreibt drei Ebenen des Umgangs mit Leistungsbewertung – und damit mit Fehlern – die zu einer kindorientierten Lernkultur beitragen (siehe Abschnitt 11.2.3). Auch im Rahmen des Unterrichtsprinzip der Aktivierung und der damit verbundenen Zurücknahme der Steuerung durch die Lehrkraft wird indirekt auf eine positive Fehlerkultur verwiesen: Offensichtliche Fehlversuche im selbstständigen Suchen nach Lösungen werden nicht vorzeitig korrigiert (siehe Abschnitt 11.1.3). Da Fehler beim Programmieren nahezu unvermeidlich sind, wird der Umgang mit ihnen auch in der Informatikdidaktik thematisiert, bspw. durch die Vermittlung konkreter Problemlösungsstrategien zur Fehlersuche und -behebung (siehe Abschnitt 12.3.2).

*Umsetzungsprinzipien*

Die Schülerinnen und Schüler arbeiten selbstständig an ihren Programmen. Auch wenn die Lehrkraft vorzeitig Fehler im Programm erkennt, greift sie nicht ein, sondern gibt ihnen die Möglichkeit, bei der Programmausführung selbst zu erkennen, dass das gewünschte Ergebnis nicht erzielt wird. Sie etabliert, dass bei Fragen zu Programmieraufgaben in einem ersten Schritt die Mitschülerinnen und Mitschüler konsultiert werden. Fehler, die im Rahmen der Programmiertätigkeit aufkommen, können – falls es sich um häufig gemachte Fehler handelt – gemeinsam im Plenum besprochen und gelöst werden. Hierbei ist zu beachten, dass darauf hingewiesen wird, dass die Fehler nicht schlimm sind, sondern dazu gehören und eine Lerngelegenheit für alle darstellen. Insbesondere bei der Umsetzung eigener Programmideen oder wenn die Schülerinnen und Schüler eigene, unvorhergesehene Lösungswege gehen, kann es vorkommen, dass die Lehrkraft die Fehlerquelle nicht sofort erkennt bzw. keine Lösung zur Hand hat. In diesem Fall wird dies offen kommuniziert und gemeinsam nach einer Lösung gesucht bzw. auf konkrete Schritte der Fehlersuche verwiesen (z. B. Lokalisieren des fehlerhaften Skripts, schrittweises Lesen des Programms, Vergleichen mit funktionierenden ähnlichen Skripten). In derartigen Situationen ermutigt die Lehrkraft die Schülerinnen und Schüler, nicht aufzugeben und gibt positive Rückmeldung, falls eine Lösung gefunden wird.

| |
|---|
| *DP 7 Die Lehrkraft orientiert sich an Zielen und fördert die Zielorientierung der Lernenden.* |

*Theoretische Fundierung*

Zielorientierung als Oberbegriff für die Ausrichtung der Lehr-Lernaktivitäten an möglichst klaren Zielen gilt als wichtiges Unterrichtsprinzip (siehe Abschnitt 11.3.3). Maras und Ametsbichler schlagen verschiedene Maßnahmen zur Zielorientierung in der Grundschule vor (ebd.). Gemäß dem Leistungsverständnis in der Grundschule ist es erforderlich, Lernsituationen so zu gestalten, dass jedes Kind erfolgreich dazulernen kann und den individuellen Lernfortschritt anzuerkennen. Gleichzeitig sollten objektive Kriterien miteinbezogen werden, um den Schülerinnen und Schülern die Einordnung der eigenen Leistung zu ermöglichen (siehe Abschnitt 11.2.2). Auch eine mögliche Leistungsbewertung sollte sich sowohl auf die individuelle als auch die kriteriale Bezugsnorm stützen (ebd.).

*Umsetzungsprinzipien*

Im Verlauf der Unterrichtssequenz kommuniziert die Lehrkraft stets die Bedeutung der einzelnen Übungen und warum sie in Hinblick auf das Programmieren wichtig sind. Dies ist besonders während der *Unplugged*-Aktivitäten von Bedeutung, die von den Schülerinnen und Schülern nicht auf Anhieb mit dem Programmieren in Verbindung gebracht werden könnten. Bei offener gehaltenen Programmieraufgaben sollten grundlegende Anforderungen an die Programme kommuniziert werden, die erfüllt werden sollen. Diese können z. B. schriftlich festgehalten oder visualisiert werden. Gegebenenfalls kann während einer Präsentation der einzelnen Programmierprojekte darauf eingegangen werden, inwiefern die Zielvorgaben umgesetzt wurden.

*DP 8 Die Lehrkraft unterstützt den Lernprozess durch Hilfestellungen.*

*Theoretische Fundierung*

In der Literatur zur Grundschulpädagogik und -didaktik wird darauf hingewiesen, dass die Unterstützung durch die Lehrkraft auch in einem Unterricht, der eine aktive und selbstgesteuerte Rolle der Lernenden vorsieht, notwendig ist (siehe Abschnitt 11.1.2). Im Kontext des entdeckenden Lernens können sich Lernhilfen z. B. auf die Strukturierung des Lernmaterials oder die Gesprächsführung im Unterricht beziehen (ebd.). In Hinblick auf die Individualisierung und Differenzierung, die für den Unterricht der Grundschule gefordert werden, spielt die kognitive Strukturierung durch Hilfestellungen eine wesentliche Rolle. Wood beschreibt in seinem Ansatz des *Contingent Support for Learning* verschiedene Ebenen der Hilfestellung, um Schülerinnen und Schüler bei Schwierigkeiten in ihrem Lernprozess zu unterstützen (siehe Abschnitt 11.4.4).

*Umsetzungsprinzipien*

Im Rahmen der Programmiertätigkeit unterstützt die Lehrkraft die Schülerinnen und Schüler beim Auftauchen von Problemen. Dabei wird die Lösung nicht direkt mitgeteilt, sondern zunächst unterschiedliche verbale Hilfestellungen gegeben. Diese geben zunächst keine spezifischen weiteren Schritte vor, z. B. könnte der Ablauf des fehlerhaften Programms schrittweise verbalisiert werden oder die Handlung könnte – falls möglich – von der Lehrkraft ausgeführt werden. Falls weitere Hilfe benötigt wird, kann die Lehrkraft sich in einer Äußerung auf einen spezifischen Schritt zur Problemlösung beziehen oder herausstellen, wo ein Fehler liegt (z. B. „Ich glaube, dass die Sequenz hier nicht passt, schau dir die Stelle einmal genauer an.“). In einem nächsten Schritt können über das Stellen von geschlossenen Fragen bestimmte Handlungsmöglichkeiten vorgegeben werden, z. B. durch Äußerungen wie „Brauchst du hier eine Endloswiederholung oder eine bedingte Wiederholung?“. In einem letzten Schritt kann die Lehrkraft demonstrieren, was als nächstes zu tun ist oder das Vorgehen in Worten beschreiben – diese Variante sollte jedoch nur als letztes Mittel gesehen werden. Neben den verbalen Hilfestellungen werden zu ausgesuchten Aufgaben Tippkarten angefertigt, die eigenständig von den Schülerinnen und Schülern zu Rate gezogen werden können und auf welche die Lehrkraft hinweisen kann.

## 14.3 Didaktische Konzeption der Unterrichtssequenz

### 14.3.1 Planungsmethodik (Unterrichtssequenz)

Zur Vorbereitung der didaktischen Konzeption der Unterrichtssequenz wurden von der Autorin Leitfadeninterviews zur Informatik in der Grundschule mit sechs Grundschullehrkräften geführt. In der Analyse zeigt sich bzgl. der Gestaltung von Lehr-Lernsituationen, dass die Schülerinnen und Schüler erkennen sollten, wie ein Computer funktioniert und wie sie dies beeinflussen können; dass sie einen Einblick bekommen sollten, wie etwas funktioniert, dass sie gut kennen, z. B. ein

Computerspiel und dass sie die Möglichkeit bekommen sollten, selbst etwas zu tun, z. B. etwas nach bestimmten Vorgaben entwickeln (Funke, Geldreich und Hubwieser 2016, S. 137 f.). Alle angesprochenen Aspekte wurden in den *Design-Principles* bereits berücksichtigt.

Ausgehend von den Überlegungen in den Kapiteln 14.1 und 14.2 wurde zunächst ein Grobentwurf entwickelt, welcher den Verlauf der einzelnen Unterrichtsphasen und -schritte zusammenfassend darstellt. Die Elemente und der Ablauf des Entwurfs wurden dabei aus den Umsetzungsprinzipien der einzelnen *Design-Principles* hergeleitet. Bspw. steht in den Umsetzungsprinzipien zu *DP 3* („Die kognitive Belastung der Lernenden soll möglichst gering gehalten werden"), dass die Blöcke der blockbasierten Programmierumgebung bzw. die algorithmischen Strukturen bereits während der Arbeit mit den *Unplugged*-Materialien eingeführt werden. Daraus wurde Unterrichtsschritt 3a abgeleitet: „Bearbeiten von Programmieraufgaben mit haptischen Programmierbefehlen". Durch dieses Vorgehen wurden automatisch alle *Design-Principles* mit Bezug zu Lernaktivitäten realisiert, was jedoch nicht bedeutet, dass diese nicht auch in einer anderen Form hätten umgesetzt werden können. Es handelt sich vielmehr um eine exemplarische Umsetzung der weitaus abstrakteren *Design-Principles*. Der Grobentwurf in Form einer unterrichtlichen Handlungslinie wurde daraufhin zu einer Feinplanung weiterentwickelt. Hierbei wurde in erster Linie auf die Musterbeispiele der einzelnen *Design-Principles* zurückgegriffen. Auch hier gilt, dass somit automatisch sämtliche Gestaltungsrichtlinien realisiert wurden. Es handelt sich allerdings auch hier lediglich um eine exemplarische Feinplanung: die *Design-Principles* und Umsetzungsprinzipien hätten zu anderen bzw. weiteren Musterbeispielen ausgearbeitet werden können, was zu einer anderen Feinplanung geführt hätte.

Während der didaktischen Konzeption der Unterrichtssequenz wurden die erarbeiteten Umsetzungsprinzipien und Musterbeispiele sowie erste Entwürfe des Ablaufs regelmäßig in Feedbackgesprächen mit Experten der Informatikdidaktik und des Unterrichtens sowie dem informatikdidaktischen Oberseminar der TUM-DDI präsentiert. Diskutiert wurde dabei zunächst, welche algorithmischen Strukturelemente in welcher Reihenfolge eingeführt werden sollen. Hierbei einigte man sich auf folgenden Ablauf: Anweisung, Sequenz, Wiederholung, bedingte Anweisung. Bei der Vorstellung der Idee eines Kontextes für die Programmieraufgaben in Scratch wurden mehrere diskutiert, wie z. B. „Maschinenzirkus", „Dschungel" und „Programmierzirkus". Der Kontext „Maschinenzirkus" wurde wieder verworfen, da er vermutlich die Erwartung wecken würde, dass Roboter programmiert würden. Die Wahl fiel schließlich auf „Programmierzirkus", da wahrscheinlich viele Schülerinnen und Schüler bereits Erfahrungen im Rahmen eines Zirkusbesuchs gemacht haben und er viele Programmieranlässe eröffnet. Außerdem ist es ein Kontext, der nicht in erster Linie mit Jungen oder Mädchen assoziiert wird. Das Feedback auf die vorgestellten Ideen war durchweg sehr positiv. Es wurde als besonders wichtig erachtet, dass die Schülerinnen und Schüler sehr zeitnah zu Beginn der Unterrichtssequenz die Lehrkraft oder sich gegenseitig programmieren, damit sie möglichst schnell einen ersten Eindruck davon bekommen, was Programmieren eigentlich ist. Die Idee, haptische Scratch-Blöcke für die *Unplugged*-Aufgaben anzufertigen, wurde als

sehr originell und kindgerecht eingeschätzt – diskutiert wurden zudem verschiedene Möglichkeiten zur Herstellung der Blöcke, z. B. aus Holz mittels eines Lasercutters oder aus laminiertem Papier. Um die Farben der Blöcke originalgetreu darzustellen, entschied man sich für das Ausdrucken und Laminieren der Blöcke. Es wurde außerdem angemerkt, dass der Einfluss der Lehrkraft im Hinblick auf die Beforschung des Unterrichtskonzepts so gering wie möglich gehalten werden sollte. Daher entschied man sich, im Scratch-Lernzirkel relativ detaillierte Anleitungen zu verwenden. Ausführlich wurde diskutiert, welche Vorgaben den Schülerinnen und Schülern für ihre eigenen Programmierprojekte gemacht werden sollten. Hier stand z. B. zur Debatte, eine Bildergeschichte zu entwerfen, welche zunächst in Scratch implementiert werden sollte und dann frei weiterentwickelt werden könnte. Da man feststellen wollte, ob die die Schülerinnen und Schüler bestimmte algorithmische Strukturelemente einsetzen können, und um sie inhaltlich nicht zu sehr einzuschränken, entschied man sich, ihnen nur wenige Vorgaben zu geben, die sich vorrangig auf die Struktur des Programms beziehen.

### 14.3.2 Entwurf einer unterrichtlichen Handlungslinie

Auf Grundlage der Überlegungen in Kapitel 14.1 und 14.2 wurde eine exemplarische Handlungslinie erarbeitet, welche den Verlauf der einzelnen Unterrichtsphasen und -schritte zusammenfassend darstellt. Der Ablauf ist jedoch nicht als starres Muster zu verstehen, sondern kann situativ abgewandelt werden. Eine tabellarische Auflistung der einzelnen Unterrichtsschritte inklusive der Herleitung aus den Umsetzungsprinzipien und Verbindung zu den Grobzielen der Unterrichtssequenz findet sich in Anhang B.1.1. Die Handlungslinie umfasst die folgenden Phasen und Kurzbeschreibungen:

(1) Thematischer Einstieg

   a) Aktivierung von Vorwissen der Schülerinnen und Schüler (DP 2)

(2) Grundlagen von Algorithmen und Programmen

   a) Formulieren von ersten Anweisungen und Verständigung über deren Eigenschaften (DP 1, DP 4)

   b) Herausarbeiten des Unterschieds zur natürlichen Sprache (DP 1)

(3) Programmieren *unplugged*

   a) Bearbeiten von Programmieraufgaben mit haptischen Programmierbefehlen (DP 1, DP 3)

   b) Kontrolle der Lösungen im Klassenraum (DP 1)

(4) Programmieren in Scratch (DP 3)

   a) Präsentation der Oberfläche von Scratch (DP 3)

   b) Angeleitetes Programmieren in Scratch und Bearbeiten von Transferaufgaben (DP 3, DP 4, DP 5)

(5) Eigene Programmierprojekte

   a) Planen eigener Programmideen (DP 5)

   b) Umsetzen der Ideen in Scratch (DP 4, DP 5)

   c) Präsentation der Ergebnisse (DP 4)

### 14.3.3 Geplanter Verlauf des Unterrichts

Der geplante Verlauf der Unterrichtssequenz wird im Folgenden als Feinplanung in Form einer tabellarischen Ablaufplanung dargestellt (siehe Tabelle 14.7). Die Spalten „Phase“ und „Kurzbeschreibung“ werden darin unverändert aus der unterrichtlichen Handlungslinie übernommen. Unter „Lernaktivitäten“ werden nach Tulodziecki, Herzig und Blömeke (2017, S. 183) „alle gedanklichen und physischen Vorgänge beim Lernenden, die in einem Lernzusammenhang auf die Auseinandersetzung mit dem jeweiligen Thema gerichtet sind“, beschrieben. Die Spalte „Lehrhandlungen“ erläutert „alle Aktivitäten der Lehrperson, die darauf zielen, Lernaktivitäten beim Lernenden anzuregen oder zu unterstützen“ (ebd.). Unter „Handlungsmuster/Sozialform“ werden die jeweiligen Ausprägungen nach Meyer (2008, S. 125 ff.) ausgewiesen. Unter „Material/Medien“ wird angegeben, welche davon in den Unterrichtsphasen verwendet werden.

*Tabelle 14.7 Tabellarische Ablaufplanung der Unterrichtssequenz*

| Phase | Kurz-beschreibung | Lernaktivitäten | Lehrhandlungen | Handlungs-muster/ Sozialform | Material/ Medien |
|---|---|---|---|---|---|
| Thematischer Einstieg | Aktivierung von Vorwissen der Schülerinnen und Schüler | Die SuS äußern ihre Assoziationen zum Begriff „Programmieren“ (→ Bsp. DP2-2). | Die Lehrkraft schreibt den Begriff „Programmieren“ an die Tafel oder ein Plakat und stellt folgende Impulsfrage: „An was denkt ihr, wenn ihr das Wort programmieren hört?“. Sie dokumentiert die Äußerungen der SuS. | Brain-storming, Unterrichts-gespräch | Tafel bzw. Plakat |

| | | | | | |
|---|---|---|---|---|---|
| Grundlagen von Algorithmen und Programmen | Formulieren von ersten Anweisungen und Verständigung über deren Eigenschaften | Die SuS programmieren die Lehrkraft mittels mündlicher Anweisungen (→ Bsp. DP1-1). Sie machen spontane Vorschläge und bekommen in der Interaktion indirekt Rückmeldung, welche Befehle korrekt formuliert wurden. Sie schlussfolgern, was die korrekten Befehle ausgemacht hat bzw. wie korrekt formulierte Befehle formuliert sein müssen. Danach spielen SuS den Roboter. | Die Lehrkraft führt nur die Befehle aus, welche die gewünschte Handlung in einzelne Schritte herunterbricht; bei zu komplexen Befehlen rührt sie sich nicht. Die Befehle werden zudem wörtlich ausgeführt, z. B. beim Befehl „gehe geradeaus“ geht die Lehrkraft solange vorwärts, bis sie irgendwo anstößt. | Unterrichtsgespräch | |
| | Herausarbeiten des Unterschieds zur natürlichen Sprache | In Kleingruppen verfassen die SuS zu einer bildlichen Anleitung sprachliche Anweisungen (→ Bsp. DP1-2, DP2-1). Sie beschreiben die Bilder möglichst genau, ohne dass ein Detaillevel vorgegeben wird. Sie stellen fest – auch in der Auseinandersetzung mit anderen Lösungen – dass die natürliche Sprache zuweilen sehr unkonkret ist und man im Alltag sehr viele Informationen aus dem Kontext schließt. | Die Lehrkraft gibt den SuS auch bei Rückfragen nicht vor, wie detailliert die Anleitung beschrieben werden muss bzw. wie viele Befehle formuliert werden müssen.<br><br>Nachdem die SuS die Anweisungen formuliert haben, wird im Plenum besprochen, ob ihnen die Aufgabe leicht/schwer gefallen ist. Zudem wird verglichen, wie viele Anweisungen jeweils von den Gruppen formuliert wurden.<br><br>Die Lehrkraft verdeutlicht, dass Computer ganz genaue Anweisungen brauchen und Programmiersprachen deshalb – im Gegensatz zur natürlichen Sprache – eindeutig sind (→ DP7). | Gruppenarbeit, Unterrichtsgespräch | bildliche Anleitungen (siehe Abschnitt B.2.1), Schreibmaterial |

| | | | | | |
|---|---|---|---|---|---|
| Programmieren *unplugged* | Bearbeiten von Programmier-aufgaben mit haptischen Programmier-befehlen | Die SuS bearbeiten in Klein-gruppen verschiedene Programmier-aufgaben, in denen der Weg einer Person/eines Tieres zum Erreichen eines bestimmten Ziels beschrieben werden muss (→ Bsp. DP1-3). Sie einigen sich zunächst auf einen Lösungsweg, zeichnen diesen auf den Arbeitsblättern ein und erstellen im Anschluss ein Programm mithilfe der haptischen Scratch-Blöcke (→ Bsp. DP3-1). | Die Lehrkraft stellt die haptischen Scratch-Blöcke vor. Zur Veran-schaulichung des Wirkprinzips können z. B. übergroße Legosteine eingesetzt werden (→ Bsp. DP2-3).<br><br>Die Lehrkraft unterstützt die SuS durch verbale Hilfestellungen (→ DP8). | Gruppen-arbeit, Lehrervortrag | Arbeits-blätter (siehe Abschnitt B.2.2), haptische Program-mier-befehle, Schreib-material |
| | Kontrolle der Lösungen im Klassenraum | Gemäß der Aufgabenstellung bestimmen die SuS jeweils ein Kind, das das Programm vorliest und abläuft. Gruppenweise laufen die SuS ihre Lösungen in einem aufgebauten Parcours ab – je nach Platz im Klassenraum, Gang, Hof (→ Bsp. DP1-3). Der Rest der Gruppe kontrolliert, ob das Programm richtig ausgeführt wird. | Im Plenum werden die ggf. unter-schiedlichen Lösungen miteinander verglichen. Falls es sich anbietet, kann die Lehrkraft zum Gebrauch algorithmischer Strukturelemente überleiten, indem sie z. B. fragt, wie man ein Programm möglichst kurz formulieren kann. | Unterrichts-gespräch | Teppich-fliesen |

| | | | | | |
|---|---|---|---|---|---|
| Programmieren in Scratch | Präsentation der Oberfläche von Scratch | | Die Lehrkraft präsentiert das Programm Scratch über den Beamer (→ Bsp. DP3-2) und stellt im Gespräch mit den SuS grundlegende Funktionsweisen vor (Blöcke kombinieren, Programm starten, Figur in Figurenleiste auswählen). | Unterrichtsgespräch | Beamer, Scratch |
| | Angeleitetes Programmieren in Scratch und Bearbeiten von Transferaufgaben | Die SuS bearbeiten einen Lernzirkel zu Scratch – den Programmierzirkus – in dem die Grundfunktionen und algorithmischen Strukturelemente nacheinander thematisiert werden (→ Bsp. DP3-3, 4-1, 4-2). Sie programmieren zunächst vorgegebene Programme in Scratch nach und bearbeiten im Anschluss dazugehörige Aufgaben (→ Bsp. DP4-3, 5-1). Je nach Computerausstattung bzw. Vorliebe können die SuS in Paaren arbeiten (→ Bsp. DP4-4). | Während des Lernzirkels führt die Lehrkraft verschiedene Schritte des Problemlösens und einhergehende Symbole ein, die für alle sichtbar im Raum platziert werden (→ Bsp. DP5-2).<br><br>Die Lehrkraft unterstützt die SuS durch verbale Hilfestellungen und verweist auf Tippkarten zu einzelnen Stationen (→ DP8). Sie greift Fragen/Probleme, die häufig vorkommen, auf und thematisiert sie im Plenum und ggf. über den Beamer (→ Bsp. DP5-4, DP6). | Einzel- oder Partnerarbeit, Lehrervortrag | Scratch, Zirkelkarten (siehe Abschnitt B.2.3), ggf. Beamer |

| | | | | | |
|---|---|---|---|---|---|
| Eigene Programmier-projekte | Planen eigener Programm-ideen | Die SuS überlegen sich eine Handlung für ihr Programm und füllen dazu die Planungsvorlage aus (→ Bsp.DP4-3, 5-3). Je nach Computerausstattung bzw. Vorliebe können die SuS in Paaren arbeiten (→ Bsp. DP4-4). | Die Lehrkraft stellt im Gespräch mit den SuS die Wichtigkeit des Planens von Programmen heraus. Sie stellt die Planungs-vorlage für die Programmier-projekte der SuS vor sowie die Vorgaben: die Programme sollen (a) mehrere Figuren beinhalten, die sich (b) bewegen; es soll (c) eine Wiederholung und (d) eine bedingte Anweisung ver-wendet werden (→ DP7). | Lehrer-vortrag, Einzel- oder Partnerarbeit | Planungs-vorlagen (siehe Abschnitt B.2.4), Karten mit Vorgaben, Schreib-material |
| | Umsetzen der Ideen in Scratch | Die SuS implementieren ihre geplanten Programme in Scratch. | Bei Fragen fordert die Lehrkraft die SuS auf, zuerst ihre Mit-schülerinnen und Mitschüler um Rat zu fragen. Komplexe Fragestellungen/Probleme werden gemeinsam mit den SuS gelöst – dabei können konkrete Schritte der Fehlersuche eingeführt werden sowie verbale Hilfe-stellungen gegeben werden (→ DP6, DP8). | Einzel- oder Partnerarbeit | Scratch |
| | Präsentation der Ergebnisse | Die SuS präsentieren ihre Ergebnisse (freiwillig) über den Beamer. Dabei erläutern sie, welche Schwierigkeiten sie bei der Umsetzung gemeistert haben (→ Bsp. DP5-4). Jedes Programm wird mit Applaus wertgeschätzt. | | Präsentation im Plenum | Beamer, Scratch |

# 15 Entwicklung eines Fortbildungskonzepts

## 15.1 Zielvorstellungen (Lehrerfortbildung)

Wie bereits bei der Entwicklung der Unterrichtssequenz werden zunächst die Ziele erläutert, die mit dem Fortbildungskonzept angestrebt werden sollen. Auf der Grundlage der Ausführungen in Kapitel 4.1 und 4.2 wird dabei auf informatische und informatikdidaktische Kompetenzen sowie affektive Merkmale der Lehrkräfte eingegangen (siehe Abbildung 4.2). Der Begriff Kompetenz wird, wie bereits in Kapitel 14.1, „als Summe von bestimmten tätigkeitsrelevanten Qualifikationen (z. B. Kenntnissen, Fertigkeiten und Fähigkeiten)“ ausgelegt (Schaper 2008, 91, siehe Abschnitt 11.1.1). Die informatischen und informatikdidaktischen Kompetenzen sind demnach als Fertigkeiten zu verstehen, die für das Unterrichten der Programmierung in der Grundschule relevant sind.

### 15.1.1 Informatische Kompetenzen

Innerhalb der Ausführungen zu den Kompetenzen zum Unterrichten des Fachs Informatik werden unterschiedliche Kompetenzmodelle und Empfehlungen bzgl. des notwendigen Fachwissens beschrieben (siehe Abschnitt 4.2). Die im Rahmen der Fortbildung zu erwerbenden informatischen Kompetenzen orientieren sich in erster Linie an den fachlichen Inhalten, die während der Unterrichtssequenz thematisiert werden. Um die Kenntnisse der Lehrkräfte zu den Grundelementen von Algorithmen zu erweitern, wird außerdem das Variablenkonzept ergänzt (siehe Abschnitt 9.1.2). Dies steht im Einklang mit den Ausführungen von Bergner u. a. (2018) zu den informatischen Fachkompetenzen auf Ebene der pädagogischen Fach- und Lehrkräfte. Für den Inhaltsbereich „Algorithmen und Programmierung “ heißt es darin:

> „Der für die Informatik essenzielle Begriff des Algorithmus, ist auch für die Konstruktion von Kompetenzerwartungen für die pädagogischen Fach- und Lehrkräfte ein wesentlicher Inhaltsbereich. [...] Hierbei sollten die wesentlichen Kontrollstrukturen eines Algorithmus wie Sequenz, bedingte Verzweigung und Schleifen/Wiederholungen ebenso kennengelernt werden wie ein elementares Variablenkonzept mit elementaren Datentypen.“ (ebd., S. 177 f.)

Innerhalb der *CSTA Standards for Computer Science Teachers* (2020) sind mehrere Aspekte des 1. Standards *CS Knowledge & Skills* für diese Arbeit relevant:

- *„**1a. Apply CS practices:** Apply CS and computational thinking practices in flexible and appropriate ways. Practices include: Fostering an Inclusive Computing Culture, Collaborating Around Computing, Communicating About Computing, Recognizing and Defining Computational Problems, Developing and Using Abstractions, Creating Computational Artifacts, and Testing and Refining Computational Artifacts.*
- ***1b. Apply knowledge of computing systems:** Apply knowledge of how hardware and software function to input, process, store, and output information within computing systems by analyzing interactions, designing projects, and troubleshooting problems.*
- ***1e. Develop programs and interpret algorithms:** Design, implement, debug, and review programs in an iterative process using appropriate CS tools and technologies. Interpret algorithms, and explain tradeoffs associated with different algorithms.“* (CSTA 2020, S. 2)

In Anlehnung an *Substandard 1a* und *1b* der *CSTA Standards* wird das Konzept *Computational Thinking* und das EVA-Prinzip in den Zielvorstellungen ergänzt. Aus den in Kapitel 14.1.1 entwickelten kognitiven Lernzielen für die Schülerinnen und Schüler und den obigen Ausführungen ergeben sich für die Lehrkräfte folgende Zielkompetenzen:

- Die Lehrkräfte führen Algorithmen in ihrer Lebenswelt bzw. der Lebenswelt der Schülerinnen und Schüler aus;
- Die Lehrkräfte beschreiben Algorithmen in Alltagssprache;
- Die Lehrkräfte entwerfen, realisieren und testen Algorithmen mit den algorithmischen Grundbausteinen Anweisung, Sequenz, Wiederholung und Verzweigung;
- Die Lehrkräfte programmieren ein für die Grundschule altersentsprechendes Informatiksystem;
- Die Lehrkräfte verwenden Variablen und Wertzuweisungen[1];
- Die Lehrkräfte ordnen Bestandteile eines Informatiksystems der Eingabe, der Verarbeitung und der Ausgabe zu[1];
- Die Lehrkräfte wenden Aspekte des *Computational Thinking* an, z. B. *Decomposition*, *Abstraction*, oder *Patterns/Generalization*;

### 15.1.2 Informatikdidaktische Kompetenzen

Das primäre Ziel der Lehrerfortbildung besteht darin, die Lehrkräfte zu befähigen, die Unterrichtssequenz zum Programmieren mit ihren Schülerinnen und Schülern durchzuführen. In einem weiteren Schritt sollen sie in die Lage versetzt werden, diese abzuwandeln oder zu erweitern und ggf. sogar

[1] Die Formulierung wurde aus den Bildungsstandards Informatik für die Sekundarstufe I der Gesellschaft für Informatik (GI 2008, S. 16 f.) entnommen.

eigene Lehr-Lernaktivitäten zum Programmieren zu planen und durchzuführen. Die entsprechenden fachdidaktischen Kompetenzen werden im Folgenden aus bestehenden Kompetenzmodellen abgeleitet, die bereits in Kapitel 4.2 vorgestellt wurden. Zunächst werden die für diese Arbeit relevanten Formulierungen aus dem informatikspezifischen Kompetenzprofil der KMK (2019), aus den Kompetenzen für Fach- und Lehrkräfte bzgl. der Gestaltung von Lernsituationen der informatischen Bildung im Elementar- und Primarbereich des Haus der kleinen Forscher (Bergner u. a. 2018) sowie der *CSTA Standards for Computer Science Teachers* (2020) miteinander abgeglichen und nach inhaltlicher Entsprechung sortiert (siehe Tabelle 15.1). Im Modell von Bergner u. a. (2018) werden bei den informatikdidaktischen Kompetenzen die Bereiche „Kompetenzen zur Planung von informatischen Lernumgebungen und Lernsituationen“ sowie „Kontextuelle informatikdidaktische Handlungskompetenz“ berücksichtigt (siehe Abbildung 15.1); in den *CSTA Standards* die Bereiche *Instructional Design* und *Classroom Practice*. Im Anschluss werden daraus informatikdidaktische Kompetenzen formuliert, die im Rahmen der Lehrerfortbildung erreicht werden sollen:

- Die Lehrkräfte sind in der Lage, Konzepte für das Erlernen einer einfachen Programmiersprache mit einem spielerischen, auch *Unplugged*-Ansatz, praktisch umzusetzen;
- Die Lehrkräfte sind in der Lage, Schülerinnen und Schüler für das Programmierenlernen zu motivieren sowie motivierende Aktivierungsmethoden und informatiktypische Sozialformen kompetenzfördernd einzusetzen;
- Die Lehrkräfte sind in der Lage, kindgemäße Software und einfache altersgemäße Informatiksysteme für das Programmieren im Primarbereich auszuwählen und sinnvoll in Lernszenarien zu integrieren;
- Die Lehrkräfte sind in der Lage, Aufgabenstellungen und Lerninhalte des Programmierens kindgemäß und auf unterschiedliche Weise vereinfacht, aber korrekt darzustellen sowie abstrakte informatische Konzepte durch verschiedene Beispiele und mittels spielerischer und explorativer Erkundungsanlässe für die Schülerinnen und Schüler erfahrbar zu machen;
- Die Lehrkräfte sind in der Lage, typische Präkonzepte, Verstehens- und Lernbarrieren der Schülerinnen und Schüler in Bezug auf das Programmieren zu identifizieren und entsprechende Formalisierungen, Abstraktionen und Interventionsmöglichkeiten einzusetzen;
- Die Lehrkräfte können auch in Gruppenarbeitssituationen mit Informatiksystemen auf Lernschwierigkeiten einzelner Kinder eingehen und gezielte individuelle Hilfestellungen geben;
- Die Lehrkräfte sind in der Lage, individuelle Leistungsstände der Schülerinnen und Schüler zu identifizieren und darauf durch situationsangemessenes Handeln zu reagieren;
- Die Lehrkräfte regen die Schülerinnen und Schüler zum selbstständigen Lernen und zur eigenständigen Begutachtung ihrer Lernergebnisse an und fördern so deren Selbstwirksamkeit;
- Die Lehrkräfte fördern durch Gruppengespräche die Reflexion der Schülerinnen und Schüler über ihr eigenes Handeln und ermöglichen so positives Feedback über die erzielten Ergebnisse;

*Tabelle 15.1 Ableitung der informatikdidaktischen Kompetenzen aus dem informatikspezifischen Kompetenzprofil der Kultusministerkonferenz (2019), der fachdidaktischen Kompetenzen für Fach- und Lehrkräfte bzgl. der Gestaltung von Lernsituationen der informatischen Bildung im Elementar- und Primarbereich des Haus der kleinen Forscher (Bergner u. a. 2018) sowie der CSTA Standards for Computer Science Teachers (2020)*

| Kompetenzprofil der Kultusminister-konferenz (KMK 2019) | Fachdidaktischen Kompetenzen für Fach- und Lehrkräfte bzgl. der Gestaltung von Lernsituationen der informatischen Bildung im Elementar- und Primarbereich (Bergner u. a. 2018) | *CSTA Standards for Computer Science Teachers (CSTA 2020)* | **Informatikdidaktische Kompetenzen** |
|---|---|---|---|
| Sie können fachdidaktische Konzepte und empirische Befunde informatikbezogener Lehr-Lernforschung und Diagnose-werkzeuge nutzen, um individuelle Denkwege und Vorstellungen von SuS je nach ihren persönlichen Voraussetzungen, Vorerfahrungen und Fähigkeiten zu analysieren, SuS für das Lernen von Informatik zu motivieren sowie individuelle Lernfortschritte zu fördern und zu bewerten (KMK 1) | Die pädagogischen Fach- und Lehrkräfte sind in der Lage, typische informatische Präkonzepte, Verstehens- und Lernbarrieren der Kinder zu identifizieren und erforderliche Formalisierungen und Abstraktionen Informatischer Prinzipien entsprechend den kognitiven Fähigkeiten der Kinder umzusetzen (HdkF 1) | *Plan instruction to foster student understanding (Plan activities that use evidence-based, CS-specific teaching strategies to develop students' conceptual understanding and proactively address student misconceptions in CS; CSTA 4f)* | **Die Lehrkräfte sind in der Lage, typische Präkonzepte, Verstehens- und Lernbarrieren der SuS in Bezug auf das Programmieren zu identifizieren und entsprechende Formalisierungen, Abstraktionen und Interventionsmöglichkeiten einzusetzen** |
| | Die pädagogischen Fach- und Lehrkräfte kennen fachspezifische Interventions-möglichkeiten und können diese situativ in informatischen Lernszenarien anwenden (z. B. mit informatischen Fehlvorstellungen, Umgang mit vorläufigen und ungenauen Fachbegriffen, Reaktion auf kindliche Denkkonstruktionen, heuristische Hilfestellungen) (HdkF 2) | | |
| | Die pädagogischen Fach- und Lehrkräfte sind in der Lage, im Rahmen ihres pädagogischen Handelns in informatischen Lernszenarien Kinder für das Lernen von Informatik zu motivieren sowie individuelle Lernfortschritte zu fördern und zu bewerten (HdkF 3) | *Plan projects that have personal meaning to students (Plan opportunities for students to create and share open-ended and personally meaningful projects; CSTA 4e)* | **Die Lehrkräfte sind in der Lage, SuS für das Programmierenlernen zu motivieren sowie motivierende Aktivierungsmethoden und informatiktypische Sozialformen kompetenzfördernd einzusetzen** |
| | Die pädagogischen Fach- und Lehrkräfte sind in der Lage, für die informatische Bildung motivierende informatiktypische Sozialformen wie z. B. Gruppenarbeit an Informatiksystemen, Rollenspiele oder *CS-Unplugged*-Methoden für die Kinder kompetenzfördernd einzusetzen (HdkF 4) | | |
| | Die pädagogischen Fach- und Lehrkräfte sind in der Lage, in kreativer und motivierender Weise Methoden zur Aktivierung der Kinder in informatischen Lernprozessen im Elementar- und Primar-bereich anzuwenden, um den Lernprozess zu steuern und zu fördern (HdkF 5) | | |
| | Die pädagogischen Fach- und Lehrkräfte sind in der Lage, unterschiedliche individuelle Leistungsstände der Kinder in informatischen Bildungsprozessen während des praktischen pädagogischen Handelns zu identifizieren und auf diese Heterogenität durch geeignetes situatives Handeln angemessen zu reagieren (z. B. Prozesshilfen, natürlich differenzierende Spiel- und Erkundungsumgebungen und Lernarrangements) (HdkF 6) | *Inform instruction through assessment (Develop multiple forms and modalities of assessment to provide feedback and support. Use resulting data for instructional decision-making and differentiation; CSTA 4g)* | **Die Lehrkräfte sind in der Lage, individuelle Leistungsstände der SuS zu identifizieren und darauf durch situations-angemessenes Handeln zu reagieren** |
| | Die pädagogischen Fach- und Lehrkräfte regen die Kinder in informatischen Lernprozessen zum selbstständigen Lernen und zur eigenständigen Begutachtung ihrer so erzielten informatischen Lösungen an und fördern damit deren Selbstwirksamkeit in informatisch-technischen Kompetenz-bereichen (HdkF 7) | *Promote student self-efficacy (Promote student self-efficacy by facilitating student creativity, choice in product and process, and self-directed learning; CSTA 5c)* | **Die Lehrkräfte regen die SuS zum selbstständigen Lernen und zur eigenständigen Begutachtung ihrer Lernergebnisse an und fördern so deren Selbstwirksamkeit** |

| | | | |
|---|---|---|---|
| | Die pädagogischen Fach- und Lehrkräfte fördern durch Gruppengespräche die Reflexion der Kinder über ihr eigenes Handeln mit Informatiksystemen und sorgen für eine kooperative Ermittlung des Lernerfolgs und für ein positives Feedback über die erzielten Ergebnisse an die Kinder (HdkF 8) | *Cultivate a positive classroom climate (Cultivate a positive classroom climate that values and amplifies varied perspectives, abilities, approaches, and solutions; CSTA 5b)*<br><br>*Support student collaboration (Provide structured opportunities for students to collaborate in CS. Develop students' ability to provide, receive, and respond to constructive feedback in the design, implementation, and review of computational artifacts; CSTA 5d)* | **Die Lehrkräfte fördern durch Gruppengespräche die Reflexion der SuS über ihr eigenes Handeln und ermöglichen so positives Feedback über die erzielten Ergebnisse** |
| | Die pädagogischen Fach- und Lehrkräfte können auch in Gruppenarbeitssituationen mit Informatiksystemen auf Lernschwierigkeiten einzelner Kinder eingehen und gezielte individuelle Hilfestellungen geben (HdkF 9) | | **Die Lehrkräfte können auch in Gruppenarbeitssituationen mit Informatiksystemen auf Lernschwierigkeiten einzelner Schülerinnen und Schüler eingehen und gezielte individuelle Hilfestellungen geben** |
| Sie kennen Möglichkeiten zur Illustration von informatischen Prinzipien, welche die visuelle, auditive und haptische Wahrnehmung ansprechen und Regeln für leichte Sprache (KMK 2) | Die pädagogischen Fach- und Lehrkräfte sind in der Lage, Aufgabenstellungen und Lerninhalte in der informatischen Bildung mit Bezug auf die Zielgruppe kindgemäß und auf unterschiedliche Weise vereinfacht, aber korrekt darzustellen sowie abstrakte informatische Konzepte (z. B. Algorithmus, Datum, Information, Variable ...) durch verschiedene Beispiele und mittels spielerischer und explorativer Erkundungsanlässe für die Kinder erfahrbar zu machen (HdkF 10) | *Design inclusive learning experiences (Use Universal Design for Learning (UDL), Culturally Relevant Pedagogy (CRP), and other techniques to support all students in successfully accessing and engaging with content; CSTA 4c)*<br><br>*Use inquiry to facilitate student learning (Use inquiry-based learning to enhance student understanding of CS content; CSTA 5a)* | **Die Lehrkräfte sind in der Lage, Aufgabenstellungen und Lerninhalte des Programmierens kindgemäß und auf unterschiedliche Weise vereinfacht, aber korrekt darzustellen sowie abstrakte informatische Konzepte durch verschiedene Beispiele und mittels spielerischer und explorativer Erkundungsanlässe für die SuS erfahrbar zu machen** |
| | Die pädagogischen Fach- und Lehrkräfte sind in der Lage, Konzepte für das Erlernen einer einfachen Programmiersprache in informatischen Lernprozessen mit einem spielerischen, auch *Unplugged*-Ansatz, praktisch umzusetzen (HdkF 11) | | **Die Lehrkräfte sind in der Lage, Konzepte für das Erlernen einer einfachen Programmiersprache mit einem spielerischen, auch *Unplugged*-Ansatz, praktisch umzusetzen** |
| Sie verfügen über ausreichende praktische Kompetenz für den Einsatz von schulrelevanter Hard- und Software, sie können insbesondere die Möglichkeiten, die sich durch den Einsatz von assistiven Technologien im Informatikunterricht eröffnen, einschätzen und bewerten (KMK 3) | Die pädagogischen Fach- und Lehrkräfte sind in der Lage, kindgemäße Software wie z. B. Programmiersprachen oder Modellierungswerkzeuge und einfache altersgemäße Informatiksysteme […] für die informatische Bildung im Elementar- und Primarbereich auszuwählen und sinnvoll in Lernszenarien zu integrieren (HdkF 12 | | **Die Lehrkräfte sind in der Lage, kindgemäße Software und einfache altersgemäße Informatiksysteme für das Programmieren im Primarbereich auszuwählen und sinnvoll in Lernszenarien zu integrieren** |

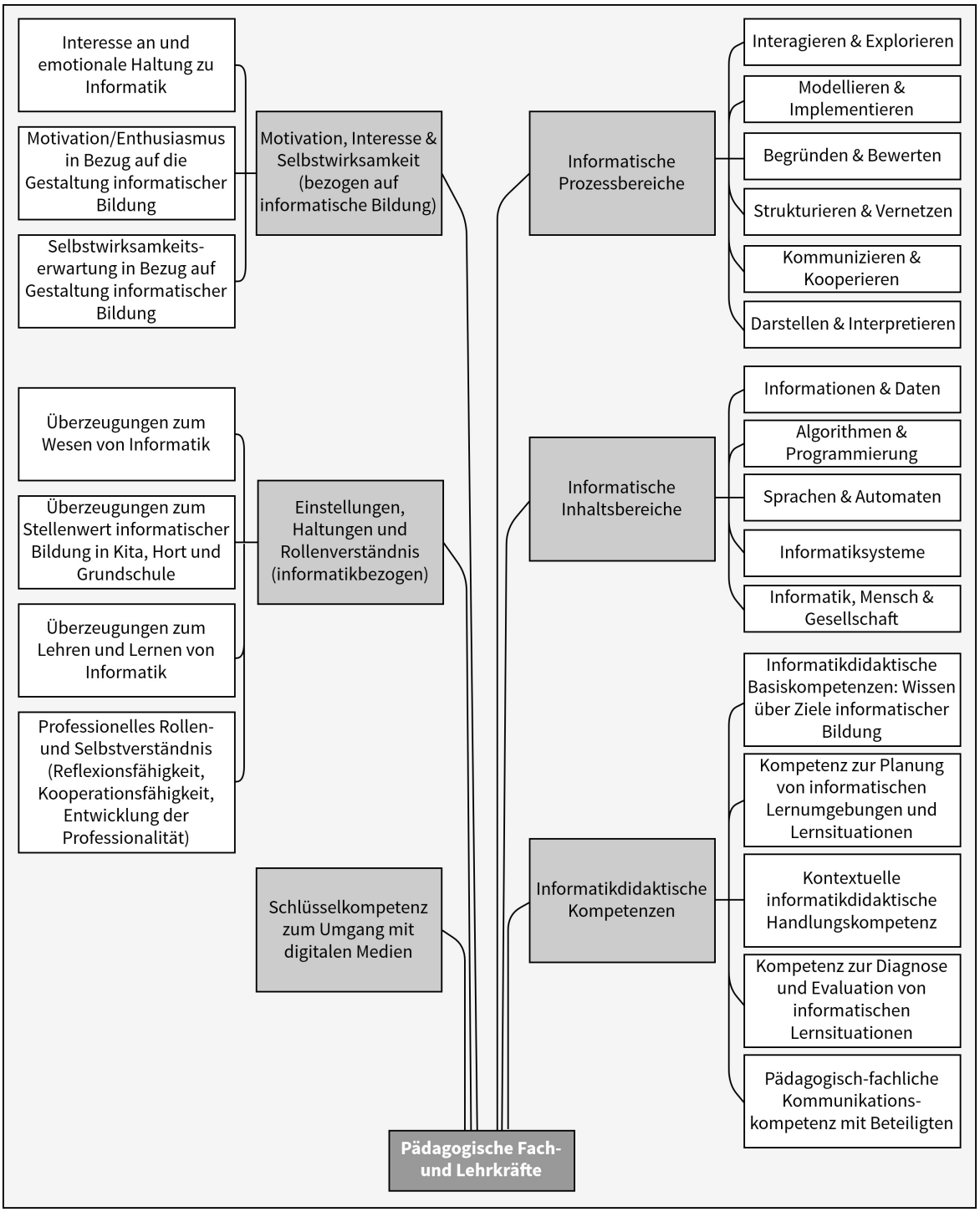



*Abbildung 15.1 Zieldimensionen früher informatischer Bildung auf Ebene der pädagogischen Fach- und Lehrkräfte aus Bergner u. a. (2018, S. 168)*

Zur weiteren Legitimation werden die informatikdidaktischen Kompetenzen in Tabelle 15.2 mit den *Aspects of Teaching and Learning* aus dem KUI-Kompetenzmodell (Hubwieser u. a. 2013b) und den Zielen für einen einführenden Kurs über die *Methods of Teaching Computer Science* von Hazzan, Lapidot und Ragonis (2011) abgeglichen (siehe Abschnitt 4.2).

*Tabelle 15.2 Abgleich der informatikdidaktischen Kompetenzen mit dem KUI-Kompetenzmodell (Hubwieser u. a. 2013b) und den Zielen für einen einführenden Kurs über die Methods of Teaching Computer Science von Hazzan, Lapidot und Ragonis (2011)*

| **Informatikdidaktische Kompetenzen** | *Aspects of Teaching and Learning (ATL)* aus dem KUI-Kompetenzmodell (Hubwieser u. a. 2013b) | Ziele für einen einführenden Kurs über die *Methods of Teaching Computer Science* (Hazzan, Lapidot und Ragonis 2011) |
|---|---|---|
| **Die Lehrkräfte sind in der Lage, Konzepte für das Erlernen einer einfachen Programmiersprache mit einem spielerischen, auch *Unplugged*-Ansatz, praktisch umzusetzen** | *Teaching Methods (ATL 7)*<br><br>*Subject-Specific Teaching Concepts (ATL 8)* | *To expose the students to a variety of computer science teaching methods* (Ziel 1)<br><br>*To enable students to master pedagogical tools for teaching computer science, including the creation of a supportive and cooperative inquiry-based learning environment* (Ziel 4) |
| **Die Lehrkräfte sind in der Lage, SuS für das Programmierenlernen zu motivieren sowie motivierende Aktivierungsmethoden und informatiktypische Sozialformen kompetenzfördernd einzusetzen** | *Teaching Methods (ATL 7)* | |
| **Die Lehrkräfte sind in der Lage, kindgemäße Software und einfache altersgemäße Informatiksysteme für das Programmieren im Primarbereich auszuwählen und sinnvoll in Lernszenarien zu integrieren** | *Media and Educational Material (ATL 10)* | |
| **Die Lehrkräfte sind in der Lage, Aufgabenstellungen und Lerninhalte des Programmierens kindgemäß und auf unterschiedliche Weise vereinfacht, aber korrekt darzustellen sowie abstrakte informatische Konzepte durch verschiedene Beispiele und mittels spielerischer und explorativer Erkundungsanlässe für die SuS erfahrbar zu machen** | *Learning Content (ATL 1)*<br><br>*Specific Teaching Elements (ATL 9)*<br><br>*Heterogeneity in the Context of Subject-Specific Learning (ATL 11)* | |
| **Die Lehrkräfte sind in der Lage, typische Präkonzepte, Verstehens- und Lernbarrieren der SuS in Bezug auf das Programmieren zu identifizieren und entsprechende Formalisierungen, Abstraktionen und Interventionsmöglichkeiten einzusetzen** | *Student Cognition (ATL 12)* | *To expose students to difficulties encountered by learners when learning different topics from the computer science curriculum* (Ziel 3) |
| **Die Lehrkräfte können auch in Gruppenarbeitssituationen mit Informatiksystemen auf Lernschwierigkeiten einzelner Kinder eingehen und gezielte individuelle Hilfestellungen geben** | *Student Cognition (ATL 12)* | *To enable students to master pedagogical skills for teaching computer science, considering different kinds of learners* (Ziel 5) |
| **Die Lehrkräfte sind in der Lage, individuelle Leistungsstände der SuS zu identifizieren und darauf durch situationsangemessenes Handeln zu reagieren** | *Student Cognition (ATL 12)* | |
| **Die Lehrkräfte regen die SuS zum selbstständigen Lernen und zur eigenständigen Begutachtung ihrer Lernergebnisse an und fördern so deren Selbstwirksamkeit** | *Teaching Methods (ATL 7)* | |
| **Die Lehrkräfte fördern durch Gruppengespräche die Reflexion der SuS über ihr eigenes Handeln und ermöglichen so positives Feedback über die erzielten Ergebnisse** | *Teaching Methods (ATL 7)* | |

### 15.1.3 Motivation, Interesse und Selbstwirksamkeit

In den Ausführungen zur Bedeutung der Lehrkraft für den Unterricht werden nicht-kognitive Merkmale bereits als Teil ihrer professionellen Handlungskompetenz genannt (siehe Abschnitt 4.1). Professionelle Überzeugungen und motivationale Orientierungen als nicht-kognitive Merkmale werden außerdem als Teil der Kompetenz zum Unterrichten des Fachs Informatik thematisiert (siehe Abschnitt 4.2). Die Relevanz des Selbstwirksamkeitsempfindens für die Vermittlung informatischer Inhalte im Kontext der Grundschule wird zudem in Kapitel 4.3 herausgestellt.

Da sich insbesondere die motivationalen Orientierungen direkt auf die alltäglichen Handlungen von Lehrkräften auswirken, werden sie in die Zielvorstellungen des Fortbildungskonzepts aufgenommen. Die fachbezogenen motivationalen Orientierungen teilen sich auf in fachbezogenen Enthusiasmus und fachbezogene Selbstwirksamkeitserwartung (siehe Abschnitt 4.2). Laut Bender u. a. (2015a, S. 10 f.) umfassen diese die folgenden Facetten:

- *Enthusiasm and interest with regard to the subject;*
- *Enthusiasm and interest with regard to teaching in the subject;*
- *Efficacy referring to the specific tasks in computer science;*
- *Efficacy referring to the profession as computer science teacher;*

Diese Facetten finden sich in abgewandelter Form auch in den Ausführungen von Bergner u. a. (2018, S. 168 ff.) wieder (siehe Abbildung 15.1). Sie werden als Ziele für diese Arbeit auf die Programmierung überführt:

- Motivation/Enthusiasmus in Bezug auf die Gestaltung und Durchführung von Lehr-Lernszenarien zur Programmierung;
- Interesse an und emotionale Haltung zur Programmierung;
- Selbstwirksamkeitserwartung in Bezug auf die Gestaltung und Durchführung von Lehr-Lernszenarien zur Programmierung;

## 15.2 *Design Principles* (Lehrerfortbildung)

Wie bereits im vorigen Kapitel wird bei der Formulierung der vorläufigen *Design Principles* auf die theoretischen Grundlagen zurückgegriffen. Die Strukturierung wird jedoch abgewandelt – anstatt der Zweiteilung in *Design Principles* mit Bezug zu Lernaktivitäten und Lehrhandlungen, werden *Design Principles* mit Bezug zu Rahmenbedingungen und Lernaktivitäten beschrieben. Die neue Dichotomie ergab sich aus den relevanten Inhalten der theoretischen Grundlagen.

### 15.2.1 *Design Principles* mit Bezug zu Rahmenbedingungen

Für die Rahmenbedingungen wird auf die Konkretisierung in Form von Musterbeispielen verzichtet, da diese organisatorischer Natur sind und nicht mit Fortbildungsmaterialien oder bestimmten Aktivitäten verbunden sind. In der Beschreibung der Umsetzungsprinzipien werden mögliche Umsetzungen der *Design Principles* mit Beispielen angereichert.

| |
|---|
| *LFB-DP 1 Die Fortbildung ermöglicht eine vertiefte Auseinandersetzung mit den Inhalten.* |

*Theoretische Fundierung*

In der Forschung zum beruflichen Lernen und zur Wirksamkeit von Fortbildungen für Lehrpersonen wird das Bereitstellen von expliziten Lernzeitfenstern als grundlegend beschrieben (siehe Abschnitt 13.1.1). Diese sollten sich zudem über einen längeren Zeitraum erstrecken, um aktives Lernen und das Erproben neuer Handlungsmuster zu ermöglichen. Empfohlen wird außerdem, Fortbildungsphasen mit Erprobungen im Unterricht zu verschränken (ebd.). Verschiedene Befunde belegen zudem, dass es von Vorteil für eine nachhaltige Auseinandersetzung mit den Inhalten ist, wenn mehrere Lehrkräfte einer Schule an einer Fortbildung teilnehmen (siehe Abschnitt 13.1.2).

*Umsetzungsprinzipien*

Die Lehrerfortbildung umfasst verschiedene Termine, die sich jeweils über mehrere Tage erstrecken. Um sich ganz auf die Fortbildungsinhalte konzentrieren zu können, wäre es ideal, wenn die Fortbildung an einem Ort stattfindet, der nicht die Schule der Teilnehmenden ist – damit diese währenddessen keine anderen Aufgaben übernehmen müssen und dadurch abgelenkt werden. Die Fortbildungstermine liegen zeitlich weit genug auseinander, um eine Erprobung der Inhalte im Unterricht zu ermöglichen. Mehrere Lehrkräfte einer Schule – mindestens zwei – nehmen an der Fortbildung teil.

*LFB-DP 2 Die Lehrkräfte erhalten eine fortlaufende Unterstützung.*

*Theoretische Fundierung*

Forschungsbefunde im Rahmen der Lehrerfortbildung zeigen, dass kontinuierliche Unterstützungsmaßnahmen sowie Coaching-Angebote die Implementierung der Inhalte im Unterricht fördern und es auch wahrscheinlicher machen, dass diese angemessen umgesetzt werden (siehe Abschnitt 13.1.3).

*Umsetzungsprinzipien*

Innerhalb der Fortbildung wird Zeit für das Besprechen möglicher Anliegen der Lehrkräfte eingeplant. Zudem erhalten sie auch über die Fortbildungsveranstaltungen hinaus die Möglichkeit, sich mit Fragen oder möglichen Problemen an die Fortbildenden zu wenden, z. B. per Mail oder Telefon. Da mehrere Lehrkräfte einer Schule an der Fortbildung teilnehmen, können sich diese zu den Inhalten austauschen sowie bei der Umsetzung im Unterricht unterstützen. Trauen sich einzelne Lehrkräfte die Umsetzung an der eigenen Schule nicht zu, können sie Unterstützung in Form von Unterrichtsbesuchen durch die Fortbildenden anfragen.

### 15.2.2 *Design Principles* mit Bezug zu Lernaktivitäten

*LFB-DP 3 Die Gestaltung des Erkenntnisgewinns der Lehrkräfte orientiert sich an dem der Schülerinnen und Schüler.*

*Theoretische Fundierung*

Das Prinzip stützt sich einerseits auf verschiedene Forschungsergebnisse im Rahmen der Lehrerbildung (siehe Abschnitte 13.2.1 und 13.2.2): Zum einen hat sich gezeigt, dass es sich positiv auf die spätere Umsetzung im Unterricht auswirkt, wenn Lehrkräfte selbst die Veränderung der ursprünglichen Vorstellungen der Lernenden erleben. Zum anderen erleben sie Fortbildungen als zufriedenstellend, wenn sie sich auf den konkreten Unterrichtsalltag beziehen. Andererseits wird es gestützt durch den Ansatz des situierten Lernens, demzufolge Lernsituationen so authentisch und anwendungsorientiert wie möglich sein sollten (siehe Abschnitt 13.3.2).

*Umsetzungsprinzipien*

Die Lehrkräfte erhalten im Rahmen der Fortbildung die Möglichkeit, dem Lernpfad der Schülerinnen und Schüler zu folgen und so selbst *Aha-Erlebnisse*, aber auch das Auftauchen möglicher Fragestellungen und Probleme zu erfahren. Zusätzlich werden den Lehrkräften beispielhafte Arbeitsergebnisse von Schülerinnen und Schülern zur Verfügung gestellt.

*Musterbeispiele*

- Beispiel LFB-DP3-1: Unterrichtsmaterialien erproben
  Die Lehrkräfte bearbeiten eine Auswahl der Unterrichtsmaterialien, die in der Unterrichtssequenz verwendet werden.
- Beispiel LFB-DP3-2: Programmierpraxis sammeln
  Die Lehrkräfte bekommen innerhalb der Fortbildung ausgiebig die Möglichkeit, Programmiererfahrung in Scratch zu sammeln. Die Fortbildenden geben Hilfestellung, wie sie dies auch innerhalb eines Unterrichtsgeschehens tun würden.
- Beispiel LFB-DP3-3: Realistische Eindrücke
  Wo es sich anbietet, werden kurze Videosequenzen aus dem Unterrichtsgeschehen gezeigt, die illustrieren, wie sich die Schülerinnen und Schüler während einer Übung verhalten bzw. wie die Lehrkraft die Übung durchführt. Hierfür eignet sich z. B. die Übung, in der die Schülerinnen und Schüler den *Lehrer-Roboter* programmieren. Zusätzlich werden im Laufe der Fortbildung Arbeitsergebnisse von Schülerinnen und Schülern gezeigt – aus den *Unplugged-* sowie den Scratch-Phasen. So werden bspw. Ergebnisse offener Programmieraufgaben präsentiert, um den Lehrkräften eine Vorstellung davon zu vermitteln, was Schülerinnen und Schüler leisten können.

*LFB-DP 4 Die Fortbildung verschränkt Fachwissenschaft und Fachdidaktik.*

*Theoretische Fundierung*

Um Programmieren zu unterrichten brauchen Lehrkräfte zum einen fundierte Fachkenntnisse, zum anderen Kenntnisse in der Didaktik der Informatik. Um die Vermittlung von fachlichen und fachdidaktischen Kenntnissen in Fortbildungen zu verknüpfen, wird auf das situierte Lernen hingewiesen (siehe Abschnitt 13.3.2). Hinsichtlich der fachlichen Ebene wird in der fachdidaktischen Literatur herausgestellt, dass nicht die bloße Vermittlung einer Programmiersprache im Vordergrund stehen sollte, sondern die Grundlagen und Kernkonzepte der Programmierung, z. B. die algorithmischen Grundstrukturen (siehe Abschnitte 12.3.1 und 13.3.1). Expertise im Programmieren stelle sich zudem nur durch Übung ein (siehe Abschnitt 12.3.2) – generell wird die aktive Auseinandersetzung mit dem Lerngegenstand in vielen Publikation zur Lehrerbildung als Voraussetzung für erfolgreiche Fortbildungsmaßnahmen gesehen (siehe Abschnitt 13.2.1).

*Umsetzungsprinzipien*

Um eine gewisse Sicherheit im Programmieren auszubilden, erhalten die Lehrkräfte mehrfach die Möglichkeit, selbst in Scratch zu programmieren. Dabei werden einerseits die algorithmischen Grundstrukturen thematisiert, andererseits werden daran fachdidaktische Prinzipien aufgezeigt.

*Musterbeispiele*

- Beispiel LFB-DP4-1: Fachinhalte
  Die algorithmischen Grundstrukturen – Anweisung, Sequenz, Wiederholung und bedingte Anweisung – werden zunächst jeweils in Pseudocode sowie Blöcken der Programmiersprache Scratch beschrieben und danach in Programmieraufgaben unterschiedlicher Schwierigkeit vertieft. Zusätzlich zu den algorithmischen Grundstrukturen wird das Variablenkonzept behandelt. Der theoretische Input wird dabei möglichst kurz gehalten – der Schwerpunkt liegt auf dem eigenständigen Bearbeiten der Aufgaben. Diese können zum Großteil auch im Unterricht eingesetzt werden. Der Algorithmusbegriff wird anhand eines Beispiels aus dem Alltag der Lehrkräfte thematisiert (siehe LFB-DP 6).

- Beispiel LFB-DP4-2: Fachdidaktik
  Die Übungsphasen während der Fortbildung sind im Sinne eines *Didaktischen Doppeldeckers* gestaltet: die Lehrkräfte erfahren Unterrichtspraktiken und -methoden als Lernende und reflektieren anschließend aus der Sicht der Lehrenden. Auf diese Art und Weise werden z. B. unterschiedliche Formate für Programmieraufgaben oder mögliche Hilfestellungen thematisiert. Die Lehrkräfte bearbeiten bspw. Programmieraufgaben in Scratch, die in Anlehnung an die *Use-Modify-Create Learning Progression* von Lee u. a. (2011) gestaltet wurden – diese wird im Anschluss vorgestellt. Um die Theorie mit der Praxis zu verknüpfen, ordnen die Lehrkräfte die Aufgaben, die sie bereits bearbeitet haben, in die Kategorien *use*, *modify* und *create* ein (siehe Abbildung 15.2). Hier arbeiteten sie zu zweit mit einem Satz Kärtchen, auf denen die Programmieraufgaben in Miniatur abgebildet sind und die sie entsprechend gruppieren und ordnen können.

- Beispiel LFB-DP4-3: Zugänge zum Programmieren
  An verschiedenen Stationen sammeln die Lehrkräfte Programmierpraxis und lernen gleichzeitig weitere Systeme zum Programmieren im Unterricht kennen: den Einplatinencomputer *Calliope mini*, den *Makey Makey*, den mit Pfeiltasten programmierbaren Bienenroboter *Bluebot*, die App *ScratchJr* sowie die programmierbaren Bausätze *LEGO WeDo* und *LEGO EV3*. An jeder Station erhalten die Lehrkräfte grundlegende Informationen zu dem jeweiligen Produkt und bearbeiten eine einführende Aufgabe. Anschließend werden im Plenum die Vor- und Nachteile der einzelnen Produkte sowie mögliche Anknüpfungspunkte zur vorgestellten Unterrichtssequenz gesammelt und diskutiert.

- Beispiel LFB-DP4-4: Fehlvorstellungen vermeiden
  Um Fehlvorstellungen zum Ablauf einzelner algorithmischer Strukturen früh abzufangen, werden Codebeispiele in Scratch gemeinsam schrittweise gelesen und nachvollzogen. Hierbei wird bspw. thematisiert, was passiert, wenn das Programm unter verschiedenen Voraussetzungen, z. B. Positionen einer Figur auf der Bühne, gestartet wird (siehe Abbildung 15.3).

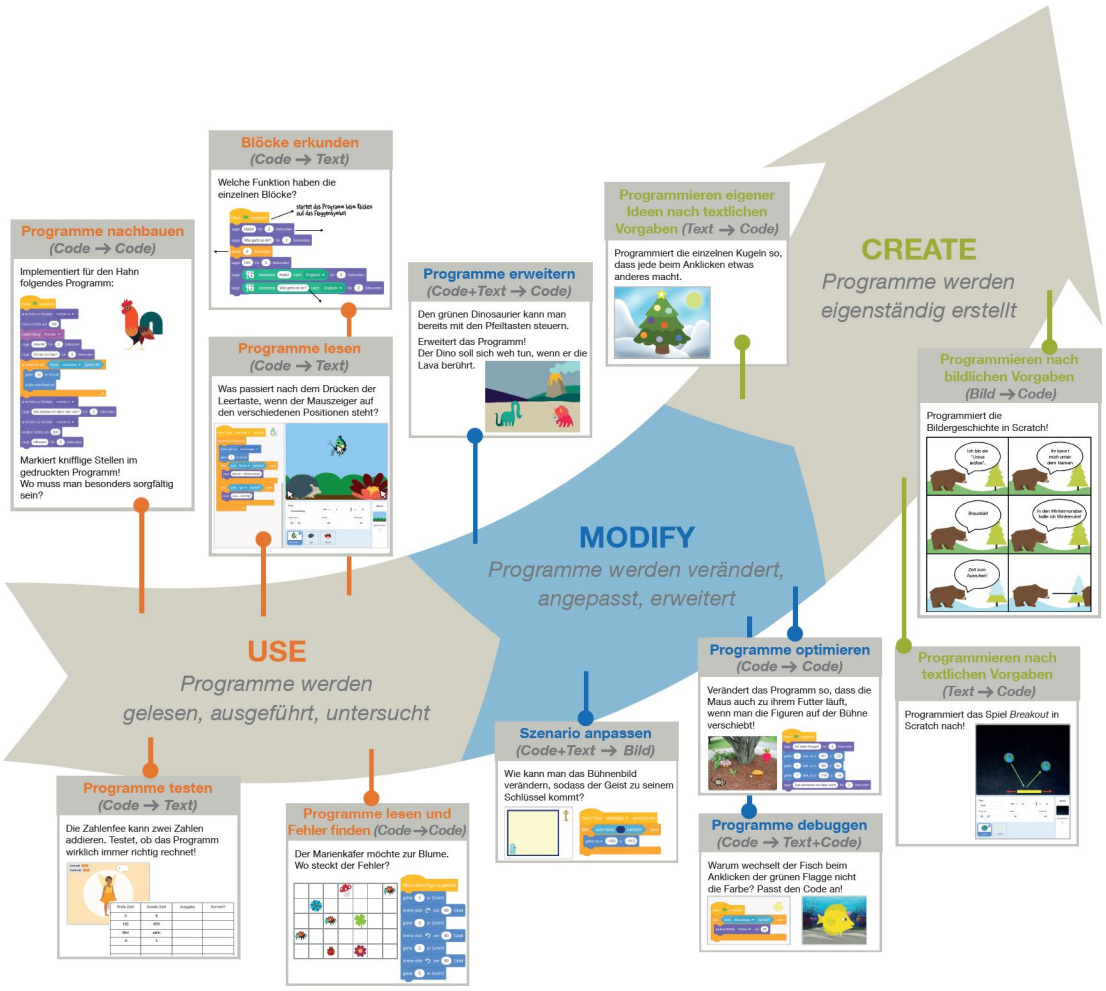



*Abbildung 15.2 Aufgabentypen in Scratch entlang des Use-Modify-Create-Ansatzes gruppiert.*


*Abbildung 15.3 Aufgabe „Was passiert in Fall 1 und 2 beim Starten des Programms?“*

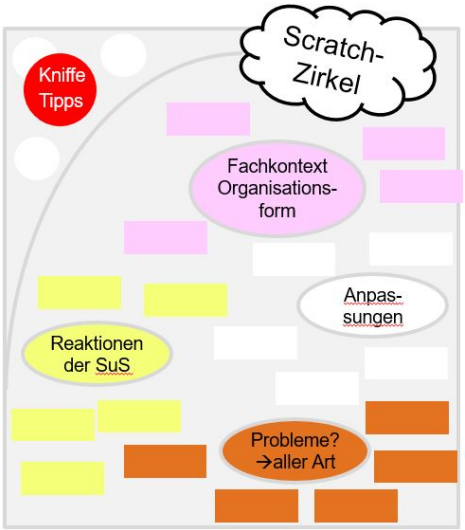



*Abbildung 15.4 Schema einer Plakatwand zum Erfahrungsaustausch*

*LFB-DP 5 In der Fortbildung werden Reflexionsprozesse über die Fortbildungsinhalte sowie die Umsetzung der Inhalte im Unterricht angeregt.*

*Theoretische Fundierung*

Die Möglichkeit, sich mit anderen Teilnehmenden über die behandelten Inhalte auszutauschen, wird in der Literatur zur Fortbildung von Lehrkräften als besonders wertvoll beschrieben (siehe Abschnitt 13.2.3). Darüber hinaus wird ein Austausch zwischen den Teilnehmenden über ihre unterrichtliche Umsetzungen der Inhalte als gewinnbringend beschrieben (ebd.). Im Rahmen des Unterrichtsansatzes des situierten Lernens ist es außerdem üblich, dass sich Lehrkräfte mit fachlichen Inhalten auseinandersetzen, bevor in einem nachgeschalteten Theorie-Input die didaktischen Hintergründe thematisiert werden (siehe Abschnitt 13.3.2).

*Umsetzungsprinzipien*

Innerhalb der Fortbildungsveranstaltungen werden Reflexionsphasen fest eingeplant. Zudem erhalten die Lehrkräfte die Möglichkeit, sich über die Erprobungen der Inhalte in ihrem Unterricht auszutauschen – dafür muss ein bestimmter zeitlicher Abstand zwischen den einzelnen Fortbildungsveranstaltungen liegen.

*Musterbeispiele*

- Beispiel LFB-DP5-1: Perspektive der Schülerinnen und Schüler
  Die Lehrkräfte bekommen die Möglichkeit, die Perspektive der Schülerinnen und Schüler einzunehmen, indem sie ausgewählte Unterrichtsmaterialien selbst bearbeiten (siehe Bsp. LFB-DP3-1). Im Anschluss wird im Plenum thematisiert, welche Schwierigkeiten dabei auftraten und welche Erkenntnisse gewonnen wurden.
- Beispiel LFB-DP5-2: Erprobung einfordern
  Im Rahmen der ersten Fortbildungsveranstaltung wird mit den Lehrkräften vereinbart, dass sie die vorgestellten Unterrichtsmaterialien bis zur nächsten Veranstaltung im Unterricht ausprobieren werden. In welcher Form sie dies tun, bleibt ihnen überlassen, z. B. können sie auch nur einzelne Teile der Unterrichtssequenz erproben oder diese bereits modifizieren bzw. erweitern.
- Beispiel LFB-DP5-3: Reflexionsphasen initiieren
  Anhand von Plakatwänden wird thematisiert, wie sich die Lehrkräfte die Umsetzung der Themen Algorithmen und Programmieren in ihrem eigenen Unterricht vorstellen können. Diese werden im Raum verteilt und die Lehrkräfte können darauf Ideen, Inspirationen und Gedanken zu verschiedenen Themen sammeln, z. B. zu möglichen kurz-, mittel- und längerfristigen Umsetzungen, Herausforderungen oder dem Bild des Programmierens und wie sich dieses bereits verändert hat.
- Beispiel LFB-DP5-4: Erfahrungsaustausch
  In der Fortbildungsveranstaltung, die sich an die eingeforderte Erprobungsphase anschließt, wird ein Erfahrungsaustausch zwischen den Lehrkräften eingeplant. Auf verschiedenen Plakatwänden

zu den einzelnen Elementen der Unterrichtssequenz sammeln die Lehrkräfte stichpunktartig, in welchem fachlichen Kontext sie etwas an ihrer Schule umsetzen konnten, welche Anpassungen sie vornahmen, wie die Reaktionen der Kinder ausfielen und welche Probleme auftraten (siehe Abbildung 15.4). Die Ergebnisse sowie *Best-Practice*-Beispiele aus den Schulen werden im Anschluss im Plenum besprochen.

- Beispiel LFB-DP5-5: Gute Programmierpraktiken
  Den Lehrkräften werden zwei Lösungen derselben Aufgabe zur Verfügung gestellt, bei denen sie Unterschiede in der Umsetzung finden sollen. Anhand der Übung werden verschiedene positive Programmierpraktiken herausgearbeitet, wie z. B. sinnvolles Benennen von Variablen, Kommentieren von Code, Entfernen unnötiger Blöcke.

*LFB-DP 6 Die Fortbildung knüpft an Bekanntes an.*

*Theoretische Fundierung*

In der informatikdidaktischen Forschung wird empfohlen, den Lehrkräften im Rahmen von Fortbildungsmaßnahmen Zeit einzuräumen, um die neuen Inhalte mit dem bestehenden Lehrplan zu verknüpfen oder konkrete Möglichkeiten zur Integration in andere Fächer aufzuzeigen (siehe Abschnitt 13.3.3). Anknüpfungspunkte sollten zudem auch auf didaktischer Ebene erarbeitet oder aufgezeigt werden (ebd.). Der Einsatz von *Unplugged*-Aktivitäten und Beispielen aus dem Alltag werden empfohlen, um vor allem Lehrkräfte ohne Vorerfahrungen im Programmieren an das Thema heranzuführen (siehe Abschnitt 12.1.1).

*Umsetzungsprinzipien*

In der Fortbildung werden Möglichkeiten zur Anknüpfung an andere Fächer oder auf didaktischer Ebene zur Grundschuldidaktik bzw. fachdidaktischen Prinzipien bestehender Grundschulfächer aufgezeigt. Im Rahmen der Fortbildung gibt es immer wieder Phasen, in denen *unplugged* gearbeitet wird.

*Musterbeispiele*

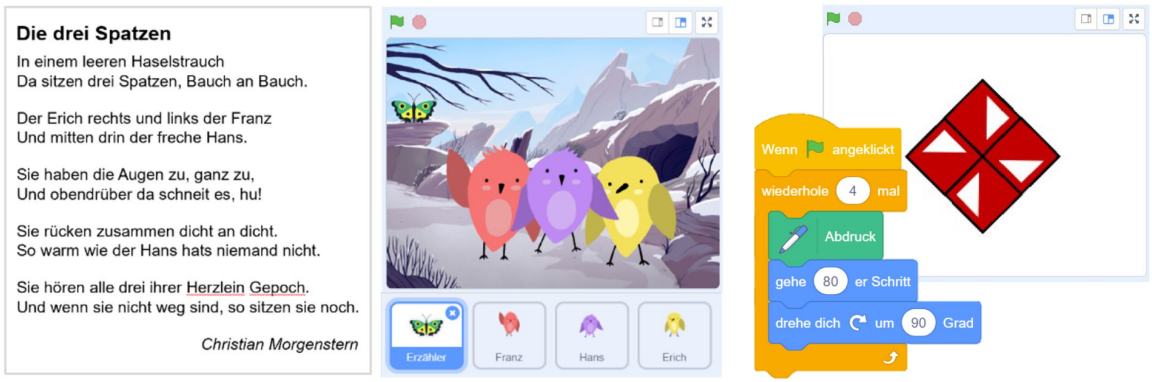
- Beispiel LFB-DP6-1: Bestehende Fächer
  Die Lehrkräfte bearbeiten verschiedene Programmieraufgaben, die im Kontext bestehender Grundschulfächer eingesetzt werden können. Zum Beispiel animieren sie ein kurzes Gedicht in Scratch oder programmieren geometrische Figuren[2] (siehe Abbildung 15.5).

- Beispiel LFB-DP6-2: Bilddiktat
  Als Einstieg in die Thematik *Algorithmus* wird eine Variante der *Stillen Post* gespielt. Jede Lehrkraft malt oben auf ein leeres Blatt Papier ein einfaches Bild und gibt es nach rechts an die Nachbarin bzw. den Nachbar weiter. Dieser beschreibt unter dem Bild, was er darauf sieht und knickt das gemalte Bild nach hinten um. Danach wird das Blatt erneut nach rechts weitergegeben, sodass

[2] Das Beispiel ist einer Aufgabe aus *ScratchMaths* entnommen – einem Projekt des University College London (https://www.ucl.ac.uk/ioe/research/projects/ucl-scratchmaths).

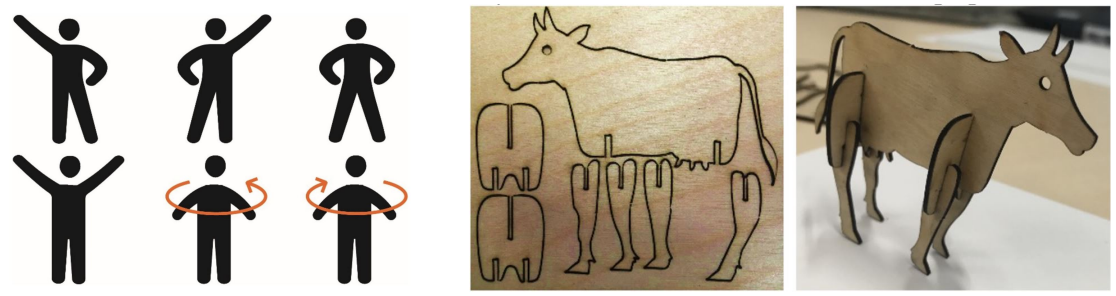
nun jeder die Beschreibung eines Bildes vor sich sieht. Zu dieser Beschreibung wird nun wieder ein Bild gemalt. Im Anschluss falten die Lehrkräfte die Blätter auf und vergleichen, wie ähnlich sich die beiden Bilder sehen oder worin sie sich unterscheiden. Falls die Bilder sehr unterschiedlich aussehen, können sie untersuchen, an was es liegt – wurde nicht genau beschrieben oder nicht genau gelesen? Anhand der Übung wird der Unterschied von Alltagsalgorithmen zu einem Algorithmus im eigentlichen Sinn herausgestellt. Eine Definition des Begriffs wird präsentiert und die einzelnen Elemente besprochen.

- Beispiel LFB-DP6-3: *Computational Thinking*
  Innerhalb der Fortbildung werden einzelne Aspekte des *Computational Thinking* thematisiert. In Kleingruppen bearbeiteten die Lehrkräfte verschiedene Stationen und präsentierten diese anschließend im Plenum. Dabei wird auf den Einsatz einer Programmiersprache verzichtet: die Lehrkräfte stellen z. B. einen Tanz-Algorithmus aus Symbolkarten mit unterschiedlichen Tanzposen auf[3] oder erstellen einen Algorithmus für das Zusammenbauen einer Holzfigur mithilfe einer Digitalkamera (siehe Abbildung 15.6).




*(a)* *(b)*

*Abbildung 15.5 Programmieranlässe in den bestehenden Fächern (a) Deutsch und (b) Mathe*



*(a)* *(b)*

*Abbildung 15.6 Unplugged-Aufgaben: (a) Tanz-Algorithmus mit Symbolen (b) Algorithmus für den Bau einer Holzfigur*

[3] Das Beispiel ist einer Aufgabe aus den „Barefoot“-Unterrichtsmaterialien entnommen – einem Angebot von „Computing at School“ (https://www.barefootcomputing.org/).

## 15.3 Didaktische Konzeption der Fortbildung

### 15.3.1 Planungsmethodik (Lehrerfortbildung)

Auf Basis der Überlegungen in Kapitel 15.1 und 15.2 wurde zunächst ein Grobentwurf entwickelt. Dabei wurde der Ansatz verfolgt, der bereits bei der Entwicklung der Unterrichtssequenz angewandt wurde (siehe Abschnitt 15.3.1).

Im Rahmen der didaktischen Konzeption der Fortbildung wurden die erarbeiteten Umsetzungsprinzipien und Musterbeispiele sowie der geplante Ablauf regelmäßig in Feedbackgesprächen mit Experten der Informatikdidaktik und Erwachsenenbildung sowie mit Experten für das Unterrichten an Grundschulen besprochen. Diskutiert wurde dabei sehr ausführlich, welche Programmiersprache im Rahmen der Fortbildung verwendet werden sollte. Während die Autorin beabsichtigte, durchgehend mit der visuellen Programmiersprache Scratch zu arbeiten, sprachen sich andere für den Einsatz der textbasierten Programmiersprache *Python* aus: Diese sei sehr übersichtlich und daher gut geeignet für die Auseinandersetzung mit den algorithmischen Grundstrukturen. Außerdem könne man auf eine große Sammlung von Materialien und Aufgaben aus der Fortbildung von Lehrkräften an Gymnasien zurückgreifen. Um die Lehrkräfte nicht mit dem Erlernen zweier Programmiersprachen zu überfordern und abzuschrecken, entschied man sich für den Einsatz von Scratch – die Programmiersprache, mit der auch die Schülerinnen und Schüler programmieren. Darüber hinaus war es auf diese Weise möglich, den Lehrkräften auch während der Erarbeitung der fachlichen Inhalte Anregungen für ihren Unterricht zu geben.

Ebenso ausführlich diskutiert wurde, ob bzw. welche Vorgaben den Lehrkräften für die erste Praxisphase gemacht werden sollten. Hier gingen die Meinungen stark auseinander: Während einige dafür plädierten, dass die Lehrkräfte die Unterrichtssequenz unverändert ausprobieren sollten, vertraten andere die Meinung, sie sollten gänzlich neue Lehr-Lernaktivitäten erarbeiten und erproben. Als Kompromiss entschied man sich, es den Lehrkräften zu überlassen, die Unterrichtssequenz oder Teile davon unverändert oder abgewandelt zu erproben und die Entwicklung eigener Lehr-Lernaktivitäten für die zweite Praxisphase einzuplanen. In diesem Zusammenhang wurde auch diskutiert, ob den Lehrkräften die Lehr-Lernmaterialien aus der Unterrichtssequenz zur Verfügung gestellt werden sollten. Dies wurde bejaht und es wurde weiter beschlossen, jeder Lehrkraft einen Satz haptischer Scratch-Blöcke bereitzustellen. Damit sollte sichergestellt werden oder zumindest die Wahrscheinlichkeit erhöht werden, dass die Unterrichtssequenz erprobt wird.

In Frage gestellt wurde, ob es sinnvoll sei, die Lehrkräfte mit zusätzlicher Software und Informatiksystemen für das Programmieren in der Grundschule zu konfrontieren, da man sie doch in erster Linie befähigen wolle, die Unterrichtssequenz umzusetzen. Die Autorin argumentierte an dieser Stelle, dass die Lehrkräfte in der Werbung und den Medien ohnehin mit Alternativen zur Programmiersprache Scratch konfrontiert würden und man ihnen deshalb die professionelle Beurteilung ebendieser ermöglichen wolle. Darüber hinaus könne man sich gut vorstellen, die Unterrichtssequenz mit weiterer Software oder Informatiksystemen sinnvoll zu ergänzen.

Die Autorin war sich zunächst unsicher, wann das Konzept *Computational Thinking* eingeführt werden sollte. Einerseits ist es für die Lehrkräfte sehr relevant, andererseits sollte in der Begleitforschung untersucht werden, ob die Lehrkräfte durch das Programmieren Veränderungen hinsichtlich einzelner Aspekte des Konzepts bei ihren Schülerinnen und Schülern feststellen können. Um beiden Argumenten gerecht zu werden, wurde entschieden, das Konzept am Ende der Lehrerfortbildung vorzustellen, da zu diesem Zeitpunkt die betreffende Datenerhebung abgeschlossen ist und sich das Konzept gut eignet, um Sinn und Bedeutung des Programmierens in der Grundschule zu diskutieren.

Zusätzlich führte die Autorin während der didaktischen Konzeption der Fortbildung einen Workshop für Lehramtsanwärterinnen und Lehramtsanwärter des Lehramts an Grundschulen am Schülerforschungszentrum Berchtesgadener Land und einen Workshop auf einer nationalen Konferenz zur informatischen Bildung[4] durch, in denen sie die entwickelte Unterrichtssequenz präsentierte. Nachdem die Teilnehmenden ausgewählte Aufgaben daraus bearbeitet hatten, wurden sie schriftlich befragt, was Grundschullehrkräfte ohne informatische Vorbildung ihrer Meinung nach können bzw. wissen müssten, um die Sequenz kompetent unterrichten zu können. Die Ergebnisse wurden danach mit dem Fortbildungskonzept abgeglichen und sind im Folgenden thematisch gruppiert aufgelistet. Die Punkte in grauer Schrift sind im Fortbildungskonzept nicht berücksichtigt – dazu wird im Anschluss Stellung bezogen.

Rückmeldungen der Lehramtsanwärterinnen und -anwärter für das Lehramt an Grundschulen:

- Fachwissenschaftliche Inhalte
  - Grundlagen des Programmierens; etwas Fachsprache;
- Fachdidaktische Inhalte
  - Didaktische Reduktion des Programmierens;
  - Grundverständnis von der Didaktik der Mathematik, da es große Schnittmenge im Bereich logischer Schlüsse gibt;
  - gut mit Scratch umgehen können und die Funktionen gut kennen; Scratch gut kennen, um die Schülerinnen und Schüler kompetent zu unterstützen und Probleme zu lösen;
- Methodische Aspekte zur Konzeption der Fortbildung
  - Unterrichtssequenz selbst durcharbeiten, um Schritte der Schülerinnen und Schüler nachzuvollziehen;

Rückmeldungen der Informatiklehrkräfte und Informatikdidaktikerinnen und -didaktiker:

- Fachwissenschaftliche Inhalte
  - Objektorientierte Sichtweise (Methoden und Attribute);
  - Algorithmusbegriff; Vergleich verschiedener Algorithmen zu einem Problem (Vielfalt von Lösungen bewusst machen);

[4] INFOS: Fachtagung des Fachausschusses Informatische Bildung in Schulen der Gesellschaft für Informatik

  - Kontrollstrukturen;
  - Programmiererfahrung über Scratch hinaus z. B. in *Python* (rudimentär);
- Fachdidaktische Inhalte
  - Einführung in den Hintergrund des Programmierens (warum sollten die Schülerinnen und Schüler etwas dazu wissen; wo begegnet uns die Programmierung im Alltag);
  - Einblick in Grundlagen der Fachdidaktik; Bewertung des Programms statt des Produkts;
  - Abgrenzung der Informatik von Medienarbeit/Arbeiten am PC;
  - Grundwissen über Programmierung mit Scratch; Wissen, dass der Kern der Aufgaben im algorithmischen Analysieren und Konstruieren liegt und die Gestaltung der Oberfläche und der Figuren nur nebensächlich ist;
  - Sicherheit mit der Programmoberfläche von Scratch;
  - Kenntnisse von Problemen, die bei Schülerinnen und Schülern in Scratch auftreten können; Fehler in Programmen schnell erkennen können; Umgang mit Fehlern;
  - Material, Tipps und Tricks
- Methodische Aspekte zur Konzeption der Fortbildung
  - aktiv selbst mit den Konzepten bzw. Beispielen aus der Unterrichtssequenz auseinandersetzen und diese aktiv durchexerzieren, um die Vorgänge zu verstehen;
  - eigene Erfahrungen über die Unterrichtssequenz hinaus mit Scratch;

Innerhalb der Fortbildung ein Grundverständnis für die Didaktik der Mathematik anzubahnen, übersteigt deren zeitlichen Rahmen. Zudem sollten alle bayerischen Grundschullehrkräfte bereits über ein solches Grundverständnis verfügen, da Mathematik in allen möglichen Kombinationen von Unterrichts- und Didaktikfächern enthalten ist[5]. Nach Rücksprache im Rahmen eines weiteren Feedbackgesprächs wird von der Einführung der objektorientierten Sichtweise abgesehen, da dies im zeitlichen Rahmen nicht angemessen erfolgen kann bzw. die Lehrkräfte dann wohl nur verwirren würde. Warum auf Programmierübungen mit einer anderen Programmiersprache als Scratch, z. B. *Python*, verzichtet wird, wurde bereits erläutert. Die Abgrenzung der Informatik von der Arbeit mit Medien oder dem Computer wird auch in Zukunft nicht explizit thematisiert, da davon ausgegangen wird, dass dies durch die Gestaltung der Fortbildung und die vielen *Unplugged*-Übungen implizit deutlich gemacht wird.

### 15.3.2 Grobentwurf

Aus den Überlegungen in Kapitel 15.1 und 15.2 wurde ein Grobentwurf entwickelt, der den Verlauf der einzelnen Phasen und Schritte der Fortbildung zusammenfassend darstellt. Wie bereits bei der didaktischen Konzeption der Unterrichtssequenz ist eine tabellarische Auflistung der einzelnen

[5] https://www.lmu.de/de/studium/studienangebot/1x1-des-lehramtsstudiums/lehramt-grundschule/index.html

Schritte einschließlich ihrer Herleitung aus den Umsetzungsprinzipien und Verbindung zu den Zielen der Fortbildung in Anhang C.1.1 nachzulesen. Insgesamt lassen sich die folgenden Phasen und Kurzbeschreibungen benennen:

(1) Thematischer Einstieg
   a) Aktivierung von Vorwissen der Lehrkräfte
   b) Kurzvorstellung der Unterrichtssequenz[6] und deren didaktischer Relevanz

(2) Programmieren in der Grundschule
   a) Bearbeiten von ausgewählten Aufgaben der Unterrichtssequenz (LFB-DP 3, LFB-DP 6)
   b) Reflexion des eigenen Lernprozesses und Sichtung der Ergebnisse von Schülerinnen und Schülern (LFB-DP 3, LFB-DP 5)

(3) Fachliche Inhalte
   a) Vorstellung des Algorithmusbegriffs (LFB-DP 6)
   b) Einführung der algorithmischen Grundstrukturen sowie des Variablenkonzepts und Bearbeiten passender Programmieraufgaben in Scratch (LFB-DP 4)

(4) Reflexion
   a) Sammeln von Möglichkeiten und Maßnahmen zur Implementierung an den Schulen der Lehrkräfte und Thematisieren von Herausforderungen (LFB-DP 5)

---

*1. Praxisphase:*
*Erprobungen der Unterrichtssequenz oder von ausgewählten Teilen der Unterrichtssequenz*

---

(5) Erfahrungsaustausch
   a) Austausch von Erfahrungen in der Umsetzung der Unterrichtssequenz (LFB-DP 5)
   b) Präsentation von *Best-Practice*-Beispielen (LFB-DP 5)

(6) Fachdidaktische Grundlagen
   a) Kennenlernen von Aufgabenformaten in Scratch (LFB-DP 4)
   b) Kennenlernen von Modellen über den Verlauf des Programmierenlernens (LFB-DP 4)
   c) Kennenlernen weiterer kindgemäßer Software und altersgemäßer Informatiksysteme zum Programmieren (LFB-DP 4)

---

*2. Praxisphase:*
*Weitere Erprobungen der Unterrichtssequenz und ggf. Entwicklung sowie Erprobung eigener Lehr-Lernaktivitäten zum Programmieren*

---

[6] Hier und in den weiteren Ausführungen bezieht sich der Begriff auf die in Kapitel 14.3 entwickelte Unterrichtssequenz.

(7) Erfahrungsaustausch
    a) Präsentation der von den Lehrkräften entwickelten Lehr-Lernaktivitäten (LFB-DP 5)

(8) Fachdidaktische Vertiefung
    a) Kennenlernen von Programmierkontexten aus bestehenden Fächern der Grundschule (LFB-DP 4, LFB-DP 6)
    b) Austausch über gute Programmierpraktiken (LFB-DP 5)
    c) Auseinandersetzen mit Aspekten des *Computational Thinking* (LFB-DP 6)

### 15.3.3 Geplanter Verlauf der Fortbildung

Nachfolgend wird der geplante Verlauf der Fortbildung als tabellarische Ablaufplanung dargestellt (siehe Tabelle 15.3). Der Aufbau der Ablaufplanung entspricht dem der Unterrichtssequenz im vorigen Kapitel (siehe Tabelle 14.7). Die Fortbildung gliedert sich in drei Fortbildungsblöcke, die jeweils durch eine mehrwöchige Praxisphase getrennt sind. Die Vorgaben für die Praxisphasen sind in der Ablaufplanung an den entsprechenden Stellen vermerkt.

*Tabelle 15.3 Tabellarische Ablaufplanung der Fortbildung*

| Phase | Kurz-beschreibung | Lernaktivität | Lehrhandlungen | Handlungs-muster / Sozialform | Material/ Medien |
|---|---|---|---|---|---|
| Thematischer Einstieg | Aktivierung von Vorwissen der Lehrkräfte | Die Lehrkräfte ergänzen den Satz „Programmieren ist für mich…“ auf Papierstreifen. Im Anschluss lesen sie diese vor. | Die Kursleitung greift die Äußerungen der Lehrkräfte auf und präsentiert Assoziationen von SuS mit dem Begriff „Programmieren“. | Einzel-Arbeit, Unterrichts-gespräch | Papier-streifen, Schreib-material |
| | Kurzvorstellung der Unterrichts-sequenz und deren didaktischer Relevanz | | Die Kursleitung stellt den grund-legenden Aufbau der Unterrichts-sequenz vor. Sie gibt einen Einblick in die Potenziale und Chancen der Programmierung in der Grundschule. | Präsentation | Beamer, PowerPoint-Präsentation |
| Program-mieren in der Grundschule | Bearbeiten von ausgewählten Aufgaben der Unterrichtssequenz | Die Lehrkräfte sehen bzw. bearbeiten verschiedene Aufgaben der Unterrichtssequenz (→ Bsp. LFB-DP3-1, LFB-DP5-1). | Die Kursleitung zeigt die Aufgaben in einer Präsentation und erläutert die Aufgaben-stellung.<br><br>Die Kursleitung stellt die Oberfläche von Scratch vor.<br><br>Sie leistet während der ganzen Fortbildungsphase Hilfestellung, so wie sie auch SuS helfen würde. | Präsentation, Gruppen-arbeit, Partnerarbeit | Beamer, PowerPoint-Präsentation, Scratch, Unterrichts-materialien, Schreib-material |
| | Reflexion des eigenen Lern-prozesses und Sichtung der Ergebnisse von SuS | Die Lehrkräfte erläutern die Schwierigkeiten und *Aha-Erlebnisse*, die sie bei der Bearbeitung der Aufgaben hatten. | Die Kursleitung führt durch die Unterrichtssequenz und zeigt ausgewählte Ergebnisse von SuS (→ Bsp. LFB-DP3-3). | Unterrichts-gespräch | Beamer, PowerPoint-Präsentation |

| | | | | | |
|---|---|---|---|---|---|
| Fachliche Inhalte | Vorstellung des Algorithmus-begriffs | Die Lehrkräfte spielen in der Gruppe eine Variante des Spiels *Stille Post* (→ Bsp. LFB-DP6-2). | Anhand der Ergebnisse sowie verschiedener Beispiele von Anweisungen in natürlicher Sprache und Pseudocode stellt die Kursleitung den Unterschied zwischen Alltagsalgorithmen und einem Algorithmus im informatischen Sinn heraus. Sie präsentiert eine Definition des Begriffs Algorithmus und erläutert dessen einzelne Elemente. | Gruppen-arbeit, Unterrichts-gespräch | Beamer, PowerPoint-Präsentation, Schreib-material |
| | Einführung der algorithmischen Grundstrukturen sowie des Variablenkonzepts und Bearbeiten passender Programmier-aufgaben in Scratch | Die Lehrkräfte bearbeiten verschiedene Programmieraufgaben in Scratch (→ Bsp. LFB-DP3-2, LFB-DP4-1).<br><br>Sie untersuchen verschiedene Codebeispiele und sagen den Programmablauf voraus (→ Bsp. LFB-DP4-4). | Anhand der Aufgaben und Beispiele in Pseudocode präsentiert die Kursleitung die algorithmischen Grund-strukturen Anweisung, Sequenz, Wiederholung, bedingte Wieder-holung und bedingte Anweisung sowie das Variablenkonzept. | Partnerarbeit, Unterrichts-gespräch | Beamer, PowerPoint-Präsentation, Aufgaben-blätter (siehe Abschnitt C.2.1), Scratch |
| Reflexion | Sammeln von Möglichkeiten und Maßnahmen zur Implementierung an den Schulen der Lehrkräfte und Thematisieren von Herausforderungen | Die Lehrkräfte notieren auf Plakatwänden ihre Eindrücke und Gedanken zu folgenden Themen: Bild des Programmierens („Programmieren ist jetzt für mich…“); kurz- mittel- und längerfristige Maßnahmen der Umsetzung des Programmierens an ihren Schulen; mögliche Herausforderungen (→ Bsp. LFB-DP5-3). | Die Kursleitung moderiert die Sichtung und Zusammenschau der Ergebnisse. | Einzelarbeit, Unterrichts-gespräch | Plakatwände, Schreib-material |

| | | | | | |
|---|---|---|---|---|---|
| *1. Praxisphase: Erprobungen der Unterrichtssequenz oder von ausgewählten Teilen der Unterrichtssequenz*<br><br>*Zwischen dem ersten und zweiten Fortbildungsblock liegen mehrere Wochen. Die Lehrkräfte sind aufgefordert, mindestens ein Element der Unterrichtssequenz in ihrem Unterricht auszuprobieren. Druckvorlagen der Unterrichtsmaterialien, Scratch-Dateien sowie ein Satz haptischer Scratch -Blöcke und eine Programmierunterlage aus Filz werden den Lehrkräften zur Verfügung gestellt. Bei Fragen oder Problemen können sie sich per Mail oder Telefon an die Kursleitung wenden (→ LFB-DP1, LFB-DP2, LFB-DP5-2). Sie sind angehalten, eine Rückmeldung zu ihren Erprobungen an die Kursleitung zu senden.* | | | | | |
| Erfahrungs-austausch | Austausch von Erfahrungen in der Umsetzung der Unterrichts-sequenz | Die Lehrkräfte sammeln ihre Eindrücke aus der ersten Praxis-phase auf Plakatwänden für die einzelnen Phasen der Unterrichts-sequenz (→ Bsp. LFB-DP5-4). Dabei notieren sie, in welchem fachlichen Kontext und Organisationsrahmen sie etwas erprobt haben, welche Anpassungen sie vorgenommen haben, wie die Reaktionen der SuS ausfielen und welche Probleme aufgetreten sind. Zusätzlich werden Kniffe und Tipps gesammelt, die sich aus der Erprobung ergeben haben. | Die Kursleitung moderiert die Sichtung und Zusammenschau der Ergebnisse. | Einzelarbeit, Unterrichts-gespräch | Plakatwände, Schreib-material |
| | Präsentation von *Best-Practice*-Beispielen | | Die Kursleitung präsentiert *Best-Practice*-Beispiele aus der vorigen Phase und den Zusendungen der Lehrkräfte. | Präsentation | Beamer, PowerPoint-Präsentation |

| | | | | | |
|---|---|---|---|---|---|
| Fach-didaktische Grundlagen | Kennenlernen von Aufgaben-formaten in Scratch | Die Lehrkräfte bearbeiten paarweise verschiedene Aufgaben in Scratch. | Die Kursleitung moderiert die Ergebnissicherung und zeigt die unterschiedlichen Aufgaben-formate der Übungen auf (→ Bsp. LFB-DP4-2). | Partnerarbeit, Unterrichts-gespräch | Beamer, PowerPoint-Präsentation Aufgaben-blätter (siehe Abschnitt C.2.2) |
| | Kennenlernen von Modellen über den Verlauf des Programmieren-lernens | Die Lehrkräfte ordnen die Aufgaben, die sie zuvor in Scratch bearbeitet hatten, in die Kategorien *use*, *modify* und *create* ein (→ Bsp. LFB-DP-4-2). Hier arbeiteten sie zu zweit mit einem Satz Kärtchen, auf denen die Programmieraufgaben in Miniatur abgebildet sind und die sie entsprechend gruppieren können. | Die Kursleitung präsentiert die *Use-Modify-Create Learning Progression* von Lee u. a. (2011). | Präsentation, Partnerarbeit | Beamer, PowerPoint-Präsentation, Aufgaben-kärtchen |
| | Kennenlernen weiterer kind-gemäßer Software und altersgemäßer Informatiksysteme zum Programmieren | Die Lehrkräfte bearbeiten in Kleingruppen verschiedene Stationen zu Möglichkeiten des Programmierens in der Grundschule (*Calliope mini*, *Makey Makey*, *Bluebot*, *ScratchJr*, *LEGO WeDo* und *LEGO EV3*; → Bsp. LFB-DP-4-3).<br><br>Jeweilige Vor- und Nachteil sowie mögliche Anknüpfungspunkte zur Unterrichtssequenz werden diskutiert. | Die Kursleitung moderiert den Austausch zu den einzelnen Informatiksystemen und Programmen. | Stationen-Lernen, Unterrichts-gespräch | Infoblätter zu Stationen (siehe Abschnitt C.2.3), *Calliope mini*, *Makey Makey*, *Bluebot*, *ScratchJr*, *LEGO WeDo*, *LEGO EV3* |

| | | | | | |
|---|---|---|---|---|---|
| *2. Praxisphase: Weitere Erprobungen der Unterrichtssequenz und ggf. Entwicklung sowie Erprobung eigener Lehr-Lernaktivitäten zum Programmieren*<br><br>*Zwischen dem zweiten und dritten Fortbildungsblock liegen mehrere Wochen. Die Lehrkräfte sind aufgefordert, die Unterrichtssequenz in ihrem Unterricht zu erproben. Bei Fragen oder Problemen können sie sich per Mail oder Telefon an die Kursleitung wenden (→ LFB-DP1, LFB-DP2, LFB-DP5-2). Sie sind angehalten, der Kursleitung ihre entwickelten Unterrichtsmaterialien zu senden.* | | | | | |
| Erfahrungs-austausch | Präsentation der von den Lehrkräften entwickelten Lehr-Lernaktivitäten | Die Lehrkräfte präsentieren ihre entwickelten Lehr-Lernaktivitäten (→ Bsp. LFB-DP5-4). | | Präsentation | Materialien der Lehrkräfte |
| Fach-didaktische Vertiefung | Kennenlernen von Programmier-kontexten aus bestehenden Fächern der Grundschule | Die Lehrkräfte bearbeiten paarweise verschiedene Aufgaben in Scratch – die Kontexte der Aufgaben entstammen Fächern der Grundschule (→ Bsp. LFB-DP6-1). | Die Kursleitung moderiert die Ergebnissicherung und holt Feedback der Lehrkräfte zu einem möglichen Einsatz der Aufgaben im Unterricht ein. | Partnerarbeit, Unterrichts-gespräch | Beamer, PowerPoint-Präsentation, Scratch, Aufgaben-blätter (siehe Abschnitt C.2.4) |
| | Austausch über gute Programmier-praktiken | Die Lehrkräfte untersuchen zwei Lösungen derselben Aufgabe nach Unterschieden in der Umsetzung. | Die Kursleitung moderiert die Ergebnissicherung und erarbeitet mit den Lehrkräften verschiedene positive Programmierpraktiken (→ Bsp. LFB-DP5-5). | Partnerarbeit, Unterrichts-gespräch | Beamer, PowerPoint-Präsentation, Scratch, Aufgaben-blätter (siehe Abschnitt C.2.5) |
| | Auseinandersetzen mit Aspekten des *Computational Thinking* | Die Lehrkräfte bearbeiten in Kleingruppen verschiedene Stationen zu einzelnen Aspekten des *Computational Thinking* (→ Bsp. LFB-DP6-3). | Die Kursleitung präsentiert das Konzept des *Computational Thinking*. | Präsentation, Gruppen-arbeit | Beamer, PowerPoint-Präsentation, Materialien für Stationen (siehe Abschnitt C.2.6) |

Teil V

# UMSETZUNG DER KONZEPTE IN DER PRAXIS

# 16 Erprobung der Unterrichtssequenz in unterrichtsnahen Lehr-Lernsituationen

Um das didaktische Design auf seine Durchführbarkeit zu testen und die empirische Evaluation vorzubereiten, wurde zunächst eine Pilotstudie durchgeführt. Sie besitzt explorativen Charakter und diente im Rahmen dieser Arbeit insbesondere der formativen Evaluation der Unterrichtssequenz mit der gewählten Zielgruppe. Die Ergebnisse, welche Auswirkungen auf die Gestaltung der Sequenz hatten, werden zusammenfassend dargestellt. Um im Anschluss die Tragfähigkeit der entwickelten Unterrichtssequenz in unterrichtsnahen Lehr-Lernsituationen zu untersuchen, wurden zwei weitere Erprobungszyklen durchgeführt. Währenddessen wurden von der Autorin und einer weiteren Forschenden in der Qualifikationsphase verschiedene Daten gesammelt, die Aufschluss darüber geben, was und wie Grundschulkinder programmieren: das gesamte Kursgeschehen wurde mit zwei Kameras gefilmt; die Bildschirme der Laptops, auf denen die Schülerinnen und Schüler programmierten, wurden mithilfe einer Bildschirmrekorder-Software aufgezeichnet; sämtliche Ergebnisse des Kurses, sowohl digital als auch analog, wurden gespeichert bzw. dokumentiert; die Schülerinnen und Schüler füllten Reflexionsbögen aus. Die Auswertung dieser Daten und daraus hervorgehende Publikationen stehen nicht im Fokus dieser Arbeit. Allerdings werden Ergebnisse der veröffentlichten Studien, die Rückschlüsse auf das Unterrichtskonzept zulassen, zusammenfassend dargestellt. Für die ersten drei Erprobungszyklen der Unterrichtssequenz wurden außerunterrichtliche Umsetzungsformate gewählt, da die Chancen, eine Genehmigung für die Durchführung von Erhebungen an öffentlichen Schulen in Bayern zu erhalten – insbesondere mit einer so umfangreichen Datenerhebung – als sehr gering eingeschätzt wurden (siehe Abschnitt 8.4). Eine grafische Übersicht über die Zyklen des DBR-Prozesses zur Entwicklung der Unterrichtssequenz und die jeweilige Datenerhebung und -auswertung kann in Kapitel A.1 eingesehen werden. Die Zyklen werden nachfolgend gemäß der methodischen Standards für *Design-Based Research* (siehe Abschnitt 7.4.2) dokumentiert.

## 16.1 Zyklus 1 (Pilotstudie)

Die Unterrichtssequenz wurde in Form eines Ferienkurses am Schülerforschungszentrum Berchtesgadener Land vom 17.05.2016 bis zum 19.05.2016 durchgeführt. Die Anmeldung zum Kurs erfolgte freiwillig und war für die Teilnehmerinnen und Teilnehmer kostenlos. Im Ferienkursprogramm des SFZ-BGL, das zu diesem Zeitpunkt an alle Schulen im Landkreis Berchtesgadener Land verschickt wurde, wurde der Kurs als *Programmierzirkus* beworben und war für Kinder zwischen acht und zehn Jahren ausgeschrieben. Der dreitägige Kurs war von 09.00 bis 15.00 Uhr angesetzt, um ei-

nerseits genügend Zeit für die Erprobung der Unterrichtssequenz zu haben und andererseits eine ausreichende Anzahl von Spielpausen integrieren zu können, die bei Ferienkursen des SFZ-BGL üblich waren. Alle Lehrhandlungen während des Kurses wurden von der Autorin durchgeführt, und eine weitere Person mit einem informatikdidaktischen Hintergrund zur Unterstützung der Datenerhebung war durchgehend anwesend. Zehn Kinder im Alter von acht bis zehn Jahren – neun Jungen und ein Mädchen – meldeten sich für den Kurs an, wobei ein Junge nur am ersten Tag und ein Junge nur am ersten und zweiten Tag teilnahm. Alle Kinder waren der Kursleitung bis zum Beginn des Kurses unbekannt. Zur Vorbereitung weiterer empirischer Studien wurden verschiedene Daten gesammelt: das gesamte Kursgeschehen wurde mit zwei Kameras gefilmt; die Bildschirme der Laptops, auf denen die Teilnehmenden programmierten, wurden mithilfe einer Bildschirmrekorder-Software aufgezeichnet; sämtliche Ergebnisse des Kurses, sowohl digital als auch analog, wurden gespeichert bzw. dokumentiert.

Die vier Phasen der Unterrichtssequenz wurden wie folgt auf die drei Kurstage verteilt:

- Tag 1: thematischer Einstieg; Grundlagen von Algorithmen und Programmen; Programmieren *unplugged*
- Tag 2: Programmieren in Scratch
- Tag 3: Eigene Programmierprojekte

Innerhalb der Pilotstudie erwiesen sich die unterrichtliche Handlungslinie und die Ablaufplanung der Unterrichtssequenz als gut durchführbar und zielführend. Die Aufgaben des ersten Tages konnten im Plenum bzw. in den Gruppen gelöst werden – das Fehlen von Computern in einem Programmierkurs irritierte jedoch einige Teilnehmende zu Beginn. Mit dem angeleiteten Programmieren im Rahmen des Lernzirkels hatten sie keine Probleme und auch die Transferaufgaben konnten überwiegend richtig gelöst werden. Einzelne Kinder hatten jedoch zu Beginn Schwierigkeiten, sich in der Ordnerstruktur zurechtzufinden und so die zu den Stationen gehörenden Scratch-Dateien zu finden. Das Planen der eigenen Programmideen fiel den Teilnehmenden schwerer, einige hätten sich gerne die Figuren und Bühnenbilder in Scratch als Anregung angesehen. Bei der Umsetzung der Ideen in Scratch entschieden sich einzelne Teilnehmende nochmal um bzw. erstellten mehrere Programme, die nicht alle im Voraus mithilfe der Vorlage geplant worden waren. Bei der Präsentation ihrer Ergebnisse über den Beamer zeigten sich die Teilnehmenden sehr stolz und freuten sich über den Applaus. Die meisten brachten eigene USB-Sticks mit, um die Programme ihren Eltern zu Hause zu zeigen und weiter daran zu arbeiten. In der Abschlussrunde äußerten die Teilnehmenden, dass sie gelernt hätten, ihrer Fantasie freien Lauf zu lassen und dass das Programmieren Spaß mache. Für zukünftige Erprobungen ergaben sich folgende Änderungsansätze:

- Beim Programmieren des *Lehrer-Roboters* lösten die Teilnehmenden zunächst Aufgaben, die von der Kursleiterin gestellt wurden, z. B. sollte das Fenster geöffnet werden. Im Anschluss wurden die Teilnehmenden aufgefordert, eigene Aufträge zu formulieren. Hierbei wurde der Auftrag gestellt, dass der *Lehrer-Roboter* eine Person fangen soll. Die Umwandlung des Auftrags in einzelne Befehle

konnte mithilfe der Kursleiterin zwar bewerkstelligt werden, nahm jedoch relativ viel Zeit in Anspruch. Zudem musste eine Wiederholung als algorithmisches Strukturelement eingesetzt werden, was zur Einführung in die Thematik ungünstig war. Daher sollten zu komplexe Aufträge in Zukunft von der Lehrkraft abgefangen werden.

- Als bildliche Anleitung, zu der die Teilnehmenden sprachliche Anweisungen schreiben sollten, wurde das Basteln eines Namensschildes mithilfe einer Buttonmaschine gewählt. Diese Handlung erwies sich für den Ferienkurs zwar als praktikabel, es kam bei der Ausführung der sprachlichen Befehle an der Buttonmaschine jedoch zu zeitlichen Verzögerungen. Für zukünftige Durchführungen sollte deshalb eine andere Handlung gewählt werden, die von den Teilnehmenden parallel ausgeführt werden kann.
- Für das Programmieren *unplugged* wurden zwei Aufgaben erstellt, in denen jeweils ein Weg durch verschiedene Parcours beschrieben werden musste. In den Visualisierungen der Parcours auf den Aufgabenblättern wurden rote Felder benutzt – einmal um darzustellen, dass sich unter dem Feld ein Gegenstand verbirgt, einmal zur Darstellung von Mauern. Die Doppelbesetzung der roten Felder führte bei den Teilnehmenden zu Irritationen und sollte in Zukunft vermieden werden, um unnötige kognitive Belastung zu vermeiden. Auf den Einsatz von Mauern in den Fragestellungen wird deshalb künftig verzichtet.
- Für die Planung der eigenen Programmierprojekte wurde den Teilnehmenden eine Vorlage in Form eines Projekt-Drehbuchs zur Verfügung gestellt. Auf dem Arbeitsblatt sollte zunächst vermerkt werden, welche Figuren etwas tun sollen, was sie tun sollen und wie sie dies erreichen können. In der Auswertung der Drehbücher stellte sich heraus, dass es den Teilnehmenden schwer fiel, die letzte Frage („Wie?“) zu beantworten. Um den Fokus der Frage auf die Umsetzung in Scratch zu lenken, könnten verschiedene Programmierbefehle aus Scratch abgebildet werden, die die Teilnehmenden ankreuzen können.
- Die Teilnehmenden wurden zunächst dazu angehalten, in ihren eigenen Programmierprojekten eine Zirkusgeschichte zu implementieren – passend zum Kontext des Lernzirkels. Da sich aber einige der Jungen sehr auf das Programmieren eines Spiels gefreut hatten, wurde das Programmierprojekt thematisch geöffnet. Da in der Abschlussrunde alle Jungen zum Ausdruck brachten, dass ihnen die Umsetzung ihrer eigenen Spielideen am besten gefallen hat, wird dies auch in Zukunft beibehalten. Auf die Zirkusgeschichte kann jedoch zurückgegriffen werden, wenn ein Kind Schwierigkeiten hat, seine eigene Programmidee zu entwickeln.

## 16.2 Zyklus 2

Die Unterrichtssequenz wurde im Schuljahr 2015/16 in Form von zwei extracurricularen Dreitageskursen in den Räumlichkeiten der Technischen Universität München durchgeführt. Diese fanden vom 27. – 29.06.2016 und vom 18. – 20.07.2016, jeweils von 09.00 bis 13.00 Uhr, statt. An den Kursen nahm je eine Schulklasse der vierten Jahrgangsstufe einer staatlichen Grundschule teil – eine aus

der ländlich geprägten Gemeinde Taufkirchen südlich von München und eine aus der Bildungsregion Hasenbergl, einem Stadtteil im Münchner Norden. Der Kontakt zu beiden Schulen wurde durch private Kontakte hergestellt. Letztere zeichnet sich aus durch einen auffallend hohen Anteil an Schülerinnen und Schülern mit Migrationshintergrund, die im Schuljahr 2012/13 mit 50 % bis unter 70 % über dem Münchner Durchschnitt von 42.6 % liegt (Landeshauptstadt München 2014, S. 13). Der Sozialindex[1] der Grundschule liegt zwischen 87.3 und 96.5 und ist demnach niedriger als der gesamtstädtische Index von 100. Die soziale Belastung ist dort also höher, was sich negativ auf individuelle Bildungsprozesse auswirken kann (ebd., S. 12 f.). Zur Grundschule in Taufkirchen konnten keine vergleichbaren Daten bzgl. der Schülerinnen und Schüler ermittelt werden. Zur Einordnung – besonders in der Abgrenzung zur anderen Schule – ist jedoch anzumerken, dass der Anteil an Einwohnerinnen und Einwohnern mit Migrationshintergrund im Jahr 2020 bei 18,3 % lag[2].

### 16.2.1 Anpassungen in der Unterrichtssequenz

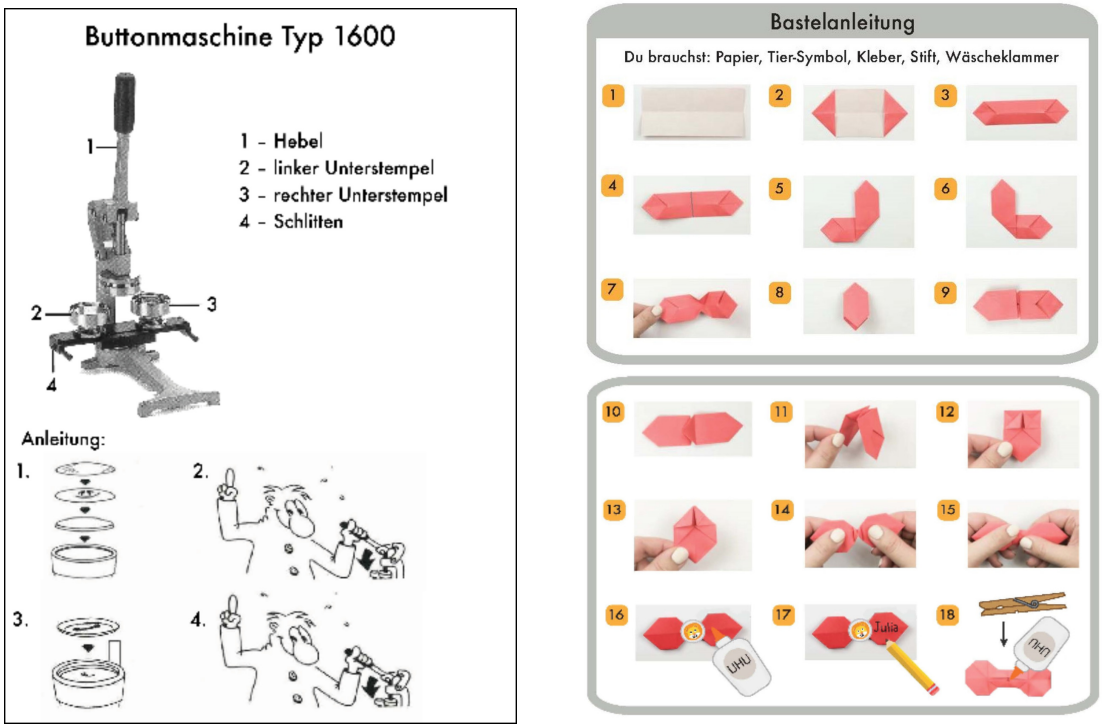
Die Unterrichtssequenz wurde für die Erprobung im zweiten Zyklus auf Grundlage der Ergebnisse der Pilotstudie wie folgt angepasst (siehe Abschnitt 16.1):

- Als bildliche Anleitung, zu der die Teilnehmenden sprachliche Anweisungen schreiben, wurde das Falten eines Namensschildes ausgewählt – eine Handlung, die alle Schülerinnen und Schüler gleichzeitig ausführen können (siehe Abbildung 16.1[3]).
- Für das Programmieren *unplugged* wurden zwei neue Aufgaben erstellt, in denen jeweils ein Weg durch verschiedene Parcours beschrieben werden muss. Diese wurden so gestaltet, dass daran das Konzept der Wiederholung (siehe Abbildung 16.3) und der bedingten Anweisung (siehe Abbildung 16.4) thematisiert werden kann.
- Die Vorlage für die Planung der eigenen Programmierprojekte wurde in zwei Punkten angepasst (siehe Abbildung 16.2). Die Leitfragen für die Punkte „Was?“ und „Wie?“ wurden angepasst, um klarer und leichter verständlich zu machen, worauf die Fragen abzielen. Für die „Wie?“-Frage wurden Programmierbefehle aus Scratch zur Auswahl eingefügt, welche die Schülerinnen und Schüler ankreuzen können.

[1] „Der Sozialindex ist eine für das Münchner Bildungsmonitoring gebildete Kennzahl, die für die soziale Belastungslage eines städtischen Gebietes steht. Er wird aus drei bevölkerungsstatistischen Faktoren (Kaufkraft der Haushalte, Ausländeranteil an der Hauptwohnsitzbevölkerung, Anteil der Haushalte mit Abitur oder Fachabitur) errechnet“ (Landeshauptstadt München 2014, S. 12).

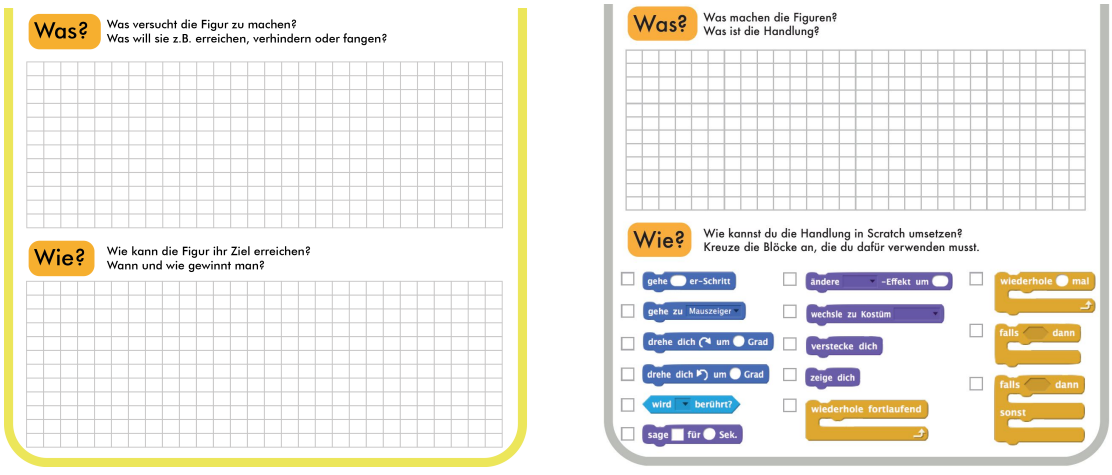
[2] https://ugeo.urbistat.com/AdminStat/de/de/demografia/dati-sintesi/taufkirchen/20178743/4

[3] Die Abbildungen von (a) sind der Anleitung zur Buttonmaschine Typ 1600 entnommen (https://www.yumpu.com/de/document/view/7489464/anleitung-zu-buttonmaschine-typ-1600-modell-ab-juli-2001), von (b) einer Origami-Anleitung (https://www.thesprucecrafts.com/easy-origami-bow-tie-tutorial-4038654).




*(a)* *(b)*

*Abbildung 16.1* *Bildliche Anleitung zum Basteln eines Namensschildes aus (a) der Pilotstudie in Zyklus 1, in der das Namensschild mittels einer Buttonmaschine hergestellt wurde, und (b) Zyklus 2, in der das Namensschild gefaltet wurden.*



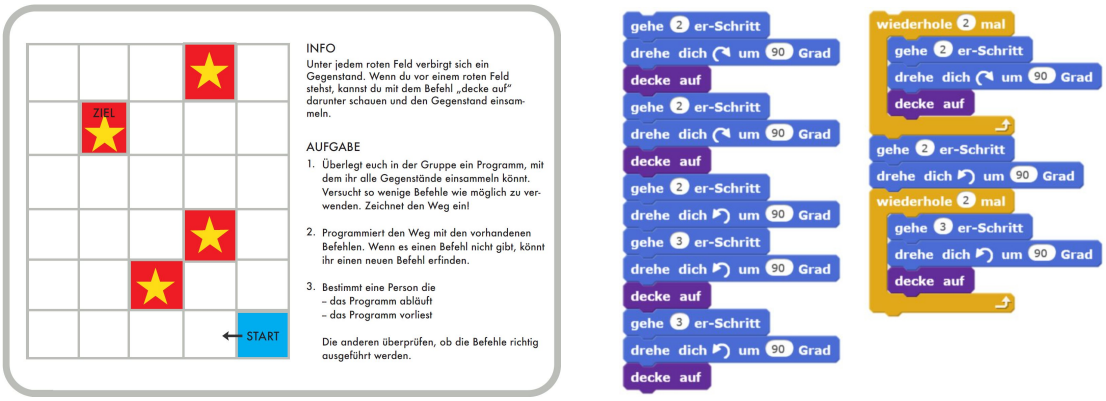

*(a)* *(b)*

*Abbildung 16.2* *Ausschnitte der Vorlage für ein Projekt-Drehbuch aus (a) der Pilotstudie und (b) dem zweiten Zyklus.*

### 16.2.2 Durchführung

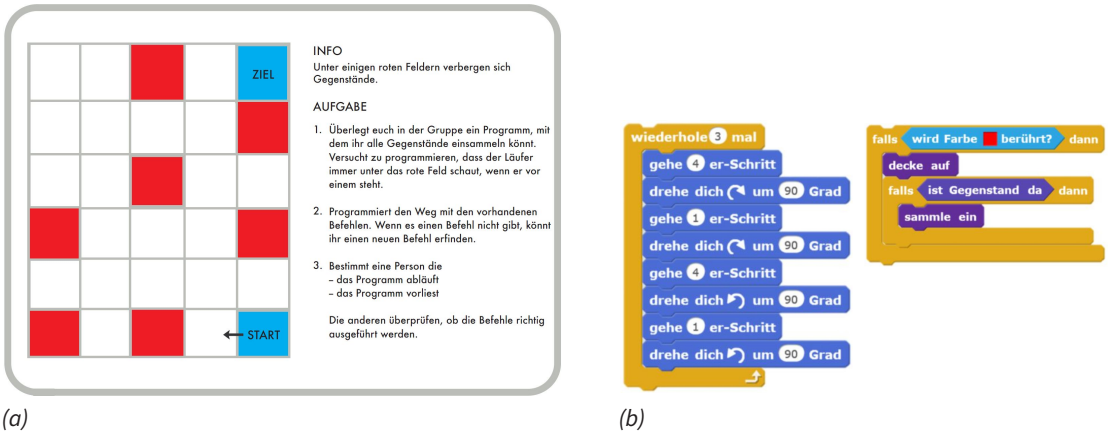
Da nicht für alle Kinder eine Einwilligungserklärung zur Anfertigung von Videoaufnahmen vorlag, wurden die Klassen für den Dreitageskurs jeweils in zwei Gruppen aufgeteilt:

- Gruppe 1 (Klasse 1, Jahrgangsstufe 4): zehn Kinder (6 Mädchen/4 Jungen); mit Videoaufzeichnung
- Gruppe 2 (Klasse 1, Jahrgangsstufe 4): neun Kinder (6 Mädchen/3 Jungen)




*(a)* *(b)*

*Abbildung 16.3 (a) Parcoursaufgabe mit Fokus auf Wiederholungen und (b) unterschiedliche Lösungsmöglichkeiten.*




*(a)* *(b)*

*Abbildung 16.4 (a) Parcoursaufgabe mit Fokus auf bedingte Anweisungen und (b) mögliche Elemente der Lösung.*

- Gruppe 3 (Klasse 2, Jahrgangsstufe 4): zehn Kinder (7 Mädchen/3 Jungen); mit Videoaufzeichnung
- Gruppe 4 (Klasse 2, Jahrgangsstufe 4): neun Kinder (6 Mädchen/3 Jungen)

In den Gruppen 1 und 3 wurden die Lehrhandlungen während des Kurses von der Autorin durchgeführt, und eine weitere Person der Arbeitsgruppe für Didaktik der Informatik war anwesend, um bei Bedarf zu unterstützen und die Datenerhebung zu beaufsichtigten. In den Gruppen 2 und 4 wurden die Lehrhandlungen hauptsächlich von einer Studierenden für das Lehramt Informatik an Gymnasien durchgeführt, und eine Person der Arbeitsgruppe war auch hier zur Unterstützung anwesend. Alle Schülerinnen und Schüler waren den Kursleiterinnen und Kursleitern bis zum Beginn der Dreitageskurse unbekannt. Die Aufteilung der vier Phasen der Unterrichtssequenz auf die drei Kurstage wurde aus der Pilotstudie übernommen:

- Tag 1: thematischer Einstieg; Grundlagen von Algorithmen und Programmen; Programmieren *unplugged*
- Tag 2: Programmieren in Scratch
- Tag 3: Eigene Programmierprojekte

### 16.2.3 Gewonnene Erkenntnisse zur Unterrichtssequenz

Innerhalb der beiden Dreitageskurse erwiesen sich die unterrichtliche Handlungslinie und die Ablaufplanung der Unterrichtssequenz erneut als gut durchführbar und zielführend. Die Aufgaben des ersten Tages konnten im Plenum bzw. in den Gruppen gelöst werden – für die Verwendung einer Wiederholung und einer bedingten Anweisung in den Lösungen der Parcoursaufgaben brauchten die Schülerinnen und Schüler jedoch teilweise Hilfestellungen durch die Kursleitung. Mit dem angeleiteten Programmieren im Rahmen des Lernzirkels hatten sie keine Probleme und auch die Transferaufgaben konnten erneut überwiegend richtig gelöst werden. Die Ordnerstruktur, in der sich die Scratch-Dateien für die Bearbeitung des Lernzirkels befanden, wurde im Vorfeld erklärt. Beim Planen der eigenen Programmideen orientierte sich etwa die Hälfte der Schülerinnen und Schüler an den Inhalten des Lernzirkels, implementierte letztlich aber nicht immer nur eine Zirkusgeschichte, sondern auch weitere, thematisch abweichende Programme. An beiden Tagen, an denen die Schülerinnen und Schüler in Scratch programmierten, arbeiteten sie so konzentriert, dass sie mehrmals gebeten werden mussten, in die Pause zu gehen. Viele äußerten, dass sie lieber weiter programmieren würden. Bei der Präsentation ihrer Programme über den Beamer schienen die Schülerinnen und Schüler sehr stolz auf ihre Ergebnisse zu sein und freuten sich über den Applaus und eine personalisierte Teilnahmeurkunde.

In dem Reflexionsbogen, den sie am Ende des ersten Kurstages ausfüllten, gaben die Schülerinnen und Schüler an, dass es ihnen am schwersten fiel, den *Lehrer-Roboter* zu steuern und Befehle in vollständigen Sätzen zu formulieren. Am meisten Spaß machten ihnen das Ausführen der Programme im Parcours und die Arbeit mit den haptischen Scratch-Blöcken. Am zweiten Tag fanden sie die Station des Lernzirkels am schwierigsten, bei der die bedingte Anweisung eingeführt wurde; am dritten Tag das Einhalten der Vorgaben, eine Wiederholung und eine bedingte Anweisung zu verwenden, und das Projekt im Voraus zu planen.

In zwei weiteren Ferienkursen am Schülerforschungszentrum Berchtesgadener Land, in denen die Unterrichtssequenz durchgeführt wurde, erprobte die Autorin weitere *Unplugged*-Aufgaben: das Programmieren von Spielen, die den Teilnehmenden aus ihrem Alltag gut bekannt sind. Ausgewählt wurden die Spiele *Fangen* und *Feuer, Wasser, Sturm*, die sich jeweils gut für das Thematisieren einer Wiederholung und bedingten Anweisung eignen. Das Erstellen von entsprechenden Programmen mit den haptischen Scratch-Blöcken hat zwar gut funktioniert, im Spielgeschehen war es jedoch nicht wirklich möglich, die Programme auf ihre Korrektheit zu überprüfen. Die Aufgaben wurden daher für den Unterrichtskontext wieder verworfen.

Der Aufbau der Unterrichtssequenz wurde nach der Erprobung als Poster auf einer internationalen Konferenz zur Informatik in der Schule[4] (Geldreich, Funke und Hubwieser 2016) präsentiert. Das mündliche Feedback im Rahmen der Postersession war sehr positiv und vor allem die Arbeit mit den haptischen Scratch-Blöcken wurde als sehr vielversprechend eingeschätzt. Die eigenen Projekte, welche die Schülerinnen und Schüler am Ende der Unterrichtssequenz programmieren, wurden

[4] ISSEP: *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*

im Anschluss an die Erprobungen ausgewertet. Die Ergebnisse wurden auf zwei Konferenzen, einer zum Informatikunterricht in der Primar- und Sekundarstufe[5] und einer zur technischen Bildung[6], als Artikel veröffentlicht und präsentiert. In einem Beitrag wurden die Scratch-Projekte anhand eines entwickelten Kategoriensystems bzgl. verschiedener Aspekte analysiert: (a) *Requirements*, (b) *Programming Concepts*, (c) *Code Organization* und (d) *Operability* (vgl. Funke, Geldreich und Hubwieser 2017). Untersucht wurde unter anderem, wie viel Prozent der Schülerinnen und Schüler die gestellten Vorgaben für das Projekt erfüllten (ebd., S. 1238): das Projekt enthält mehrere Figuren (94 %), die sich bewegen (98 %), und mindestens eine Wiederholung (78 %) und bedingte Anweisung (56 %). Programmiert wurden vorrangig Animationen (30 % aller Programme) und Geschichten (61 % aller Programme). In einer weiteren Analyse wurde untersucht, wie sich die Projekte von Mädchen und Jungen unterscheiden (Funke und Geldreich 2017). Ein Ergebnis dieser Studie ist, dass 90 % aller programmierten Spiele von Jungen programmiert wurden und 79 % der Geschichten von Mädchen. Für die Unterrichtssequenz heißt das, dass es beibehalten wird, den Schülerinnen und Schülern für ihre Projekte keine inhaltlichen Vorgaben zum Thema des Projekts, wie z. B. „Programmiert eine Zirkusgeschichte!“ zu machen.

Für zukünftige Erprobungen ergaben sich folgende Änderungsansätze:

- Das Brainstorming zu Beginn des Kurses verlief in manchen Gruppen sehr schleppend. Um die Schülerinnen und Schüler anzuregen, könnte man hier z. B. mit visuellen Impulsen arbeiten.
- Das Falten einer Schleife als Namensschild erwies sich zwar als praktikabel, die Beschreibung des Vorgangs benötigte durch die hohe Anzahl einzelner Arbeitsschritte jedoch relativ viel Zeit. Außerdem waren die sprachlichen Anweisungen alle sehr ähnlich – häufig musste eine Kante oder Ecke auf eine andere Kante oder Ecke gefaltet werden, was die Schülerinnen und Schüler mit der Zeit etwas zu langweilen schien. Eine kürzere, etwas abwechslungsreichere Anleitung wäre an dieser Stelle sinnvoller.
- Um die Schülerinnen und Schüler besser für den Einsatz der algorithmischen Strukturen der Wiederholung und bedingten Anweisung in Scratch vorzubereiten, sollten diese ausführlicher beim Programmieren *unplugged* behandelt werden. Dazu sollten die Aufgaben im Parcours so gestaltet werden, dass die Verwendung der Strukturen für die Schülerinnen und Schüler – auch ohne Hilfestellung – ersichtlicher wird.
- Die Arbeitsanweisungen und Aufgaben im Lernzirkel im DIN A4-Format wirkten manchmal etwas unhandlich, da der Platz bei der Arbeit am Computer eher knapp war. Außerdem befanden sich an einigen Arbeitsanweisungen des Lernzirkels sehr viele einzelne Arbeitsschritte auf einer Seite, wodurch einige Schüler durcheinander kamen. Darüber hinaus waren einige Schülerinnen und Schüler schneller mit dem Lernzirkel fertig als ihre Mitschülerinnen und Mitschüler. In Zukunft könnte für die Arbeitsanweisungen ein anderes Format gewählt werden und zusätzliche Aufgaben erstellt werden.

[5] WiPSCE: *Workshop in Primary and Secondary Computing Education*
[6] EDUCON: *The IEEE Global Engineering Education Conference*

## 16.3 Zyklus 3

Im Jahr 2017 wurde die Unterrichtssequenz in Form von vier extracurricularen Dreitageskursen in den Räumlichkeiten der Technischen Universität München durchgeführt. Diese fanden vom 19. – 21.06.2017, 28. – 30.06.2017, vom 12. – 14.07.2017 und vom 19. – 21.07.2017, jeweils von 09.00 bis 12.15 Uhr, statt. Drei vierte Klassen und eine dritte Klasse nahmen an den Kursen teil. Sie stammten alle aus der Grundschule in der Bildungsregion Hasenbergl, die bereits im vorherigen Zyklus teilgenommen hat. Den Kontakt zu den Lehrkräften stellte die Lehrerin her, die mit ihrer Klasse bereits im Vorjahr teilgenommen hatte.

### 16.3.1 Anpassungen in der Unterrichtssequenz

Entsprechend der Erkenntnisse des zweiten Zyklus wurde die Unterrichtssequenz für die Erprobung im dritten Zyklus wie folgt angepasst:

- Im Rahmen des thematischen Einstiegs wird nach dem Brainstorming anhand von verschiedenen Bildern thematisiert, wo sich Programme in der Umgebung der Schülerinnen und Schüler verstecken (siehe Abbildung 16.5). Gemeinsam stellen sie Vermutungen an, was auf den Bildern programmiert sein könnte. Die unterrichtliche Handlungslinie und Ablaufplanung werden jeweils an der entsprechenden Stelle um den Punkt „Herstellen eines Bezugs zum Alltag der Schülerinnen und Schüler“ ergänzt.
- Als bildliche Anleitung wurde eine Anleitung zum Basteln eines Namensschildes erstellt, die deutlich weniger Handlungsschritte umfasst (siehe Abbildung 16.6). Zusätzlich wurde darauf geachtet, dass sie in der Anzahl der sprachlichen Befehle mehr Spielraum lässt und die einzelnen Bastelschritte mehr unterschiedliche Begrifflichkeiten enthalten können, z. B. Moosgummi, Filz, Muffinförmchen, Tonpapier.
- Für das Programmieren *unplugged* wurden zwei neue Aufgaben erstellt, in denen jeweils ein Weg durch verschiedene Parcours beschrieben werden muss. Als Neuerung wurden die Aufgaben in kleine Geschichten eingebettet, z. B. „Der Zirkusdirektor hat eine Aufgabe für den Affen: Wenn er ‚Los!‘ ruft, soll er entlang der roten Felder zur Banane laufen.“ Auf diese Weise können die Erprobungen der Lösungen in kurze Rollenspiele eingebettet werden. Die Aufgaben wurden zudem mit der Absicht gestaltet, dass sich der Einsatz einer Wiederholung (siehe Abbildung 16.8) und bedingten Anweisung (siehe Abbildung 16.9) mehr anbietet. Darüber hinaus wurde eine Knobelaufgabe als Zusatzaufgabe erstellt, in der den Schülerinnen und Schüler die erlaubte Anzahl der Blöcke zur Lösung einer Aufgabe vorgegeben wurde (siehe Abbildung 16.7).
- Für die Arbeitsanweisungen im Lernzirkel wurde das DIN A5-Format gewählt, um die Seiten übersichtlicher und bei der Arbeit mit dem Computer handlicher zu gestalten (siehe Abbildung 16.10). Die Seiten einer Station wurden mithilfe einer Öse zusammengehalten. Außerdem wurden vier Zusatzaufgaben gestaltet, welche die Schüler nach Fertigstellung des Lernzirkels in Scratch bearbeiten konnten.

- Da in diesem Zyklus erstmals eine dritte Klasse an der Unterrichtssequenz teilnehmen würde, wurde für alle Unterrichtsmaterialien eine besonders leserliche Schriftart ausgewählt. Die Schrift *Sassoon Primary* wurde im Rahmen eines britischen Forschungsprojektes für Kinderbücher und Unterrichtsmaterialien entwickelt und ist den Ergebnissen zufolge für Kinder leicht zu lesen[7].

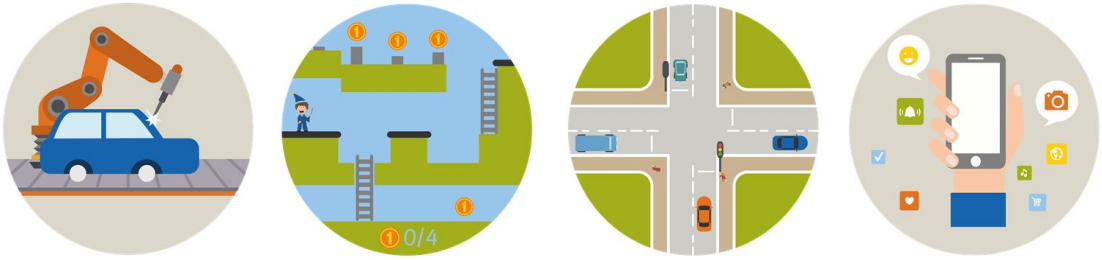



*Abbildung 16.5 Visuelle Impulse „Wo verstecken sich überall Programme?“*

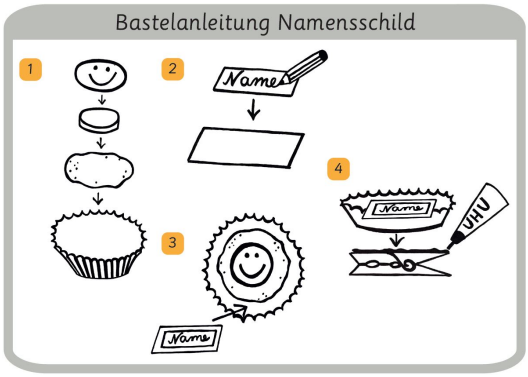



*Abbildung 16.6 Bildliche Anleitung zum Basteln eines Namensschildes aus Zyklus 3*

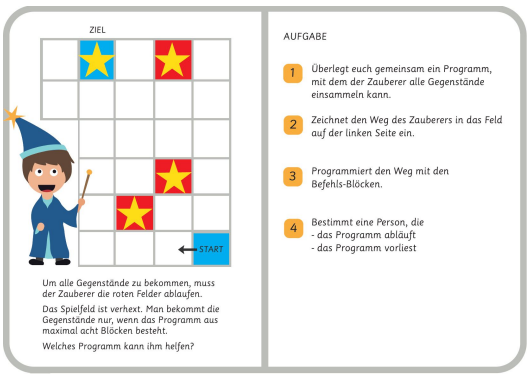
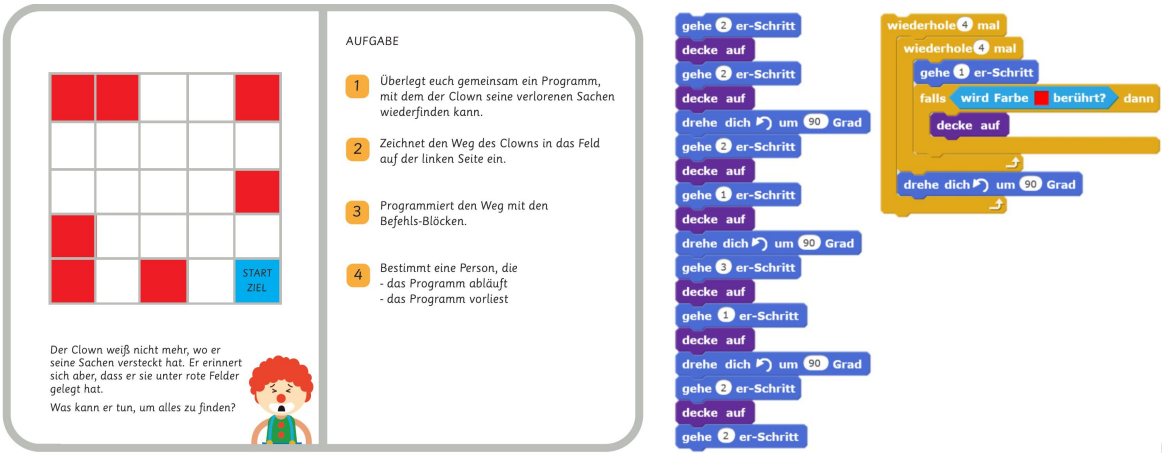



*Abbildung 16.7 Zusätzliche Parcoursaufgabe für schnelle Schülerinnen und Schüler*


*(a)* *(b)*

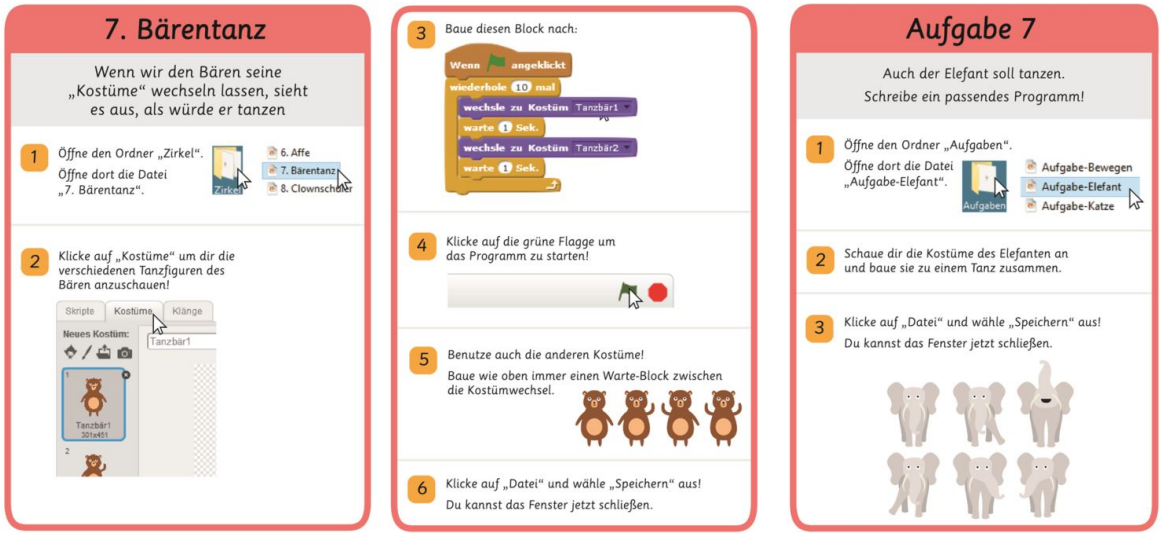
*Abbildung 16.8 (a) Parcoursaufgabe mit Fokus auf Wiederholungen und (b) unterschiedliche Lösungsmöglichkeiten.*

[7] http://www.identifont.com/show?3XD




(a) (b)

*Abbildung 16.9* *(a) Parcoursaufgabe mit Fokus auf bedingte Anweisungen und (b) unterschiedliche Lösungsmöglichkeiten.*




(a) (b)

*Abbildung 16.10* *Arbeitsanweisungen für den Lernzirkel mit (a) einem Programm, das nachprogrammiert wird und (b) einer passenden Aufgabe.*

### 16.3.2 Durchführung

Da, wie im Vorjahr, nicht für alle Kinder eine Einwilligungserklärung zur Anfertigung von Videoaufnahmen vorlag, wurden die Klassen für den Dreitageskurs erneut jeweils in zwei Gruppen aufgeteilt:

- Gruppe 1 (Klasse 1, Jahrgangsstufe 4): acht Kinder (4 Mädchen/4 Jungen); mit Videoaufzeichnung
- Gruppe 2 (Klasse 1, Jahrgangsstufe 4): acht Kinder (3 Mädchen/5 Jungen)
- Gruppe 3 (Klasse 2, Jahrgangsstufe 3): neun Kinder (4 Mädchen/5 Jungen); mit Videoaufzeichnung
- Gruppe 4 (Klasse 2, Jahrgangsstufe 3): neun Kinder (6 Mädchen/3 Jungen)
- Gruppe 5 (Klasse 3, Jahrgangsstufe 4): neun Kinder (3 Mädchen/6 Jungen); mit Videoaufzeichnung

- Gruppe 6 (Klasse 3, Jahrgangsstufe 4): acht Kinder (5 Mädchen/3 Jungen)
- Gruppe 7 (Klasse 4, Jahrgangsstufe 4): acht Kinder (4 Mädchen/4 Jungen); mit Videoaufzeichnung
- Gruppe 8 (Klasse 4, Jahrgangsstufe 4): sieben Kinder (4 Mädchen/3 Jungen)

In den Gruppen 1, 4, 5 und 7 wurden alle Lehrhandlungen während des Kurses von der Autorin durchgeführt. Die Kurse für Gruppe 2, 3 und 8 wurden jeweils von verschiedenen Personen der Arbeitsgruppe für Didaktik der Informatik geleitet. In Gruppe 6 wurden die Lehrhandlungen von einer Studierenden für das Lehramt Informatik an Gymnasien durchgeführt. Alle Schülerinnen und Schüler waren den Kursleiterinnen und Kursleitern bis zum Beginn der Dreitageskurse unbekannt. Die vier Phasen der Unterrichtssequenz wurde wie im Vorjahr auf die drei Kurstage verteilt.

### 16.3.3 Gewonnene Erkenntnisse zur Unterrichtssequenz

Innerhalb der Kursdurchführungen erwies sich die didaktische Konzeption der Unterrichtssequenz als gut durchführbar und zielführend. Die *Unplugged*-Aufgaben konnten ohne größere Schwierigkeiten gelöst werden: die Schülerinnen und Schüler stellten rege Vermutungen an, was auf den visuellen Impulsen programmiert wird; die neu erstellte Bastelanleitung für das Namensschild konnte von den Schülerinnen und Schülern in eine überschaubare Anzahl an sprachlichen Befehlen überführt werden; bei der Bearbeitung der Wiederholungs-Parcoursaufgabe kam der Großteil der Gruppen von alleine oder mit nur kleinen Hilfestellungen auf die Lösungsvariante mit der Wiederholung. Bei der Bearbeitung der Parcoursaufgabe für die Einführung der bedingten Anweisung entwickelten viele Gruppen eine Lösung mit einer bedingten Anweisung, die jedoch nicht vollständig korrekt war. Jedoch konnte beim Ablaufen der Lösungen im Parcours problemlos die Lösungsvariante mit der bedingten Anweisung erarbeitet werden. Hier war es hilfreich, wenn die Kursleitung im Parcours die fehlerhaften Lösungen der Schülerinnen und Schüler der richtigen Variante kontrastierend gegenüberstellte. Mit dem angeleiteten Programmieren im Rahmen des Lernzirkels hatten sie keine Probleme und auch die Transferaufgaben wurden erneut meist richtig gelöst. Auffällig war, dass in den beiden Gruppen der dritten Klasse vermehrt Schülerinnen und Schüler mehr Zeit zur Fertigstellung des Lernzirkels benötigten. Außerdem hatten Einzelne Probleme, einen Doppelklick auszuführen. Nach eigenen Angaben nutzten sie zuvor fast ausschließlich Tablets oder Handys. Das neue Format der Arbeitsanweisungen hat sich gut bewährt. Beim Planen der eigenen Programmideen orientierten sich nur noch ca. 40 % der Schülerinnen und Schüler an den Inhalten des Lernzirkels. Wie bereits in Zyklus 2 wollten die meisten Schülerinnen und Schüler am zweiten und dritten Kurstag lieber weiter programmieren, als in die Pause zu gehen. Bei der Präsentation ihrer Programme über den Beamer schienen die Schülerinnen und Schüler sehr stolz auf ihre Ergebnisse zu sein und freuten sich über den Applaus und eine personalisierte Teilnahmeurkunde.

In dem Reflexionsbogen, den sie am Ende des ersten Kurstages ausfüllten, gaben die meisten Schülerinnen und Schüler an, dass sie die Parcoursaufgaben, insbesondere die Aufgabe mit der

bedingten Anweisung, am schwierigsten fanden – gleichzeitig machten diese vielen aber auch am meisten Spaß. Am zweiten und dritten Tag hätten sich einige Schülerinnen und Schüler mehr Zeit für die Arbeit am Lernzirkel und an ihrem eigenen Projekt gewünscht. Am dritten Kurstag machte ihnen am meisten Spaß, dass sie selbst etwas Eigenes programmieren durften. Einige Schülerinnen und Schüler äußerten außerdem, dass ihnen die Präsentation ihrer Programme am meisten Spaß machte.

Der Aufbau der Unterrichtssequenz wurde nach den Erprobungen innerhalb eines Workshops für Lehramtsanwärterinnen und Lehramtsanwärter des Lehramts an Grundschulen am Schülerforschungszentrum Berchtesgadener Land und innerhalb eines Workshops auf einer nationalen Konferenz zur informatischen Bildung[8] präsentiert. Das mündliche Feedback im Rahmen der Workshops war sehr positiv, wobei die *Unplugged*-Aufgaben des ersten Tages besonders hervorgehoben wurden.

Die Ergebnisse der ersten Phase der Unterrichtssequenz, in der die Schülerinnen und Schüler ihre Assoziationen zum Begriff *Programmieren* äußern und überlegen, was auf verschiedenen Bildern programmiert sein könnte wurden nach den Erprobungen ausgewertet und auf einer Konferenz zum Informatikunterricht in der Primar- und Sekundarstufe[9] als Artikel veröffentlicht und präsentiert (Geldreich, Simon und Starke 2019). Hier zeigte sich, dass sowohl Jungen als auch Mädchen mit einer fast ähnlichen Anzahl von Wortäußerungen an der Assoziationsfrage teilnahmen (ebd., S. 4). Bei der Bilderaufgabe jedoch beteiligten sich die Jungen mit fast doppelt so vielen Äußerungen (ebd., S. 6). Für die Analyse der Programmierprozesse der Schülerinnen und Schüler bei der Implementierung ihrer eigenen Ideen wurden zwei unterschiedliche Diagrammtypen als Visualisierungswerkzeuge entwickelt und einzelne Bildschirmaufnahmen damit untersucht. Die Ergebnisse wurden auf der gleichen Konferenz als Artikel veröffentlicht und präsentiert (Simon, Geldreich und Hubwieser 2019). Hierbei zeigte sich, dass Skripte bzw. Figuren inklusive Skript in Scratch-Projekten gelöscht wurden, ohne jemals ausgeführt zu werden. Darüber hinaus zeigte sich, dass häufig die Skripte mehrerer Figuren bearbeitet wurden und erst dann das Programm ausgeführt wurde. Für zukünftige Erprobungen ergaben sich folgende Änderungsansätze:

- Die Bilder, zu denen die Schülerinnen und Schüler in der ersten Phase der Unterrichtssequenz überlegen sollten, was auf ihnen programmiert ist, scheinen eher Jungen anzusprechen. Die darauf abgebildeten Themen – Verkehr, Computerspiele, Fabriken – könnten angepasst oder erweitert werden, um Mädchen mehr anzusprechen. Gemäßt der Ausführungen in Kapitel 12.1.4 könnten hier Bilder bzw. Themen ausgewählt werden, die Mädchen aus ihrem Alltag kennen.

- Im Rahmen der Arbeit an den eigenen Programmierprojekten könnte man die Schülerinnen und Schüler erneut auf verschiedenen Schritte des Problemlösen aufmerksam machen oder gezielt ansprechen, wann oder wie oft man ein Programm am besten ausführen sollte.

[8] INFOS: Fachtagung des Fachausschusses Informatische Bildung in Schulen der Gesellschaft für Informatik
[9] WiPSCE: *Workshop in Primary and Secondary Computing Education*

## 16.4 Übersicht der Änderungen

Im Folgenden werden die Änderungen an der Unterrichtssequenz, die im Rahmen der Erprobung in unterrichtsnahen Lehr-Lernsituationen durchgeführt wurden, tabellarisch dargestellt (siehe Tabelle 16.1). Zur weiteren Strukturierung der Änderungen werden die Entscheidungsfelder des *Berliner Modells* (siehe Kapitel 12) herangezogen: Intentionen, Fachinhalte, Methoden und Medien/Materialien. Zu jeder Änderung wird die jeweilige Argumentation aufgeführt sowie deren Quelle. Es wird bspw. vermerkt, ob diese der Analyse bestimmter Unterrichtsergebnisse entstammt. Zusätzlich wird anhand der Schritte der Unterrichtssequenz dargestellt, wie viele Änderungen pro Entscheidungsfeld durchgeführt wurden (siehe Tabelle 16.2).

*Tabelle 16.1 Übersicht der Änderungen an der Unterrichtssequenz nach Zyklus*

| Erprobungszyklen der Unterrichts-sequenz | | Änderungsansätze | Begründung (Quelle der Begründung) |
|---|---|---|---|
| Zyklus 1 | Methoden | Programmierprojekt wird thematisch geöffnet. Es muss keine Zirkus-geschichte implementiert werden. | Umsetzung eigener Spielideen gefiel den Jungen am besten (Auswertung Abschlussrunde). |
| | | Handlung, die in bildlicher Anleitung dargestellt ist, sollte von den SuS parallel ausgeführt werden können. | Es kam zu zeitlichen Verzögerungen an der Buttonmaschine (Durchführung). |
| | Medien/ Materialien | In den Parcoursaufgaben werden die roten Felder nicht mehr zur Visuali-sierung von Mauern eingesetzt. | Unterschiedliche Bedeutung der roten Felder führte zu Irritationen bei den SuS (Durchführung). |
| | | Abschnitt „Wie?“ auf der Planungsvorlage für die eigenen Programmier-projekte der SuS wird angepasst. | Wenige SuS konnten die Frage beantworten (Auswertung Planungsvorlagen). |

| | | | |
|---|---|---|---|
| Zyklus 2 | Fachinhalte | Visuelle Impulse für den thematischen Einstieg werden angefertigt, die Programme in der Umgebung der SuS darstellen. So kann direkt ein Bezug zu deren Alltag hergestellt werden. | Brainstorming verlief teilweise sehr schleppend (Durchführung). |
| | Medien/ Materialien | Bildliche Anleitung zum Basteln des Namensschilds wird neu angefertigt. Diese sollte weniger Schritte umfassen und unterschiedlichere Begrifflichkeiten enthalten. | Beschreibung des bildlichen Ablaufs dauerte sehr lange und die Anweisungen waren insgesamt sehr ähnlich (Durchführung). |
| | | Die Parcoursaufgaben werden angepasst, sodass die Verwendung der algorithmischen Strukturen für die SuS ersichtlicher wird und somit thematisiert werden kann. | Die SuS konnten einige Parcoursaufgaben nur mit Hilfestellungen lösen (Durchführung).<br><br>Nur 56% der SuS waren in der Lage, eine bedingte Anweisung in ihrem Programmierprojekt zu verwenden (Analyse Scratch-Projekte). |
| | | Für die Arbeitsanweisungen des Lernzirkels wird ein handlicheres Format gewählt. | Arbeitsanweisungen des Lernzirkels waren für die SuS unhandlich und wurden durch die vielen Arbeitsschritte unübersichtlich (Durchführung). |

| | | | |
|---|---|---|---|
| Zyklus 3 | Fachinhalte | Im Rahmen der Arbeit an den Programmier-projekten kann thematisiert werden, wann bzw. wie oft man ein Programm ausführen sollte. | SuS löschen Skripte, ohne sie jemals ausgeführt zu haben (Analyse Programmierprozesse). |
| | Medien/ Materialien | Visuelle Impulse für den thematischen Einstieg sollten angepasst werden, dass Bilder bzw. Themen aus dem Alltag von Mädchen vorkommen. | Jungen beteiligen sich mit fast doppelt so vielen Äußerungen als Mädchen (Analyse Brainstorming). |

*Tabelle 16.2 Übersicht der Änderungen an den einzelnen Schritten der Unterrichtssequenz*

| Schritte der Unterrichtssequenz | Änderungen (Zyklus) | | | |
|---|---|---|---|---|
| | Intentionen | Fachinhalte | Methoden | Medien/Materialen |
| Thematischer Einstieg | | x (Z2) | | x (Z3) |
| Grundlagen von Algorithmen | | | x (Z1) | x (Z2) |
| Programmieren unplugged | | | | x (Z1, Z2) |
| Programmieren in Scratch | | | | x (Z2) |
| Eigene Programmierprojekte | | x (Z3) | x (Z1) | x (Z1) |

# 17 Erprobung der Unterrichtssequenz im Grundschulunterricht

Um die Praxistauglichkeit der Unterrichtssequenz sowie deren Auswirkungen und Gelingensbedingungen zu untersuchen, wurde ein weiterer Erprobungszyklus im Rahmen eines offiziellen Pilotversuchs an bayerischen Grundschulen durchgeführt (siehe Abbildung A.1). Dieser fand im Rahmen des Projekts *AlgoKids – Algorithmen für Kinder* statt, einem Kooperationsprojekt zwischen der Professur für Didaktik der Informatik und dem Staatsministerium für Unterricht und Kultus (siehe Abschnitt 8.4), das von der Autorin geleitet wurde. Innerhalb des Projekts wurde die Sequenz bzw. Elemente daraus von insgesamt 41 Grundschullehrkräften erprobt. Währenddessen wurden von der Autorin verschiedene Daten gesammelt (siehe Abschnitt 8.5), die in drei Teilstudien ausgewertet werden.

## 17.1 Zyklus 4

Die Unterrichtssequenz wurde von Mai 2018 bis November 2019 im Rahmen des Projekts *AlgoKids – Algorithmen für Kinder* von 41 Grundschullehrkräften an ihren jeweiligen Schulen erprobt. Ab Februar 2018 konnten sich alle staatlichen Grundschulen im Rahmen einer bayernweiten Ausschreibung des Bayerischen Staatsministeriums für Unterricht und Kultus für die Teilnahme am Projekt bewerben. Die Auswahl der Schulen erfolgte durch das Ministerium unter Berücksichtigung der für Schulversuche üblichen Repräsentativitätskriterien, wie ihrer Größe und Lage (einzügig-mehrzügig; Stadt-Land; Einbezug aller Regierungsbezirke, siehe Abbildung 17.1), sowie ihren Erfahrungen im Bereich der digitalen Bildung und ihrer digitalen Ausstattung. Laut Staatsministerium gab es zahlreiche Bewerbungen, die das zur Verfügung stehende Platzkontingent von 20 Projektschulen deutlich überstiegen – genaue Angaben liegen der Autorin jedoch nicht vor. Die ausgewählten Schulen wurden im April 2018 vom Ministerium über ihre Zulassung informiert und aufgefordert, jeweils zwei Lehrkräfte für die erste Fortbildungsveranstaltung anzumelden. Insgesamt 40 Lehrkräfte wurden somit von der der Autorin fachlich betreut – eine weitere Lehrkraft kam im Laufe des Projekts hinzu. Die Lehrkräfte wurden für die Teilnahme an den Fortbildungsveranstaltungen freigestellt. Insgesamt nahmen 39 weibliche und zwei männliche Lehrkräfte am Projekt teil. Das Alter der Lehrkräfte bewegte sich zum Projektstart zwischen unter dreißig Jahren und über fünfzig Jahren; die Initiative zur Teilnahme am Projekt war relativ gleichmäßig zwischen den Lehrkräften, ihrer jeweiligen Schulleitung oder beiden verteilt[1] (siehe Tabelle 17.1). Die Berufserfahrung der Lehrkräfte reichte von drei bis 39 Jahren, wobei fast die Hälfte der Teilnehmenden angab, über drei

[1] Von zwei Lehrkräften liegen hierzu keine Daten vor.

bis zehn Jahre Berufserfahrung zu verfügen. Über die Hälfte der Lehrkräfte (n=27) gab zu Beginn an, überhaupt keine Vorkenntnisse in Informatik zu besitzen, dreizehn hatten Informatik zumindest vorübergehend als Wahl- oder Pflichtfach in der Schule.


*Abbildung 17.1 Übersicht über die Verteilung der Projektschulen auf Bayern und seine Regierungsbezirke.*

*Tabelle 17.1 Verteilung des Alters und der Initiative zur Projektteilnahme der teilnehmenden Lehrkräfte*

| Alter | Anzahl der Lehrkräfte | Initiative zur Teilnahme (eigene / der Schulleitung / gleichmäßig verteilt) |
|---|---|---|
| unter 30 Jahre | 8 | 2 / 1 / 5 |
| 30 bis 40 Jahre | 17 | 5 / 6 / 5 |
| 41 bis 50 Jahre | 9 | 1 / 3 / 4 |
| über 50 Jahre | 7 | 4 / 2 / 1 |
| | 41 | 12 / 12 / 15 |

### 17.1.1 Anpassungen in der Unterrichtssequenz

Entsprechend der Erkenntnisse des dritten Zyklus und des neuen Erprobungssettings wurde die Unterrichtssequenz im vierten Zyklus wie folgt angepasst:

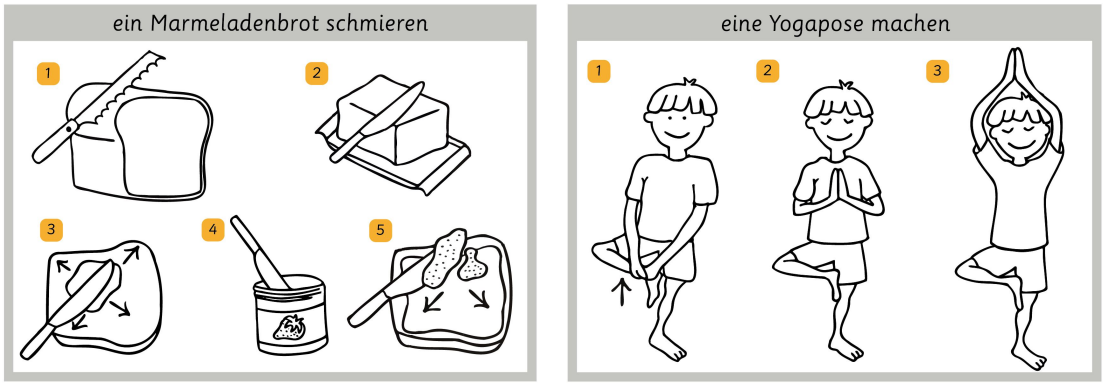
- Mit der Absicht, den Schülerinnen und Schülern in der ersten Phase der Unterrichtssequenz weitere Anregungen zur Verortung der Programmierung in ihrem Alltag zu geben, wurden die bestehenden Bilder durch ein Poster der *Stiftung Haus der kleinen Forscher* ergänzt (siehe Abbildung 17.2). Dieses wurde von der Stiftung als Beilage zu ihrer Zeitschrift „Forscht mit!“[2] veröffentlicht.
- Da die Grundschullehrkräfte ihre Schülerinnen und Schüler kennen – im Gegensatz zu den Kursleiterinnen und -leitern in früheren Zyklen – ist das Basteln eines Namensschildes zu Beginn der Unterrichtssequenz nicht sehr naheliegend. Daher wurden zwei neue bildliche Anleitungen angefertigt (siehe Abbildung 17.3).


*Abbildung 17.2 Wimmelbild: Wo versteckt sich ein Informatiksystem/Programmierung?*

### 17.1.2 Durchführung

Die Lehrkräfte wurden in drei mehrtägigen Veranstaltungen nach dem Konzept der Autorin fortgebildet (siehe Abschnitt 15.3 und Kapitel 18). Bereits nach der ersten Fortbildungsveranstaltung im Mai bzw. Juni 2018 erprobten die Lehrkräfte die Unterrichtssequenz erstmals mit ihren Schülerinnen

[2] https://www.haus-der-kleinen-forscher.de/de/praxisanregungen/begleitende-materialien/magazin-forscht-mit




*(a)* *(b)*

*Abbildung 17.3 Bildliche Anleitung zum (a) Schmieren eines Marmeladenbrots und (b) Einnehmen einer Yogapose aus Zyklus 3.*

und Schülern. Dabei waren sie nicht auf ein bestimmtes Umsetzungsformat beschränkt. Sie sollten die Unterrichtssequenz so in ihren Unterrichtsalltag einbauen, wie es ihnen in ihrem jeweiligen Schulkontext am erfolgversprechendsten erschien. Die Implementierung konnte dazu in weiten Grenzen skaliert werden, z. B. als mehrtägiger, zusammenhängender Block wie in den Erprobungen an der TUM, innerhalb einzelner Tage oder verteilt auf ein bis zwei Wochenstunden über ein halbes oder ganzes Schuljahr. Die Lehrkräfte konnten sich jederzeit telefonisch oder per Mail an die Autorin wenden, um offene Fragen zu klären oder Feedback zu ihren eigenen Unterrichtsideen einzuholen. Für alle in der ursprünglichen Unterrichtssequenz benötigten Materialien wurden ihnen Druckvorlagen und die entsprechenden Scratch-Programme zur Verfügung gestellt. Darüber hinaus erhielten alle Lehrkräfte einen Satz haptischer Programmierbefehle sowie eine Filzmatte, auf der sie mit den Befehlen programmieren konnten. Die TUM stellte zudem ein mobiles Set von zwanzig Tablet-Computern zur Verfügung, das die Projektschulen in vier Fällen ausgeliehen haben.

## 17.2 Teilstudie 1: Unterrichtssequenz

Ziel der Studie ist es, die Praxistauglichkeit der Unterrichtssequenz sowie mögliche Anpassungen oder Erweiterungen durch die Lehrkräfte zu untersuchen:

**(RQ 2-d)** Wie implementieren Lehrkräfte der Grundschule das Unterrichtskonzept in der Praxis, wandeln es ab oder erweitern es?

Für die empirische Untersuchung werden neunzehn explorative Interviews analysiert, die mit insgesamt 32 Lehrkräften im Laufe der Erprobung der Unterrichtssequenz geführt wurden. Ergänzt werden sie durch Unterrichtsmaterialien, welche die Lehrkräfte zum Unterrichten des Programmierens angefertigt hatten. Die Datenerhebung wird in Kapitel 8.5 detailliert beschrieben. Der Ablauf des explorativen Interviews kann in Anhang A.2 eingesehen werden. Folgende Teilfragen werden untersucht:

- (RQ 2-d-1) In welchem Format implementieren die Lehrkräfte die Unterrichtssequenz?
- (RQ 2-d-2) Welche Erfahrungen machen die Lehrkräfte bei der Erprobung der Unterrichtssequenz?
- (RQ 2-d-3) Inwiefern wandeln die Lehrkräfte bestehende Elemente der Unterrichtssequenz ab?
- (RQ 2-d-4) Inwiefern erweitern die Lehrkräfte die Unterrichtssequenz um zusätzliche Elemente?

### 17.2.1 Methodik

Als Auswertungsmethode wurde die inhaltlich strukturierende qualitative Inhaltsanalyse nach Kuckartz (2016, S. 97 ff.) gewählt (siehe Abschnitt 8.6.1). Die deduktiv-induktive Vorgehensweise wurde wie folgt umgesetzt: Nach der Sichtung zweier Transkripte wurde aus den Forschungsfragen für den ersten Codierprozess verschiedene Hauptkategorien und Subkategorien abgeleitet und definiert (siehe Tabelle 17.2 und 17.3). In einem nächsten Schritt wurde das Datenmaterial anhand des Kategoriensystems codiert. Die Codiereinheit wurde dabei so groß gewählt, dass der jeweilige Inhalt auch ohne weiteren Kontext zu verstehen war. Jede Codierung entspricht einer Aussage zu einer Haupt- bzw. Subkategorie, was dazu führt, dass in einem einzelnen Interviews mehrere Codiereinheiten der gleichen Kategorie zugewiesen werden können. Zur Codierung der Interviews und Unterrichtsmaterialien wurde MAXQDA verwendet, eine Software zur computergestützten qualitativen und *Mixed-Methods*-Datenanalyse[3].

Nach der Codierung eines Viertels des Datenmaterials (fünf Interviews und 18 Unterrichtsmaterialien) wurde überprüft, ob das Kategoriensystem angepasst werden musste, was der Fall war (siehe Tabelle 17.3, rote Markierung). Im Anschluss wurden sämtliche Textstellen und Bilddateien, die in einer Subkategorie der Hauptkategorie „Unterrichtssequenz“ codiert wurden, zwei neuen Hauptkategorien zugewiesen: (a) *Umsetzung der Elemente der Unterrichtssequenz* mit den Subkategorien *unverändert* und *modifiziert* sowie (b) *Feedback* mit den Subkategorien *positiv*, *neutral* und *negativ*. Feedback bezieht sich hier zum einen auf generelle Äußerungen über das Unterrichtselement, zum anderen auf Äußerungen, wie der Einsatz des Elements im Unterricht verlief. Die codierten Segmente der Hauptkategorie *Format der Umsetzung* wurden zusammengestellt und am Material codiert. Sämtliche Textstellen und Bilddateien der Hauptkategorien *Erweiterung der Unterrichtssequenz* und *Format der Umsetzung* wurden ebenfalls zusammengestellt und am Material codiert. Da es aus terminlichen Gründen nicht möglich war, mit drei Testschulen ein Interview zu führen, wurden sie per Mail gefragt, in welchem Format sie die Unterrichtssequenz getestet haben, um die Daten zu vervollständigen. Die Ergebnisse der qualitativen Inhaltsanalyse werden ergänzt durch ausgewählte Items des Fragebogen FB-LK4 (siehe Abschnitt A.3.4), den die Lehrkräfte nach der dritten Fortbildungsveranstaltung ausgefüllt haben.

[3] https://www.maxqda.com/de/

*Tabelle 17.2 Deduktive Kategorienbeschreibung zu Forschungsfrage RQ 2-d*

| Name | Beschreibung | Beispiel |
|---|---|---|
| Unterrichtssequenz (Element 1-11) | Äußerungen, die sich auf Element 1-11 der Unterrichtssequenz beziehen oder Materialien für ebendieses. | „Und dann diese Aufgaben mit dem Marmeladenbrot, mit den Schritten, [...] da haben wir an der Tafel glaube ich auch festgehalten, wie viele Schritte die unterschiedlichen Gruppen hatten. Sind dann eben auf das eingegangen, dass man das alles nicht voraussetzen darf. Dass man wirklich jede Kleinigkeit festhalten muss.“ (S13-1_20190716: Pos. 18) |
| Format der Umsetzung | Äußerungen, welche die formale Umsetzung der Unterrichtssequenz an der Schule beschreiben. | „Wir waren ja jetzt zu zweit bei mir drin. Da war das jetzt echt ok. Aber wenn ich mir denke, ich wäre jetzt alleine gewesen, da hätte ich es echt schwierig gefunden.“ (S2-1_20190508: Pos. 92)<br><br>„Ich hab’s jetzt nicht im Block gemacht das Ganze, in drei Tagen. Sondern ich hab’s immer wieder in einzelnen Unterrichtsstunden vorgenommen.“ (S9-1_20190718: Pos. 10) |
| Erweiterung der Unterrichtssequenz | Äußerungen, die eine Lehr-Lernhandlung beschreiben, die nicht Teil der Unterrichtssequenz ist, oder Materialien für ebendiese. | „Und ich habe mir einen Cozmo gekauft und ich habe jetzt auch einen Cozmo mit in der AG. Und Kinder, wenn sie jetzt das Pflichtprogramm gemacht hatten, sind wir dann ja langsam durch. Dann können sie an eigenen Projekten arbeiten. Oder eben abwechselnd alle drei Wochen dann auch mit Cozmo programmieren.“ (S15-1_20190529: Pos. 56) |

### 17.2.2 Ergebnisse

Die Ergebnisse der qualitativen Inhaltsanalyse werden im Folgenden anhand der einzelnen Forschungsfragen der Teilstudie dargestellt.

*(RQ 2-d-1) In welchem Format implementieren die Lehrkräfte die Unterrichtssequenz?*

Die Unterrichtssequenz wurde an den Schulen mit einer Ausnahme mit Schülerinnen und Schülern der Klassenstufe vier erprobt (n=19[4]). Zwei Drittel der Schulen arbeiteten – größtenteils zusätzlich – mit Kindern der Klassenstufe drei (n=13). Die Mehrheit der Schulen implementierte die Unterrichtssequenz im Klassenverband (n=17), ein Viertel der Schulen bot teilweise zusätzlich eine Arbeitsgemeinschaft (AG) an (n=5), zu der zwischen sechs und sechzehn Kinder zugelassen wurden. An einer Schule wurden Elemente der Sequenz lediglich punktuell mit einzelnen Kindern ausprobiert.

[4]An dieser Stelle entsprechen die Zahlenwerte von *n* nicht der Anzahl codierten Aussagen, sondern der Anzahl der Schulen. Der Maximalwert ist demnach *20*.

*Tabelle 17.3 Kategoriensystem für ersten Codierprozess (Anpassungen rot markiert) und Anzahl der vergebenen Codes.*

| Hauptkategorie | Zwischenebene | Subkategorie (erster Entwurf) | Subkategorie (angepasst) |
|---|---|---|---|
| Unterrichts-sequenz | Thematischer Einstieg | Brainstorming | Aktivierung Vorwissen (10) |
| | | Programmierung im Alltag | |
| | Grundlagen von Algorithmen und Programmen | *Lehrer-Roboter* | *Lehrer-Roboter* (21) |
| | | Bildliche Anweisungen umwandeln | Bildliche Anweisungen umwandeln (11) |
| | Programmieren unplugged | Parcoursaufgaben bearbeiten | Parcoursaufgaben bearbeiten und ablaufen (59) |
| | | Parcoursaufgaben ablaufen | |
| | Programmieren in Scratch | Einführung Scratch | Einführung Scratch (10) |
| | | Lernzirkel Scratch | Lernzirkel Scratch (43) |
| | Eigene Programmier-projekte | Planen von Programmen | Planen von Programmen (13) |
| | | Programmideen umsetzen | Programmideen umsetzen (27) |
| | | Programme präsentieren | Programme präsentieren (6) |
| Format der Umsetzung (31) | | | |
| Erweiterung der Unterrichts-sequenz (89) | | | |

Weiterhin ist anzumerken, dass in sieben Interviews berichtet wurde, dass die Unterrichtssequenz oder zumindest die Elemente, in denen in Scratch programmiert wird, innerhalb der Schule als *Teamteaching* durchgeführt wurden (d. h. die Lehrkräfte unterrichteten gemeinsam). Die Ergebnisse des Fragebogens zeigen, dass mehr als die Hälfte der Lehrkräfte bei der Planung und Umsetzung des Programmierens im Unterricht zusammengearbeitet hat (siehe Abbildung 17.4). Begründet wurde dies in erster Linie damit, dass die Lehrkräfte dadurch mehr Sicherheit im Umgang mit der technischen Ausstattung und dem Lösen von Problemen/Fragen in Scratch bekamen – vor allem, wenn sie nicht die Möglichkeit hatten, mit ihrer Klasse über einen längeren Zeitraum zu programmieren. Zudem empfanden einige Lehrkräfte die Umsetzung des Programmierens alleine als anstrengend:

> *Lehrkraft 1:* „Aber insgesamt hätte ich mir das, glaube ich, letztes Jahr schwieriger vorgestellt. Dass es die Kinder nicht so gut selber hinkriegen und dass wir dann noch viel mehr haben. Also am Anfang war es glaube ich wirklich, weil man sich selber orientieren musste. Gold war, dass wir zu zweit waren für die 16 Kinder. Ich meine, das ist schon ein Oberluxusschlüssel. Das kriegen wir natürlich sonst in keiner AG. [...]“
> *Lehrkraft 2:* „Aber ich glaube das braucht's schon. Also mit 16 alleine, ich hab's ja vorhin schon gesagt. Also das ist dann schon anstrengend.“
> *Lehrkraft 1:* „Wenn mal wirklich einer von uns krank war oder so, dann war es schon hart.“ S6-1_20190527: Pos. 299-301

*(RQ 2-d-2) Welche Erfahrungen machen die Lehrkräfte bei der Erprobung der Unterrichtssequenz?*
In den Interviews konnte eine Vielzahl von Textsegmenten identifiziert werden, in denen sich die Lehrkräfte positiv oder negativ über Elemente der Unterrichtssequenz bzw. deren Erprobung mit Schülerinnen und Schülern äußerten (siehe Abbildung 17.5 und 17.6, Spalten 2 und 4). Diese werden im Folgenden anhand des Ablaufs der Unterrichtssequenz zusammenfassend dargestellt.

Das Programmieren des *Lehrer-Roboters* zu Beginn der Unterrichtseinheit wurde als sehr motivierend für die Schülerinnen und Schülern beschrieben, sie arbeiteten sehr konzentriert und der Lerneffekt war nach Ansicht der Lehrkräfte groß. Der Umgang mit den Winkelwerten im Rahmen der Drehung des Roboters stellte keine Probleme da. Viele Lehrkräfte berichteten, dass der Einstieg großen Spaß bereitete:

> „Und wie gesagt, ich glaube auch gerade dieser Anfang mit dem Lehrerroboter. Den Lehrer zu programmieren, das ist schon gut angekommen. Ganz viel Lachen und da war gleich schon eine positive Atmosphäre eigentlich. War ein guter Einstieg."
>
> S13-1_20190716: Pos. 80

Auch das Umwandeln der bildlichen Anleitungen in sprachliche Befehle schien den Schülerinnen und Schülern großen Spaß zu machen. Das Programmieren *unplugged* im Rahmen der Parcoursaufgaben wurde als sehr gut umsetzbar in Partner- oder Gruppenarbeit und sehr effektiv beschrieben. Für die meisten Lehrkräfte stellte der Teil ein unverzichtbarer, wichtiger Schritt in der Unterrichtssequenz dar, da er die Arbeit in Scratch vorentlastete und die Schülerinnen und Schüler sich sehr intensiv mit ihren Programmen auseinandersetzten:

> „Und da hatte ich auch den Eindruck, dass ihnen das die Angst, oder den Respekt vor dem Programm ein bisschen genommen hat. Weil sie die Blöcke erstens kannten und wussten, wie der aussieht. Und wussten eigentlich, was der macht."
>
> S15-1_20190529: Pos. 26

> „Wenn ein Fehler da ist, ist er nicht so leicht wieder wegzuschieben, wie am Computer. [...] ich glaube, man muss sich intensiver damit auseinandersetzen nochmal."
>
> S8-1_20191130: Pos. 171

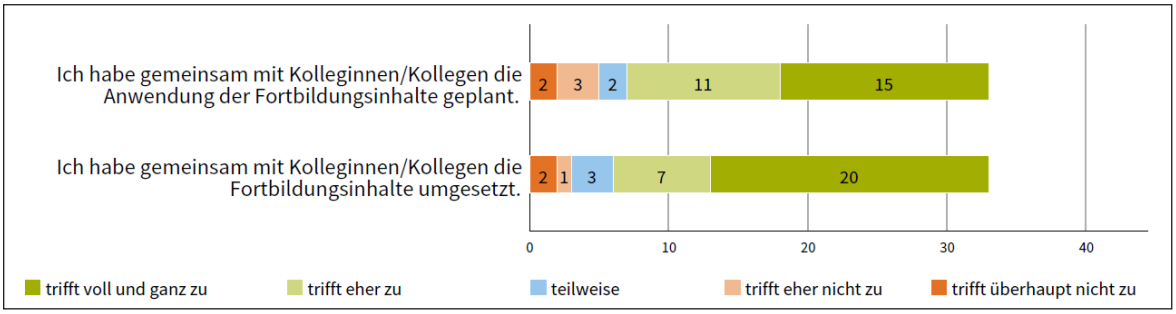



*Abbildung 17.4 Angaben zur Umsetzung der Fortbildungsinhalte mit Kolleginnen und Kollegen (n=33; FB-LK4)*

| Codesystem | positiv | neutral | negativ | unverändert | modifiziert |
|---|---|---|---|---|---|
| Aktivierung Vorwissen | 1 | 5 | 2 | 9 | 1 |
| Lehrer-Roboter | 11 | 3 | 1 | 10 | 11 |
| Bildliche Anweisungen umwandeln | 6 | 3 | | 4 | 8 |
| Parcoursaufgaben bearbeiten und ablaufen | 20 | 5 | 11 | 24 | 35 |
| Einführung Scratch | 4 | 4 | | 4 | 7 |
| Lernzirkel Scratch | 23 | 14 | 12 | 28 | 16 |
| Planen von Programmideen | 1 | 1 | 7 | 9 | 4 |
| Programmideen umsetzen | 11 | 5 | 13 | 20 | 7 |
| Programme präsentieren | 6 | | | 5 | 1 |

*Abbildung 17.5 Numerische Code-Relation-Darstellung über alle Transkripte und Unterrichtsmaterialien*

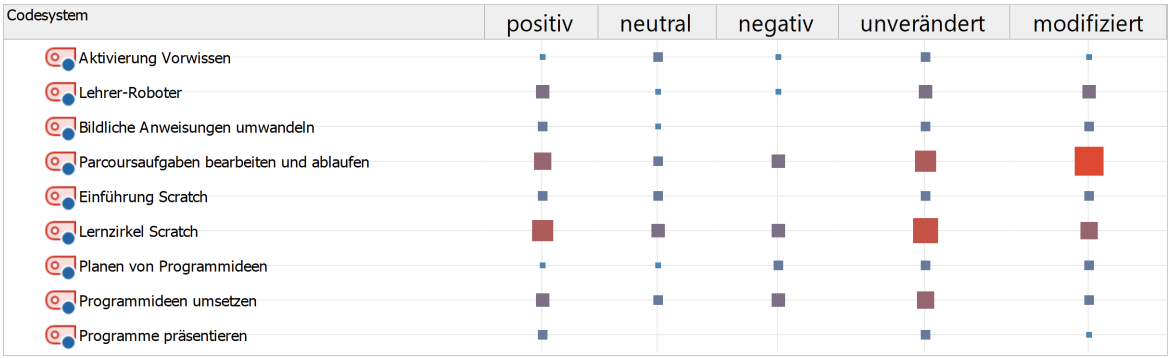



*Abbildung 17.6 Grafische Code-Relation-Darstellung über alle Transkripte und Unterrichtsmaterialien*

Gleichzeitig berichteten manche Lehrkräfte, dass die Schülerinnen und Schüler bei der Lösung der Parcoursaufgabe zur bedingten Anweisung Probleme hatten und man sich hier eine leichtere Aufgabe gewünscht hätte. Zudem kam es beim Ablaufen der Programme im Parcours zum Teil zu Verzögerungen. Der Lernzirkel zu Scratch hat sich in der Umsetzung an den meisten Schulen bewährt: die Materialien wurden als sehr gut strukturiert, für die Schülerinnen und Schüler gut nachvollziehbar und ihrem Leistungsniveau entsprechend beschrieben. Es fiel den Lehrkräften leicht, auf die Fragen der Schülerinnen und Schüler zu antworten und diese schienen großen Spaß zu haben. Das Programmieren im Team hat sich bewährt:

> „Ich habe dann manchmal gestaunt [...] wie schnell die die Aufgaben wirklich richtig gelöst haben. Das war wirklich erstaunlich. Und es handelt sich [dabei] um eine Klasse, die normalerweise Schwierigkeiten bei so Sachen hat, beim Zusammenarbeiten. Da stehen sich oft bei Arbeitszeit in Gruppen [...] die Egoismen gegenüber und es ist ein ‚ich und ich und ich'. Das war weg. Ich habe wirklich nur noch gestaunt. Das war weg. Die haben wunderbar zusammengearbeitet." S17-1_20180723: Pos. 9

Gleichzeitig wurde jedoch auch beobachtet, dass sich einzelne Schülerinnen und Schüler beim Programmieren im Team oder zu dritt zurücknahmen. Zudem berichteten einige Lehrkräfte, dass sich die Fähigkeiten der Schülerinnen und Schüler im Umgang mit den technischen Geräten sehr unterschieden. Von einer Lehrkraft wurde berichtet, dass die Schülerinnen und Schüler im Laufe des Lernzirkels den Wunsch äußerten, freier zu arbeiten. Zum Planen eigener Programmideen

äußerten verschiedene Lehrkräfte, dass es für die Schülerinnen und Schüler sehr schwer war und diese lieber direkt mit dem Programmieren beginnen wollten. Die geplanten Handlungen waren teilweise zu hoch gegriffen und konnten nur schwer in einzelne Schritte zerlegt werden:

> *Lehrkraft 1:* „Und wenn es zum ersten eigenen Projekt ging [...] da waren schnell mal zehn Finger oben. "
> *Lehrkraft 2:* „Da war die Vorstellung und das Können einfach komplett auseinandergelaufen. Oder die Kinder konnten einfach das, was sie sich vorgestellt haben, das ging einfach noch nicht. So weit waren die dann noch nicht." S6-1_20190527: Pos. 22-23

Das Umsetzen eigener Programmideen fiel den Schülerinnen und Schüler teilweise schwer – vor allem im Vergleich zu der Arbeit im Lernzirkel. Sie gaben teilweise sehr schnell auf und verwarfen ihre ursprüngliche Programmidee. Zudem beschäftigten sie sich mitunter sehr lange mit den Figuren und Hintergründen in Scratch oder entdeckten neue Funktionen, die sie von ihrer ursprünglichen Idee ablenkten. Die Lehrkräfte waren zudem vermehrt in der Situation, dass sie auf die Fragen der Schülerinnen und Schüler nicht direkt eine Antwort geben konnten:

> „Also das war halt teilweise für mich dann schon schwierig, da jetzt einen Fehler zu finden, oder so. Da habe ich mich dann selber reindenken müssen. Oder wir waren dann beide [...] dagestanden: Wo ist jetzt da der Fehler? Warum funktioniert [das nicht]? Also das war dann für mich auch interessant. Und für die Kinder toll, dass sie dann auch selber was rausgefunden haben, was ich jetzt nicht gleich entdeckt habe [...] Und das war für die Kinder toll. [...] Aber ich find's dann auch OK, wenn man zu den Kindern sagt ‚Mein Gott, dann müssen wir jetzt schauen. Da müssen wir jetzt miteinander überlegen. Ich weiß das jetzt auch nicht gleich.'" S1-1_20190515: Pos. 76

Einige Lehrkräfte berichteten jedoch auch davon, dass die Schülerinnen und Schüler sich viel gegenseitig halfen und es durch das Programmieren im Team relativ wenige Meldungen gab. Zudem äußerten sich die meisten Lehrkräfte positiv über die tollen Ergebnisse sowie Ideenvielfalt und Spaß der Schülerinnen und Schüler. Bei der Präsentation ihrer Programme zeigten diese sich sehr stolz auf ihre Ergebnisse und interessierten sich teilweise sehr für die Programme ihrer Mitschülerinnen und Mitschüler:

> „In meiner Klasse war es dann so, wir haben am Schluss in der letzten Stunde gemeinsam alle Projekte angeschaut und die, die das Programm gemacht haben, haben gesagt ‚was war schwer/was war leicht' und die anderen Kinder durften dazu noch Fragen stellen. Ich habe mich dann völlig rausgehalten. Hat wunderbar funktioniert. Und es war erstaunlich, wer was gefragt hat. Kinder, von denen ich das nie erwartet hätte. Wie zum Beispiel die [Anna], die ist immer ganz piepsig, und die hat dann wirklich gezielt [...] nach einzelnen Programmierschritten nachgefragt. [...] Dann haben sie das Programm eingeblendet von ihrem Projekt, wo man dann schön alles zeigen konnte." S17-1_20180723: Pos. 16

*(RQ 2-d-3) Inwiefern wandeln die Lehrkräfte bestehende Elemente der Unterrichtssequenz ab?*
Die Mehrheit der Lehrkräfte gab im Fragebogen FB-LK4 an, die Fortbildungsinhalte ohne größere Änderungen angewandt zu haben. Allerdings gab nur die Hälfte der Befragten an, dass sie die Inhalte in ihre bestehenden schulischen Aktivitäten integrieren konnten. Über die Hälfte der befragten Lehrkräfte gab an, die Fortbildungsinhalte[5] angepasst, weiterentwickelt oder sogar eigene Ideen dazu entwickelt zu haben (siehe Abbildung 17.7). Ein ähnliches Bild zeigt sich hinsichtlich der Materialien aus der Fortbildung: alle befragten Lehrkräfte gaben an, zumindest einen Teil des Materials im Unterricht verwendet zu haben. Etwa dreiviertel gaben an, es inhaltlich oder methodisch weiterentwickelt zu haben (siehe Abbildung 17.8).

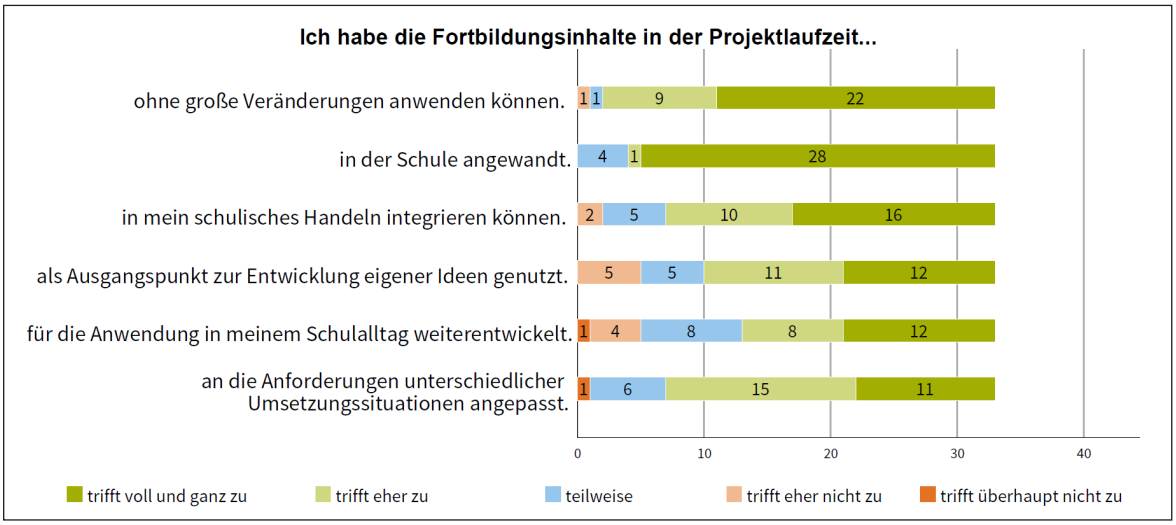



*Abbildung 17.7 Angaben zur Umsetzung der Fortbildungsinhalte (n=33; FB-LK4)*

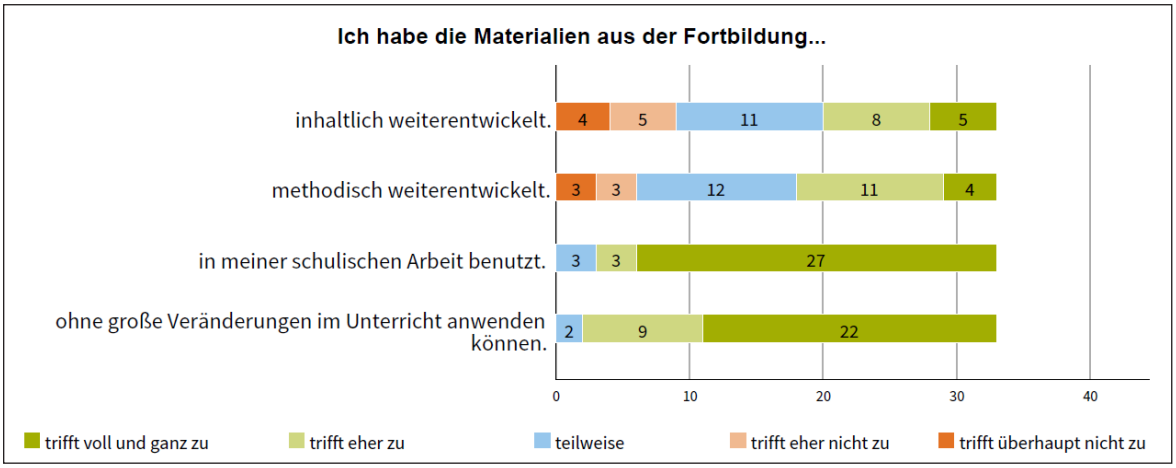
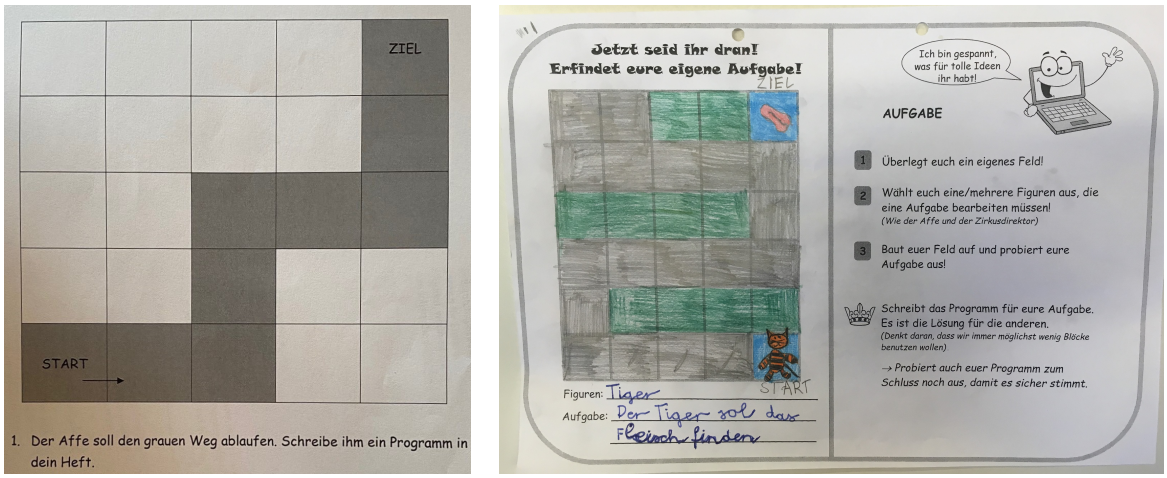



*Abbildung 17.8 Angaben zur Verwendung und Anpassung der Fortbildungsmaterialien (n=33; FB-LK4)*

Die Interviews zeigen, dass alle Elemente der Unterrichtssequenz modifiziert wurden, wobei es hier starke Unterschiede in den Häufigkeiten gibt (siehe Abbildung 17.5 und 17.6, rechte Spalte). Die Abwandlungen der bestehenden Elemente der Unterrichtssequenz werden im Folgenden dem Ablauf der Sequenz folgend zusammengefasst.

[5]Unter *Fortbildungsinhalte* fallen sämtliche Strukturelemente des Unterrichtens der Programmierung in der Grundschule: Intentionen, Fachinhalte, Methoden, Medien/Materialien.




*(a)* *(b)*

*Abbildung 17.9* *Zusätzliche Parcoursaufgaben: (a) Variation einer bestehenden Aufgabe und (b) Schülerinnen und Schüler erarbeiten eigene Aufgaben.*

Beim Programmieren des *Lehrer-Roboters* arbeiteten einige Lehrkräfte bereits mit den haptischen Scratch-Blöcken, z. B. um ein Programm für die Lehrkraft zu erstellen oder um ein Programm für die Schülerinnen und Schüler vorzugeben. Außerdem sprachen die meisten Lehrkräfte an dieser Stelle die Winkelwerte an, die während der gesamten Sequenz vorkamen (90 Grad, 180 Grad). Einzelne Lehrkräfte ließen die Schülerinnen und Schüler hier bereits die Anweisungen für den Roboter aufschreiben. Für die Umwandlung der bildlichen Anleitungen in sprachliche Befehle haben einige Lehrkräfte Anweisungen auf Papierstreifen vorbereitet, welche die Schülerinnen und Schüler lediglich in die richtige Reihenfolge bringen mussten. Damit die sprachlichen Befehle besser auf ihre Richtigkeit hin kontrolliert werden können, brachten einige Lehrkräfte Utensilien mit, z. B. Toastbrot und Messer. Bezüglich des Bearbeitens und Ablaufens der Parcoursaufgaben wurden Veränderungen genannt: Hier erstellten die Lehrkräfte zusätzliche – zumeist leichtere – Aufgaben (siehe Abbildung 17.9); fertigten zum Erstellen der Programme haptische Scratch-Blöcke in einem kleineren Format an oder ließen die Schülerinnen und Schüler die Anweisungen aufschreiben; das Ablaufen der Lösungen erfolgte z. B. im Pausenhof, mit kleinen Spielfiguren auf Arbeitsblättern oder durch die Lehrkraft mit einer Spielfigur über die Dokumentenkamera (siehe Abbildung 17.10). Auch das Vergleichen der Lösungen der Schülerinnen und Schüler wurde von den Lehrkräften thematisiert (siehe Abbildung 17.11). Zur Einführung der Programmierumgebung Scratch überlegten sich einige Lehrkräfte einführende Arbeitsaufträge oder Quizfragen, welche die Schülerinnen und Schüler lösen mussten. Andere ließen sie die Programmierumgebung zunächst komplett frei erkunden. Eine Lehrkraft überführte die *Unplugged*-Parcoursaufgaben in Scratch:

> „Dieses Katze und Affe und Zaubertrick, das hatte ich ja noch in digitalisierter Form. [...] Und das war super, weil sie eigentlich dann von Scratch her nur die Befehle suchen mussten und die Aufgabe kein Problem mehr war. [...] Der ganze Denkweg war klar, und jetzt nur – wie setze ich es um? Und das war mega als Zwischenschritt, weil sie dann viel leichter weitermachen konnten." S3-2_20191124: Pos. 28-34

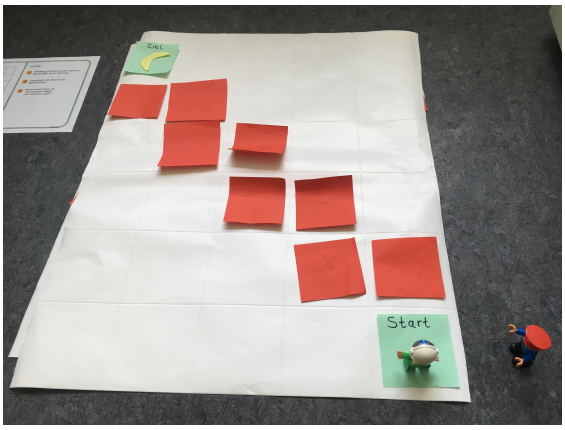



*Abbildung 17.10 Ablaufen der Lösungen mit Figuren*


*Abbildung 17.11 Vergleich der Programme an der Tafel*

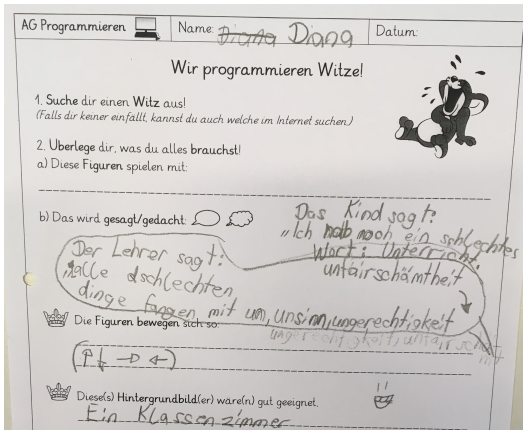
Den Lernzirkel zu Scratch wandelten die Lehrkräfte auf ganz unterschiedliche Weise ab. Zum einen gab es Lehrkräfte, die den Lernzirkel zunächst über den Beamer anleiteten. Zum anderen gab es Lehrkräfte, die ihren Schülerinnen und Schülern mehr Offenheit in der Umsetzung ließen, z. B. indem diese ihre Programme an den einzelnen Stationen erweitern und personalisieren konnten oder das Gelernte in kleinen Zwischenprojekten anwenden sollten. Einzelne Lehrkräfte wählten einzelne Stationen des Lernzirkel aus oder orientierten sich nur an den darin enthaltenen Lernschritten:

> „Ich habe es etwas anders umgesetzt. Habe mich zwar inhaltlich an die Karten gehalten – so von den Schritten – habe es aber immer unabhängig von den Karten gemacht. Ich habe immer einen gebundenen Teil gehabt, wo ich zum Beispiel gesagt habe ‚Heute ist das Ziel, eine Figur von links nach rechts zu bewegen'. Aber sie durften sich immer Hintergründe und alles frei dazu wählen, Figuren auch." S17-1_20180723: Pos. 12

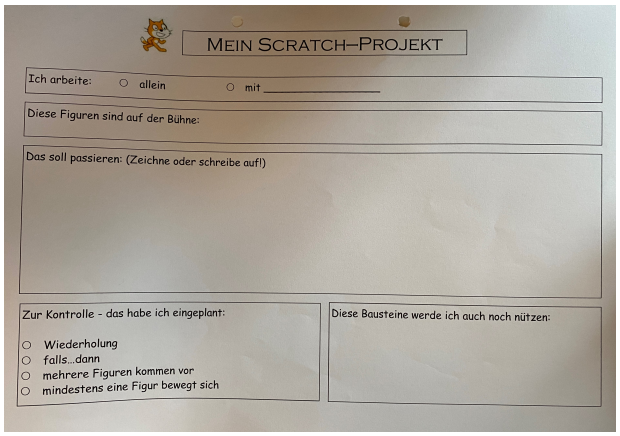
Einige Lehrkräfte reicherten die Stationen des Lernzirkels durch zusätzliche Aufgaben in Scratch oder *Unplugged*-Übungen mit den haptischen Scratch-Blöcken an. Zur Binnendifferenzierung ließ eine Lehrkraft einzelne leistungsschwächere Schülerinnen und Schüler mit fertigen Scratch-Programmen arbeiten, die sie zunächst nur anpassen sollten. Um besser mit den Fragen umgehen zu können, die während des Lernzirkels aufkamen, führte eine Lehrkraft ein, diese auf Karteikärtchen zu sammeln und danach im Plenum zu beantworten:

> „Ich habe meinen dann so Karteikarten gegeben. Die sollten aufschreiben irgendwie, welche Probleme sie hatten. Und dann haben wir halt irgendwann [...] unterbrochen. Und dann haben wir uns die Karten hergenommen und haben geklärt, was da denn los war. Und dadurch hast du einfach einen Sprachanlass." S2-1_20190508: Pos. 129

Zur Planung von Programmideen der Schülerinnen und Schüler fertigten einige Lehrkräfte neue Vorlagen an, von denen einige einen Handlungsrahmen vorgaben und andere inhaltlich frei gestaltet waren (siehe Abbildung 17.12). Für die Umsetzung eigener Programmideen gaben einige Lehrkräfte ihren Schülerinnen und Schülern als Vorgabe weitere Blöcke, die im Programm beinhaltet sein sollten, oder konkrete Handlungsbeschreibungen, die zunächst umgesetzt und im Anschluss erweitert werden sollten.




(a)




(b)

*Abbildung 17.12 Weitere Planungsvorlagen: (a) mit thematischer Vorgabe und (b) thematisch frei.*

*(RQ 2-d-4) Inwiefern erweitern die Lehrkräfte die Unterrichtssequenz um zusätzliche Elemente?*
Die Interviews zeigen, dass die Lehrkräfte die Unterrichtssequenz auf verschiedenste Weise erweitert haben (siehe Abbildung 17.13). Zur Strukturierung der Darstellung der Ergebnisse werden die vier Entscheidungsfelder des Berliner Modells herangezogen.

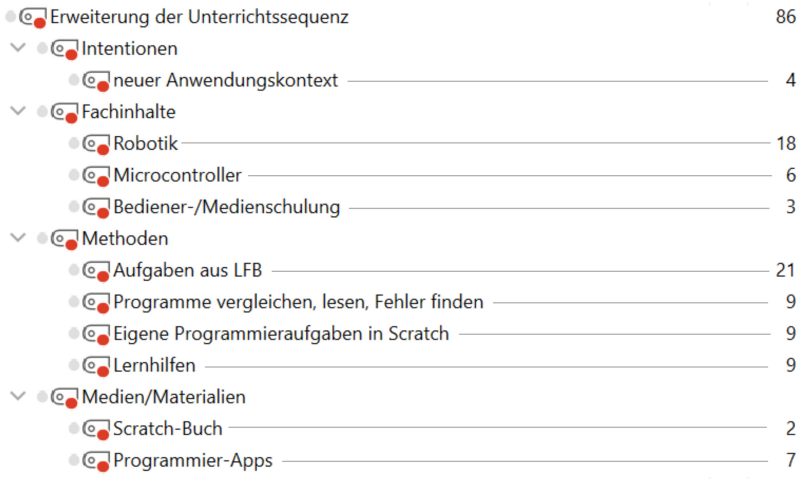



*Abbildung 17.13 Induktiv erstellte Subkategorien der Hauptkategorie „Erweiterung der Unterrichtssequenz“ und Anzahl der codierten Segmente*

Methoden

Viele Lehrkräfte reicherten die Unterrichtssequenz durch Aufgaben aus den Fortbildungen an, z. B. durch das Bilddiktat (siehe Beispiel LFB-DP4-5) oder unterschiedliche Programmieraufgaben in Scratch. Letztere wurden in erster Linie ausgewählt, da sie offener gestaltet waren als die Aufgaben aus dem Lernzirkel und von den Schülerinnen und Schülern individueller gelöst werden konnten. Einige Lehrkräfte entwickelten eigene Programmieraufgaben, die sich sehr unterschieden. Teilweise ergänzten sie den Lernzirkel oder wurden stattdessen eingesetzt, in einem anderen Fall wurde eine Projektaufgabe entwickelt, die arbeitsteilig von der ganzen Klasse bearbeitet wurde. Darin wurden einzelne Elemente eines Stadtrundgangs von jeweils einer Gruppe gestaltet und programmiert und im Anschluss von der Lehrkraft zu einem Programm zusammengefügt

(siehe Abbildung 17.14). Die Lehrkräfte erweiterten die Unterrichtssequenz zudem durch Aufgaben, in denen Programme verglichen, gelesen oder Fehler darin gefunden werden mussten (siehe Abbildung 17.16). Darüber hinaus wurde in den Interviews von unterschiedlichen Maßnahmen zur Unterstützung des Lernprozesses der Schülerinnen und Schüler berichtet. Diese bezogen sich z. B. ganz konkret auf einzelne Aufgaben (siehe Abbildung 17.18), die Orientierung in Scratch (siehe Abbildung 17.17) oder die Modellierung von Lösungen. In Bezug auf Letzteres wurden an einer Schule sehr gute Erfahrungen damit gemacht, die Lösungen von Aufgaben zunächst in Pseudocode zu beschreiben.


*Abbildung 17.14 Klassenprojekt in Scratch*

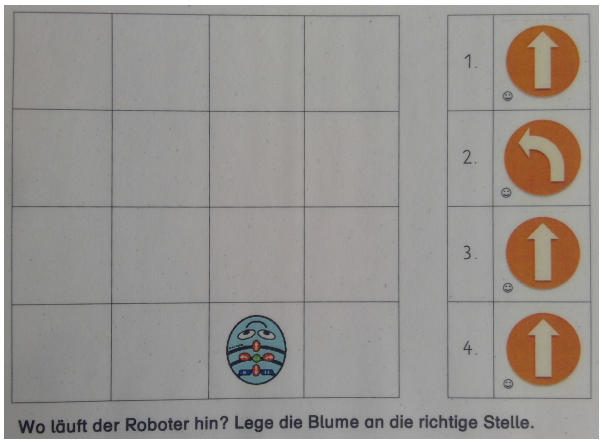



*Abbildung 17.15 Aufgabe für die Arbeit mit dem BlueBot*

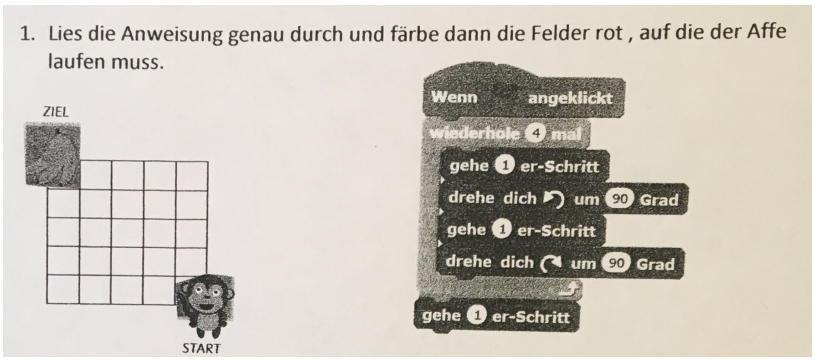


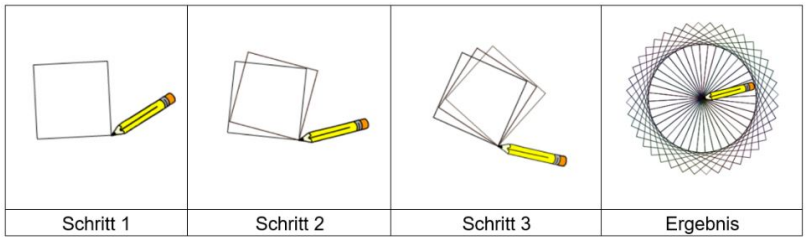

*Abbildung 17.16 Erstellte Aufgabe zum Lesen von Programmen: Der Weg des Affens muss von den Schülerinnen und Schülern eingezeichnet werden.*


*Abbildung 17.17 Blockbereiche in Scratch*

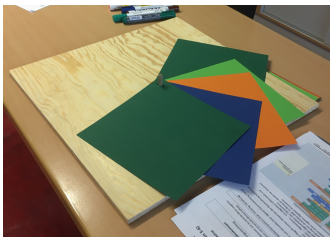
### Medien/Materialien

Zwei Lehrkräfte arbeiteten zur Differenzierung im Rahmen der Unterrichtssequenz mit einem Buch zur Programmierung in Scratch, aus dem sich schnelle Schülerinnen und Schüler ein Projekt zum nachprogrammieren heraussuchen konnten. Zur Vorbereitung der Programmiersequenz in den Jahrgangsstufen 1/2 wurden außerdem die Programmier-Apps *ScratchJr*[6] und *Ronjas Roboter*[7]

[6] http://www.scratchjr.org/
[7] https://www.meine-forscherwelt.de/spiel/ronjas-roboter




*(a)*



*(b)*

*Abbildung 17.18 (a) Aufgabe in Scratch, in der sich auch mehreren Quadraten ein Ornament formt, und (b) Verdeutlichung des Prozesses an einem Modell.*

eingesetzt. Für den Einsatz von *ScratchJr* entwickelte eine Lehrkraft umfangreiche Materialien, die sich in ihrem Aufbau an dem Lernzirkel der Unterrichtssequenz orientierten.

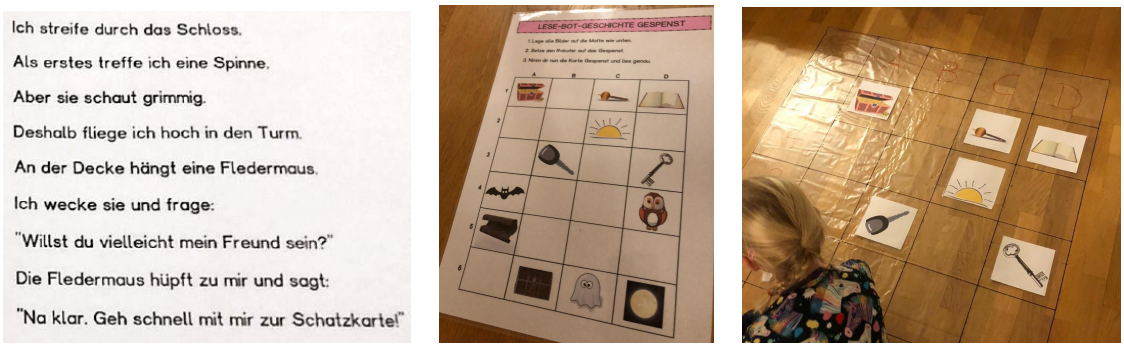
Fachinhalte

Mehrere Schulen, die im Klassenverbund programmierten, entschieden sich nach der zweiten Fortbildung dazu, *BlueBots*[8] – eine Variante des *BeeBots* – anzuschaffen, und diese in Zukunft zur Vorbereitung der Unterrichtssequenz oder bereits in den Jahrgangsstufen 1/2 einzusetzen. Dafür wurden Materialien aus dem Handel, aber auch selbst entwickelte Materialien entwickelt (siehe Abbildung 17.15). Eine Schule arbeitete bereits vor dem Besuch der ersten Fortbildung mit den Bienenrobotern und einem *Cubetto*[9]. Zwei Schulen, die ausschließlich eine AG anboten, schafften im Laufe des Projekts einen programmierbaren Spielzeugroboter (*Cozmo*[10]) und einen Satz von programmierbaren Bausätzen (*LEGO WeDo*[11]) an. Eine weitere Schule schaffte einen Satz *MBots*[12] an, entschied sich nach einer ersten Sichtung aber dafür, diesen vorrangig an der angegliederten Mittelschule einzusetzen. Im Bereich der Microcontroller wurde der *Calliope mini* und *MakeyMakey* ergänzend eingesetzt. Letzterer kann mittels Scratch eingesetzt werden, für die Programmierung des *Calliope mini* wurde der Editor *MakeCode*[13] verwendet. Einzelne Lehrkräfte bereiteten die Unterrichtssequenz durch eine Einheit zum Umgang mit Medien und dem Computer vor.

Intentionen

Zwei Lehrkräfte wandten das Programmieren in gänzlich neuen Kontexten an: im Fach *Deutsch als Zweitsprache* und im Fach Deutsch im Lernbereich Lesen. Um die Faszination, die von Robotern ausgeht, als Motivation für das Lesen auszunutzen, wurde das Format der *Lese-Bot-Geschichte* entwickelt (siehe Abbildung 17.19). Hier müssen die Schülerinnen und Schüler Texte genau lesen, um den richtigen Weg in einem Feld zu finden.

[8] https://www.b-bot.de/produkte/bee-bots/blue-bot/
[9] https://www.primotoys.com/
[10] https://www.golem.de/news/anki-cozmo-ausprobiert-niedlicher-programmieren-lernen-und-spielen-1706-128571.html
[11] https://education.lego.com/de-de/products/lego-education-wedo-2-0-set/45300#wedo-20-set
[12] https://www.makeblock.com/steam-kits/mbot
[13] https://makecode.calliope.cc/




*(a)* *(b)* *(c)*

*Abbildung 17.19* *Die Schülerinnen und Schüler müssen (a) einen Text lesen, (b) den Weg einer Figur einzeichnen und (c) diesen mit einem Roboter überprüfen.*

### 17.2.3 Zusammenfassung

Aus der Auswertung der Interviews ergeben sich verschiedene Erkenntnisse, die im Folgenden dargestellt werden.

*Format der Umsetzung*
Die Unterrichtssequenz wurde in den Jahrgangsstufen drei und vier, überwiegend im Klassenverband, erprobt. Ein Viertel der Schulen bot – teilweise zusätzlich – eine AG an. Gut ein Drittel der Lehrkräfte führte die Unterrichtssequenz im *Teamteaching* durch.

*Übungen ohne Computereinsatz*
Die *Unplugged*-Übungen wurden als sehr motivierend, spaßbringend, gut umsetzbar und effektiv beschrieben. Die Parcoursaufgaben stellten sich zum Teil als zu schwer heraus, was von vielen Lehrkräften durch das Erstellen eigener Aufgaben kompensiert wurde. Darüber hinaus entwickelten sie Ansätze, die es den Schülerinnen und Schülern ermöglichten, ihre haptischen Programme zeitgleich abzulaufen, sowie Aufgaben, bei denen Programme untersucht werden mussten.

*Programmieren in Scratch*
Das Programmieren in Scratch bereitete den Schülerinnen und Schülern großen Spaß. Hier bewährte sich vor allem das angeleitete Programmieren und bearbeiten konkreter Aufgaben. Viele Lehrkräfte kombinierten den Lernzirkel mit Programmieraufgaben aus der Fortbildung oder entwickelten zusätzliche eigene Aufgaben, die mehr Offenheit in der Umsetzung zuließen. Vielen Schülerinnen und Schülern fiel es schwer, ihre eigenen Programmideen zu planen und sich an die ursprüngliche Idee zu halten. Dennoch hatten sie viel Spaß bei der Arbeit an ihren eigenen Projekten und die meisten Ergebnisse waren sehr zufriedenstellend.

*Arbeitsweise der Schülerinnen und Schüler*
Viele Lehrerinnen und Lehrer berichteten, dass sich die Schülerinnen und Schüler gegenseitig oft geholfen haben. Das Programmieren im Team oder in der Gruppe funktionierte gut, wobei vereinzelt auch festgestellt wurde, dass sich einzelne Schülerinnen und Schüler dabei zurücknahmen.

*Zusätzliche Systeme*
An einzelnen Schulen wurde die Unterrichtssequenz durch zusätzliche Systeme ergänzt, die im Rahmen der Fortbildungen vorgestellt wurden. Im Klassenverband wurden vorrangig Systeme eingesetzt, welche die Unterrichtssequenz vorbereiten, wie *BlueBots* und die App *ScratchJr*. Alternative Systeme als nächster Schritt in der Unterrichtssequenz wurden, mit einer Ausnahme, nur in Arbeitsgemeinschaften eingesetzt.

### 17.2.4 Diskussion

Im Fokus der Studie stand die Praktikabilität der Unterrichtssequenz sowie die Identifikation von Anpassungen und Erweiterungen durch die Lehrkräfte. Der Einsatz explorativer Interviews und die Ergänzung durch von den Lehrkräften erstellte Unterrichtsmaterialien erwiesen sich für die Beantwortung der Forschungsfragen als zufriedenstellend. Dies gilt ebenso für die Methode der inhaltlich strukturierenden qualitativen Inhaltsanalyse.

Durch die inhaltliche Offenheit der Interviews und Verzicht auf einen Interviewleitfaden unterschieden sich die Transkripte zum Teil erheblich in ihren inhaltlichen Schwerpunktsetzungen sowie ihrer Dauer. Das führte dazu, dass nicht in allen Interviews alle Aspekte der Forschungsfragen in der gleichen Tiefe thematisiert wurden. Die Quantifizierung der Ergebnisse in Form der Anzahl der codierten Segmente pro Haupt- oder Subkategorie ist demnach zu vernachlässigen und dient lediglich zur groben Orientierung und Offenlegung der Methodik. Da zudem Dopplungen in den Interviews und Unterrichtsmaterialien nicht entfernt wurden, kam es vor, dass eine Abwandlung oder Erweiterung der Unterrichtssequenz im Rahmen des Interviews codiert wurde und zusätzlich das dazugehörige Unterrichtsmaterial. In diesem Kontext ist außerdem darauf hinzuweisen, dass mit den Lehrkräften von drei am Projekt teilnehmenden Schulen kein Interview geführt werden konnte. Im Rahmen der Analyse der Interviews wurde in dieser Teilstudie auf die Codierung durch einen zweiten Codierer verzichtet, da der Zweitcodierer für die Auswertung eine sehr gute Kenntnis der Unterrichtssequenz hätte haben müssen – insbesondere um entsprechende Anpassungen und Erweiterungen zu identifizieren. Es sind daher keine Angaben zur Intercoder-Reliabilität möglich. Da es sich bei den Kategorien jedoch ausschließlich um Fakten- und Themencodes handelt, die wenig Spielraum für Interpretation lassen, wird dies als weniger problematisch angesehen.

In Bezug auf die Grobziele, die der Gestaltung der Unterrichtssequenz zugrunde lagen (siehe Abschnitt 14.1) kann nun geprüft werden, ob die bislang nicht berücksichtigten Kompetenzformulierungen aus den Kompetenzerwartungen für informatische Bildung im Primarbereich der GI (GI 2019), den Zieldimensionen für eine frühe informatische Bildung der *Stiftung Haus der kleinen Forscher* (2018) und den *CSTA K12 Computer Science Standards* (2017) zukünftig berücksichtigt werden sollten:

- Die Schülerinnen und Schüler vergleichen Algorithmen unter Verwendung der Fachsprache (GI 6): Das Vergleichen verschiedener Lösungen einer Aufgabe wurde beim Programmieren *unplugged* umgesetzt und auch konkret von Lehrkräften in den Interviews thematisiert. Da die

Handlung jedoch auch als Feinziel des bisherigen Grobziels „Die Schülerinnen und Schüler entwerfen, realisieren und testen Algorithmen mit den algorithmischen Grundbausteinen Anweisung, Sequenz, Wiederholung und Verzweigung“ angesehen werden kann, wird die Kompetenzformulierung nicht als Grobziel aufgenommen.

- Die Kinder erklären gelesene Handlungsvorschriften und -abläufe für die Steuerung eines altersentsprechenden Informatiksystems (HdkF 2): Das Lesen und Untersuchen von vorgegebenen Programmen wurde nach der Erprobung im Grundschulunterricht in den Lernaktivitäten der Unterrichtssequenz ergänzt. Da die Kompetenzformulierung bisher nicht in den Grobzielen berücksichtigt wird, wird das Grobziel „Die Schülerinnen und Schüler beschreiben Algorithmen in ihrer Alltagssprache“ erweitert zu „Die Schülerinnen und Schüler lesen, beschreiben und untersuchen Algorithmen und Programme in ihrer Alltagssprache“.
- Die Kinder entwerfen eine Vorschrift zur Verschlüsselung von Nachrichten (Daten) mit altersentsprechenden Verfahren (z. B. Skytale) (HdkF 3): Die Kompetenzformulierung ist in Anbetracht der didaktischen Konzeption der Unterrichtssequenz nicht sinnvoll.
- Die Kinder wenden ein gegebenes Verfahren zur Lokalisierung einer fehlerhaften Stelle an (HdkF 5) bzw. *Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops* (CSTA 7): Das Lokalisieren von Fehlern in Programmen wird sowohl in der Beschreibung der *Design Principles* als auch in der Ablaufplanung der Unterrichtssequenz berücksichtigt und wurde auch von den Lehrkräften thematisiert. Da die Handlung jedoch auch als Feinziel des bisherigen Grobziels „Die Schülerinnen und Schüler entwerfen, realisieren und testen Algorithmen mit den algorithmischen Grundbausteinen Anweisung, Sequenz, Wiederholung und Verzweigung“ angesehen werden kann, wird die Kompetenzformulierung nicht als Grobziel aufgenommen.
- *Model the way programs store and manipulate data by using numbers or other symbols to represent information* (CSTA 2): Die Kompetenzformulierug ist in Anbetracht des zeitlichen Umfangs der Unterrichtssequenz nicht sinnvoll.
- *Give attribution when using the ideas and creations of others while developing programs* (CSTA 6): Das Wiederverwenden von bestehenden Programmen und damit verbundene Fragen des Urheberrechts wird im Rahmen der Unterrichtssequenz und deren didaktischer Konzeption nicht explizit erwähnt und wurde auch von den Lehrkräften nicht thematisiert.
- *Using correct terminology, describe steps taken and choices made during the iterative process of program development* (CSTA 8): Die Schülerinnen und Schüler entwickeln in der Unterrichtssequenz zwar Programmierprojekte, diese sind in der Regel jedoch nicht sehr umfangreich. Da ein mehrstufiger, iterativer Entwicklungsprozess darüber hinaus aus zeitlichen Gründen nicht möglich ist, wird die Kompetenzformulierung nicht als Grobziel aufgenommen.

Um Rückschlüsse auf die *Design Principles* ziehen zu können, die der Gestaltung der Unterrichtssequenz zugrunde lagen, wird in einem nächsten Schritt zusammengestellt, welche Gestaltungsprinzipien bzw. Umsetzungen davon als positiv thematisiert wurden (markiert mit „+“) bzw. welche in eigenen Anpassungen und Erweiterungen aufgegriffen wurden (markiert mit „neu“):

- Die Lernenden begegnen den Lerninhalten in unterschiedlichen Repräsentationsformen (DP 1):
  - Programmieren des *Lehrer-Roboters* (+)
  - Umwandeln von bildlichen Anleitungen zu sprachlichen Anweisungen (+)
  - Bearbeiten von Programmieraufgaben im Parcours bzw. Ablaufen der Lösungen mit Spielfiguren (+/neu)
  - Einsatz von programmierbaren Robotern (neu)
- Die Unterrichtssequenz führt vom Bekannten zum Unbekannten (DP 2):
  - Aufgreifen der Vorgangsbeschreibung (+)
  - Einsatz des Bilddiktats als abgewandelte Form des Spiels *Stille Post* (neu)
  - Einsatz der Parcoursaufgaben zur Erleichterung des Einstiegs in Scratch (neu)
- Die kognitive Belastung der Lernenden soll möglichst gering gehalten werden (DP 3):
  - Programmieren mit haptischen Scratch-Blöcken (+)
  - Lesen und Untersuchen von vorgegebenen Programmen (neu)
  - Einführung der Programmierumgebung durch die Lehrkraft (+)
  - angeleitetes Programmieren in Scratch (+)
- Die Aufgaben sind motivierend und ermöglichen aktives sowie kooperatives Lernen (DP 4):
  - Schnelle Erfolgserlebnisse durch Anstieg der Aufgabenschwierigkeit (+)
  - Möglichkeit zum Personalisieren der Programme (+)
  - Gemeinsames Programmieren in der Gruppe oder im Team (+)
  - Bearbeiten von Projektaufgaben im Klassenverband (neu)
  - Einsatz von programmierbaren Robotern und Microcontrollern (neu)
- Die Lernenden durchlaufen verschiedene Schritte des Problemlösens (DP 5):
  - Verknüpfung von angeleitetem Programmieren und Transferaufgaben (+)
  - Modellieren von Lösungen mithilfe von Pseudocode (neu)
  - Schülerinnen und Schüler helfen sich gegenseitig (+)
  - Planungsphase durch Vorlagen anleiten (+/neu)
- Die Lehrkraft fördert eine positive Fehlerkultur (DP 6):
  - Thematisieren von Fragen und Problemen im Plenum (+/neu)
- Die Lehrkraft ergreift Maßnahmen zur Zielorientierung (DP 7):
  - Planungsphase durch Vorlagen anleiten (+/neu)
- Die Lehrkraft unterstützt den Lernprozess durch Hilfestellungen (DP 8):
  - Modellieren von Lösungen mithilfe von Pseudocode (neu)
  - Visualisieren von Aufgabenstellungen (neu)
  - Maßnahmen zur Binnendifferenzierung, z. B. Lesen und Untersuchen von vorgegebenen Programmen (neu)

Die Auflistung zeigt, dass die Kernelemente aller *Design Principles* von den Lehrkräften als positiv bewertet wurden bzw. indirekt durch Anpassungen und Erweiterungen bestätigt wurden.

## 17.3 Teilstudie 2: Auswirkungen auf Schülerinnen und Schüler

Ziel der Studie ist es, die Auswirkungen des Unterrichtskonzepts auf Schülerinnen und Schüler zu untersuchen:

**(RQ 1-d)** Welche Auswirkungen hat das Unterrichtskonzepts auf Schülerinnen und Schüler?

Für die empirische Untersuchung werden explorative Interviews analysiert, die mit den Lehrkräften im Laufe der Erprobung der Unterrichtssequenz geführt wurden (siehe Abschnitt 8.5). Es handelt sich also um Einschätzungen der Lernfortschritte und des Verhaltens der Schülerinnen und Schüler. Von einer Untersuchung der Fragestellung, z. B. im Sinne einer Kompetenzmessung, wird an dieser Stelle abgesehen. Verschiedene Analysen der Programmierergebnisse der Schülerinnen und Schüler wurden bereits im zweiten Zyklus der Erprobung der Unterrichtssequenz durchgeführt (siehe Abschnitt 16.2).

### 17.3.1 Methodik

Zur Klärung der Forschungsfrage wurde eine Teilmenge der Daten aus der ersten Studie untersucht. Die transkribierten Interviews wurden anhand eines weiteren Kategoriensystems erneut codiert, das durch die inhaltlich strukturierende qualitative Inhaltsanalyse nach Kuckartz (2016, S. 97 ff.) entwickelt wurde (siehe Abschnitt 8.6.1). Nach der Sichtung zweier Transkripte wurde für den ersten Codierprozess in Anlehnung an die Zieldimensionen für frühe informatische Bildung der Stiftung *Haus der kleinen Forscher* (Bergner u. a. 2018, siehe Abschnitt 2.3) folgende Hauptkategorien festgelegt: (1) Übergreifende Basiskompetenzen, (2) Motivation, Interesse und Selbstwirksamkeit bzgl. des Programmierens, und (3) Informatische Kompetenzen. In einem nächsten Schritt wurde das Datenmaterial anhand der Hauptkategorien codiert. Die Codiereinheit wurde dabei so groß gewählt, dass der jeweilige Inhalt auch ohne weiteren Kontext zu verstehen war. Zur Codierung der Interviews wurde MAXQDA verwendet, eine Software zur computergestützten qualitativen Daten- und Textanalyse. Nach fünf Interviews wurde geprüft, ob die Hauptkategorien angepasst werden mussten, und die Hauptkategorie „Informatische Kompetenzen“ wurde umbenannt in „Lernergebnisse“. Die Kategorie sollte Kenntnisse oder Fähigkeiten umfassen, die explizit durch Elemente der Unterrichtssequenz erworben wurden. Da die Lehrkräfte auch über den Erwerb mathematischer Fähigkeiten berichteten, wie z. B. den Umgang mit Winkeln und dem Koordinatensystem, wurde die Kategorie umbenannt (siehe Tabelle 17.4). Das gesamte Datenmaterial wurde im Anschluss von zwei Codierern anhand der drei Hauptkategorien codiert, dabei ergab sich eine prozentuale Übereinstimmung von 80.51% und ein Intercoder-Koeffizient Kappa nach Brennan und Prediger (1981) von 0.76, was als substanzielle Übereinstimmung angesehen wird. Nachdem die jeweils nicht übereinstimmenden Segmente gesichtet und ggf. umcodiert wurden, wurden die codierten Segmente der Hauptkategorien am Material codiert, woraus verschiedene Subkategorien resultierten (siehe Abbildung 17.20). Die gesamten codierten Segmente wurden im Anschluss erneut mit dem ausdifferenzierten Kategoriensystem codiert. Die Ergebnisse der qualitativen Inhaltsanalyse werden ergänzt durch ausgewählte Items des Fragebogens FB-LK4 (siehe Abschnitt A.3.4), den die Lehrkräfte nach der dritten Fortbildungsveranstaltung ausgefüllt haben.

*Tabelle 17.4 Deduktive Kategorienbeschreibung zu Forschungsfrage RQ 1-d*

| Name | Beschreibung | Beispiel |
|---|---|---|
| Übergreifende Basiskompetenzen | Äußerungen, die Auswirkungen der Unterrichtssequenz auf Fähigkeiten beschreiben, die nicht ausschließlich Zielkompetenzen informatischer Bildung sind, z. B. kognitive Kompetenzen, sprachliche Kompetenzen, soziale Kompetenz. | „Wenn ich mir anschaue, wie unsicher die am Anfang da saßen und auch ganz große Probleme hatten, das zu versprachlichen [...]. Und das sehe ich jetzt gar nicht mehr. Wenn ich zu [meinen Schülern] hingehe, die können mir sagen, was sie schon ausprobiert haben. Das heißt, da ist auf einer sprachlichen Ebene wirklich auch viel passiert [...]."<br>(S5-1_20190515: Pos. 102) |
| Motivation, Interesse und Selbstwirksamkeit bzgl. des Programmierens | Äußerungen, die Rückschlüsse auf die Motivation, das Interesse und die Selbstwirksamkeit der Schülerinnen und Schüler erlauben. | „Aber ansonsten waren alle total begeistert. Die wollten gar nicht aufhören."<br>(S1-1_20190515: Pos. 49)<br><br>„Und ich finde auch, dass die ja schnell Erfolgserlebnisse auch haben. Also vor allem auch, wenn sie das erste Mal selber was machen, dann ist es ja schon ein Erfolg, wenn [die Figur] von A nach B geht. Und dann begrüßen sie sich. Und das hat geklappt. Und dann ist das schon so ‚Ich habe das jetzt schon geschafft. Und jetzt schaffe ich das nächste dann.'"<br>(S12-1_20190627: Pos. 151) |
| Lernergebnisse | Äußerungen, die Auswirkungen der Unterrichtssequenz auf Kenntnisse oder Fähigkeiten beschreiben, die explizit durch Elemente der Unterrichtssequenz erworben wurden. | „Also diese Selbstreflexion kommt immer mehr. Also wir können jetzt schon immer öfter sagen ‚Lies nochmal dein Skript' [...]. Also vielleicht sehen sie es nicht beim selber Programmieren, aber wir müssen nicht mehr den Fehler sagen und erklären, sondern sie kommen jetzt wirklich selber dran."<br>(S6-1_20190527: Pos. 27)<br><br>„Also es war wirklich, die haben das sofort kapiert mit den 90 Grad. Richtung, überhaupt kein Problem."<br>(S3-1_20191124: Pos. 28) |

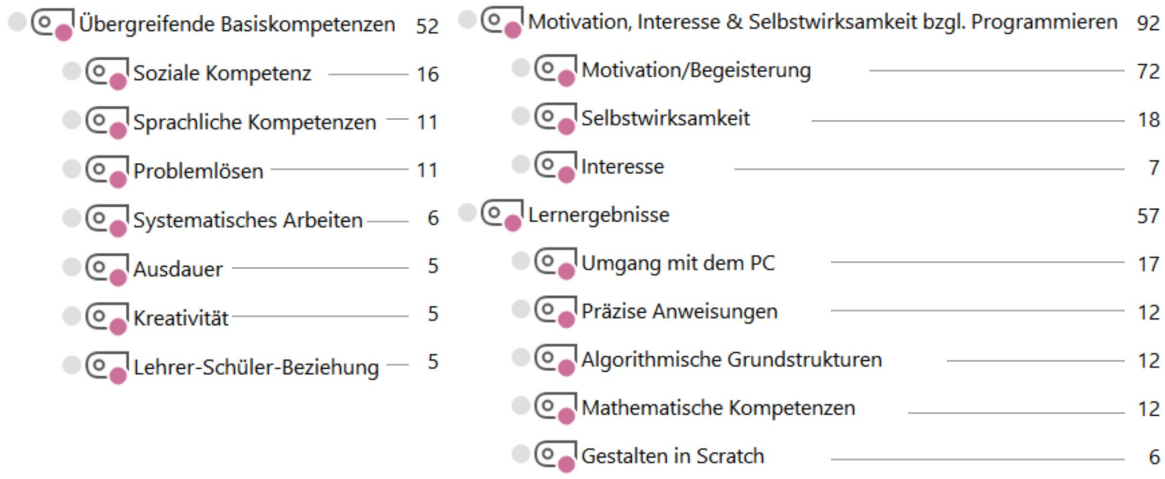



*Abbildung 17.20 Deduktiv-induktiv erstelltes Kategoriensystem zu Auswirkungen des Programmierens auf die Schülerinnen und Schüler und Anzahl der codierten Segmente*

### 17.3.2 Ergebnisse

Insgesamt wurden 201 codierte Segmente erfasst, die sich auf 15 Subkategorien verteilen (siehe Abbildung 17.20). Einzelne Codiereinheiten wurden mehreren Subkategorien zugewiesen, sodass die Summe der vergebenen Codes in den Subkategorien die der Hauptkategorien übersteigt. Die Ergebnisse der qualitativen Inhaltsanalyse werden im Folgenden anhand von Ankerbeispielen nach den Hauptkategorien gegliedert dargestellt.

*Übergreifende Basiskompetenzen*

Der Hauptkategorie konnten insgesamt 52 Textstellen zugeordnet werden, aus denen sieben Subkategorien generiert wurden.

Soziale Kompetenz: In neun Interviews wurde von einer positiven Auswirkung der Unterrichtseinheit zur Programmierung auf das Verhalten der Schülerinnen und Schüler sowie den Klassenzusammenhalt berichtet. Auch im Fragebogen FB-LK4 stellten vierzehn Lehrkräfte positive Auswirkungen auf das soziale Klima in der Klasse fest (siehe Abbildung 17.21). Die Lehrkräfte erwähnten die sehr gute Zusammenarbeit zwischen Schülerinnen und Schüler beim Programmieren selbst und beim Klären von Fragen:

> „Und ich finde auch, der Klassenzusammenhalt hat dadurch wirklich gewonnen, weil eben Schwächere den Stärkeren helfen konnten und weil sie endlich auch einmal irgendwo ein Selbstbewusstsein und ein Selbstvertrauen bekamen."
>
> S11-1_20190705: Pos. 71

Sprachliche Kompetenzen: In neun Interviews, die mit Lehrkräften von acht Schulen geführt wurden, wurde neben dem souveränen Verwenden von Fachbegriffen von einer Verbesserung des Sprachbewusstseins berichtet:

> „Ich finde zum Beispiel meine Klasse, dass die beim Schreiben, alleine beim Texte schreiben, haben die sich so verbessert alle, ohne dass wir da jetzt irgendwie konkret was gearbeitet haben." S12-1_20190627: Pos. 210

Problemlösen: In sieben Interviews berichteten die Lehrkräfte, dass die Schülerinnen und Schüler zunehmend versuchten, Probleme allein und im Team zu lösen, was ihnen auch immer mehr gelang:

> „Für mich ist dieses Weiterdenken und selber Nachdenken, wo hängt es jetzt bei mir, oder wo muss ich was anders erstellen, oder anders vorgehen [...]. Dieses, was ich mir für das Lernen aneignen muss, egal in welchem Fach. [...] Und ich finde, da trägt das jetzt schon viel bei." S6-1_20190527: Pos. 166

Systematisches Arbeiten: In fünf Gesprächen berichteten Lehrkräfte von einer Veränderung der Schülerinnen und Schüler hin zu einem planvollen, strukturierten Vorgehen. Dabei wurde sowohl das Fokussieren auf das Wesentliche erwähnt, als auch das Berücksichtigen aller notwendigen Schritte:

> „Und ich glaube, es hilft auch Schülern mal ein bisschen strukturierter zu denken. Weil da müssen sie sich wirklich einen Plan im Kopf machen, und sie können schon ausprobieren, aber trotzdem muss man sich das genau überlegen. Und das ist halt schon mal eine freiere Art sich das genau zu überlegen, das ist ja oft im Unterricht ja nicht so gegeben. Da ist ja irgendwas im Buch, oder [irgendwas] vorgegeben durch die Lehrkraft. Und so freies Entscheiden, wie handle ich jetzt und was muss ich machen, das ist schon gut, finde ich." S7-1_20190524: Pos. 72

Ausdauer: Von Auswirkungen auf das Durchhaltevermögen wurde in vier verschiedenen Interviews berichtet. Die meisten Lehrkräfte konnten positive Veränderungen bei ihren Schülerinnen und Schülern feststellen:

> „Und jetzt, das machen die jetzt schon. Und du merkst, die sind einfach dran [...]. Die wollen noch dran arbeiten und das immer noch weiter perfektionieren. Also nicht alle, aber ganz, ganz viele. Die sind da auch sehr hartnäckig." S6-1_20190527: Pos. 29

Kreativität: In vier Interviews wurde berichtet, dass ihre Schülerinnen und Schüler ihre Fantasie beim Programmieren in Scratch ausgelebt haben und viele eigene Ideen entwickelten:

> „Ich war überrascht, wie gut die Schüler da eigentlich dabei waren. Und auch überrascht von der jetzigen vierten Klasse, wie die selber auf die Ideen gekommen sind." S7-1_20190524: Pos. 191

Lehrer-Schüler-Beziehung: Neben einem verbesserten Sozialverhalten zwischen den Schülerinnen und Schülern wurde in fünf Interviews auch ein positiver Effekt auf das Verhältnis zur Lehrkraft geschildert. Eine positive Auswirkung auf die Lehrer-Schüler-Interaktion wurde auch von siebzehn Lehrern im Fragebogen FB-LK4 angegeben (siehe Abbildung 17.22). Die gemeinsame Suche nach Lösungen und die Erkenntnis, dass auch Lehrkräfte nicht alles wissen, brachten einander näher:

„Also ich war ganz oft beim Kind gestanden und habe gesagt: ‚Keine Ahnung'. Aber das war auch toll, weil auch für's Kind. Weil die dann einfach gemerkt haben, ‚OK, der Lehrer ist auch nicht perfekt'. Und man ist da so ein Stückchen zusammengewachsen, weil man dann selber gemeinsam überlegt hat, wie könnte denn das jetzt gehen? Und manchmal ist dann der Schüler drauf gekommen, manchmal man selber. Also das war so eine Zusammenarbeit und man ist so schön nah an den Schüler hingekommen."

S7-1_20190524: Pos. 192

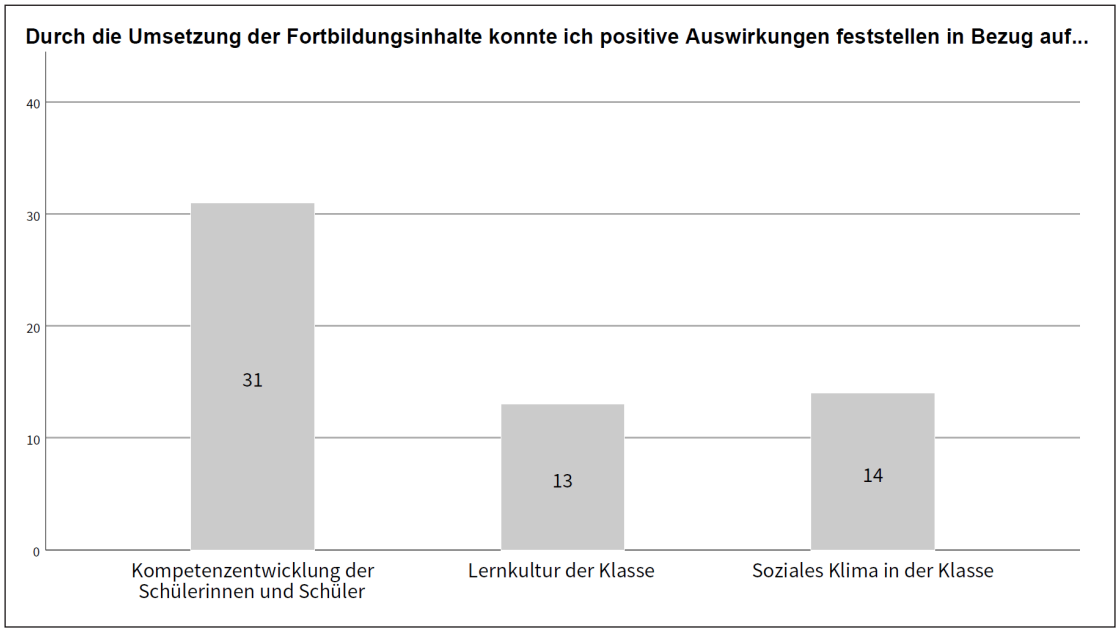



*Abbildung 17.21 Angaben zu Auswirkungen der Umsetzung der Fortbildungsinhalte auf die Schülerinnen und Schüler (n=33; FB-LK4)*

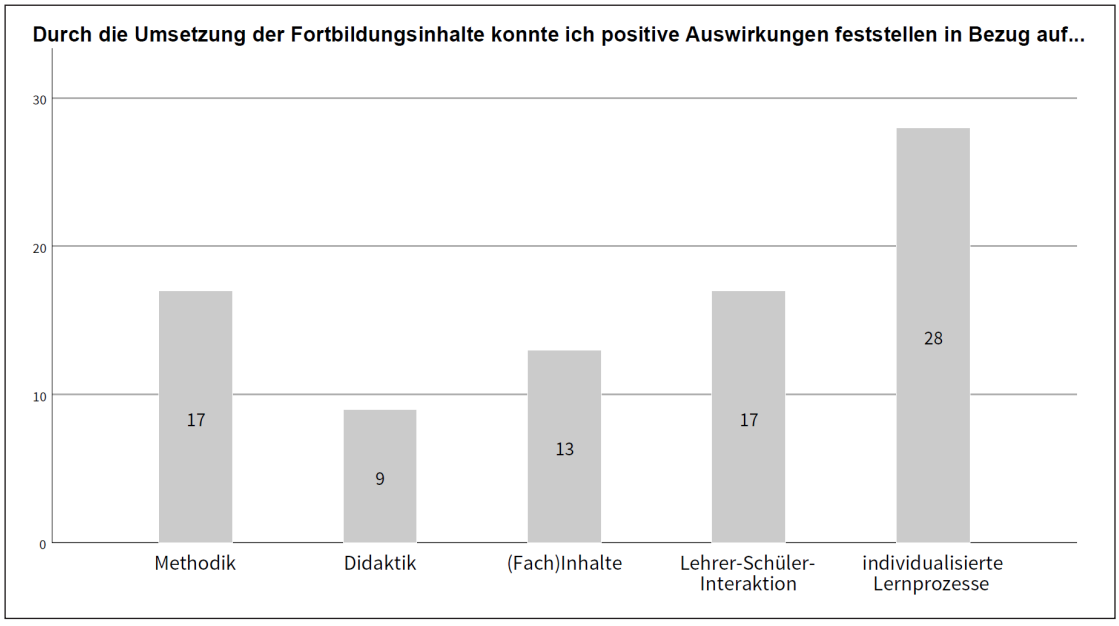



*Abbildung 17.22 Angaben zu Auswirkungen der Umsetzung der Fortbildungsinhalte auf den Unterricht (n=33; FB-LK4)*

*Motivation, Interesse und Selbstwirksamkeit bezüglich des Programmierens*
Insgesamt konnten 92 Textstellen der Hauptkategorie zugeordnet werden, aus denen drei Subkategorien entwickelt wurden. Diese wurden nahezu unverändert aus der Notation der Hauptkategorie übernommen.

Motivation/Begeisterung: In allen Interviews wurde berichtet, dass die Schülerinnen und Schüler bis auf wenige Ausnahmen durchweg sehr motiviert waren – sowohl bei den *Unplugged*-Übungen als auch beim Programmieren in Scratch. Es wurde außerdem berichtet, dass Schülerinnen und Schüler, die sich sonst sehr zurückhielten, sich plötzlich sehr motiviert beteiligten. Ein Unterschied zwischen Mädchen und Jungen wurde in dieser Hinsicht nicht festgestellt:

> *Lehrkraft 1:* „Also die [Klasse] war ganz begeistert, die war wirklich total dabei. "
> *Lehrkraft 2:* „Und auch bei den *unplugged* Sachen. Ja."
> *Lehrkraft 1:* „Wo ich ganz, also wirklich, wo ich schon überlegt habe, [...] wenn wir jetzt da kommen, wir machen hier Programmieren und dann ohne Computer. [...] Aber gar nicht" S7-1_20190524: Pos. 44-46

> „Also ich war ganz positiv überrascht, dass wirklich alle in der Klasse der Sache positiv gegenüber standen und sich alle dafür begeistern ließen, durch die Bank. Das war für mich schon überraschend." S11-1_20190705: Pos. 69

> „Die Schüler waren alle sehr interessiert und das schöne finde ich war, dass auch Kinder plötzlich total aktiv waren, die sich sonst eher sehr zurückhalten. Die [Schülerin] bekommt sonst das ganze Jahr den Mund nicht auf und war ständig am Fragen und war aktiv. Denen hat das wirklich total Spaß gemacht." S17-1_20180723: Pos. 4

Selbstwirksamkeit: In zehn Gesprächen wurde von positiven Auswirkungen auf das Selbstbewusstsein und die Selbstwirksamkeit der Kinder berichtet. Dabei wurden explizit positive Effekte auf Mädchen sowie leistungsschwächere Schülerinnen und Schüler angesprochen.

> „Also das habe ich jetzt auch gemerkt, dass, ja, viele Mädchen da dem Ganzen schon positiv gegenüber gestanden sind. Und das von sich aus jetzt eigentlich so auch als neue oder als versteckte Fähigkeit für sich entdeckt haben. ‚Das macht mir Spaß, das ist was, was ich auch kann, [...] wo ich auch ein Erfolgserlebnis habe.' [...] So ein Grundgefühl dann auch." S4-1_20190409: Pos. 340

Interesse: In drei Interviews wurde berichtet, dass einzelne Kinder ein tieferes Interesse für das Programmieren entwickelt hatten und diese sich z. B. auch privat einschlägige Bücher gekauft und sich in der *Scratch Online Community* angemeldet hatten. Eine Lehrkraft, die eine AG zum Programmieren anbot, berichtete, dass bei ihr vorrangig die Jungen privates Interesse entwickelten:

> „Die Jungs sind aber tatsächlich die, die sich zuhause mehr damit beschäftigen. Die sich anmelden, sich das Programm runterladen, sich Bücher besorgen [...]. Oder mir erzählen, sie haben zuhause dieses und jenes Projekt da sich schon überlegt und das wollen sie machen." S15-1_20190529: Pos. 131

*Lernergebnisse*

Neben den implizit vermittelten Basiskompetenzen und den Auswirkungen auf die affektiven Merkmale der Schülerinnen und Schüler beschrieben die Lehrkräfte auch Kenntnisse und Fähigkeiten, die explizit in der Unterrichtssequenz erworben wurden. Insgesamt wurden 57 Textstellen identifiziert, aus denen fünf Subkategorien entwickelt wurden. Im Fragebogen FB-LK4 gaben zudem 31 Lehrkräfte an, dass sie durch die Umsetzung der Fortbildungsinhalte positive Auswirkungen auf die Kompetenzentwicklung der Schülerinnen und Schüler feststellen konnten (siehe Abbildung 17.21).

Umgang mit dem PC: In elf Interviews wurde von den Lehrkräften angesprochen, dass die Schülerinnen und Schüler ihre Fähigkeiten im Umgang mit dem Computer vertiefen konnten, z. B. das Hoch- und Herunterfahren des Rechners, die Handhabung von Maus und Tastatur oder das Nutzen einer Ordnerstruktur:

> „Und das nächste war auch im Umgang mit dem PC. Speichern auf dem USB-Stick. Mal einen Ordner wechseln. Mal ein Laufwerk wechseln. Das kriegen die jetzt schon ganz anders hin, als vorher." S11-1_20190705: Pos. 27

Präzise Anweisungen: Insbesondere durch verschiedene *Unplugged*-Aufgaben konnten die Lehrkräfte ihren Schülerinnen und Schülern vermitteln, dass es beim Programmieren wichtig ist, genaue Anweisungen zu geben. Dies wurde in zehn Interviews thematisiert:

> „Ich fand auch dieses, dass irgendwie der Lehrer soll programmiert werden als Roboter an die Tür zu gehen und irgendwie die Türklinke zu drücken. Und du hast halt mit Absicht genau das gemacht, was sie gesagt haben. Und da finde ich, war auch der Lerneffekt auch der größte. [...] Da kamen wir sofort zu diesem ‚gehe einen Schritt' und so. Das war echt super." S5-1_20190515: Pos. 20

Algorithmische Grundstrukturen: Durch das Bearbeiten verschiedener Programmieraufgaben erlangten die Schülerinnen und Schüler ein grundlegendes Verständnis für algorithmische Strukturen, wie der Wiederholung und Bedingung. Dies wurde in acht Interviews, die an sieben Schulen geführt wurden, thematisiert:

> „Was unsere schon toll machen, finde ich, ist das mit den Bedingungen und Wiederholungen." S8-2_20191130: Pos. 13

Mathematische Kompetenzen: In acht Gesprächen berichteten Lehrkräfte von Auswirkungen auf die mathematischen Kenntnisse ihrer Schülerinnen und Schüler. Winkel und das Koordinatensystem wurden als Lerninhalte genannt, in denen die Kinder durch die Unterrichtssequenz Kenntnisse erworben hatten:

> „Und es bringt auch allgemein für die verschiedensten Dinge was. Einmal mit den Winkeln im Vorfeld schon. Dann wie wir die Figuren gemacht haben, die Dreiecke, Fünfecke. Da war dann klar, ich meine man darf jetzt nicht mit dem mathematischen Begriff Winkelsumme in [Klasse drei arbeiten]. Aber du musst überlegen, wenn du 360 Grad Winkel hast, wie teilst du den auf, dass es ein Fünfeck, ein Viereck, ein Dreieck wird. Und da sind die auf viele Dinge von selbst gekommen. Und auch welche Positionen. Es ist ja ein Koordinatensystem und ich habe das mit ihnen thematisiert und die kennen auch den Begriff Koordinatensystem natürlich nicht. Aber ich denke mal in der sechsten Klasse, oder in der fünften [...] tun sich die einfach leichter."
>
> S11-1_20190705: Pos. 20

Gestalten in Scratch: Eine Fertigkeit, die in acht Interviews angesprochen wurde, ist das konkrete Gestalten von Programmen in Scratch sowie das Implementieren eigener Ideen:

> „Ich hätte nie gedacht [...] dass die Kinder das [Programmieren des Spiels] so leicht können. Das hätte ich nicht gedacht, dass das [...] auch teilweise so perfekt ist. Wenn ich mir gerade das Projekt von [der Schülerin] ankucke, denke ich mir, ja besser hätte ich es eigentlich nicht machen können. So wie die Kinder über sich herauswachsen, das finde ich super, super beeindruckend." S6-1_20190527: Pos. 228

### 17.3.3 Zusammenfassung und Diskussion

Der Fokus dieser Teilstudie liegt auf den Auswirkungen, die das Programmieren im Unterricht auf Schülerinnen und Schüler haben kann. Durch die Auswertung der explorativen Interviews konnten verschiedene Effekte auf die Schülerinnen und Schüler herausgearbeitet werden. Dabei handelt es sich jedoch nur um subjektive Einschätzungen der Lehrkräfte, Rückschlüsse auf die tatsächlichen Auswirkungen können deshalb nur bedingt getroffen werden. Weitere Erhebungen im Rahmen der unterrichtlichen Umsetzungen, die eine unmittelbare Untersuchung der Auswirkungen des Programmierens auf die Schülerinnen und Schüler ermöglicht hätten, konnten aufgrund der restriktiven Bestimmungen des bayerischen Staatsministeriums für Unterricht und Kultus für die Forschung an Schulen nicht durchgeführt werden (siehe Abschnitt 8.4). Die Ergebnisse deuten jedoch darauf hin, dass eine frühe informatische Bildung im Allgemeinen bzw. das Programmieren im Speziellen das Potenzial hat, ein breites Spektrum an Kompetenzen bei den Schülerinnen und Schülern anzusprechen. Auffällig ist, dass die Lehrkräfte mehr von der Auswirkung auf übergreifende Basiskompetenzen und affektive Merkmale berichten als von fachspezifischen

Lernfortschritten. Die Subkategorien *Systematisches Arbeiten* und *Problemlösen* legen außerdem eine Verbindung zum Konzept des *Computational Thinking* nahe, bei dem das strukturierte Lösen von Problemen im Mittelpunkt steht (vgl. Kapitel 2.1). Darüber hinaus bestätigten sich die Hoffnungen, dass es möglich ist, durch informatische Bildung im Primarbereich ein positives Bild der Informatik zu vermitteln und Schülerinnen und Schüler in ihrem informatischen Selbstkonzept zu stärken (ebd.). Die Grundstruktur des Modells der Stiftung *Haus der kleinen Forscher* zu den Zieldimensionen früher informatischer Bildung hat sich in diesem Kontext als valide erwiesen.

Mit der Erhebungsmethode des explorativen Interviews war es möglich, im Gespräch mit den Lehrkräften eine angenehme Atmosphäre zu schaffen, in der sie ihre Erfahrungen und Ansichten frei teilen konnten. Jedoch ist anzumerken, dass darin nicht explizit nach den Auswirkungen auf die Schülerinnen und Schüler gefragt wurde. Bei der Auswertung der Interviews war es deshalb teilweise schwierig zu unterscheiden, ob die Lehrkräfte eine bereits im Vorfeld vorhandene Kompetenz oder eine durch die Unterrichtssequenz ausgebildete oder weiterentwickelte Kompetenz beschrieben. Es war zudem nur in Einzelfällen möglich, die Lehrkräfte der jeweiligen Schulen einzeln zu interviewen. Die Redeanteile waren jedoch auch in den Gruppeninterviews sehr ausgeglichen und es wurden durchaus Meinungen geäußert, die sich von denen der jeweils anderen Lehrkraft unterschieden. Als Auswertungsmethode für die Daten hat sich die inhaltlich strukturierende qualitative Inhaltsanalyse bewährt.

Die Ergebnisse der Studie stehen in Einklang mit anderen Forschungsergebnissen (siehe Abschnitt 3.2). So konnten bspw. die Ergebnisse von Duncan, Bell und Atlas (2017), Greifenstein, Graßl und Fraser (2021) und Murmann u. a. (2018) in großen Teilen repliziert werden. Darüber hinaus wurde das Potenzial von Scratch zur Förderung des mathematischen Denkens bereits von Taylor, Harlow und Forret (2010) beschrieben, die Fähigkeit zum Problemlösen und *Computational Thinking* von Gökçe und Yenmez (2022). Die berichteten Auswirkungen des Programmierens auf die affektiven Eigenschaften der Schülerinnen und Schüler legen die Entwicklung eines positiven Selbstkonzepts in Bezug auf das Programmieren nahe. Interessant wäre hier zu untersuchen, ob sich dies, wie in der Untersuchung von Diethelm u. a. (2020), auf die Informatik im Allgemeinen erweitern lässt, oder nur auf das Programmieren beschränkt.

In Bezug auf die Ziele, die der Gestaltung der Unterrichtssequenz zugrunde lagen (siehe Abschnitt 14.1), liegen mehrere Schlussfolgerungen nahe. Die affektiven Zieldimensionen – Motivation und Lernfreude im Umgang mit der Programmierung, Interesse an der Programmierung und Selbstwirksamkeitserwartung im Umgang mit der Programmierung – wurden zumindest teilweise an den Schulen erreicht. Über Motivation und Lernfreude wurde in allen Interviews berichtet, von einer gesteigerten Selbstwirksamkeitserwartung die Hälfte. Die Ausbildung von Interesse einzelner Schülerinnen und Schüler an der Programmierung wurde nur in drei Interviews angesprochen. Hier ist anzumerken, dass Motivation und Lernfreude vermutlich am leichtesten im Unterrichtsgeschehen zu identifizieren sind, da sie sich auf auszuführende Handlung beziehen (vgl. Schiefele und Schaffner 2020, S. 165). Interesse hingegen ist schwieriger durch Beobachtung zu ermitteln

und erfordert möglicherweise Erzählungen von Schülerinnen und Schülern, die über die Schule hinausgehen. Die kognitiven Lernziele, welche für die Unterrichtssequenz formuliert wurden (siehe Tabelle 14.5), sind in den Subkategorien der Hauptkategorie „Lernergebnisse“ wiederzufinden, wurden allerdings nur in weniger als der Hälfte der Interviews angesprochen.

## 17.4 Teilstudie 3: Gelingensbedingungen

Ziel der Studie ist es, Gelingensbedingungen für das Programmieren in der Grundschule abzuleiten:

**(RQ 3-a)** Mit welchen Herausforderungen sind die Lehrkräfte während der Erprobung des Programmierens in der Grundschule konfrontiert?

**(RQ 3-b)** Welche Gelingensbedingungen für das Programmieren in der Grundschule ergeben sich aus der Erprobung in der Praxis?

Zur Beantwortung der Forschungsfragen werden explorative Interviews ausgewertet, die mit Lehrkräften im Laufe der Erprobung der Unterrichtssequenz geführt wurden (siehe Abschnitt 8.5). Der Fokus liegt dabei auf den Herausforderungen, mit denen Grundschullehrkräfte bei der Erprobung der Unterrichtssequenz in der Praxis konfrontiert waren. Aus den identifizierten Herausforderungen werden in einem nächsten Schritt Gelingensbedingungen für das Programmieren in der Grundschule abgeleitet.

### 17.4.1 Methodik

Zur Klärung der Forschungsfrage wurden die explorativen Interviews untersucht, die im Rahmen von Feedbackgesprächen mit den Lehrkräften im Projekt *AlgoKids* geführt wurden (siehe Abschnitt 8.5). Die Interviewtranskripte wurden in Anlehnung an die inhaltlich strukturierende qualitative Inhaltsanalyse nach Kuckartz (2016, S. 97 ff.) (siehe Abschnitt 8.6.1) codiert. Nach der Sichtung zweier Transkripte wurde für den Codierprozess eine deduktive Kategorienbildung ausgewählt, für deren Umsetzung ein Kategoriensystem nach Vorlage des *Angebots-Nutzungs-Modell* von Seidel (2014) entwickelt wurde (siehe Abschnitt 4.1).

In einem nächsten Schritt wurde das Datenmaterial anhand der Kategorien codiert (siehe Tabelle 17.5). Die Codiereinheit wurde dabei so groß gewählt, dass der jeweilige Inhalt auch ohne weiteren Kontext zu verstehen war. Zur Codierung der Interviews wurde MAXQDA verwendet, eine Software zur computergestützten qualitativen Daten- und Textanalyse. Nach fünf Interviews wurde geprüft, ob das Kategoriensystem angepasst werden musste, was nicht der Fall war. Die Ergebnisse der qualitativen Inhaltsanalyse werden ergänzt durch ausgewählte Items der Fragebögen FB-LK 1 (siehe Abschnitt A.3.1) und FB-LK4 (siehe Abschnitt A.3.4), den die Lehrkräfte vor der ersten und nach der dritten Fortbildungsveranstaltung ausgefüllt haben.

*Tabelle 17.5 Deduktives Kategoriensystem zu Forschungsfrage RQ 3-a und Anzahl der vergebenen Codes*

| Hauptkategorie | Subkategorie | Beispiel |
|---|---|---|
| Rahmenbedingungen | Kontext des Bildungssystem (20) | „Man müsste halt den Lehrplan dementsprechend wieder irgendwo anpassen. Das ist halt das Problem, dass die Stunden, die man dafür hernimmt, die bleiben bei anderen Fächern hängen." (S6-1_20190527: Pos. 136) |
| | Kontext des Fachs (38) | „Das müsste verbindlicher sein und es müsste auch einfach in der Stundentafel mit drin stehen." (S5-1_20190515: Pos. 64) |
| | Kontext der Schule (51) | „Da muss man sagen, ist es trotzdem ein riesen Zeitding. Wir haben an den Grundschulen keinerlei Systembetreuung mit großer Stundenanrechnung oder sowas." (S3-1_20191124: Pos. 28) |
| | Kontext des Kollegiums (26) | „Oder Beamer, irgendwas , das gar nichts damit zu tun hat. Ihr macht die Programmier-AG, ihre kennt euch mit den PCs und dem ganzen Zeug aus." (S6-1_20190527: Pos. 96) |
| Unterricht | Kontext der Klasse (16) | „Ja, weil wir jetzt schon echt besondere Klassen haben. Wenn du jetzt Klassen mit weniger Migrationshintergrund hast, dann ist es wahrscheinlich gar nicht mal so." (S2-1_20190508: Pos. 51) |
| | Lehrerkompetenzen (43) | „Trotzdem immer noch so, kann ich das den Kindern auch so vermitteln? Kann ich das so rüberbringen? Wie geht das vom didaktischen her? Wie organisiere ich das alles?" (S4-1_20190409: Pos. 283) |
| | Lehrprozesse im Unterricht (37) | „Da konnte ich dann aber nie mehr irgendwas dazwischen sagen. Weil dann waren die weg. Jeder hat dann irgendwie geschrien: ‚Äh, ich brauch sie, ich kann's nicht!' Ich konnte aber nicht mehr eine allgemeine Anweisung geben." (S3-1_20191124: Pos. 130) |
| | Individuelle Lernaktivitäten (46) | „Natürlich machen die Kinder, was sie können, oder was funktioniert. Und wenn ich jetzt will, dass die Personen miteinander sprechen, aber es funktioniert nicht – ja, dann machen wir halt was anderes." (S1-1_20190515: Pos. 118) |
| | Lernergebnisse (8) | „Zu sagen, man nimmt da ganz viele Stunden her und dann kommt keine so Bewertung raus. Aber an sich, als Lehrer hat man nicht das Gefühl, dass man diese Bewertung unbedingt noch bräuchte. Aber man hat das Gefühl es wird erwartet, dass es die gibt." (S6-1_20190527: Pos. 135) |
| Schülerinnen und Schüler | Lernumwelt (15) | „Ich habe halt Kinder, die kannten sich mit Laptops aus, ich hatte Kinder, die kannten Scratch schon und ich hatte Kinder, die haben noch nie irgendwas in der Hand gehabt." (S3-2_20191124: Pos. 4) |
| | Individuelle Voraussetzungen (38) | „Ich finde schon, dass man da auch, dass die vielleicht schon alle motiviert sind, aber dass man halt schon auch Kinder hat, die kommen da an ihre Grenzen. Und die [...] können es einfach nicht durchdenken." (S12-1_20190627: Pos. 53) |

### 17.4.2 Ergebnisse

Insgesamt wurden 338 codierte Segmente erfasst, die sich auf 11 Subkategorien verteilen (siehe Tabelle 17.5). Die Ergebnisse der qualitativen Inhaltsanalyse werden im Folgenden anhand von Ankerbeispielen nach den Hauptkategorien gegliedert dargestellt.

*Rahmenbedingungen*
Die Hauptkategorie fast vier Subkategorien zusammen, denen insgesamt 135 Textstellen zugeordnet werden konnten.

Kontext des Bildungssystem: In insgesamt elf Interviews wurde über Herausforderung im Zusammenhang mit dem Bildungssystem berichtet, vor allem hinsichtlich des Übertritts auf die weiterführenden Schulen. Aufgrund der Vielzahl der übertrittsrelevanten Themen erwies es sich für viele Lehrkräfte als schwierig, dem Programmieren ausreichend Zeit einzuräumen und regelmäßig in den Unterricht zu integrieren:

> „Und dann frägt man sich, wo kommt das jetzt noch rein, oder wo sollen wir das reinschieben. Der Lehrplan ist eh schon so voll, was lassen wir dafür weg. Und gerade, überleg mal, mit unseren Klassen dieses Jahr, die sind teilweise einfach auch so leistungsschwach [...]." S6-1_20190527: Pos. 136

Dies zeigte sich auch in der Auswertung des Fragebogens FB-LK4, in dem 23 Lehrkräfte angaben, dass mangelnde Zeit die Umsetzung der Fortbildungsinhalte erschwerte (siehe Abbildung 17.23). Aus diesem Grund entschieden sich einzelne Lehrkräfte, das Programmieren als Projekt nach dem Übertritt anzubieten oder es in einer AG zu verorten:

> „Zum Schuljahresende hin in Deutsch, Mathe und so, da sind wir eh immer unter Druck. Da hat man bis zum Übertritt schon echt fast den ganzen Stoff durch. Und dann ist das am Schuljahresende dann eigentlich eher so, dass man Luft hat. Vor dem Übertritt das durchzuziehen, hätte ich wirklich die Muse auch nicht. Da wäre der Druck zu groß gewesen, das hätten wir nicht geschafft. [..] Und da jetzt mal zwölf Unterrichtseinheiten herzunehmen, das ist schon." S9-1_20190718: Pos. 94-95

> „[...] wenn man eine vierte Klasse hat in Bayern ist es kaum im normalen Unterricht unterzubringen. In der AG super, aber wirklich im Alltagsunterricht kaum möglich zu machen, dass man das irgendwie unterbringt. Und das ist halt schade einfach."
> S6-1_20190527: Pos. 136

Kontext des Fachs: In vierzehn Interviews wurden Herausforderung im Kontext des Fachs *Programmieren* bzw. dem Fehlen eines Fachs thematisiert. Die Lehrkräfte verwiesen vor allem auf die fehlende Legitimation des Fachs und schwierige Verortung in der Stundentafel:

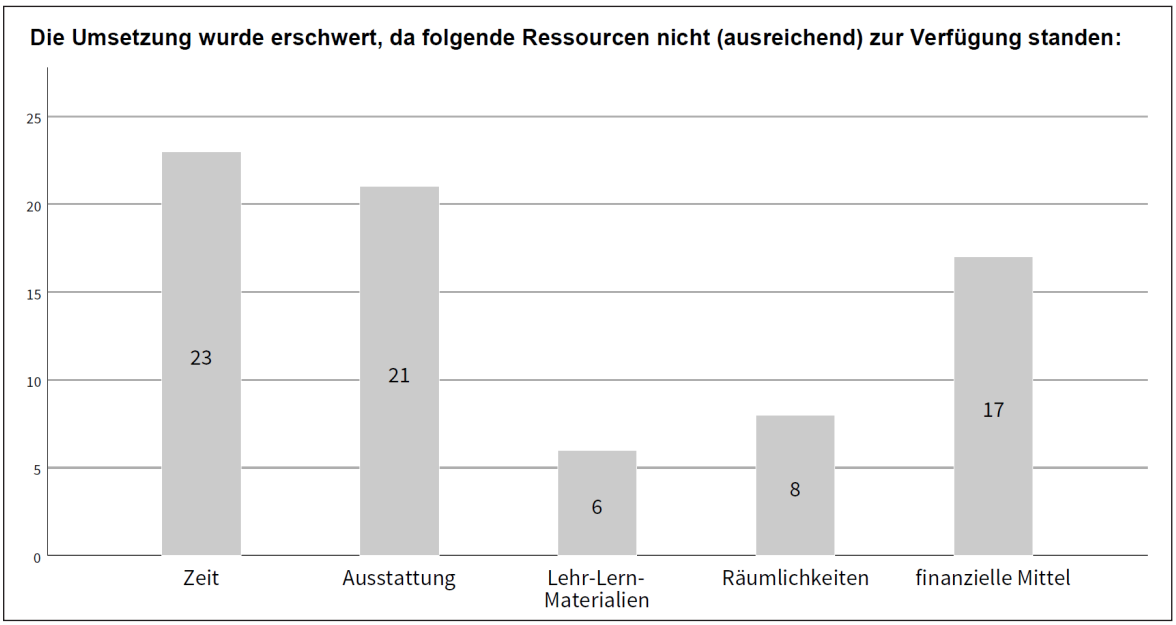



*Abbildung 17.23 Angaben zu Hindernissen bei der Umsetzung der Fortbildungsinhalte (n=33; FB-LK4)*

> „Aber es geht mir auch nicht darum, wo es verortet ist. Sondern es geht mir einfach darum, dass es seinen Platz kriegt und jetzt nicht so [...] ‚Es ist für jedes Fach wichtig und deshalb geben wir euch keine Stunden dafür, sondern deshalb gießt ihr mal mit der Gießkanne irgendwie drüber.' [...] So haut es halt nicht hin."
>
> S11-1_20190705: Pos. 79

Als weitere Herausforderung wurde die inhaltliche Abgrenzung der Unterrichtssequenz von den Inhalten der Mittelschule genannt, in der das Unterrichtsfach Informatik in Bayern zum Schuljahr 2019/20 einstündig im Stundenplan verankert wurde:

> „[...] Es gibt ja einfach an den weiterführenden Schulen einfach die Berührungspunkte, wo die Kinder irgendwie damit konfrontiert werden und dann nicht vollkommen unvorbereitet da kommen. Auf der anderen Seite habe ich dann auch wieder bei einer Fortbildung gehört von einer Mittelschullehrerin, die Informatik unterrichtet, die gesagt hat [...] sie fangen bei Scratch an. Da habe ich gedacht, OK, wir auch. Es wäre halt gut, wenn das irgendwie stringent wäre. Dann irgendwie klar wäre, OK, bis zu dem Punkt arbeiten die Grundschulen und da können die weiterführenden Schulen anknüpfen und weiterarbeiten." S5-1_20190515: Pos. 79

KONTEXT DER SCHULE: Als eine der größten und belastendsten Herausforderungen wird die technische Ausstattung und Infrastruktur an den einzelnen Schulen empfunden. In dreizehn Interviews berichteten die Lehrkräfte von nicht funktionierenden oder fehlenden Geräten sowie einer fehlenden oder problembehafteten Systembetreuung. Auch im Fragebogen FB-LK4 geben 21 Lehrkräfte an, dass die Umsetzung der Fortbildungsinhalte durch die Ausstattung erschwert wurde (siehe Abbildung 17.23). Bereits zu Beginn der Fortbildung bemängelte mehr als die Hälfte der Lehrkräfte die technische Ausstattung ihrer jeweiligen Schule (siehe Abbildung 17.24). Selbst wenn

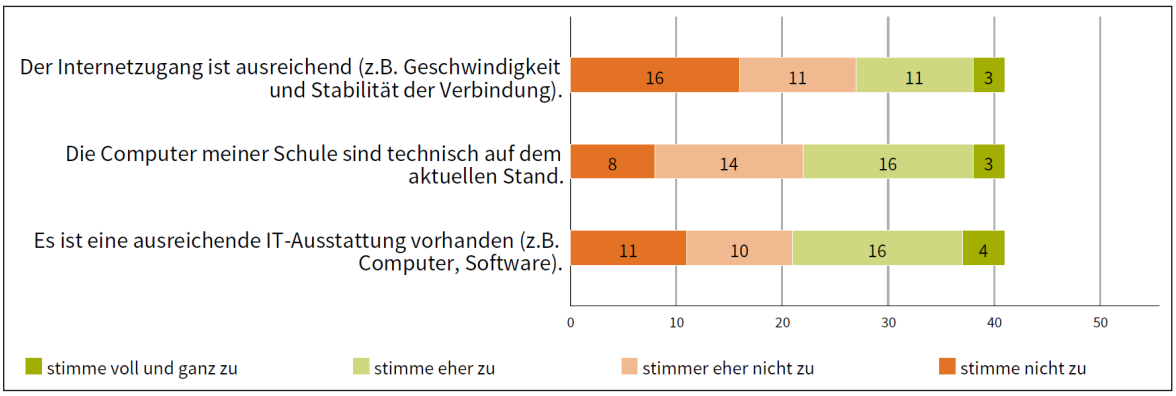



*Abbildung 17.24 Angaben zur technischen Ausstattung an den Schulen (n=41; FB-LK1)*

eine entsprechende Ausstattung vorhanden war, klagten viele Lehrkräfte über technische Probleme bei der Nutzung, die zu Verzögerungen im Unterrichtsverlauf führten. Dazu gehörten: Probleme beim Speichern von Dateien, überlastete Server, fehlende Zugriffsrechte für Lehrkräfte, fehlendes WLAN in den Klassenräumen, nur bedingt funktionierende Mäuse, ständige Updates und nicht hochfahrbare Computer. Die technischen Probleme führten zu Unsicherheiten, die insbesondere in Verbindung mit den neuen Fachinhalten als Belastung empfunden wurden:

> „Ja, das ist halt einfach echt frustrierend. Weil eigentlich ist es ja nicht so, [...] dass es alles total *easy peasy* ist und man macht das quasi mit links. Sondern man hat ja quasi schon das Thema was irgendwie neu ist und wo man irgendwie noch unerfahren ist.[...] Und dann gibt es eben diese technischen Probleme.“ S12-1_20190627: Pos.

KONTEXT DES KOLLEGIUMS: Im Zusammenhang mit dem Kollegium wurden in zehn Interviews Herausforderungen geschildert. Hier wurde beklagt, dass man plötzlich als Ansprechperson für sämtliche IT-Fragen fungieren musste, wenig Interesse oder Engagement des Kollegiums wahrgenommen wurde und man die alleinige Verantwortung für den Bereich der Programmierung erhielt:

> „Man ist irgendwie so jetzt plötzlich die AlgoKids-Fachfrau, ja? [...] ‚Ja nächstes Jahr macht ihr auf jeden Fall wieder Programmieren.‘ [...] Das ist jetzt so wie irgendwie, ich habe nicht schnell genug weggeschaut. Oder ich habe vor zwei Jahren mir gedacht, ja ist doch nett, so im Sinne der Medienbildung. Man engagiert sich mal ein bisschen und nimmt ein bisschen was mit. Und jetzt plötzlich ist man hier irgendwie [...] voll zuständig für’s ganze Programmieren, habe ich das Gefühl.
> Ich finde es total spannend. Ich mache es auch so gerne in dem Rahmen. Aber ich denke mir so, wenn das jetzt noch mehr wird und ich irgendwie auch noch dazu verdonnert werde. Bis jetzt habe ich es ja irgendwie freiwillig gemacht. Da finde ich es irgendwie schwierig, dass man jetzt echt so einen Status aufoktroyiert bekommt, nur weil ich mich damals bereit erklärt habe, im Sinne der Medienbildung mich fortzubilden.“
>
> S5-1_20190515: Pos. 256-258, 298

*Unterricht*

Die Hauptkategorie Unterricht umfasst insgesamt 150 Textstellen, die fünf Subkategorien zugeordnet wurden.

Kontext der Klasse: In sechs Interviews wurden Herausforderungen in Zusammenhang mit den unterrichteten Klassen beschrieben, z. B. fehlende Vorerfahrung in der Gruppenarbeit oder einen hohen Anteil an Schülerinnen und Schülern mit Migrationshintergrund und damit verbunden vermehrt sprachliche Probleme. Um besser mit den hohen Schülerzahlen zurecht zu kommen, wurde an einigen Schulen im Team unterrichtet:

> „Also wir machen es generell in anderen Fächern auch. Es ist häufiger, dass wir was zusammen machen. Weil es einfach schon den Einzelnen entlastet. Und wir machen es auch oft so stärkend. Jeder kann was zu einem bestimmten Teil besser. Und dann ist es schon auch eine Entlastung. Gerade so bei Scratch fand ich es schon gut, es auf zwei aufzuteilen." S14-1_20190312: Pos. 62

Lehrerkompetenzen: Herausforderungen in Bezug auf die eigenen Kompetenzen als Lehrperson wurden in fünfzehn Interviews genannt. Hier wurden vor allem Unsicherheiten und Zweifel in Bezug auf die eigene Qualifizierung genannt und Befürchtungen geäußert, Fragen im Unterricht nicht beantworten zu können oder mit Kindern konfrontiert zu werden, die einen höheren Wissensstand haben als man selbst:

> „[...] Wo ich jetzt glaube ich überfordert dann wäre, wenn die selber was entwickeln. Wenn die dann da Fragen haben und da wüsste ich wahrscheinlich nicht auf alles eine Antwort." S2-1_20190508: Pos. 191

Darüber hinaus wurde berichtet, dass man sich nach längeren Pausen im Programmieren erst wieder in die Arbeit mit Scratch hineinfinden musste:

> „[...] Ich habe mich ja wirklich eigentlich mit Scratch gar nicht mehr beschäftigt, muss ich zugeben. Und ich habe dann eben vor ein paar Wochen wieder dann angefangen. Und also ich musste plötzlich wieder. Jetzt zum Beispiel vorhin auch, das war ja was ganz Einfaches, dass die nur auf die Figur klicken. Das ist mir dann nicht mehr eingefallen. Ja. Also da habe ich mir echt ein bisschen schwergetan. [...] Wenn man's dann macht, kommt man eigentlich doch wieder relativ schnell dann rein."
> S10-1_20190607: Pos. 6

Lehrprozesse im Unterricht: In 15 Interviews wurden sehr unterschiedliche Herausforderungen in Bezug auf Lehrprozesse geschildert, wie z. B. Schwierigkeiten bei der Formulierung von Arbeitsaufträgen während der Arbeit am Computer oder das Erkennen von Schülerinnen und Schülern, die gerade etwas anderes am Computer tun. Einige Lehrkräfte taten sich beim freieren Programmieren in Scratch schwer damit, ein gewisses Maß an Ernsthaftigkeit durchzusetzen:

> „Aber wo bremst man sie dann wirklich so aus, und sagt Stopp jetzt. Also ich möchte das jetzt schon machen, weil ich möchte einfach, dass die nächste Ebene, auch einfach ein bisschen an Ernsthaftigkeit dann zunimmt. Dass man sagt, ja OK, es geht nicht nur drum hier Halligalli Spaß mit den Figuren zu betreiben, sondern einfach auch sich wirklich zu überlegen, wie schaffe ich es, dass... Genau, also das habe ich dann schon immer wieder eingefordert und sie haben es, ja doch, und dann haben sie auch wirklich zusammengearbeitet. [...] Und also doch, die Sachen, die ich ihnen vorgegeben hatte, haben sie schon gemacht. Aber sobald sie das hatten, haben sie schon wieder – voller Fantasie." S4-1_20190409: Pos. 84

Individuelle Lernaktivitäten: Ebenfalls sehr unterschiedlich waren die Schilderungen von Herausforderungen bzgl. individueller Lernaktivitäten. In dreizehn Interviews nannten die Lehrkräfte bspw. die vielen Fragen der Schülerinnen und Schüler, ihr ständiger Drang, auf die Computer von anderen Gruppen zu schauen, oder Probleme beim sinnentnehmenden Lesen sowie mangelnde Geduld – vor allem bei den Jungen. Beim freieren Programmieren bemerkten einige Lehrkräfte, dass die Schülerinnen und Schülern weniger mit dem Programmieren als vielmehr mit der Gestaltung des Projekts in Scratch beschäftigt waren:

> „Wir haben jetzt beim zweiten Mal, dass wir es mit ihnen gemacht haben, da war es dann so, dass wir [...] sie wieder frei was machen lassen wollten. Und da haben wir dann aber festgestellt, dass viele eher sich die ganzen Hintergründe anschauen wollten und welche Figuren gibt es überhaupt. Also es war dann wirklich mehr so ein Bild gestalten sage ich jetzt mal, als jetzt tatsächlich irgendwie einen Befehl tatsächlich zu programmieren. " S13-1_20190716: Pos. 58

Lernergebnisse: In vier Interviews wurde benannt, dass man die Überprüfung des Erwerbs fachlicher Kompetenzen als herausfordernd sehe und sich generell frage, ob eine Bewertung der Leistungen überhaupt zielführend sei:

> „Weil im Endeffekt ist es meistens so, dass man Unterrichtsinhalte hat und das Ziel dann irgendwann eine Probe ist, in der man die abprüft. Und das ist jetzt ja hier zum Beispiel nicht gegeben. [...] Eigentlich blöd, dass man immer alles nur auf diese Leistungserhebungen ausrichtet, aber im Endeffekt ist es halt doch so. [...] Ich meine, das sind wichtige Fähigkeiten, die sie da lernen, aber das prüfst du jetzt konkret nicht irgendwie ab. Und dann ist halt die Frage, wo ordnest du das zu, dass du dir die Zeit im Wochenplan dafür nehmen kannst." S13-1_20190716: Pos. 116

*Schülerinnen und Schüler*

Der Hauptkategorie konnten insgesamt 53 Textstellen zugeordnet werden, die sich auf zwei Subkategorien aufteilen.

Lernumwelt: In Hinblick auf die Lernumwelt der Schülerinnen und Schülern wurden die unterschiedliche Mediennutzung und damit verbunden die unterschiedlichen Vorerfahrungen als herausfordernd beschrieben. Hier wurde in neun Interviews von einzelnen Kindern berichtet, die Zuhause keinen Computer nutzen dürfen, und anderen, die im Gegensatz dazu zuhause mit dem Vater – selbst ein Informatiker – Legoroboter programmieren. Es wurde zudem im Kontext einer möglichen Bewertung des Programmierens angemerkt, dass es für manche Schülerinnen und Schüler nicht möglich wäre, zuhause das Programmieren in Scratch zu üben. Betont wurde außerdem, das vor allem die Mädchen zuhause weniger Berührungspunkte mit dem Programmieren oder Computern im Allgemeinen haben:

> „Es wird besser, aber es ist schon eher die Scheu am Anfang. Und auch wirklich eher Mädchen würde ich sagen, die sagen Anfang der dritten Klasse, sie saßen noch nie an einem Computer. Sie wissen nicht, wie das geht. “ S15-1_20190529: Pos. 145

Individuelle Voraussetzungen: In fünfzehn Interviews wurden die individuellen Voraussetzungen der Schülerinnen und Schüler als Herausforderung beschrieben. In erster Linie wurden hier die unterschiedlichen Vorerfahrungen im Umgang mit dem Computer genannt. Während einige Schülerinnen und Schüler keine oder nur geringe Vorkenntnisse hatten, konnten andere bereits problemlos Programme öffnen, Dateien speichern und selbstständig am Computer arbeiten. Darüber hinaus bemerkten die Lehrkräfte erhebliche Unterschiede in der Geduld, Frustrationstoleranz und Anstrengungsbereitschaft ihrer Schülerschaft. Auch hinsichtlich der sprachlichen und kognitiven Kompetenzen wurden Unterschiede festgestellt:

> „Ich glaube es gibt schon Kinder, die von ihrem Verständnis her noch nicht so weit sind. Auch in der vierten Klasse noch nicht. Die einfach schon mit der Welt in der vierten Klasse – Mathe, Deutsch, HSU – an ihr Maximum ihrer Entwicklung hinkommen. “ S15-1_20190529: Pos. 108

### 17.4.3 Interpretation

Ziel der Teilstudie ist das Ableiten von Gelingensbedingungen für das Programmieren in der Grundschule. Durch die Auswertung der explorativen Interviews konnten in einem ersten Schritt Herausforderungen identifiziert werden, mit denen die Lehrkräfte im Rahmen des Projekts *AlgoKids* konfrontiert waren. Hier wurden verschiedenste herausfordernde Bereiche genannt, welche die Rahmenbedingungen des Unterrichts, den Unterricht selbst und die Schülerinnen und Schüler betreffen. Als Gelingensbedingung zu vernachlässigen sind die Aspekte, die Schülerinnen und Schüler und den Kontext der Klasse betreffen, da diese nicht beeinflusst werden können. Gleiches gilt für die Herausforderungen im Kontext des Kollegiums, da sich diese zum einen schwer beeinflussen lassen und zum anderen einige Aspekte genannt wurden, die sich spezifisch auf den Projektkontext beziehen. Aus den verbleibenden Herausforderungen lassen sich folgende Gelingensbedingungen ableiten:

*Gelingensbedingung 1: Verankerung im Lehrplan*
Die fehlende Legitimation des Programmierens durch den Lehrplan und der damit einhergehende Mangel an zeitlichen Ressourcen beunruhigte viele Lehrer und wurde vereinzelt konkret als Grund genannt, warum das Programmieren in der Grundschule scheitern könnte. Eine Anbindung der Unterrichtssequenz an den bestehenden Lehrplan oder eine Erweiterung des Lehrplans könnten hier Abhilfe schaffen. Zu klären bleibt allerdings, ob die Programmierergebnisse der Schülerinnen und Schüler bewertet werden sollten.

*Gelingensbedingung 2: Technische Infrastruktur*
Die technische Ausstattung führte an vielen Schulen zu Problemen in der Umsetzung der Unterrichtssequenz und stellte einen erheblichen Stressfaktor für viele Lehrkräfte dar. Eine angemessene technische Infrastruktur inklusive Systembetreuung ist für das Programmieren in der Grundschule unabdingbar. Hier ist jedoch anzumerken, dass das Projekt *AlgoKids* durchgeführt wurde, bevor die Gelder des *DigitalPakts Schule* zur Förderung der digitalen Bildungsinfrastruktur an bayerischen Schulen ausgeschüttet wurden.

*Gelingensbedingung 3: Fortbildung von Lehrkräften*
Einige Lehrkräfte zweifelten – trotz der Fortbildungsveranstaltungen innerhalb des Projekts – an ihren fachlichen sowie fachdidaktischen Kompetenzen und merkten an, im Programmieren schnell aus der Übung zu kommen. Hier zeigt sich, dass eine umfassende Fortbildung der Lehrkräfte unbedingt notwendig ist. Hilfreich wären zudem Fortbildungsimpulse und Anregungen im kleinerem Umfang, die von den Lehrkräften je nach Bedarf genutzt werden können.

*Gelingensbedingung 4: Adäquate Unterrichtsmaterialien*
Die Lehrkräfte schilderten einige Herausforderungen im Zusammenhang mit Lehrprozessen und Lernaktivitäten, insbesondere in Bezug auf das freiere Programmieren. Diese können durch Anpassungen in der Unterrichtssequenz abgeschwächt werden. Es zeigt sich, dass geeignete Unterrichtsmaterialien für eine erfolgreiche Umsetzung des Programmierens in der Grundschule notwendig sind.

### 17.4.4 Diskussion

Die Erhebungsmethode des explorativen Interviews erwies sich als geeignete Methode zur Datenerhebung. Es ist anzumerken, dass nicht explizit nach Herausforderungen gefragt wurde – ein visueller Impuls in der zweiten Phase der Interviews bezog sich jedoch auf erlebte Belastungen (siehe Abschnitt A.2). Als Auswertungsmethode für die Daten hat sich die inhaltlich strukturierende qualitative Inhaltsanalyse bewährt. Gleiches gilt für das Kategoriensystem, das nach Vorlage des *Angebots-Nutzungs-Modell* von Seidel (2014) entwickelt wurde. Sämtliche Herausforderungen, die geschildert wurden, konnten im Kategoriensystem verortet werden.

Die Ergebnisse der Studie stimmen mit anderen nationalen und internationalen Untersuchungen überein (siehe Abschnitt 2.4). So berichten auch Magenheim u. a. (2018b) von einer unzureichenden technischen Infrastruktur an den Grundschulen und davon, dass sich die Lehrkräfte nicht

immer in der Lage sahen, alle Fragen der Schülerinnen und Schüler zu beantworten. Auch im Abschlussbericht der Explorationsstudie von Murmann u. a. (2018) zum Einsatz des *Calliope mini* in der Grundschule werden Probleme mit der Technik beschrieben. Greifenstein, Graßl und Fraser (2021) berichten ebenfalls über Herausforderungen hinsichtlich der Infrastruktur sowie mangelndes Fachwissen der Lehrkräfte und Herausforderungen hinsichtlich der Schülerinnen und Schüler, z. B. fehlende Geduld. Auffallend ist, dass nur 17% der befragten 200 Lehrkräfte von Herausforderungen im Zusammenhang mit Zeit und Lehrplänen berichteten. Auch in der Studie von Sentance und Csizmadia (2017) werden keine Herausforderungen hinsichtlich einer fehlenden Legitimation der informatischen Inhalte beschrieben. Dies lässt sich durch die Tatsache erklären, dass 70% der befragten Lehrkräfte in der Studie von Greifenstein, Graßl und Fraser im Vereinigten Königreich Großbritannien und Nordirland lebten, und von Sentance und Csizmadia ausschließlich Lehrkräfte in Großbritannien befragt wurden – in beiden Fällen sind informatische Inhalte bereits in den Curricula der Grundschule verankert (siehe Abschnitt 2.2). Auch im Abschlussbericht des Projekts *Informatische Bildung und Technik in der Grundschule* benennen Breiter u. a. (2020, S. 76) die technische Infrastruktur an den Schulen als Gelingensbedingung für die erfolgreiche Durchführung des Projekts und das Testen der darin entwickelten Unterrichtsmaterialien. Die weiteren Gelingensbedingungen, die im Rahmen dieser Teilstudie formuliert wurden, finden sich in den Forderungen von Haselmeier u. a. (2016) wieder. Diese wurden zum damaligen Zeitpunkt jedoch nicht empirisch hergeleitet.

## 17.5 Gewonnene Erkenntnisse zur Unterrichtssequenz

Die Ergebnisse des vierten Zyklus zeigen, dass die formulierten Lernziele für die Zielgruppe angemessen sind. Zudem erwies sich die unterrichtliche Handlungslinie in der Erprobung der Unterrichtssequenz durch die Lehrkräfte als gut durchführbar und zielführend. Auch die zugrundeliegenden *Design Principles* konnten bestätigt werden und die jeweiligen Umsetzungsprinzipien können durch die Arbeit der Lehrkräfte erweitert werden. Die erweiterte Ablaufplanung wurde von vielen Lehrkräften in einigen Punkten angepasst und erweitert. Von einer durchgängigen Ablaufplanung der Unterrichtssequenz wird deshalb im weiteren Verlauf der Arbeit abgesehen. Stattdessen werden verschiedene Umsetzungsmöglichkeiten der unterrichtlichen Handlungslinie zusammengestellt. Für die abschließende Darstellung des Unterrichtskonzepts in Anhang B.1.2 ergeben sich die nachfolgenden Änderungs- bzw. Erweiterungsansätze. Zusätzlich werden in Kapitel 19.3 mögliche Anknüpfungspunkte zum bayerischen Lehrplan der Grundschule aufgezeigt.

- Die *Unplugged*-Aufgaben werden inhaltlich und methodisch erweitert.
- Das Programmieren in Scratch wird durch Transferaufgaben aus der Fortbildung ergänzt.
- Beim Einsatz des Programmierens im Team wird auf eine entsprechende Anleitung verwiesen.
- Das Umsetzen der eigenen Programmierprojekte wird durch mögliche inhaltliche Vorgaben ergänzt.
- Mögliche Anknüpfungspunkte zu zusätzlichen Systemen werden aufgezeigt.

# 18 Erprobung des Fortbildungskonzepts

Um die Praxistauglichkeit und Auswirkungen des Fortbildungskonzepts zu untersuchen, wurde ein Erprobungszyklus im Rahmen des Projekts *AlgoKids – Algorithmen für Kinder* durchgeführt (siehe Abbildung 8.3). In dieser Zeit wurden von der Autorin unterschiedliche Daten gesammelt (siehe Abschnitt 8.5), die in zwei Teilstudien ausgewertet werden: explorative Interviews und vier verschiedene Fragebögen. Es handelt sich dabei um die gleichen Interviews, die bereits mit anderen inhaltlichen Schwerpunkten in Kapitel 17 ausgewertet wurden.

## 18.1 LFB-Zyklus 1

Das Fortbildungskonzept wurde von Mai 2018 bis November 2019 in Zusammenarbeit mit der Akademie für Lehrerfortbildung und Personalführung (ALP)[1] erprobt. Das Staatsministerium für Unterricht und Kultus wählte die teilnehmenden Grundschulen aus, die wiederum schulintern zwei Lehrkräfte zur Teilnahme bestimmten (siehe Abschnitt 17.1).

### 18.1.1 Durchführung

Die Lehrkräfte wurden in drei mehrtägigen Fortbildungsveranstaltungen an der ALP in Dillingen und der Technischen Universität in München entsprechend dem erarbeiteten Konzept der Autorin geschult (siehe Abschnitt 15.3). Eine grafische Übersicht des zeitlichen Ablaufs findet sich in Anhang A.1. Die ALP war für die Organisation der ersten beiden Veranstaltungen verantwortlich, z. B. für die Raumbuchung, Koordination der Übernachtungen der Teilnehmenden und technische Infrastruktur; durchgeführt wurden die Veranstaltungen von der Autorin und einem weiteren Mitarbeiter der TUM-DDI. Darüber hinaus war entweder eine Fachreferentin für die Grundschule oder ein Fachreferent für E-Learning der ALP fast durchgehend anwesend. Die dritte Veranstaltung wurde von der Autorin und demselben Mitarbeiter durchgeführt – erstere übernahm auch deren Organisation. In allen Veranstaltungen wurde jeder Lehrkraft für die praktischen Übungen ein Laptop zur Verfügung gestellt.

Um eine optimale Betreuung sicherzustellen, wurden die Lehrkräfte für die ersten beiden Veranstaltungen in zwei Gruppen aufgeteilt, die in ihrer Zusammensetzung variierten. Die abschließende dritte Fortbildung fand für beide Gruppen gemeinsam statt, um die Erkenntnisse aus dem Projekt *AlgoKids* unter allen beteiligten Lehrkräften diskutieren zu können. Die Lehrkräfte wurden

[1] Die Akademie für Lehrerfortbildung und Personalführung (ALP) ist die zentrale Fortbildungseinrichtung für Lehrkräfte, Funktionsträger und pädagogische Führungskräfte in Bayern und hat darüber hinaus einen Beratungsauftrag im Bereich der Informationstechnik und der Medienpädagogik. Sie untersteht dem Bayerischen Staatsministerium für Unterricht und Kultus (https://alp.dillingen.de/akademie/grundlagen-unserer-arbeit/auftrag/).

zwar für die Teilnahme an den Veranstaltungen freigestellt, konnten jedoch in einigen Fällen aus schulinternen oder privaten Gründen nicht an einzelnen Veranstaltungen teilnehmen:

- Fortbildung 1 an der ALP (insgesamt 11 Stunden Arbeitszeit):
  - Gruppe 1 (16 Lehrkräfte): 07. – 09.05.2018
  - Gruppe 2 (23 Lehrkräfte): 06. – 08.06.2018
- Fortbildung 2 an der ALP (insgesamt 11 Stunden Arbeitszeit):
  - Gruppe 1 (22 Lehrkräfte): 12. – 14.12.2018
  - Gruppe 2 (12 Lehrkräfte): 17. – 19.12.2018
- Fortbildung 3 an der TUM (insgesamt 8,5 Stunden Arbeitszeit):
  - Gesamtgruppe (33 Lehrkräfte): 21. – 22.11.2019

## 18.2 Teilstudie 4: Fortbildungskonzept

Ziel der Studie ist es, die Praxistauglichkeit des Fortbildungskonzepts zu untersuchen:

**(RQ 2-c)** Wie bewährt sich das entwickelte Fortbildungskonzept in der Praxis?

Für die empirische Untersuchung werden explorative Interviews analysiert, die mit den Lehrkräften Im Laufe der Erprobung des Fortbildungskonzepts geführt wurden (siehe Abschnitt 8.5). Es ergeben sich folgende Teilfragen:

- (RQ 2-c-1) Wie äußern sich die Lehrkräfte bezüglich der Rahmenbedingungen der Fortbildung?
- (RQ 2-c-2) Wie äußern sich die Lehrkräfte bezüglich der Lernaktivitäten der Fortbildung?

### 18.2.1 Methodik

Die Interviews wurden mittels der inhaltlich strukturierenden qualitativen Inhaltsanalyse nach Kuckartz (2016, S. 97 ff.) ausgewertet (siehe Abschnitt 8.6.1). In einem ersten Codierprozess wurden sämtliche Textstellen codiert, die sich auf das Fortbildungskonzept beziehen. Die Codiereinheit wurde dabei so groß gewählt, dass der jeweilige Inhalt auch ohne weiteren Kontext zu verstehen war. Zur Codierung der Interviews wurde MAXQDA verwendet, eine Software zur computergestützten qualitativen und *Mixed-Methods*-Datenanalyse. Im Anschluss wurden alle codierten Segmente zusammengestellt und am Material codiert.

### 18.2.2 Ergebnisse

Insgesamt wurden 91 codierte Segmente erfasst, die sich auf 5 Subkategorien verteilen (siehe Tabelle 18.1). Einzelne Codiereinheiten wurden mehreren Subkategorien zugewiesen, sodass die Summe der vergebenen Codes in den Subkategorien die der Hauptkategorie übersteigt. Die Ergebnisse der qualitativen Inhaltsanalyse werden im Folgenden anhand der beiden Forschungsfragen der Teilstudie dargestellt.

*Tabelle 18.1 Induktive Kategorienbeschreibung zu Forschungsfrage RQ 2-c sowie Anzahl der vergebenen Codes und Anzahl der Interviews mit codierten Segmenten*

| Name | Beschreibung | Beispiel |
|---|---|---|
| Format der Fortbildung (32/11) | Äußerungen, die sich auf das Format der Fortbildung beziehen, z. B. die Dauer. | „Und ich glaube auch, diese Begeisterung dafür entwickelt sich auch nur durch diese persönliche Zusammenarbeit und Auseinandersetzung damit. Und nicht – du hockst alleine daheim vorm Computer. Also das kann ich mir überhaupt nicht vorstellen.“ (S13-1_20190716: Pos. 204) |
| Praxisphase (17/8) | Äußerungen, die sich auf die Praxisphasen innerhalb der Fortbildung beziehen. | „Auch diesen *Break* zu haben und auszuprobieren, weil es einfach was ganz anderes ist, wenn ich dann auf einmal allein davor sitze, oder halt mit Schülern. Und dann auch mit ganz anderen Fragen wieder zurückkomme zu dem zweiten Block.“ (S5-1_20190515: Pos. 311) |
| Unterrichts-material (17/8) | Äußerungen, die sich darauf beziehen, dass innerhalb der Fortbildung Unterrichtsmaterialien zur Verfügung gestellt wurden. | „Ich glaube aber, sonst wäre die Hemmschwelle vielleicht auch noch größer gewesen. [...] Das war jetzt wirklich für uns keine große Arbeit. Man konnte das so hernehmen und umsetzen.“ (S5-1_20190515: Pos. 24) |
| Programmier-praxis (18/10) | Äußerungen, die sich darauf beziehen, dass die Lehrkräfte innerhalb der Fortbildung selbst programmiert haben. | „Auch so generell auch die Sachen. Also nur lesen glaube ich funktioniert überhaupt nicht beim Programmieren. Sondern man muss es glaube ich auch gemacht haben. Ja.“ (S12-1_20190627: Pos. 290) |
| Auswahl der Inhalte (20/8) | Äußerungen, die sich auf die Auswahl der behandelten Themen beziehen. | „Und was ich auch echt schön fand, den Überblick, was ist sonst noch so auf dem Markt. Das fand ich schon echt nochmal interessant.“ (S14-1_20190312: Pos. 152) |

*(RQ 2-c-1) Wie äußern sich die Lehrkräfte bezüglich der Rahmenbedingungen der Fortbildung?*
Die Durchführung der Fortbildung als Präsenzveranstaltung und die Teilnahme als Tandem wurde von den Lehrkräften sehr positiv aufgenommen. Durch die Übernachtung am Fortbildungsort konnten sie sich voll und ganz auf das Thema einlassen, ohne durch andere Dinge abgelenkt zu werden:

> „Ja, dann auch die Fortbildungstage, da sind uns dann auch einfach so viele Ideen gekommen, weil man hat sich doch konstant nur mit dem beschäftigt. Und der Schulalltag so hintenansteht, oder man den nicht so präsent im Kopf hat. Und dann waren wir nur damit irgendwie beschäftigt.“ S6-1_20190527: Pos. 370

Es wurde bezweifelt, dass die Inhalte im Rahmen eines Selbstlernkurses vermittelt werden können bzw. dass der Zeitaufwand in gleichem Maße betrieben worden wäre. Zudem waren einige Lehrkräfte der Meinung, dass sich innerhalb eines Selbstlernkurses nicht die gleiche Motivation und Lernfreude eingestellt hätte.

Die Praxisphase zwischen der ersten und zweiten Fortbildungsveranstaltung wurde als sehr positiv empfunden. Diesbezüglich wurde zum einen genannt, dass sich die Lehrkräfte ohne die Pause von der Menge an Inhalten überfordert gefühlt hätten, zum anderen ergaben sich aus der Erprobung in der Praxis Fragen, die in der zweiten Veranstaltung aufgegriffen werden konnten. Im Austausch mit den Anderen erhielten die Lehrkräfte zudem viele Anregungen für die eigene Praxis. Darüber hinaus wurde berichtet, dass dadurch eine gewisse Verbindlichkeit hergestellt wurde, Elemente der Unterrichtssequenz im eigenen Unterricht zu erproben:

> „Der Termin kommt immer näher, jetzt muss man was machen. Jetzt muss man es einbauen, ob es geht oder nicht." S4-1_20190409: Pos. 214

*(RQ 2-c-2) Wie äußern sich die Lehrkräfte bezüglich der Lernaktivitäten der Fortbildung?*
Die Lehrkräfte bezeichneten es als unerlässlich, dass während der Fortbildung Unterrichtsmaterialien zur Verfügung gestellt wurden. Einerseits konnten sie ohne lange Vorbereitungen im Unterricht mit dem Programmieren beginnen, andererseits wurde stark bezweifelt, dass sie ohne Anregungen in der Lage gewesen wären, eigenes Material zu entwickeln:

> „[...] Ich habe es im Endeffekt genau so in einer Sequenz gemacht, wie ihr uns es im Endeffekt beigebracht habt. Ich weiß nicht, ob ich mir das selber so erarbeiten hätte können. Das war ja ganz strukturiert, kleinschrittig – wie kann man vorgehen, viele Beispiele dazu. Ich weiß nicht, ob das überhaupt anders möglich gewesen wäre, wenn man sich's selber erarbeitet. " S1-1_20190515: Pos. 153

Die Bereitstellung von Unterrichtsmaterialien ohne ein entsprechendes Fortbildungsangebot wurde jedoch als nicht zielführend eingeschätzt. Das eigene Durchlaufen der Unterrichtssequenz innerhalb der Fortbildung wurde positiv gesehen, da man auf diese Weise den Lernprozess der Schülerinnen und Schüler verstehen und Sicherheit für das Unterrichten gewinnen konnte. Letzteres, weil viele Fragen und Probleme, die im Unterricht auftauchen, bereits während der Fortbildung geklärt werden konnten.

Die Lehrkräfte empfanden es darüber hinaus als sehr sinnvoll und notwendig, dass sie im Rahmen der Fortbildung die Möglichkeit hatten, sich intensiv mit Scratch zu beschäftigen und eigene Programmiererfahrungen zu sammeln. Positiv wurde zudem bewertet, dass das Niveau der Programmieraufgaben über Grundschulniveau lag, da sie sich gut gerüstet fühlten, die im Vergleich leichteren Aufgaben der Unterrichtssequenz zu meistern. Es war zudem möglich, sich im Sinne eines Modelllernens an den Hilfestellungen der Kursleitung zu orientieren und Anregungen für die eigene Unterrichtspraxis zu erhalten:

> „[Es] war dann doch für mich irgendwie überraschend [...], dass man dann merkt, wenn man es dann macht, dass man eigentlich doch ganz gut aufgestellt ist. Durch

> diese Fortbildungen einfach schon. Dass man den Kindern die Fragen größtenteils doch einigermaßen kompetent beantworten kann, dass man ihnen kleine Tipps geben kann, wie sie da hinkommen. Und ich habe mich dann komischerweise auch manchmal zurückversetzt, wie ich in der Situation war und irgendwas wollte und dann dich gefragt habe. Genau. Und dann habe ich mir auch mal gedacht, nein ich darf jetzt nicht ihr die Maus wegnehmen und das selber machen." S4-1_20190409: Pos. 285

Insgesamt war die Resonanz auf die Auswahl der Fortbildungsinhalte sehr positiv. Bei der zweiten Fortbildungsveranstaltung wurden insbesondere die flexibleren Aufgabenformate und das Kennenlernen weiterer Möglichkeiten zum Programmieren in der Grundschule positiv hervorgehoben. Für letzteres hätten sich einzelne Lehrkräfte mehr Zeit gewünscht. In einem Interview wurde die Idee geäußert, in zukünftigen Fortbildungen eigene Ideen zur Integration des Programmierens in den Unterricht zu entwickeln und auszutauschen:

> *Lehrkraft 1:* „Also ich fände es total gut, wenn man eben so Anwendungsbeispiele erarbeiten würde. Oder dass man einfach sagt, wo kann ich das Programmieren richtig in den Unterricht integrieren. Zum Beispiel ein Projekt im Sachunterricht. Was könnte ich programmieren, zum Beispiel eine Wald-App oder irgendwas. Wo man sagt, OK, man macht so eine Art Quiz oder sowas. Also dass man wirklich tatsächlich so Anknüpfungspunkte entwickelt, so ganz konkret. "
> *Lehrkraft 2:* „Wenn man sich vorher vielleicht was überlegt und man würde dann trotzdem zu einem festen Zeitpunkt irgendwie zusammenkommen, dass man sich dann austauschen kann. Also dass man schon so ein bisschen auch die Pflicht hat, OK, bis dahin muss ich vielleicht mir schon mal was überlegt haben [...]. Dass man da so einen ganz großen Pool an Möglichkeiten bekommt." S7-1_20190524: Pos. 161,165

### 18.2.3 Zusammenfassung und Diskussion

Der Fokus dieser Teilstudie liegt auf den Äußerungen der Lehrkräfte bzgl. des Fortbildungskonzepts und seiner Praxistauglichkeit. Durch die Auswertung der explorativen Interviews konnten darin Meinungen sowohl hinsichtlich der Rahmenbedingungen als auch der Lernaktivitäten der Fortbildung identifiziert werden. In Bezug auf die Rahmenbedingungen bewerteten die Lehrkräfte vor allem die Durchführung als Präsenzveranstaltung, die Teilnahme im Tandem und die Praxisphase zwischen der ersten und zweiten Fortbildungsveranstaltung als positiv. Bzgl. der Lerninhalte wurde die Arbeit mit Unterrichtsmaterialien, das Sammeln von Programmierpraxis sowie Kennenlernen von unterschiedlichen Aufgabenformaten und anderer Möglichkeiten zum Programmieren in der Grundschule positiv bewertet.

In Bezug auf die Erhebungsmethode ist anzumerken, das darin nicht explizit nach dem Fortbildungskonzept gefragt wurde. Da dies nur in zwölf von neunzehn Interviews thematisiert wurde,

besteht hier kein Anspruch auf Vollständigkeit. Darüber hinaus sind die Erhebungszeitpunkte von Relevanz – zwei Interviews wurden vor der zweiten Fortbildungsveranstaltung geführt, die restlichen siebzehn zwischen der zweiten und dritten (siehe Abschnitt 8.5.2). Es konnten darin also keine Einschätzungen zur dritten Veranstaltung getroffen werden. Als Auswertungsmethode für die Daten hat sich die inhaltlich strukturierende qualitative Inhaltsanalyse bewährt.

Um Rückschlüsse auf die *Design Principles* ziehen zu können, die der Gestaltung des Fortbildungskonzepts zugrunde lagen, wird in einem nächsten Schritt zusammengestellt, welche Gestaltungsprinzipien bzw. Umsetzungen davon als positiv thematisiert wurden:

- Die Fortbildung ermöglicht eine vertiefte Auseinandersetzung mit den Inhalten (LFB-DP 1):
  - verschiedene mehrtägige Termine
  - Durchführung nicht an der Schule der Teilnehmenden
  - zeitlicher Abstand zwischen Terminen um Erprobung zu ermöglichen
  - mehrere Lehrkräfte einer Schule nehmen teil
- Die Lehrkräfte erhalten eine fortlaufende Unterstützung (LFB-DP 2):
  - Besprechen von Fragen, die sich während Erprobung ergeben
  - mehrere Lehrkräfte einer Schule nehmen teil
- Die Lehrkräfte folgen dem Erkenntnisgewinn der Schülerinnen und Schüler (LFB-DP 3):
  - Bearbeiten von Unterrichtsmaterialien
  - Möglichkeiten, Programmierpraxis zu sammeln
  - Hilfestellung durch Kursleitung
- Die Fortbildung verschränkt Fachwissenschaft und Fachdidaktik (LFB-DP 4):
  - Eigenständiges Bearbeiten von Programmieraufgaben
  - Kennenlernen von verschiedenen Aufgabenformaten
  - Kennenlernen weiterer Systeme zum Programmieren im Unterricht
- In der Fortbildung werden Reflexionsprozesse angeregt (LFB-DP 5):
  - Bearbeiten von Unterrichtsmaterialien
  - Einfordern der Erprobung im Unterricht
  - Erfahrungsaustausch anleiten
- Die Fortbildung knüpft an Bekanntes an (LFB-DP 6):
  - Wunsch nach konkreter Anknüpfung an den Lehrplan

In der Auflistung zeigt sich, dass sämtliche Kernelemente der *Design Principles* LFB-DP 1-5 von den Lehrkräften als positiv bewertet wurden. Die praktische Umsetzung von LFB-DP 6 erfolgte vorrangig in der dritten Fortbildungsveranstaltung. Da diese zum Erhebungszeitpunkt noch nicht stattgefunden hatte, konnten sich die Lehrkräfte dazu nicht äußern. Der Wunsch nach konkreten Anknüpfungspunkten zum Lehrplan der Grundschule, der in einem Interview artikuliert wurde, wurde darin berücksichtigt.

## 18.3 Teilstudie 5: Auswirkungen auf Lehrkräfte

Ziel der Studie ist es, die Auswirkungen des Fortbildungskonzepts auf Lehrkräfte zu untersuchen:

**(RQ 2-e)** Welche Auswirkungen hat das Fortbildungskonzept auf Lehrkräfte?

Die Fragestellung wird sowohl qualitativ als auch quantitativ untersucht. Zum einen werden explorative Interviews analysiert, die mit den Lehrkräften im Laufe der Erprobung des Fortbildungskonzepts geführt wurden, zum anderen werden Fragebögen ausgewertet, die von den Lehrkräften während des Erprobungszeitraums ausgefüllt wurden (siehe Abschnitt 8.5). Letztere beinhalten Items in Bezug auf verschiedene Aspekte: (1) Sicherheit im Umgang mit Computern und Computeranwendungen, (2) Einschätzung der Programmierfähigkeit und (3) Selbstwirksamkeit in Bezug auf die Umsetzung der Fortbildungsinhalte im Unterricht. Bei beiden Erhebungen handelt es sich also um Selbsteinschätzungen der Auswirkungen durch die Lehrkräfte. Von einer Untersuchung der Fragestellung, z. B. im Sinne einer Kompetenzmessung, wird an dieser Stelle abgesehen. Es ergeben sich folgende Teilfragen:

- (RQ 2-e-1) Von welchen Auswirkungen des Fortbildungskonzepts berichten die Lehrkräfte?
- (RQ 2-e-2) Wie wirkt sich das Fortbildungskonzept auf die Sicherheit der Lehrkräfte im Umgang mit Computern und Computeranwendungen aus?
- (RQ 2-e-3) Wie wirkt sich das Fortbildungskonzept auf die Selbstwirksamkeitserwartung der Lehrkräfte in Bezug auf ihre Programmierfähigkeiten aus?
- (RQ 2-e-4) Wie wirkt sich das Fortbildungskonzept auf die Selbstwirksamkeitserwartung der Lehrkräfte in Bezug auf das Umsetzen der Fortbildungsinhalte im Unterricht aus?

### 18.3.1 Methodik

Die Interviews wurden durch eine inhaltlich strukturierende qualitative Inhaltsanalyse nach Kuckartz (2016, S. 97 ff.) ausgewertet (siehe Abschnitt 8.6.1). In einem ersten Codierprozess wurden sämtliche Textstellen codiert, die sich auf die Auswirkungen der Fortbildung beziehen. Die Codiereinheit wurde dabei so groß gewählt, dass der jeweilige Inhalt auch ohne weiteren Kontext zu verstehen war. Zur Codierung der Interviews wurde MAXQDA verwendet, eine Software zur computergestützten qualitativen und *Mixed-Methods*-Datenanalyse. Im Anschluss wurden alle codierten Segmente zusammengestellt und am Material codiert. Die Ergebnisse der qualitativen Inhaltsanalyse werden ergänzt durch ausgewählte Items des Fragebogen FB-LK1 (siehe Abschnitt A.3.1) und FB-LK4 (siehe Abschnitt A.3.4), den die Lehrkräfte nach der ersten und dritten Fortbildungsveranstaltung ausgefüllt haben. Die Analyse der Fragebögen anhand von Mittelwerten und Standardabweichungen sowie Mittelwertvergleichen mithilfe des $t$-Tests für abhängige Stichproben und der einfaktoriellen Varianzanalyse mit Messwiederholung wurden mit dem Statistikanalyseprogramm SPSS 29 durchgeführt. Die angegebenen Mittelwerte reichen von eins bis fünf, wobei eins den kleinstmöglichen und fünf den größtmöglichen Wert darstellt. Zur Berechnung der Mittelwertindizes der einzelnen Skalen wurden invers formulierte Items invertiert.

### 18.3.2 Ergebnisse

Die Ergebnisse der Teilstudie werden im Folgenden anhand der einzelnen Forschungsfragen dargestellt.

*(RQ 2-e-1) Von welchen Auswirkungen des Fortbildungskonzepts berichten die Lehrkräfte?*
Insgesamt wurden 43 codierte Segmente erfasst, die sich auf 4 Subkategorien verteilen (siehe Tabelle 18.2). Einzelne Codiereinheiten wurden mehreren Subkategorien zugewiesen, sodass die Summe der vergebenen Codes in den Subkategorien die der Hauptkategorie übersteigt. Die Ergebnisse der qualitativen Inhaltsanalyse werden im Folgenden anhand von Ankerbeispielen dargestellt.

*Tabelle 18.2 Induktive Kategorienbeschreibung zu Forschungsfrage RQ 2-e-1 sowie Anzahl der vergebenen Codes und Anzahl der Interviews mit codierten Segmenten*

| Name | Beschreibung | Beispiel |
|---|---|---|
| Spaß und Motivation (19/8) | Äußerungen, die Rückschlüsse auf Veränderungen in der Motivation und im Spaß am Programmieren oder am Unterrichten der Programmierung erlauben. | „Ich glaube so überhaupt, dass es mir Spaß macht mit dem Programmieren habe ich schon neu an mir entdeckt. Hätte ich jetzt gar nicht gedacht.“ (S6-1_20190527: Pos. 252) |
| Selbstwirksamkeitserwartung (14/9) | Äußerungen, die Rückschlüsse auf die Veränderung der Selbstwirksamkeitserwartung bzgl. des Programmierens oder des Unterrichtens der Programmierung erlauben. | „Ja, ich finde dadurch, dass wir das ja auch alles ausprobiert haben und dadurch, dass wir das wirklich, also wir haben es ja wirklich fast 1:1 übernommen. [...] Also ich habe mich echt sicher gefühlt.“ (S16-1_20180724: Pos. 240) |
| Wissen und Fertigkeiten (12/8) | Äußerungen, die Auswirkungen der Fortbildung auf Wissen und Fertigkeiten der Lehrkräfte beschreiben. | „Ich würde sagen, ein Erfolgserlebnis wäre ‚Hurra! Ich kann programmieren.‘ Nein, auch dass man wirklich was dazu gelernt hat und ein ganz neuer Bereich einem eröffnet worden ist, den man hoffentlich weitergeben kann.“ (S5-1_20190515: Pos. 255) |
| Interesse (6/4) | Äußerungen, die Rückschlüsse auf die Veränderung des Interesses am Programmieren oder am Unterrichten des Programmierens erlauben. | „Und das ist für einen selber so schön gewesen das zu machen und [...] dass das einem Spaß macht, wo man [...] schon auch als Frau tatsächlich die Einstellung hatte: ‚Nee, interessiert mich gar nicht.‘ Doch! Interessiert mich schon! Ich möchte programmieren, ich möchte das verstehen!“ (S15-1_20190529: Pos. 253) |

Spass und Motivation: In acht Interviews berichteten die Lehrkräfte, dass sie nach der Teilnahme an den Fortbildungen sehr motiviert waren, das Programmieren im Unterricht zu erproben und dass sie festgestellt haben, dass ihnen die Materie Spaß macht. Dies bestätigte sich auch in der Auswertung des Fragebogens FB-LK4. Hier gaben nahezu alle befragten Lehrkräfte an, dass sie

motiviert waren, die Inhalte der Fortbildung umzusetzen, und dass ihnen die Umsetzung der Fortbildungsinhalte Freude bereitete (siehe Abbildungen 18.1 und 18.2). Einige Lehrkräfte betonten, dass sie dies im Vorfeld nicht erwartet hatten und mit einer eher negativen Einstellung in die erste Fortbildung gegangen waren:

> „Weil ich bin ja eigentlich auch so ein bisschen negativ hingefahren. Habe ich gedacht, was soll ich da. Bin ja eingesprungen so ungefähr. War ja nicht der Kandidat, der da eigentlich zuerst hinfahren sollte. Und dann dachte ich mir schon: ‚Oh, meine Güte. Programmieren, ich und Programmieren.' [...] Aber dadurch, dass wir das dann von euch bekommen haben, dass wir da wirklich eine Hilfestellung gerade am Anfang hatten mit Material und so. Hat das dann auch wirklich ganz schnell sich gewandelt in Spaß haben, in Interesse, in Neugier und so weiter." S4-1_20190409: Pos. 247-249

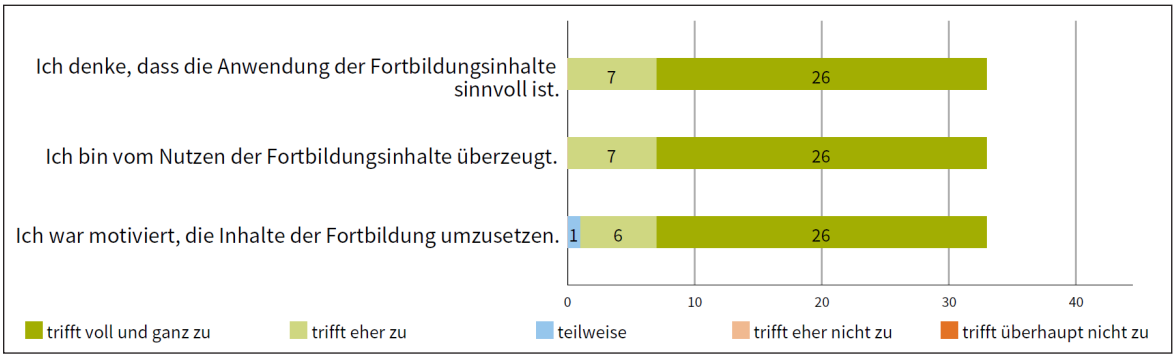



*Abbildung 18.1 Angaben zur Motivation und Überzeugungen (n=33; FB-LK4)*

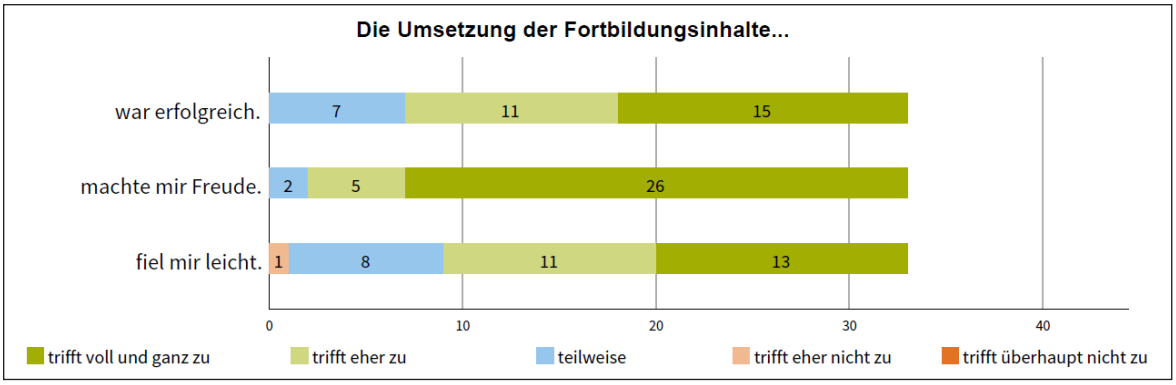



*Abbildung 18.2 Angaben zum Empfinden der Umsetzung der Fortbildungsinhalte (n=33; FB-LK4)*

SELBSTWIRKSAMKEITSERWARTUNG: In neun Interviews wurde berichtet, dass die Lehrkräfte sich durch die Fortbildungen sehr sicher im Programmieren gefühlt hatten und sich zutrauten, die Thematik im Unterricht zu behandeln – insbesondere, da sie die Unterrichtsmaterialien selbst ausprobiert hatten. Einige Lehrkräfte berichteten, dass sie sich jedes Mal nach den Fortbildungen wesentlich sicherer fühlten und sich daher regelmäßige Fortbildungsangebote wünschten. Die Lehrkräfte einer Schule berichteten, dass sie sich bereits nach der ersten Fortbildungsveranstaltung sehr sicher gefühlt hätten und die zweite Fortbildung nicht unbedingt benötigten. An einer anderen

Schule berichteten die Lehrkräfte wiederum, dass sie sich vor allem nach der zweiten Fortbildung sehr sicher fühlten, da sie darin vermehrt schwierigere Aufgaben bewältigt hatten. Darüber hinaus wurde als hilfreich empfunden, dass in der Fortbildung thematisiert wurde, dass auch die Kursleitung nicht immer auf alle Fragen eine Antwort hat:

> „Nee, aber ich glaube, da habt ihr uns eigentlich auch ein gutes Gefühl gegeben. Weil ihr ja auch gesagt habt, wenn ihr etwas nicht findet, dann naja, googeln wir halt mal. Beziehungsweise der [Kursleiter] hat das ja auch ein paar Mal gesagt. Er macht das dann auch so. Und da dachte ich mir, naja, wenn der das so macht. [...] Und eigentlich klappt es echt, man kuckt dann halt. Und es geht dann meistens auch."
>
> S14-1_20190312: Pos. 74

Wissen und Fertigkeiten: In acht Interviews wurde thematisiert, dass man durch die Fortbildungen Kompetenzen bzgl. des Programmierens und dessen praktischer Umsetzung im Unterricht erworben hat. Dabei sprachen einige Lehrkräfte auch an, dass sie im Vorfeld keine Vorstellung über das Programmieren in der Grundschule gehabt hatten:

> „Ich habe ja überhaupt nicht gewusst, dass es sowas quasi so kindgerecht überhaupt gibt. Also für mich war programmieren immer irgendwas mit schwarzem Bildschirm und ich gebe da ein – XY 1212. [...] Von daher war das der totale Aha-Effekt, was es da gibt, oder was es überhaupt für Möglichkeiten gibt, und wie kindgerecht das auch alles sein kann. Also wie gesagt, als wir da hingekommen sind, also ich war so ein unbeschriebenes weißes Blatt Papier." S13-1_20190716: Pos. 164-166

Zwei Lehrkräfte stellten zudem fest, dass sich ihre Arbeitsweise im Laufe des Projekts verändert hat:

> *Lehrkraft 1:* „Ich merke das schon an mir, dass ich früher viel chaotischer in der ganzen Arbeit an sich war. Dass ich aber jetzt nicht unbedingt, weil ich mir das vorgenommen habe, oder so. Dass ich den Eindruck habe, dass ich strukturierter arbeite."
> *Lehrkraft 2:* „[...] Bei den Kindern – [...] dass man vielleicht auch selber nochmal drüber nachdenkt: ‚Vielleicht sage ich ihnen das auch nochmal. Vielleicht fehlt da ein Schritt.' Weil mir ist es klar, dass der Schritt dann noch kommt, aber vielleicht ist es ihnen gar nicht so klar, dass das noch kommt." S12-1_20190627: Pos. 214-215

Die Ergebnisse der Interviews werden auch durch die Auswertung des Fragebogens FB-LK1 bestätigt, den die Lehrkräfte nach der ersten Fortbildungsveranstaltung ausfüllten. Alle Lehrkräfte gaben an, in der Fortbildung viel gelernt zu haben und die Inhalte mindestens teilweise verstanden zu haben (siehe Abbildung 18.3). Zudem war es allen Lehrkräften durch die Teilnahme mindestens teilweise möglich, die Fortbildungsinhalte an der Schule anzuwenden sowie geeignete Ziele für die Umsetzung zu definieren und notwendige Arbeitsschritte für die Umsetzung zu planen (siehe Abbildung 18.4).

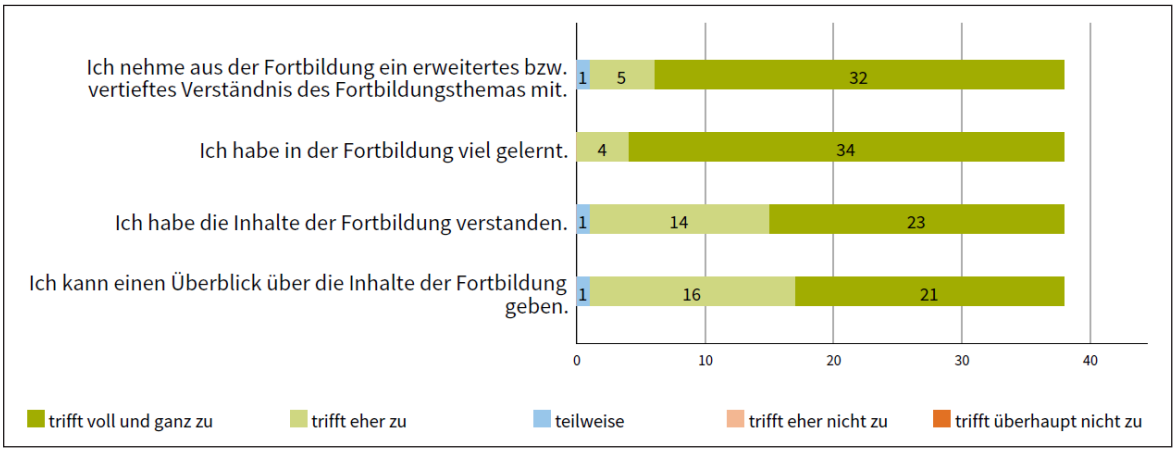



*Abbildung 18.3 Angaben zum Wissenszuwachs nach der ersten Fortbildungsveranstaltung (n=38; FB-LK2)*

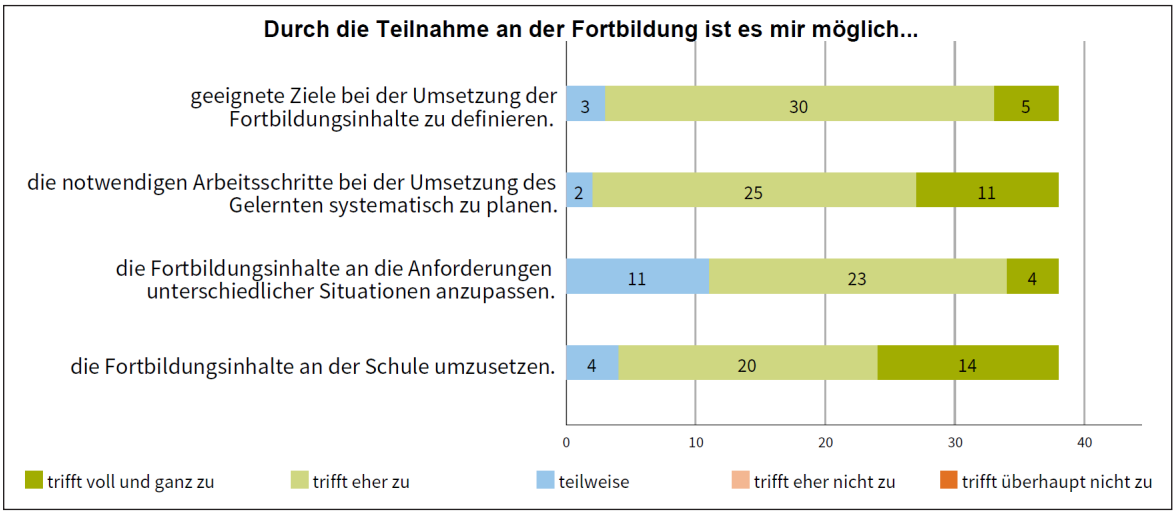



*Abbildung 18.4 Angaben zur Befähigung zur Anwendung der Fortbildungsinhalte nach der ersten Fortbildungsveranstaltung (n=38; FB-LK2)*

INTERESSE: Vier Lehrkräfte berichteten, dass sich im Laufe des Projekts *AlgoKids* Interesse für das Programmieren und dessen Umsetzung in der Grundschule entwickelt hat:

> „Also für mich? Wahnsinnig spannend. Ich hätte nie gedacht, dass ich mich überhaupt für sowas interessieren kann. Und ich bin auf alle Fälle so motiviert. Ich habe viel dazu gelernt. Ich habe echt rumrecherchiert ohne Ende." S10-1_20190607: Pos. 186

*(RQ 2-e-2) Wie wirkt sich das Fortbildungskonzept auf die Sicherheit der Lehrkräfte im Umgang mit Computern und Computeranwendungen aus?*

Die Sicherheit der Lehrkräfte im Umgang mit Computern und Computeranwendungen wurde zu Beginn und zum Ende des Projekts *AlgoKids* in den Fragebögen FB-LK1 und FB-LK4 erhoben. Dazu wurde die Computerängstlichkeitsskala COMA verwendet, die der revidierten Fassung des Inventar zur Computerbildung (INCOBI-R) entnommen wurde (Richter, Naumann und Horz 2010). Die subjektive Sicherheit im Umgang mit dem Computer wird darin als Abwesenheit von

Computerängstlichkeit definiert (ebd., S. 24). Da nicht alle teilnehmenden Lehrkräfte an jeder Erhebung teilnehmen konnten, wurden einige Datensätze für die Analyse ausgeschlossen. Die bereinigte Stichprobe beträgt $n$=33. Abbildung 18.5 zeigt die einzelnen Items der Skala sowie die Verteilung der Antworten zu beiden Messzeitpunkten.

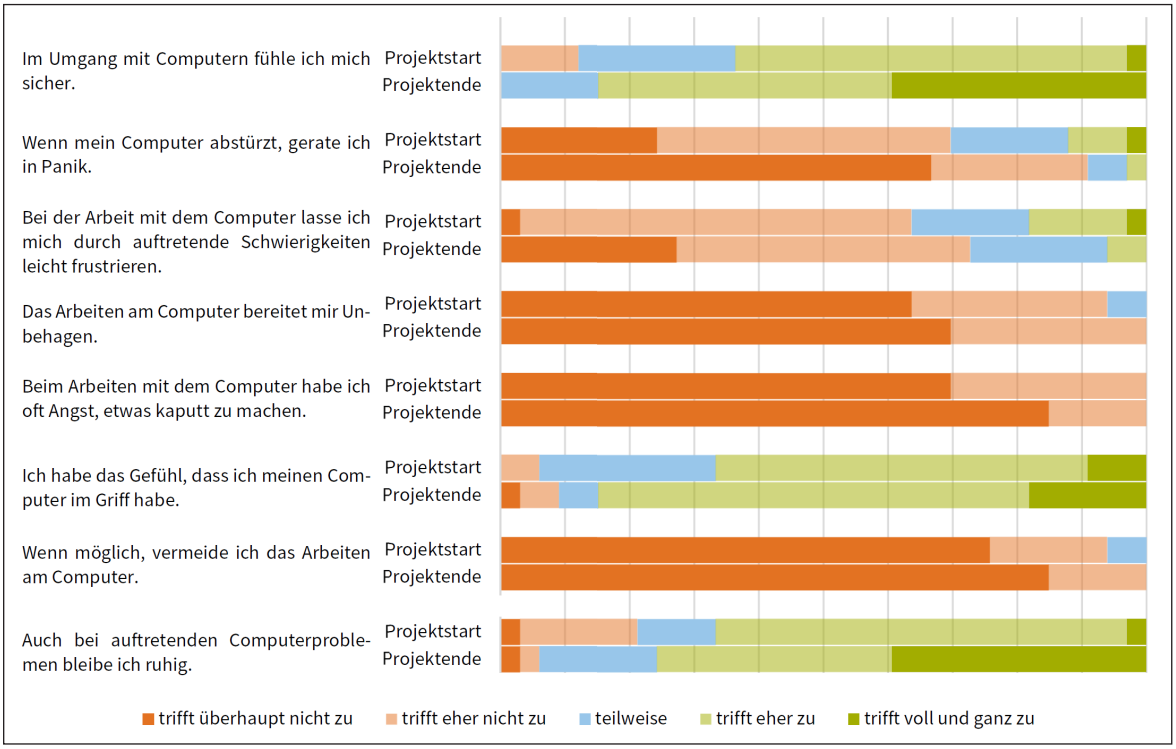



*Abbildung 18.5 Angaben zur Sicherheit im Umgang mit Computern und Computeranwendungen (n=33; FB-LK1 und FB-LK4)*

Um zu untersuchen, ob sich die Computerängstlichkeit der Lehrkräfte nach der Teilnahme am Projekt *AlgoKids* vermindert hat, werden die Mittelwerte der Skala zum Messzeitpunkt 1 und 2 untersucht (siehe Abbildung 18.6). Um zu überprüfen, ob ein signifikanter Abfall der Computerängstlichkeit feststellbar ist, wird ein $t$-Test für abhängige Stichproben durchgeführt. Voraussetzung hierfür ist zum einen, dass die Differenz zwischen den gepaarten Werten normalverteilt ist. Laut dem Kolmogorov-Smirnov-Test ist die Differenz auf einem 5%-Signifikanzniveau normalverteilt ($p$ = 0.20), ebenso laut dem Shapiro-Wilk-Test ($p$ = 0.56). Zum anderen müssen die Daten intervallskaliert sein. Die Beantwortung der Items zur Computerängstlichkeit erfolgt über eine 5-stufige Likert-Skala – die Items sind also ordinalskaliert. Laut Sedlmeier und Renkewitz (2008, S. 65) ist es jedoch üblich, Messungen mit Rating-Skalen als intervallskallierte Daten zu interpretieren.

Es zeigt sich, dass die Teilnahme am Projekt *AlgoKids* einen statistisch signifikanten Einfluss auf die Computerängstlichkeit hat ($t$ = 6.147, $p$ < 0.001, $n$ = 33). Nach der Projektteilnahme ($M$ = 1.61, $SD$ = 0.44) weisen die Lehrkräfte eine signifikant geringere Computerängstlichkeit auf, als vor der Projektteilnahme ($M$ = 2.01, $SD$ = 0.40). Die Effektstärke nach Cohen (1988) liegt bei $r$ = 0,73 und entspricht damit einem starken Effekt. Der Logik von Richter, Naumann und Horz (2010) folgend, weisen sie eine gesteigerte Sicherheit im Umgang mit Computern und Computeranwendungen auf.

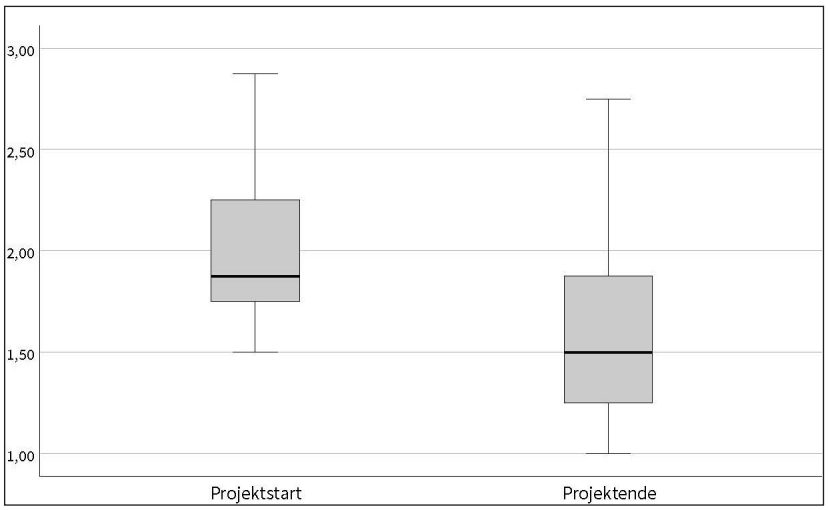



*Abbildung 18.6 Mittelwerte der Selbsteinschätzung zur Computerängstlichkeit (n=33; FB-LK1 und FB-LK4)*

*(RQ 2-e-3) Wie wirkt sich das Fortbildungskonzept auf die Selbstwirksamkeitserwartung der Lehrkräfte in Bezug auf ihre Programmierfähigkeiten aus?*

Die Selbstwirksamkeitserwartung der Lehrkräfte bzgl. ihrer Programmierfähigkeit wurde nach der ersten Fortbildung im Fragebogen FB-LK2, vor der zweiten Fortbildung und somit nach der ersten Praxisphase in Fragebogen FB-LK3 sowie nach der dritten Fortbildung in Fragebogen FB-LK4 erhoben. Dazu wurde die *Computer Programming Self-Efficacy Scale (CPSES)* von Tsai, Wang und Hsu (2017) aus dem Englischen ins Deutsche übersetzt und um zwei projektspezifische Items ergänzt. Da nicht alle teilnehmenden Lehrkräfte an jeder Erhebung teilnehmen konnten, wurden einige Datensätze für die Analyse ausgeschlossen. Die bereinigte Stichprobe beträgt $n$=25. Abbildung 18.7 zeigt ausgewählte Items der Skala sowie die Verteilung der Antworten zu den Messzeitpunkten.

Um zu untersuchen, ob bzw. wie sich die Selbstwirksamkeitserwartung der Lehrkräfte hinsichtlich ihrer Programmierfähigkeit im Laufe des Projekts *AlgoKids* verändert hat, werden die Mittelwerte der Skala zu drei verschiedenen Messzeitpunkten untersucht (siehe Abbildung 18.8). Um zu überprüfen, ob eine signifikante Veränderung feststellbar ist, wird eine einfaktorielle Varianzanalyse mit Messwiederholung durchgeführt. Voraussetzung hierfür ist, dass die Daten intervallskaliert sind. Wie bereits bei der Untersuchung der vorigen Forschungsfrage, werden die Items als intervallskalierte Daten interpretiert, obwohl sie über eine 5-stufige Likert-Skala erfasst wurden. Zudem müssen die Residuen der abhängigen Variable innerhalb jedes Messzeitpunktes normalverteilt sein. Nach dem Kolmgorov-Smirnov-Test ist dies bei allen Messungen der Fall ($p1$ = 0.200, $p2$ = 0.200, $p3$ = 0.113), ebenso nach dem Shapiro-Wilk-Test ($p1$ = 0.405, $p2$ = 0.889, $p3$ = 0.201). Dies bestätigt sich sowohl bei der Sichtung der jeweiligen Histogramme als auch bei der Berechnung der Quotienten aus der Kurtosis und deren Standardfehler sowie der Schiefe und deren Standardfehler. Die letzte Voraussetzung für die Anwendung der einfaktoriellen Varianzanalyse ist die Sphärizität, die mittels des Mauchly-Test auf Sphärizität bestimmt wird. Dieser ist für die vorliegenden Daten nicht signifikant, sodass Sphärizität angenommen wird ($Mauchly\text{-}W(2)$ = 0.933, $p$ = 0.451).

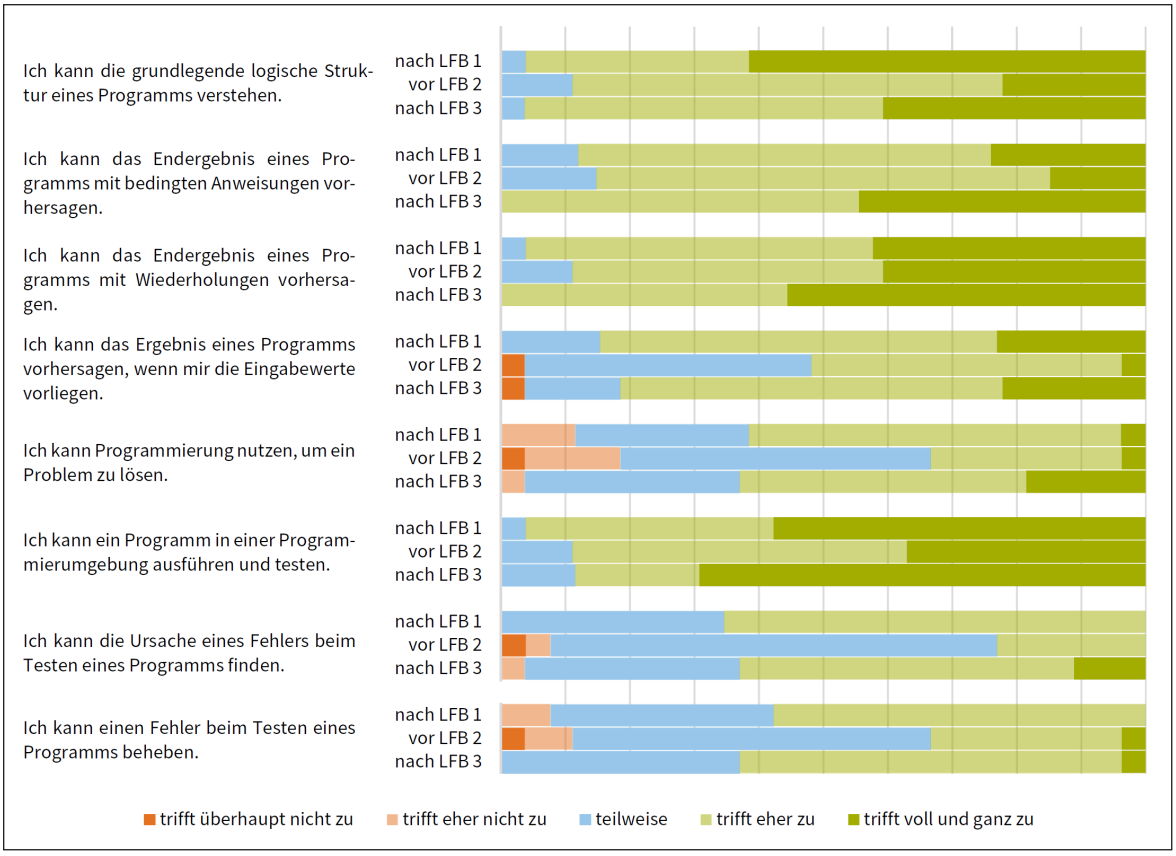



*Abbildung 18.7 Angaben zur eigenen Programmierfähigkeit (n=25; FB-LK2, FB-LK3 und FB-LK4)*

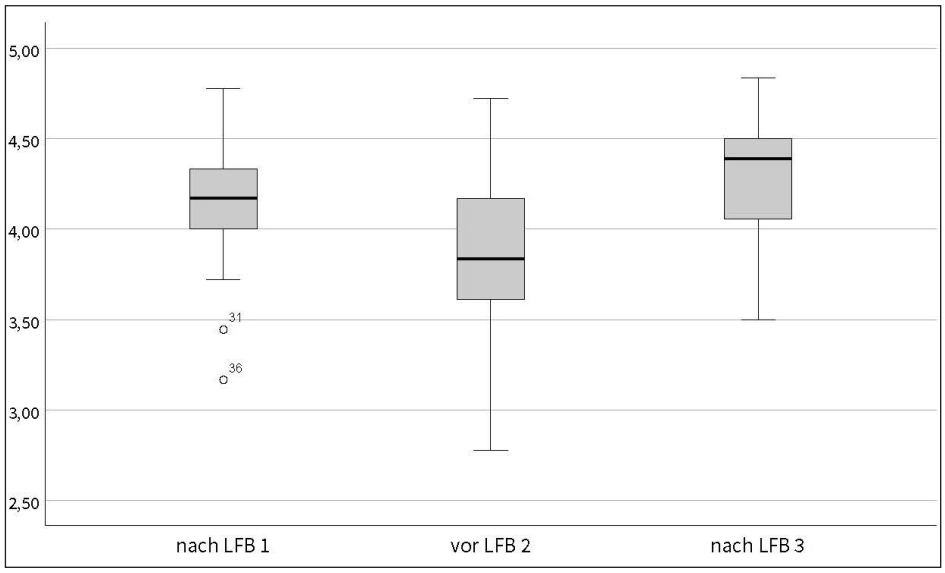



*Abbildung 18.8 Mittelwerte der Selbsteinschätzung zur eigenen Programmierfähigkeit (n=25; FB-LK2, FB-LK3 und FB-LK4)*

Es zeigt sich, dass sich die Selbstwirksamkeitserwartung der Lehrkräfte bzgl. ihrer Programmierfähigkeit zu den Messzeitpunkten signifikant unterscheidet ($F$(2,48) = 17.871, $p$ = <0,001, $\eta_p^2$ = 0.427, $n$ = 25). Bonferroni-korrigierte paarweise Vergleiche zeigen, dass die Selbstwirksamkeit bzgl. der Programmierfähigkeit nach der ersten Fortbildung ($M$ = 4.111, $SD$ = 0.391) signifikant größer ist als vor der zweiten Fortbildung ($M$ = 3.797, $SD$ = 0.481). Vor der zweiten Fortbildung ist sie wiederum signifikant geringer als nach der dritten Fortbildung ($M$ = 4.265, $SD$ = 0.372). Zwischen dem ersten und dem dritten Messzeitpunkt unterscheidet sich die Selbstwirksamkeit nicht signifikant. Die Effektstärke $f$ nach Cohen (1988) liegt bei 0.86 und entspricht einem starken Effekt.

*(RQ 2-e-4) Wie wirkt sich das Fortbildungskonzept auf die Selbstwirksamkeitserwartung der Lehrkräfte in Bezug auf das Umsetzen der Fortbildungsinhalte im Unterricht aus?*
Die Selbstwirksamkeitserwartung der Lehrkräfte bzgl. der Umsetzung der Fortbildungsinhalte wurde nach der ersten Fortbildung im Fragebogen FB-LK2, vor der zweiten Fortbildung und somit nach der ersten Praxisphase in Fragebogen FB-LK3 sowie nach der dritten Fortbildung in Fragebogen FB-LK4 erhoben. Dazu wurde eine Skala entworfen, die in Anlehnung an Jerusalem u. a. (2009) konstruiert wurde. Da nicht alle teilnehmenden Lehrkräfte an jeder Erhebung teilnehmen konnten, wurden einige Datensätze für die Analyse ausgeschlossen. Die bereinigte Stichprobe beträgt $n$=26. Abbildung 18.9 zeigt die einzelnen Items der Skala sowie die Verteilung der Antworten zu den Messzeitpunkten.

Um zu untersuchen, ob bzw. wie sich die Selbstwirksamkeitserwartung der Lehrkräfte hinsichtlich der Umsetzung der Fortbildungsinhalte im Laufe des Projekts *AlgoKids* verändert hat, werden die Mittelwerte der Skala zu drei verschiedenen Messzeitpunkten untersucht (siehe Abbildung 18.10). Um zu überprüfen, ob eine signifikante Veränderung feststellbar ist, wird eine einfaktorielle Varianzanalyse mit Messwiederholung durchgeführt. Voraussetzung hierfür ist, dass die Daten intervallskaliert sind. Wie bereits bei der Untersuchung der beiden vorangegangenen Forschungsfragen, werden die Items als intervallskalierte Daten interpretiert, obwohl sie über eine 5-stufige Likert-Skala erfasst wurden. Zudem müssen die Residuen der abhängigen Variable innerhalb jedes Messzeitpunktes normalverteilt sein. Nach dem Kolmgorov-Smirnov-Test ist dies nur bei der ersten Messung der Fall ($p1$ = 0.109, $p2$ = 0.38, $p3$ = <0.001), laut dem Shapiro-Wilk-Test bei der ersten und zweiten Messung ($p1$ = 0.460, $p2$ = 0.95, $p3$ = 0.006). Die Quotienten aus der Kurtosis und deren Standardfehler sowie der Schiefe und deren Standardfehler liegen jedoch nicht unter -2 und über +2, sodass dennoch von einer Normalverteilung ausgegangen werden kann. Dies bestätigt sich auch bei der Sichtung der jeweiligen Histogramme. Die letzte Voraussetzung ist die Sphärizität, die mittels des Mauchly-Test auf Sphärizität bestimmt wird. Dieser ist für die vorliegenden Daten nicht signifikant, sodass Sphärizität angenommen wird ($Mauchly$-$W(2)$ = 0.877, $p$ = 0.208).

Es zeigt sich, dass sich die Selbstwirksamkeitserwartung der Lehrkräfte bzgl. der Umsetzung der Fortbildungsinhalte zu den Messzeitpunkten signifikant unterscheidet ($F$(2,50) = 9.399, $p$ = <0,001, $\eta_p^2$ = 0.273, $n$ = 26). Bonferroni-korrigierte paarweise Vergleiche zeigen, dass die Selbstwirksamkeit

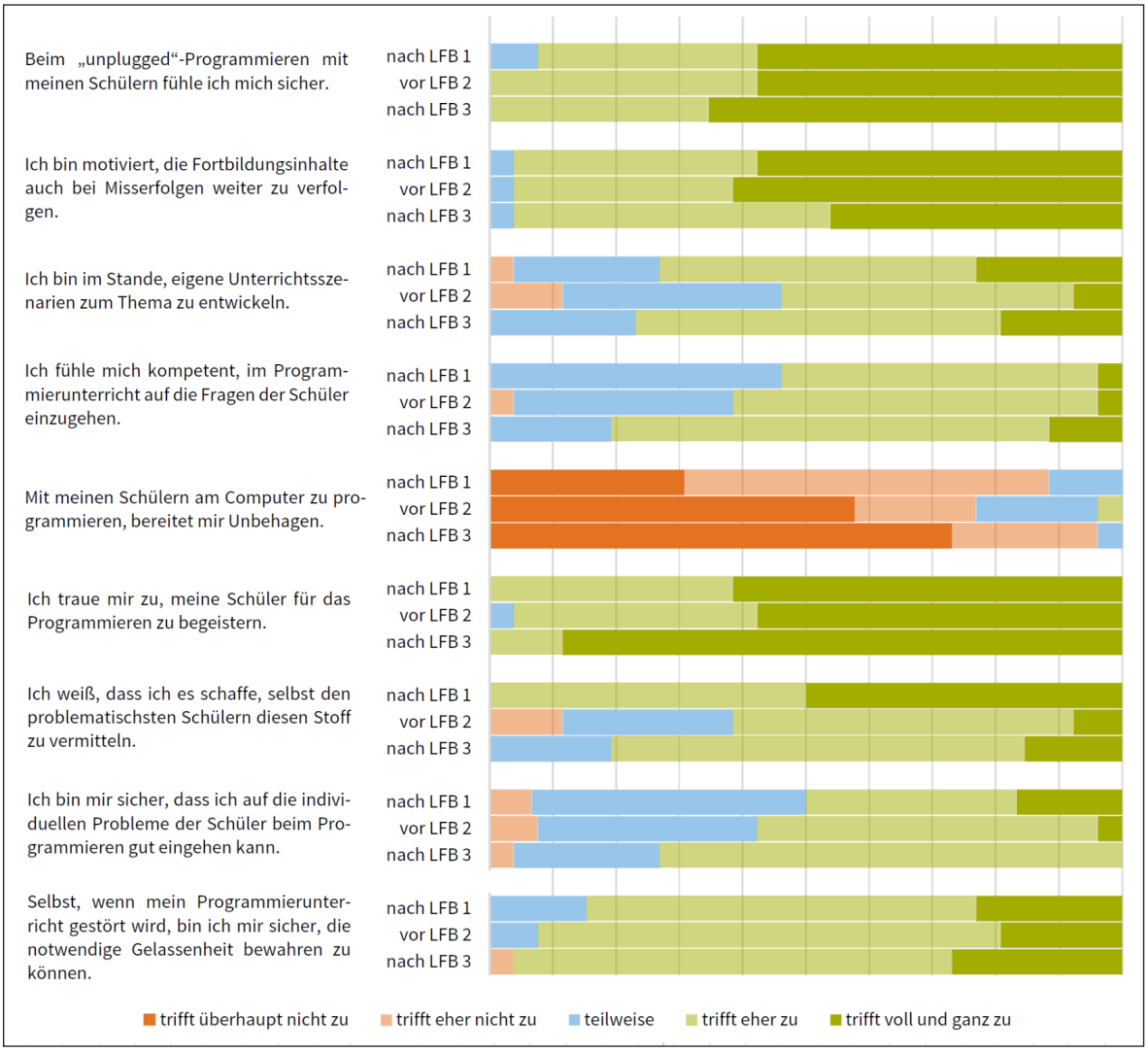



*Abbildung 18.9 Angaben zur Umsetzung der Fortbildungsinhalte im Unterricht (n=26; FB-LK2, FB-LK3 und FB-LK4)*

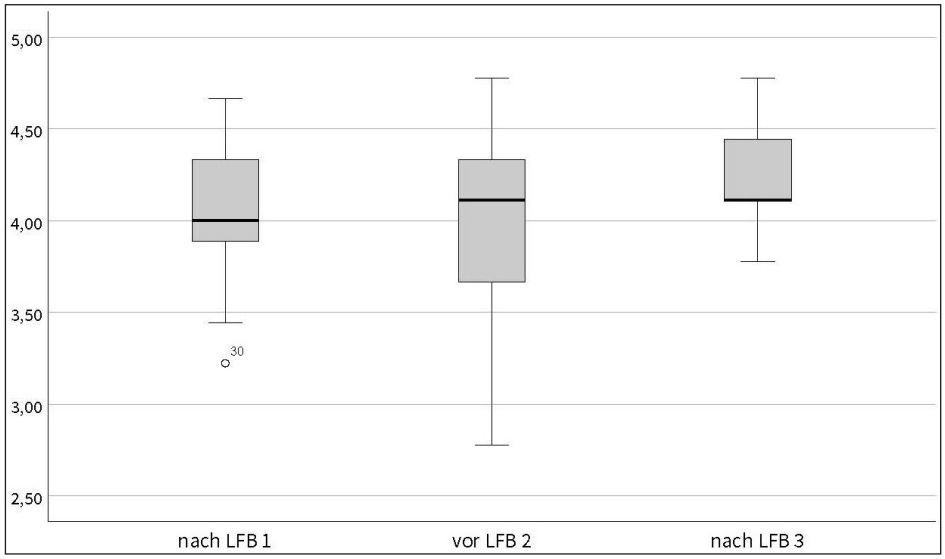



*Abbildung 18.10 Mittelwerte der Selbsteinschätzung zur Umsetzung der Fortbildungsinhalte (n=26; FB-LK2, FB-LK3 und FB-LK4)*

bzgl. der Umsetzung der Fortbildungsinhalte nach der ersten Fortbildung ($M$ = 4.034, $SD$ = 0.366) und vor der zweiten Fortbildung ($M$ = 4.038, $SD$ = 0.479) signifikant geringer ist als nach der dritten Fortbildung ($M$ = 4.265, $SD$ = 0.275). Zwischen den ersten beiden Messzeitpunkten unterscheidet sich die Selbstwirksamkeitserwartung nicht signifikant. Die Effektstärke $f$ nach Cohen (1988) liegt bei 0.61 und entspricht einem starken Effekt.

### 18.3.3 Zusammenfassung und Diskussion

Der Fokus dieser Teilstudie liegt auf den Auswirkungen, die das Fortbildungskonzept auf die teilnehmenden Lehrkräfte hatte. In einem ersten Schritt konnten durch die Auswertung der explorativen Interviews bereits verschiedene Effekte identifiziert werden. Diese beziehen sich zum Großteil auf affektive Merkmale – Motivation, Selbstwirksamkeit und Interesse bzgl. des Programmierens und dessen Umsetzung im Unterricht. Durch die Analyse der Fragebögen konnten Teile diese Ergebnisse bestätigt und weitere Auswirkungen identifiziert werden: eine signifikante Steigerung der Sicherheit im Umgang mit Computern und Computeranwendungen, eine signifikante Steigerung der Selbstwirksamkeit bzgl. der Umsetzung der Fortbildungsinhalte sowie eine Steigerung der Selbstwirksamkeitserwartung bzgl. der eigenen Programmierfähigkeit, die jedoch kein Signifikanzniveau erreicht hat.

Die Steigerung der Sicherheit im Umgang mit Computern und Computeranwendung ist nicht überraschend, da die Lehrkräfte im Rahmen der Fortbildungen und Erprobung des Unterrichtskonzepts viel am Computer gearbeitet haben und sich dadurch vermutlich eine gewisse Übung eingestellt hat. Betrachtet man den Verlauf der Selbstwirksamkeit bzgl. der Programmierfähigkeit, so fällt auf, dass diese im Verlauf der ersten Praxisphase stark abfällt. Dies könnte darauf zurückzuführen sein, dass die Lehrkräfte in dieser Zeit erste Erprobungen der Unterrichtssequenz durchgeführt haben und viele Fragen bzw. Unsicherheiten in der Programmierumgebung Scratch aufgekommen sind. Eine andere Erklärung könnte die relativ lange Zeitspanne zwischen den ersten beiden Messzeitpunkten sein. Es ist möglich, dass die Lehrkräfte nach sechs Monaten mit einigen der in den Items verwendeten Begriffe nicht mehr vertraut waren und sich daher tendenziell schlechter bewerteten. Nach Durchlaufen des restlichen Fortbildungskonzepts, d. h. der zweiten und dritten Fortbildung sowie der zweiten Praxisphase, stieg die Selbsteinschätzung der Programmierfähigkeit wieder deutlich an. Dies lässt sich vermutlich durch Lern- und Übungseffekte erklären. Die Selbstwirksamkeit bzgl. der Umsetzung der Fortbildungsinhalte hat sich im Verlauf der ersten Praxisphase nicht signifikant verändert. Nach Durchlaufen des restlichen Fortbildungskonzepts konnte jedoch ein signifikanter Anstieg ermittelt werden. Dies lässt sich vermutlich ebenfalls durch Lern- und Übungseffekte erklären. Hinzuweisen ist außerdem auf das Item *„Ich bin im Stande, eigene Unterrichtsszenarien zum Thema zu entwickeln“*. Im Rahmen der explorativen Interviews berichteten viele Lehrkräfte, dass sie die Unterrichtssequenz in der ersten Praxisphase unverändert erprobten und erst im Laufe der zweiten Praxisphase erweiterten.

Hinsichtlich der Erhebungsmethode der explorativen Interviews ist anzumerken, das darin nicht explizit nach den Auswirkungen des Fortbildungskonzepts gefragt wurde. Da diese nur in vierzehn von neunzehn Interviews thematisiert wurde, besteht hier kein Anspruch auf Vollständigkeit. Bei der Bewertung der quantitativen Ergebnisse ist zu berücksichtigen, dass die Veränderung der abhängigen Variablen auch auf andere zeitabhängige Ursachen zurückgeführt werden könnte (vgl. Bortz und Döring 2002, S. 558) oder durch Reifungsprozesse verursacht werden könnte (ebd., S. 504). Zudem ist nicht eindeutig festzustellen, welchen Beitrag die Fortbildungsveranstaltungen und welchen Beitrag die Praxisphasen daran geleistet haben. Es liegt nahe, dass sich die erhobenen Konstrukte auch durch die Praxisphasen verändert haben. Es ist nicht auszuschließen, dass dies auch bei Lehrkräften der Fall wäre, die nicht an den Fortbildungen teilgenommen haben und z. B. lediglich mit einer Handreichung und den Unterrichtsmaterialien arbeiteten. Der Einsatz einer weiteren Experimentalgruppe mit unterschiedlichem Treatment bzw. einer Kontrollgruppe hätte hier für Klarheit sorgen können. Dies war jedoch aufgrund der Anlage des Projekts *AlgoKids* nicht möglich.

Die Ergebnisse der Studie stehen tendenziell in Einklang mit anderen Forschungsergebnissen (siehe Abschnitt 4.3). So konnten z. B. die Ergebnisse von Duncan, Bell und Atlas (2017) sowie Rich u. a. (2017b) in Teilen bestätigt werden. Hier ist jedoch darauf hinzuweisen, dass die Studien grundlegend verschieden aufgebaut waren. Der Annahme von Standl und Schlomske-Bodenstein (2021), dass Selbstwirksamkeit ein relativ stabiles Konstrukt ist und sich nicht leicht verändern lässt, kann auf Basis der vorliegenden Daten nicht entsprochen werden.

In Hinblick auf die Ziele, die der Gestaltung des Fortbildungskonzepts zugrunde lagen (siehe Abschnitt 15.1), lassen sich mehrere Schlussfolgerungen ziehen. Die affektiven Zieldimensionen – Motivation und Enthusiasmus in Bezug auf die Gestaltung und Durchführung von Lehr-Lernszenarien zur Programmierung, Interesse an und emotionale Haltung zur Programmierung, und Selbstwirksamkeitserwartung in Bezug auf die Gestaltung und Durchführung von Lehr-Lernszenarien zur Programmierung – wurden zumindest teilweise erreicht. In zwölf von neunzehn Interviews wurde mindestens eine der Zieldimensionen konkret angesprochen. Das Erreichen der informatischen und informatikdidaktischen Lernziele wurde in neun Interviews thematisiert. Da die Lehrkräfte jedoch im Allgemeinen keine Probleme hatten, die Programmieraufgaben während den Fortbildungen zu lösen sowie die Unterrichtssequenz mit ihren Schülerinnen und Schülern zu erproben bzw. diese sogar zu erweitern, ist auch in diesen Bereichen von einem Kompetenzzuwachs auszugehen. Auch in Bezug auf das Fortbildungskonzept lassen sich Rückschlüsse ziehen. Da die Selbstwirksamkeitserwartung der Lehrkräfte in Hinblick auf ihre Programmierfähigkeit nach der ersten Fortbildung im Mittel signifikant abfiel (siehe Abbildung 18.8), ist mindestens eine weitere Fortbildungsveranstaltung zu empfehlen. Gleiches gilt für die Selbstwirksamkeitserwartung bzgl. der Umsetzung der Fortbildungsinhalte (siehe Abbildung 18.10). Diese steigt zwar im Mittel, das Minimum der Werte sinkt jedoch – es gibt also Lehrkräfte, die ihre Selbstwirksamkeitserwartung nach der ersten Praxisphase geringer einschätzen als vorher. Aufgrund des Designs der Erhebung

können keine Aussagen getroffen werden, ob die zweite Fortbildung, die zweite Praxisphase oder die dritte Fortbildung für den Anstieg der Werte verantwortlich sind.

## 18.4 Gewonnene Erkenntnisse zum Fortbildungskonzept

Die Ergebnisse des ersten Zyklus zeigen, dass die formulierten Lernziele für die Zielgruppe angemessen sind. Darüber hinaus erwies sich die Handlungslinie des Fortbildungskonzepts bei dessen Erprobung als gut durchführbar und zielführend – lediglich die Anknüpfungspunkte an den Lehrplan sollten bereits in der ersten Fortbildungsveranstaltung thematisiert werden. Die zugrundeliegenden *Design Principles* konnten ebenfalls bestätigt werden. Da in den Teilstudien ein Mehrwert der zweiten Praxisphase nicht eindeutig festgestellt werden konnte, werden in der abschließende Darstellung des Fortbildungskonzepts in Anhang C.1.2 zwei Varianten der Handlungslinie präsentiert: eine Handlungslinie mit einer Praxisphase und eine mit zwei Praxisphasen. Auf eine durchgängige Ablaufplanung des Fortbildungskonzepts wird, wie im Fall der Unterrichtssequenz, im weiteren Verlauf der Arbeit verzichtet.

Teil VI

# FOLGERUNGEN

# 19 Empfehlungen für die Umsetzung des Programmierens in der Grundschule

Im folgenden Kapitel werden die Ergebnisse aller Erprobungen und Teilstudien zusammengeführt zu einer abschließenden Darstellung der bereits operationalisierten *Design Principles* der Unterrichtssequenz und des Fortbildungskonzepts. Die im Vorfeld formulierten *Design Principles* werden nicht angepasst, es sind lediglich verschiedene Operationen auf der Ebene der Umsetzungsprinzipien und Musterbeispiele vorzunehmen. Diese werden durch unterschiedliche Ziffern gekennzeichnet (siehe Tabelle 19.1, vgl. Hiller 2017, S. 325). Da in der dritten Teilstudie die Verankerung des Programmierens im Lehrplan bzw. Anknüpfung daran als Gelingensbedingung identifiziert wurde, werden zusätzlich Anknüpfungspunkte des Programmierens zum LehrplanPLUS für die bayerische Grundschule aufgezeigt. Die Überarbeitung der didaktischen Konzeption der Unterrichtssequenz und des Fortbildungskonzepts befinden sich im Anhang (siehe Anhang B.1.2 und C.1.2).

*Tabelle 19.1 Operationen zur Weiterentwicklung der Design-Principles*

| Nummer | Beschreibung der Operationen, die an den Umsetzungsprinzipien bzw. Musterbeispielen durchgeführt wurden |
|---|---|
| 1 | bleibt unverändert erhalten |
| 2 | wurde aufgrund der Erprobungen in unterrichtsnahen Lehr-Lernsituationen modifiziert |
| 3 | wurde aufgrund der Erprobungen im Grundschulunterricht modifiziert |
| 4 | wurde aufgrund der Erprobungen in unterrichtsnahen Lehr-Lernsituationen neu eingeführt |
| 5 | wurde aufgrund der Erprobungen im Grundschulunterricht neu eingeführt |
| 6 | wurde aufgrund der Erprobung der Fortbildung modifiziert |
| 7 | wurde aufgrund der Erprobung der Fortbildung neu eingeführt |
| 8 | wurde gestrichen |

Zur Darstellung der *Design Principles* wird auf die von Euler (2014b) vorgeschlagene Grundstruktur zurückgegriffen (siehe Tabelle 7.2). Die Aussagen zum jeweiligen Kontext beziehen sich im Fall der Unterrichtssequenz auf die Ergebnisse der ersten und dritten Teilstudie, im Fall des Fortbildungskonzepts auf die Informationen zum vierten Erprobungszyklus der Unterrichtssequenz. Die Zielvorstellungen werden jeweils den Kapiteln zur Entwicklung der Unterrichtssequenz (Kapitel 14.1) und des Fortbildungskonzepts (Kapitel 15.1) entnommen und entsprechend der Ergebnisse von Teilstudie drei und fünf angepasst. Anstatt der Nomenklatur von Euler, in der Leit- und Umsetzungsprinzipien unterschieden werden, werden die in dieser Arbeit bereits eingeführten Begriffe der Umsetzungsprinzipien und Musterbeispiele verwendet.

## 19.1 Überarbeitete *Design Principles* (Unterrichtssequenz)

**KONTEXT**

Organisationale und soziale Rahmenbedingungen

- Sequenz kann als Projekt, in einzelnen Unterrichtsstunden oder als AG durchgeführt werden.
- Für das Programmieren in Scratch ist die Umsetzung im *Teamteaching* von Vorteil.
- In Scratch können SuS in Partnerarbeit programmieren.

Individuelle Lernvoraussetzungen

- SuS haben unterschiedliche Vorerfahrungen im Umgang mit Computern.
- SuS weisen Unterschiede in Geduld, Frustrationstoleranz und Anstrengungsbereitschaft auf.
- Es gibt Unterschiede in den sprachlichen und kognitiven Kompetenzen der SuS.

**ZIELVORSTELLUNGEN**

Informatische Kompetenzen (3)

- Die SuS führen Algorithmen in ihrer Lebenswelt aus;
- Die SuS lesen, beschreiben und untersuchen Algorithmen und Programme in ihrer Alltagssprache;
- Die SuS entwerfen, realisieren und testen Algorithmen mit den algorithmischen Grundbausteinen Anweisung, Sequenz, Wiederholung und Verzweigung;
- Die SuS programmieren ein altersentsprechendes Informatiksystem;

Affektive Merkmale (1)

- Motivation und Lernfreude im Umgang mit der Programmierung;
- Interesse an der Programmierung;
- Selbstwirksamkeitserwartung im Umgang mit der Programmierung;

**DP1 Die Lernenden begegnen den Lerninhalten in unterschiedlichen Repräsentationsformen**

Umsetzungsprinzipien

- Die SuS begegnen den algorithmischen Grundstrukturen durch eigenständig ausgeführte Handlungen, durch graphische und bildliche Darstellungen sowie durch Sprache und Zeichen (1).
- Die SuS nutzen eine kindgerechte block-basierte Programmierumgebung (1).
- Die SuS programmieren kindgerechte Roboter (5).

Musterbeispiele

- Die SuS programmieren den *Lehrer-Roboter* mittels mündlicher Anweisungen. Dieser führt die Anweisungen wörtlich aus und bewegt sich nicht bei zu allgemeinen Befehlen (1).
- Die SuS verfassen sprachliche Anweisungen zu einer bildlichen Anleitung (1).
- Die SuS programmieren sich gegenseitig, um Aufgaben in einem Parcours zu lösen. Sie kontrollieren die Programme beim Ablaufen in einem aufgebauten Parcours oder mittels Spielfiguren auf den Arbeitsblättern (3).
- Die SuS programmieren in Scratch oder ScratchJr (3).
- Die SuS programmieren BeeBots oder BlueBots (5).

Begründung

- Grundschulkinder profitieren von einer Kopplung ihrer Denkoperationen an konkrete Handlungen/Wahrnehmungen.
- Lernprozesse vollziehen sich wesentlich bei den Übergängen zu anderen Repräsentationsformen eines Lerngegenstands.
- Es ist kindgemäß, den Unterrichtsgegenstand zu veranschaulichen, indem man ihn auf verschiedenen Stufen der Konkretisierung/Abstraktion einbindet.
- *Unplugged*-Materialien stellen kindgerechten Zugang zur Informatik dar.
- Visualisierungen und Rollenspiele sind förderlich zur Ausbildung der *Notional Machine*.

| **DP2 Die Unterrichtssequenz führt vom Bekannten zum Unbekannten** | |
|---|---|
| Umsetzungsprinzipien<br>• Anknüpfen an den Alltag und das Vorwissen der SuS durch vertraute Beispiele und Aufgabenformate (1).<br>• Besonders zu Beginn werden Aufgaben eingesetzt, die in ähnlicher Form aus anderen Schulfächern oder Alltagssituationen bekannt sind (1). | Begründung<br>• Komplexe und abstrakte Ideen werden am besten durch vertraute Beispiele eingeführt.<br>• Auch jüngere Kinder sind zum schlussfolgerndem Denken in der Lage, wenn sich Aufgaben auf Konzepte beziehen, die ihnen bekannt sind und konkret dargeboten werden. |
| Musterbeispiele<br>• In Form eines Brainstormings wird gesammelt, was die SuS bereits über das Programmieren wissen (1).<br>• Anhand von Bildern oder eines Wimmelbildes wird thematisiert, wo sich Programme in der Umgebung der SuS verstecken (4).<br>• Als Einstieg in die Auseinandersetzung mit dem Algorithmusbegriff wird eine Abwandlung des Spiel *Stille Post* gespielt, in der die SuS nacheinander reihum ein Bild malen, dieses beschreiben und im Anschluss nach der Beschreibung wieder ein Bild malen (5).<br>• Bei der Umwandlung der bildlichen Anleitungen in schriftliche Befehle verweist die Lehrkraft auf die Textform der Vorgangsbeschreibung – falls diese bereits bekannt ist (1).<br>• Die Parcoursaufgaben werden zum Einstieg in Scratch verwendet, da der Lösungsweg bereits klar ist und die SuS sich auf die Bedienung des Programms konzentrieren können (5). | |

| **DP3 Die kognitive Belastung der Lernenden soll möglichst gering gehalten werden** | |
|---|---|
| Umsetzungsprinzipien<br>• Die algorithmischen Grundstrukturen und Scratch-Blöcke werden bereits in den *Unplugged*-Aufgaben eingeführt (1).<br>• Beim Programmieren in Scratch wird zunächst mit vorgegebenen Programmen gearbeitet. Diese werden in einem nächsten Schritt angepasst und erweitert, bis schließlich eigene Programmideen umgesetzt werden (1). | Begründung<br>• Durch unnötige kognitive Belastung wird das Arbeitsgedächtnis entlastet.<br>• Lerninhalte sollten auf algorithmische Grundstrukturen und allgemeine Herangehensweisen zur Lösung eines Problems begrenzt werden.<br>• Reduktion der kognitiven Belastung bei Programmieraufgaben durch *Fading Worked Examples* und im Rahmen der Unterrichtsgestaltung durch den Ansatz *PRIMM* bzw. die *Use-Modify-Create Learning Progression*. |
| Musterbeispiele<br>• Die SuS programmieren *unplugged* mit haptischen Programmierblöcken auf Filzbahnen und an der Tafel (1).<br>• Die Programmierumgebung Scratch wird über den Beamer gemeinsam betrachtet und grundlegende Funktionsweisen thematisiert oder kleine Forscheraufträge für die SuS vergeben (3).<br>• Die SuS bearbeiten einen Lernzirkel, in dem die Grundfunktionen von Scratch und algorithmischen Grundstrukturen nacheinander thematisiert werden. Sie programmieren erst ein Programm nach und bearbeiten danach eine passende Aufgabe (1).<br>• Die SuS lesen und interpretieren vorgegebene Programme und suchen z. B. nach Fehlern (5). | |

| **DP4 Die Aufgaben sind motivierend und ermöglichen aktives sowie kooperatives Lernen** | |
|---|---|
| Umsetzungsprinzipien<br>• Für die Programmieraufgaben werden Kontexte gewählt, die an die persönlichen Erfahrungen der SuS anknüpfen, lebendige und unterhaltsame Vermittlung sowie breites Spektrum an Aufgaben ermöglichen (1).<br>• Die SuS nehmen von Beginn an eine aktive Rolle im Unterrichtsgeschehen ein – lange lehrerzentrierte Phasen werden vermieden (1).<br>• Die Aufgaben ermöglichen Erfolgserlebnisse, können personalisiert werden und knüpfen, wenn möglich, an Alltagserfahrungen an (1).<br>• Die *Unplugged*-Aufgaben und Programmieraufgaben am Computer werden in Gruppen oder in Partnerarbeit bearbeitet (1).<br>• Die SuS bearbeiten eine Projektaufgabe im Klassenverbund (5).<br>• Die SuS programmieren kindgerechte Roboter und Microcontroller (5). | Begründung<br>• Förderung der Lernmotivation durch Beachtung des *ARCS-Modell*, der Selbstbestimmungstheorie der Motivation, der Dimensionen zur Gestaltung motivierender Programmieraufgaben sowie genderspezifischer Unterschiede.<br>• Computer und programmierbare Roboter üben besondere Anziehungskraft auf Kinder aus.<br>• Die aktive Auseinandersetzung mit dem Lerngegenstand ist lernförderlich.<br>• Soziale Interaktion kann im Sinne der *Zone der proximalen Entwicklung* lernförderlich sein. |
| Musterbeispiele<br>• Als Kontext für die Programmieraufgaben wird das Thema *Zirkus* gewählt, das an Alltagserfahrungen der SuS anknüpft und viele Programmieranlässe bietet (1).<br>• Die Programmieraufgaben werden zunehmend anspruchsvoller, dass die SuS relativ schnell erste Erfolgserlebnisse haben (1).<br>• Innerhalb der Programmieraufgaben haben die SuS die Möglichkeit, ihre Programme zu personalisieren oder sogar eigene Programmierideen umzusetzen (1).<br>• Die *Unplugged*-Aktivitäten werden in Gruppenarbeit gelöst, beim Programmieren in Scratch können die SuS in Paaren zusammenarbeiten. Das *Pair Programming* sollte thematisiert und eingeübt werden (3).<br>• Die SuS programmieren arbeitsteilig ein Klassenprojekt, z. B. einen Rundgang durch den Heimatort. Die Lehrkraft führt die einzelnen Programme am Ende zusammen (5).<br>• Die SuS programmieren Microcontroller, wie z. B. den *Calliope mini*, oder Roboter, z. B. *Lego WeDo* (5). | |

| **DP5 Die Lernenden durchlaufen verschiedene Schritte des Problemlösens** | |
|---|---|
| Umsetzungsprinzipien<br>• Aufgaben werden so konzipiert, dass sie durch Schlussfolgern gelöst werden können (1).<br>• Die SuS werden mit Aufgaben konfrontiert, für die keine offensichtliche Lösungsroutine abgerufen werden kann (1).<br>• Einzelne Schritte des Problemlösens werden im Plenum thematisiert (1). | Begründung<br>• Die Anbahnung von Problemlösen ist zentrale Aufgabe der Grundschule.<br>• Schlussfolgerndes Denken kann durch Angebote, die analoges Schließen erfordern, angeregt werden. |

| | |
|---|---|
| Musterbeispiele<br>• Zur Lösung der Aufgaben des Lernzirkels können Lösungsansätze aus den *Worked Examples* herangezogen werden (1).<br>• Die Lösungen komplexerer Aufgaben wird im Plenum mithilfe von Pseudocode modelliert (5).<br>• Die SuS planen ihre Programmideen mithilfe von entsprechenden Vorlagen. Diese können thematisch frei sein oder eine grobe Handlung vorgeben (3).<br>• Die Lösung schwieriger Aufgaben bzw. das entsprechende Vorgehen werden im Plenum thematisiert (1). | • Elaboriertes Üben kann eine flexible Anwendung des Wissens und Könnens fördern.<br>• Das Entwickeln von Problemlösefähigkeit ist Ziel des Informatikunterrichts. |

| **DP6 Die Lehrkraft fördert eine positive Fehlerkultur** | |
|---|---|
| Umsetzungsprinzipien<br>• Auch wenn die Lehrkraft Fehler im Programm sieht, gibt sie den SuS die Chance, sie bei der Programmausführung selbst zu erkennen (1).<br>• Bei Fragen in Scratch werden zuerst andere SuS konsultiert (1).<br>• Häufige Fehler werden gemeinsam im Plenum besprochen und können vorher auf Zetteln gesammelt werden (3).<br>• Falls die Lehrkraft Fehler nicht sofort lösen kann, kommuniziert sie dies offen und sucht gemeinsam mit den SuS nach Lösungen (1). | Begründung<br>• Fehler geben Einblick in den Lernstand und Denkweise der SuS.<br>• Fehler sind beim Programmieren unvermeidlich, deswegen sollten Problemlösestrategien zur Fehlersuche und -behebung vermittelt werden. |

| **DP7 Die Lehrkraft orientiert sich an Zielen und fördert die Zielorientierung der Lernenden** | |
|---|---|
| Umsetzungsprinzipien<br>• Die Lehrkraft kommuniziert, besonders während den *Unplugged*-Aktivitäten, die Bedeutung der einzelnen Übungen (1).<br>• Bei offeneren Programmieraufgaben werden die grundlegenden Anforderungen an die Programme kommuniziert (1).<br>• Bei der Präsentation von Programmen wird thematisiert, inwieweit die Vorgaben eingehalten wurden (1). | Begründung<br>• Zielorientierung ist ein wichtiges Unterrichtsprinzip.<br>• Lernsituationen sollten so gestaltet werden, dass jedes Kind erfolgreich dazulernen kann.<br>• Objektive Kriterien sollten zur Einordnung der Leistung der SuS einbezogen werden. |

| **DP8 Die Lehrkraft unterstützt den Lernprozess durch Hilfestellungen** | |
|---|---|
| Umsetzungsprinzipien<br>• Bei Problemen teilt die Lehrkraft die Lösung nicht direkt mit, sondern gibt verbale Hilfestellungen (1).<br>• Die Lösungen komplexerer Aufgaben wird im Plenum mithilfe von Pseudocode modelliert (5).<br>• Wenn möglich, werden Aufgabenstellungen mit haptischen Hilfsmitteln oder Rollenspielen visualisiert (3).<br>• Zu ausgewählten Aufgaben werden Tippkarten angefertigt, die von den SuS eigenständig verwendet werden können (1).<br>• Leistungsschwächere SuS können zunächst vorgegebene Programme untersuchen, bevor sie selbst programmieren (5). | Begründung<br>• Unterstützung durch die Lehrkraft ist im Unterricht der Grundschule notwendig.<br>• Kognitive Strukturierung durch Hilfestellungen spielt im Unterricht der Grundschule eine wesentliche Rolle. |

## 19.2 Überarbeitete *Design Principles* (Lehrerfortbildung)

**KONTEXT**

Individuelle Lernvoraussetzungen

- Über die Hälfte der Lehrkräfte (n=27) gab zu Beginn an, überhaupt keine Vorkenntnisse in Informatik zu besitzen, dreizehn hatten Informatik zumindest vorübergehend als Wahl- oder Pflichtfach in der Schule.
- Die Berufserfahrung der Lehrkräfte im Jahr 2018 reichte von drei bis 39 Jahren, wobei fast die Hälfte der Teilnehmenden angab, über drei bis zehn Jahre Berufserfahrung zu verfügen.
- Die Initiative zur Teilnahme war relativ gleichmäßig zwischen den Lehrkräften, ihrer jeweiligen Schulleitung oder beiden verteilt.

**ZIELVORSTELLUNGEN**

Informatische Kompetenzen (3)

- Die Lehrkräfte (LK) führen Algorithmen in ihrer Lebenswelt bzw. der Lebenswelt der SuS aus;
- Die LK lesen, beschreiben und untersuchen Algorithmen und Programme in ihrer Alltagssprache;
- Die LK entwerfen, realisieren und testen Algorithmen mit den algorithmischen Grundbausteinen Anweisung, Sequenz, Wiederholung und Verzweigung ;
- Die LK programmieren ein für die Grundschule altersentsprechendes Informatiksystem;
- Die LK verwenden Variablen und Wertzuweisungen;
- Die LK ordnen Bestandteile eines Informatiksystems der Eingabe, der Verarbeitung und der Ausgabe zu;
- Die LK wenden Aspekte des *Computational Thinking* an, z. B. *Decomposition*, *Abstraction*, oder *Patterns/-Generalization*;

Informatikdidaktische Kompetenzen (1)

- Die LK sind in der Lage, Konzepte für das Erlernen einer einfachen Programmiersprache mit einem spielerischen, auch *Unplugged*-Ansatz, praktisch umzusetzen;
- Die LK sind in der Lage, SuS für das Programmierenlernen zu motivieren sowie motivierende Aktivierungsmethoden und informatiktypische Sozialformen kompetenzfördernd einzusetzen;
- Die LK sind in der Lage, kindgemäße Software und einfache altersgemäße Informatiksysteme für das Programmieren im Primarbereich auszuwählen und sinnvoll in Lernszenarien zu integrieren;
- Die LK sind in der Lage, Aufgabenstellungen und Lerninhalte des Programmierens kindgemäß und auf unterschiedliche Weise vereinfacht, aber korrekt darzustellen sowie abstrakte informatische Konzepte durch verschiedene Beispiele und mittels spielerischer und explorativer Erkundungsanlässe für die SuS erfahrbar zu machen;
- Die LK sind in der Lage, typische Präkonzepte, Verstehens- und Lernbarrieren der SuS in Bezug auf das Programmieren zu identifizieren und entsprechende Formalisierungen, Abstraktionen und Interventionsmöglichkeiten einzusetzen;
- Die LK können auch in Gruppenarbeitssituationen mit Informatiksystemen auf Lernschwierigkeiten einzelner Kinder eingehen und gezielte individuelle Hilfestellungen geben;
- Die LK sind in der Lage, individuelle Leistungsstände der SuS zu identifizieren und darauf durch situationsangemessenes Handeln zu reagieren;
- Die LK regen die SuS zum selbstständigen Lernen und zur eigenständigen Begutachtung ihrer Lernergebnisse an und fördern so deren Selbstwirksamkeit;
- Die LK fördern durch Gruppengespräche die Reflexion der SuS über ihr eigenes Handeln und ermöglichen so positives Feedback über die erzielten Ergebnisse;

| Affektive Merkmale (1) |
|---|
| • Motivation/Enthusiasmus in Bezug auf die Gestaltung und Durchführung von Lehr-Lernszenarien zur Programmierung;<br>• Interesse an und emotionale Haltung zur Programmierung;<br>• Selbstwirksamkeitserwartung in Bezug auf die Gestaltung und Durchführung von Lehr-Lernszenarien zur Programmierung; |

| **LFB-DP 1 Die Fortbildung ermöglicht eine vertiefte Auseinandersetzung mit den Inhalten** | |
|---|---|
| Umsetzungsprinzipien<br>• Die Lehrerfortbildung umfasst verschiedene, jeweils mehrtägige Termine (1).<br>• Damit sich die Lehrkräfte auf die Fortbildungsinhalte konzentrieren können, ist es ideal, wenn die Fortbildung nicht an der Schule der Teilnehmenden stattfindet (1).<br>• Die Fortbildungstermine liegen weit genug auseinander, um eine Erprobung der Inhalte im Unterricht zu ermöglichen (1). | Begründung<br>• Bereitstellen von expliziten Lernzeitfenstern ist grundlegend für Wirksamkeit von Fortbildungen.<br>• Längere Lernzeitfenster ermöglichen Erprobung neuer Handlungsmuster.<br>• Es wird empfohlen, Fortbildungsphasen mit Erprobungen zu verschränken. |

| **LFB-DP 2 Die Lehrkräfte erhalten eine fortlaufende Unterstützung** | |
|---|---|
| Umsetzungsprinzipien<br>• Innerhalb der Fortbildung wird Zeit für das Besprechen möglicher Anliegen der Lehrkräfte eingeplant (1).<br>• Die Lehrkräfte erhalten auch über die Fortbildungsveranstaltungen hinaus die Möglichkeit, sich mit Fragen oder Problemen an die Fortbildenden zu wenden (1).<br>• Mehrere Lehrkräfte einer Schule nehmen teil, damit sie sich zu den Inhalten austauschen und bei der Umsetzung im Unterricht unterstützen können (1). | Begründung<br>• Kontinuierliche Unterstützungsmaßnahmen sowie Coaching-Angebote fördern die Implementierung der Fortbildungsinhalte im Unterricht und erhöhen die Wahrscheinlichkeit, dass diese angemessen umgesetzt werden. |

| **LFB-DP 3 Die Gestaltung des Erkenntnisgewinns der Lehrkräfte orientiert sich an dem der SuS** | |
|---|---|
| Umsetzungsprinzipien<br>• Die Lehrkräfte erhalten die Möglichkeit, dem Lernpfad der SuS zu folgen (1).<br>• Es werden beispielhafte Arbeitsergebnisse von SuS zur Verfügung gestellt (1). | Begründung<br>• Erleben der Veränderung der ursprünglichen Vorstellungen wirkt sich positiv auf die spätere Umsetzung im Unterricht aus.<br>• Lehrkräfte erleben Fortbildungen als zufriedenstellend, wenn sie sich auf den konkreten Unterrichtsalltag beziehen.<br>• Nach dem Ansatz des situierten Lernens sollten Lernsituationen so authentisch und anwendungsorientiert wie möglich sein. |
| Musterbeispiele<br>• Die Lehrkräfte bearbeiten selbst eine Auswahl der Unterrichtsmaterialien (1).<br>• Die Lehrkräfte bekommen die Möglichkeit, Programmiererfahrung in Scratch zu sammeln (1).<br>• Die Fortbildenden geben Hilfestellung, wie sie dies auch innerhalb des Unterrichts tun würden (1). | |

| **LFB-DP 4 Die Fortbildung verschränkt Fachwissenschaft und Fachdidaktik** | |
|---|---|
| Umsetzungsprinzipien<br>• Die Lehrkräfte erhalten die Möglichkeit, selbst in Scratch zu programmieren (1).<br>• Die algorithmischen Grundstrukturen werden thematisiert (1).<br>• Fachdidaktische Prinzipien werden aufgezeigt (1). | Begründung<br>• Nach dem Ansatz des situierten Lernens sollten Lernsituationen sollten so authentisch und anwendungsorientiert wie möglich sein.<br>• Kernkonzepte der Programmierung sollten im Vordergrund stehen.<br>• Expertise im Programmieren stellt sich nur durch Übung ein. |
| Musterbeispiele<br>• Die algorithmischen Grundstrukturen werden in Pseudocode sowie der Programmiersprache Scratch beschrieben und in Aufgaben vertieft. Zusätzlich wird das Variablenkonzept behandelt (1).<br>• Die Lehrkräfte erfahren Unterrichtspraktiken und –methoden als Lernende und reflektieren anschließend aus der Sicht der Lehrenden. Sie bearbeiten z. B. Programmieraufgaben, die in Anlehnung an die *Use-Modify-Create Learning Progression* gestaltet wurden (1).<br>• Die Lehrkräfte lernen an verschiedenen Stationen weitere Systeme zum Programmieren im Unterricht kennen. Anschließend werden diese besprochen (1).<br>• Um Fehlvorstellungen zu vermeiden, werden Codebeispiele in Scratch gemeinsam schrittweise gelesen und nachvollzogen (1). | |

| **LFB-DP 5 In der Fortbildung werden Reflexionsprozesse über die Fortbildungsinhalte sowie die Umsetzung der Inhalte im Unterricht angeregt** | |
|---|---|
| Umsetzungsprinzipien<br>• Innerhalb der Fortbildungsveranstaltungen werden Reflexionsphasen eingeplant (1).<br>• Die Lehrkräfte erhalten die Möglichkeit, sich über die Erprobungen der Inhalte im Unterricht auszutauschen (1). | Begründung<br>• Möglichkeit, sich mit anderen über Inhalte auszutauschen, ist besonders wertvoll und gewinnbringend.<br>• Im dem Ansatz des situierten Lernens setzen sich Lehrkräfte mit fachlichen Inhalten auseinander, bevor in einem Theorie-Input die didaktischen Hintergründe thematisiert werden. |
| Musterbeispiele<br>• Lehrkräfte bekommen durch Bearbeiten der Unterrichtsmaterialien die Möglichkeit, die Perspektive der SuS einzunehmen, und besprechen im Anschluss, welche Schwierigkeiten und Erkenntnisse aufgetreten sind (1).<br>• Lehrkräfte untersuchen zwei Lösungen derselben Aufgabe nach Unterschieden in der Umsetzung und erarbeiten daran positive Programmierpraktiken (1).<br>• Bei der ersten Fortbildungsveranstaltung wird vereinbart, dass (Teile der) Unterrichtsmaterialien bis zur nächsten Veranstaltung erprobt werden (1).<br>• Anhand von Plakatwänden wird thematisiert, wie die Inhalte im eigenen Unterricht umgesetzt werden könnten (1).<br>• In der Veranstaltung nach der Praxisphase wird Erfahrungsaustausch eingeplant und moderiert (1). | |

| **LFB-DP 6 Die Fortbildung knüpft an Bekanntes an** | |
|---|---|
| Umsetzungsprinzipien<br>• Es werden Möglichkeiten zur Anknüpfung an andere Fächer oder zur Grundschuldidaktik bzw. fachdidaktischen Prinzipien bestehender Grundschulfächer aufgezeigt. (1).<br>• Es gibt Phasen, in denen *unplugged* gearbeitet wird (1). | Begründung<br>• Lehrkräften sollte Zeit eingeräumt werden, die neuen Inhalte mit dem bestehenden Lehrplan zu verknüpfen, oder Möglichkeiten zur Integration in andere Fächer sollten aufgezeigt werden.<br>• Einsatz von *Unplugged*-Aktivitäten und Beispielen aus dem Alltag werden empfohlen, um Lehrkräfte an das Thema heranzuführen. |
| Musterbeispiele<br>• Lehrkräfte bearbeiten Programmieraufgaben, die im Kontext bestehender Grundschulfächer eingesetzt werden können (1).<br>• Lehrkräfte erarbeiten Ideen für Programmierprojekte, die in bestehenden Fächern verortet werden können (7).<br>• Um den Unterschied von Alltagsalgorithmen zu einem Algorithmus im eigentlichen Sinn zu thematisieren, spielen die Lehrkräfte eine Variante des Spiels *Stille Post*, in der sie nacheinander reihum ein Bild malen, dieses beschreiben und im Anschluss nach der Beschreibung wieder ein Bild malen (1).<br>• Innerhalb der Fortbildung werden Aspekte des *Computational Thinking* ohne den Einsatz einer Programmiersprache thematisiert (1). | |

## 19.3 Bezüge zum LehrplanPLUS

Die Themen Algorithmen und Programmierung können mit verschiedenen bestehenden Kompetenzbereichen des LehrplanPLUS verknüpft werden und diese vertiefen oder erweitern.

### 19.3.1 Mathematik

Im Fachprofil der Grundschule für das Fach Mathematik sind im LehrplanPLUS die prozessbezogenen Kompetenzen *Modellieren*, *Probleme lösen* und *Darstellungen verwenden* beschrieben, die Schülerinnen und Schüler in Verbindung mit den inhaltsbezogenen Kompetenzen erwerben sollen (StMBKWK 2014, S. 106 ff.). Alle diese Kompetenzen werden beim Programmieren bzw. *Computational Thinking* adressiert: Um ein Problem zu lösen, wird ein Algorithmus aufgestellt, in verschiedenen Darstellungen (z.B. Symbolen, Grafiken) beschrieben und schließlich in einer Programmiersprache implementiert. In den Kompetenzformulierungen für die dritte und vierte Jahrgangsstufe steht dazu:

Die Schülerinnen und Schüler ...
- erweitern und verkürzen Sachsituationen, um Zusammenhänge zu erfassen und zu erklären, und beschaffen sich ggf. geeignete, noch fehlende Informationen (LB 1: Zahlen und Operatoren, TB 1.3 Sachsituationen und Mathematik in Beziehung setzen[1]).
- entnehmen relevante Daten aus verschiedenen Darstellungsformen und übertragen die Daten in geeignete andere Darstellungsformen (LB 4: Daten und Zufall, TB 4.1 Daten erfassen und strukturiert darstellen).

[1] LB steht für Lernbereich, TB steht für Teilbereich.

### 19.3.2 Deutsch

Im Fach Deutsch findet man für das Programmieren in allen Kompetenzbereichen Anknüpfungsmöglichkeiten (StMBKWK 2014, S. 42). Beim gemeinsamen Lösen von Problemen artikulieren die Schülerinnen und Schüler ihre Ergebnisse und Lösungswege. Sie setzen sich genau damit auseinander, wie Befehle formuliert sein müssen, damit sie das gewünschte Ergebnis zur Folge haben, und strukturieren beim Planen von Programmen ihre Ideen. In den Kompetenzformulierungen für die dritte und vierte Jahrgangsstufe heißt es dazu:

Die Schülerinnen und Schüler ...

- führen Lerngespräche, in denen sie ihre Lernstrategien beschreiben, über Arbeitsergebnisse und Lösungswege sprechen, die Zusammenarbeit bewerten oder Feedback an ein Team geben. Sie bewerten eigene Lernergebnisse im Vergleich mit denen anderer und ziehen Schlüsse für ihr eigenes Lernen (LB 1: Sprechen und Zuhören, TB 1.4 Über Lernen sprechen).
- veranschaulichen Abfolgen und Zusammenhänge im Text durch einfache Darstellungen (LB 2: Lesen – mit Texten und weiteren Medien umgehen, TB 2.4 Texte erschließen).
- unterscheiden Textarten, indem sie typische Elemente und Funktionen herausarbeiten: erzählende und poetische Texte, sachliche Texte, Gebrauchstexte. Sie übertragen denselben Stoff in andere Textsorten oder mediale Darstellungsformen und beschreiben dabei die Besonderheiten des jeweiligen Mediums (LB 2: Lesen – mit Texten und weiteren Medien umgehen, TB 2.1 Über Leseerfahrung verfügen).
- verfassen eigene informierende, beschreibende Texte und achten dabei auf eine reihende Darstellung sowie eine logische Anordnung der Informationen. Sie nutzen vor dem Schreiben Methoden zur Sammlung und Ordnung von Wortmaterial, Informationen, Begründungen und Schreibideen (LB 3: Schreiben, TB 3.2 Texte planen und schreiben).
- nehmen zentrale Anregungen für die Überarbeitung auf und setzen sich dazu jeweils ein konkretes Überarbeitungsziel (LB 3: Schreiben, TB 3.3 Texte überarbeiten).
- beschreiben anhand von Beispielen Gemeinsamkeiten und Unterschiede von Sprachen und Schriftsystemen im eigenen Umfeld und nutzen ihre Einsichten zur Erweiterung ihrer Sprachbewusstheit. Sie beschreiben und vergleichen Aspekte konzeptioneller Mündlichkeit und konzeptioneller Schriftlichkeit (LB 4: Sprachgebrauch und Sprache untersuchen und reflektieren, TB 4.2 Gemeinsamkeiten und Unterschiede von Sprache entdecken).
- beschreiben und bewerten Ursachen und Wirkungen von gelingender Verständigung. Sie untersuchen, welche sprachlichen Mittel genutzt werden, um bestimmte Wirkungen zu erreichen (LB 4: Sprachgebrauch und Sprache untersuchen und reflektieren, TB 4.1 Sprachliche Verständigung untersuchen).

### 19.3.3 Heimat- und Sachunterricht

Die Gesellschaft für Didaktik des Sachunterrichts (GDSU 2013, S. 63 f.) weist darauf hin, dass es die Aufgabe des Sachunterrichts ist, Kinder darin zu unterstützen, ihre technisch geprägte Umwelt sachgerecht zu verstehen, sie sich bildungswirksam zu erschließen und sich darin zu orientieren, mitzuwirken und zu handeln. Im LehrplanPLUS steht im Fachprofil des Heimat- und Sachunterrichts in Bezug zum Gegenstandsbereich *Technik und Kultur*, dass Schülerinnen und Schüler technische Errungenschaften als Grundlage unserer Kultur und Arbeitswelt kennenlernen (StMBKWK 2014, S. 85). In den Kompetenzformulierungen für die dritte und vierte Jahrgangsstufe steht dazu:

Die Schülerinnen und Schüler ...

- beschreiben die Entwicklung eines technischen Alltagsgegenstandes und erklären die jeweiligen Auswirkungen auf unsere Lebenswelt (LB 4: Zeit und Wandel, TB 4.2 Dauer und Wandel).
- erklären anhand von Beispielen, welche Prinzipien bei einfachen technischen Erfindungen zu einer Arbeitserleichterung führen und inwiefern sie Kulturleistungen möglich machen (LB 6: Technik und Kultur, TB 6.1 Arbeit, technische und kulturelle Entwicklung).

Anhand von Informatiksystemen, zum Beispiel eines Computers, kann thematisiert werden, welche Rolle er im Alltag der Schülerinnen und Schüler spielt und diesen beeinflusst. Gemeinsam kann erarbeitet werden, wie ein Informatiksystem bestimmte Arbeiten erleichtern kann.

### 19.3.4 Kunst & Werken und Gestalten

In Programmierumgebungen wie Scratch können Schülerinnen und Schüler kreativ und konstruktiv eigene Projekte entwerfen und umsetzen. Das Gestalten findet sich sowohl in den Kompetenzbeschreibungen im Fach Kunst als auch beim Werken und Gestalten wieder:

Die Schülerinnen und Schüler ...

- bauen mit geeigneten Materialien und Techniken Modelle oder gestalten fantasievolle Szenen, um die Wechselbeziehung zwischen Darstellungsabsicht und Gestaltung zu erkennen (Fach Kunst, LB 5: Fantasiewelten).
- beschreiben und unterscheiden Wirkungen von Gestaltungselementen und -prinzipien auf den Betrachter, finden dafür Beispiele aus Natur, Kunsthandwerk oder Design und nutzen ihre Erkenntnisse für eigene Gestaltungsvorhaben. Sie planen im Hinblick auf die Funktion der Gestaltung eigene Gestaltungsideen. Sie stellen ihre Skizzen unter Verwendung von Fachbegriffen vor und entwickeln sie im Austausch mit anderen weiter. Sie bewerten gemeinsam mit Mitschülerinnen und Mitschülern ihr Werkstück konstruktiv unter ästhetischen Gesichtspunkten und leiten daraus Erkenntnisse für künftige Gestaltungsprozesse ab (Fach Werken und Gestalten, LB 1: Gestaltungselemente und Gestaltungsprinzipien).

### 19.3.5 Medienbildung

Im LehrplanPLUS ist die Medienbildung als übergreifendes Bildungs- und Erziehungsziel verankert (StMBKWK 2014, S. 35). Schülerinnen und Schüler sollen Kenntnisse und Fertigkeiten erwerben, um sachgerecht, selbstbestimmt und verantwortungsvoll in einer multimedial geprägten Gesellschaft zu handeln. Weiter bedeutet das, dass Medien bewusst und reflektiert für private und schulische Zwecke genutzt werden sollen. Besonders in Hinblick auf den Kompetenzrahmen zur Medienbildung an bayerischen Schulen (ISB 2017) wird deutlich, dass das Programmieren von Informatiksystemen einen erheblichen Beitrag zur Medienbildung leisten kann:

Die Schülerinnen und Schüler ...

- handhaben Informatiksysteme zielorientiert, durchdringen ihre Funktionsweisen sowie grundlegenden Prinzipien und setzen sie zur Bewältigung neuer Herausforderungen ein (Bereich 1: Basiskompetenzen).
- strukturieren, modellieren und bereiten Daten und Informationen auf (Bereich 2: Suchen und Verarbeiten).
- verwenden analoge und digitale Werkzeuge zur effektiven Gestaltung kollaborativer und individueller Lernprozesse und teilen Resultate mit anderen (Bereich 3: Kommunizieren und Kooperieren).
- setzen Werkzeuge zur Realisierung verschiedener Medienprodukte ein (Bereich 4: Produzieren und Präsentieren).
- analysieren und bewerten Gestaltungsmittel, Strukturen und Wirkungsweisen von Informatiksystemen (Bereich 5: Analysieren und Reflektieren).

# 20 Abschließende Bemerkungen

Die Grundschule als erste gemeinsame Schule ermöglicht allen Kindern eine grundlegende schulische Bildung. Diese sollte nicht nur den Aufbau von Medienkompetenzen, sondern auch den Aufbau informatischer Kompetenzen umfassen. Zwar gibt es bereits eine Vielzahl von Unterrichtsmaterialien für diesen Zweck, doch sind diese oft weder in ein didaktisches Konzept eingebettet noch umfassend evaluiert. Wie Lehrkräfte der Grundschule in die Lage versetzt werden können, Materialien zu informatischen Themen im Unterricht einzusetzen und sich dies auch zuzutrauen, wird meist vernachlässigt. Diese Arbeit ging daher von einer mehrperspektivischen Zielsetzung aus: In einem ersten Schritt sollte ein Unterrichtskonzept und Materialien für die Grundschule entwickelt werden, die nicht nur auf der Informatik und Informatikdidaktik, sondern auch auf entwicklungspsychologischen sowie grundschuldidaktischen und -pädagogischen Erkenntnissen basiert. Hierfür wurde exemplarisch das Themengebiet der Programmierung ausgewählt. Darüber hinaus sollte ein passendes Fortbildungskonzept für Grundschullehrkräfte ausgestaltet werden. Für die Entwicklung und Erprobung der beiden Konzepte wurde der Forschungsansatz der *Design-Based Research* gewählt. Die Begleitforschung legt den Fokus auf die Überarbeitung der beiden Konzepte, die Auswirkungen auf die Schülerinnen und Schüler bzw. Lehrkräfte sowie Gelingensbedingungen für das Programmieren in der Grundschule.

## 20.1 Zusammenfassung der Ergebnisse

Ziel dieser Studie war, zu untersuchen, wie das Programmieren in der Grundschule implementiert werden kann. Dabei wurden drei verschiedene Ebenen betrachtet: *Unterricht*, *Lehrkräfte* und *Rahmenbedingungen*. Im Folgenden werden die Ergebnisse dieser Arbeit anhand der einzelnen Forschungsfragen zusammengefasst.

*RQ 1 Wie kann das Programmieren im Unterricht der Grundschule behandelt werden?*

(1-a) Welche Ziele sind in einem Unterrichtskonzept zum Programmieren in der Grundschule anzustreben?

Ausgehend von der fachlichen Klärung des Programmierens sowie dem Stand der informatischen Bildung im Primarbereich wurden Ziele auf verschiedenen Ebenen formuliert: Lernziele im Sinne informatischer Kompetenzen und Ziele bzgl. affektiver Merkmale (siehe Abschnitt 14.1). Diese konnten aufgrund der Erprobungen der Unterrichtssequenz als angemessen für die Zielgruppe der Schülerinnen und Schüler der dritten und vierten Jahrgangsstufe eingestuft werden (siehe Abschnitt 17.3).

(1-b) Welche Prinzipien zur Entwicklung eines Unterrichtskonzepts zum Programmieren in der Grundschule ergeben sich aus der Analyse begrifflicher, theoretischer und empirischer Grundlagen in den Bereichen der Entwicklungspsychologie, Grundschulpädagogik und -didaktik sowie Fachdidaktik und -wissenschaft der Informatik?

Auf der Grundlage einer umfassenden Literaturrecherche in den genannten Themenbereichen (siehe Teil I und III) wurden acht *Design Principles* entwickelt, die sich einerseits auf die Gestaltung von Lernaktivitäten und andererseits auf die Gestaltung der darauf bezogenen Lehrhandlungen beziehen (siehe Abschnitt 14.2). Die entwickelten Gestaltungsprinzipien wurden sowohl durch die Erprobung in unterrichtsnahen Lehr-Lernsituationen (siehe Kapitel 16) als auch durch die Erprobung im Grundschulunterricht (siehe Kapitel 17) validiert (siehe Abschnitt 19.1).

(1-c) Wie können diese Prinzipien in einem konkreten Unterrichtskonzept zum Programmieren in der Grundschule umgesetzt werden?

Die entwickelten *Design Principles* wurden zunächst in einem mehrstufigen Operationalisierungsprozess zu jeweiligen Umsetzungsprinzipien und Musterbeispielen konkretisiert (siehe Abschnitt 14.2). Auf dieser Basis wurde im Anschluss eine prototypische Unterrichtssequenz konstruiert, deren didaktische Konzeption den Entwurf einer unterrichtlichen Handlungslinie sowie ein Artikulationsschema mit Beschreibungen von Lernaktivitäten und Lehrhandlungen umfasst (siehe Abschnitt 14.3). Die entwickelte Unterrichtssequenz wurde zunächst im Rahmen einer Pilotstudie und zwei weiteren Erprobungszyklen mit 115 Schülerinnen und Schülern der dritten und vierten Jahrgangsstufe in unterrichtsnahen Lehr-Lernsituationen erprobt und weiterentwickelt (siehe Abschnitte 16.1, 16.2 und 16.3). Im Anschluss wurde die Unterrichtssequenz von 41 Grundschullehrkräften in ihrem Unterricht erprobt, angepasst und erweitert (siehe Abschnitt 17.1). Die daraus resultierende didaktische Konzeption der Unterrichtssequenz ist im Anhang nachzulesen (siehe Abschnitt B.1.2).

(1-d) Welche Auswirkungen hat das Unterrichtskonzept auf Schülerinnen und Schüler?

Für die Untersuchung dieser Frage wurden Lehrkräfte interviewt, welche die Unterrichtssequenz mit Schülerinnen und Schülern der dritten und vierten Jahrgangsstufe erprobt hatten. Die Interviews wurden im Anschluss mithilfe der inhaltlich strukturierenden qualitativen Inhaltsanalyse ausgewertet (siehe Abschnitt 17.3). In der Analyse konnten verschiedene Effekte auf die Schülerinnen und Schüler herausgearbeitet werden, die sich vorrangig auf die Veränderung von übergreifenden Basiskompetenzen und affektiven Merkmalen bezogen. Untersuchungen, die sich mit der Analyse der Programmierergebnisse von Schülerinnen und Schülern befassten, wurden im Rahmen des zweiten Erprobungszyklus der Unterrichtssequenz durchgeführt (siehe Abschnitt 16.2), werden in dieser Arbeit jedoch nur referenziert.

*RQ 2 Wie können Lehrkräfte der Grundschule ohne einschlägige Vorkenntnisse befähigt werden, das Programmieren im Unterrichts zu behandeln?*

(2-a) Welche Prinzipien zur Entwicklung eines Fortbildungskonzepts ergeben sich aus dem Unterrichtskonzept sowie aus der Analyse theoretischer und empirischer Grundlagen im Bereich der Lehrerbildung und der Fachdidaktik der Informatik?

Ausgehend von der Unterrichtssequenz und einer umfassenden Literaturrecherche in den genannten Themenbereichen (siehe Teil III) wurden sechs *Design Principles* entwickelt, die sich zum einen auf die Rahmenbedingungen und zum anderen auf die Lernaktivitäten eines Fortbildungskonzeptes beziehen (siehe Abschnitt 15.2). Diese Gestaltungsprinzipien wurden durch die Erprobung mit insgesamt 41 Lehrkräften der Grundschule validiert (siehe Abschnitt 19.2).

(2-b) Wie können diese Prinzipien in einem konkreten Fortbildungskonzept zum Programmieren in der Grundschule umgesetzt werden?

In Entsprechung zum Vorgehen bei der Entwicklung der Unterrichtssequenz wurden die Gestaltungsprinzipien zu jeweiligen Umsetzungprinzipien und Musterbeispielen ausgestaltet (siehe Abschnitt 15.2). Darauf aufbauend wurde ein Grobentwurf sowie ein Artikulationsschema des geplanten Verlaufs der Fortbildung entwickelt und mit 41 Lehrkräften erprobt. Die daraus resultierende didaktische Konzeption des Fortbildungskonzepts ist im Anhang nachzulesen (siehe C.1.2).

(2-c) Wie bewährt sich das entwickelte Fortbildungskonzept in der Praxis?

Zur Beantwortung der Fragestellung wurde das Fortbildungskonzept mit 41 Grundschullehrkräften erprobt (siehe Abschnitt 18.1). Im Rahmen dieser Erprobung wurden explorative Interviews mit den Lehrkräften geführt, die anschließend mithilfe einer inhaltlich strukturierenden qualitativen Inhaltsanalyse in Hinblick auf Äußerungen zum Fortbildungskonzept ausgewertet wurden (siehe Abschnitt 18.2). Die Ergebnisse zeigen, dass sowohl die Gestaltung der Rahmenbedingungen als auch die Lernaktivitäten des Fortbildungskonzepts sehr positiv bewertet wurden.

(2-d) Wie implementieren Lehrkräfte der Grundschule das Unterrichtskonzept in der Praxis, wandeln es ab oder erweitern es?

Zur Klärung der Fragestellung wurde die Unterrichtssequenz in einem ersten Schritt von den 41 Grundschullehrkräften, die im Vorfeld fortgebildet wurden, in ihrem Unterricht erprobt. In einem zweiten Schritt wurden diese in Form von explorativen Interviews und Fragebögen zur Umsetzung befragt (siehe Abschnitt 8.5). Die Interviews wurden hinsichtlich der Unterrichtssequenz mittels der inhaltlich strukturierenden qualitativen Inhaltsanalyse untersucht und durch ausgewählte Items der Fragebögen ergänzt (siehe Abschnitt 17.2). Im Fokus stand dabei das Format, in dem die Lehrkräfte die Unterrichtssequenz erprobten, die Erfahrungen, die sie dabei machten sowie die Frage, wie sie die Unterrichtssequenz modifizierten und erweiterten.

(2-e) Welche Auswirkungen hat das Fortbildungskonzept auf Lehrkräfte?

Zur Untersuchung der Fragestellung wurden die Lehrkräfte, die an der Fortbildung teilnahmen, interviewt und in Fragebögen zum Fortbildungskonzept befragt. Letztere wurden zu verschiedenen Zeitpunkten der Erprobung erhoben (siehe Abschnitt 8.5). Die Interviews wurden mittels einer inhaltlich strukturierenden qualitativen Inhaltsanalyse auf Äußerungen der Lehrkräfte zu den Auswirkungen der Fortbildung hin untersucht und durch ausgewählte Items der Fragebögen ergänzt (siehe Abschnitt 18.3). Hier wurden zum Großteil Auswirkungen bzgl. affektiver Merkmale thematisiert – Motivation, Selbstwirksamkeit und Interesse bzgl. des Programmierens und dessen Umsetzung im Unterricht. Durch die quantitative Analyse der Fragebögen konnten Teile dieser Ergebnisse bestätigt und weitere Auswirkungen identifiziert werden: eine signifikante Steigerung der Sicherheit im Umgang mit Computern und Computeranwendungen, eine signifikante Steigerung der Selbstwirksamkeit bzgl. der Umsetzung der Fortbildungsinhalte sowie eine Steigerung der Selbstwirksamkeitserwartung bzgl. der eigenen Programmierfähigkeit, die jedoch kein Signifikanzniveau erreicht hat.

*RQ 3 Welche Rahmenbedingungen wären für das Programmieren in der Grundschule wünschenswert?*

(3-a) Mit welchen Herausforderungen sind die Lehrkräfte während der Erprobung des Programmierens in der Grundschule konfrontiert?

Um die Forschungsfrage zu beantworten wurden in einem ersten Schritt Herausforderungen hinsichtlich des Programmierens in der Grundschule identifiziert. Zu diesem Zweck wurden explorative Interviews analysiert, die mit Grundschullehrkräften geführt wurden, welche zum einen an der Fortbildung teilgenommen und zum anderen das Programmieren im Unterricht erprobt hatten. In der Analyse konnten Herausforderungen in Bezug auf die Rahmenbedingungen, den Unterricht und die Schülerinnen und Schüler herausgearbeitet werden.

(3-b) Welche Gelingensbedingungen für das Programmieren in der Grundschule ergeben sich aus der Erprobung in der Praxis?

Aus den identifizierten Herausforderungen wurden in einem nächsten Schritt vier Gelingensbedingungen für das Programmieren in der Grundschule abgeleitet: (1) Verankerung im Lehrplan, (2) Technische Infrastruktur, (3) Fortbildung von Lehrkräften und (4) adäquate Unterrichtsmaterialien. Da eine Verankerung im Lehrplan momentan nicht abzusehen ist, wurden Möglichkeiten einer Anbindung zum LehrplanPLUS der bayerischen Grundschule erarbeitet (siehe Abschnitt 19.3).

## 20.2 Beitrag der Arbeit

Im Rahmen der Arbeit wurden verschiedene Lernangebote und Konzepte entwickelt, die nun für Unterrichtspraktiker und Fortbildende zur Verfügung stehen. Auf Ebene des Unterrichts sind hier zum einen die Zielvorstellungen und *Design Principles* zu nennen, zum anderen deren konkrete didaktische Ausarbeitung in Form der angereicherten unterrichtlichen Handlungslinie. Darüber hinaus wurden konkrete Unterrichtsmaterialien entwickelt, die unter der ausgezeichneten *Creative Commons*-Lizenz online zugänglich sind[1]. Die Materialien umfassen verschiedenste *Unplugged*-Aufgaben, Programmieraufgaben inklusive der dazugehörigen Scratch-Dateien und eine Methodik, um mit den Schülerinnen und Schülern *unplugged* mit haptischen Scratch-Blöcken zu programmieren. Auf Ebene der Lehrkräfte bzw. ihrer Fortbildung sind ebenfalls die jeweiligen Zielvorstellungen, *Design Principles* und die konkrete didaktische Ausarbeitung in Form der angereicherten Handlungslinie zu nennen.

Zum Zeitpunkt der Veröffentlichung dieser Arbeit wurden die *Design Principles* für die Gestaltung der Unterrichtssequenz von der Autorin bereits zur Gestaltung des Programmierkapitels eines Schulbuchs für die Mittelschule herangezogen (Boesch, Geldreich und Glatz 2020). Die *Design Principles* für die Gestaltung des Fortbildungskonzepts und die Unterrichtsmaterialien wurden darüber hinaus der Akademie für Lehrerfortbildung und Personalführung zur Verfügung gestellt und dienten als Grundlage für die dortige Multiplikatorenausbildung für das Fach Informatik in der Mittelschule.

Im Anschluss an das Projekt *AlgoKids – Algorithmen für Kinder* wurde im Auftrag des Bayerischen Staatsministeriums für Unterricht und Kultus ein Arbeitskreis am Staatsinstitut für Schulqualität und Bildungsforschung gegründet, der von der Autorin begleitet wird. Der Arbeitskreis erarbeitet auf Basis der von der Autorin entwickelten Konzepte und Forschungsergebnisse ein Konzept für die Ausweitung des Programmierens in der Grundschule in Form einer Multiplikatorenausbildung an der ALP. Dem Arbeitskreis wurden sämtliche Fortbildungsmaterialien zur Verfügung gestellt, die im Rahmen dieser Arbeit entwickelt wurden. Diese umfassen verschiedene *Unplugged*-Aufgaben, Handreichungen, Programmieraufgaben inklusive der dazugehörigen Scratch-Dateien, entsprechende PowerPoint-Präsentationen sowie Materialien für die Stationenarbeit zu *Computational Thinking* und Zugängen zum Programmieren in der Grundschule.

Die Teilstudien dieser Arbeit stellen einen ersten Schritt zur empirischen Validierung der Kompetenzen für informatische Bildung im Primarbereich in Bezug auf den Inhaltsbereich Algorithmen (GI 2019) sowie die Zieldimensionen informatischer Bildung im Elementar- und Primarbereich auf Ebene der Kinder und Lehrkräfte dar. Die empirischen Ergebnisse sind darüber hinaus für künftige Interventionen und für die Forschung zum Programmieren in der Grundschule relevant , da sie zur Klärung von Teilaufgaben der *Didaktischen Rekonstruktion* genutzt werden können (siehe Kapitel 12).

[1] https://www.edu.sot.tum.de/ddi/fuer-lehrkraefte/grundschule/

Die vorliegende Arbeit leistet außerdem verschiedene Beiträge zur Forschungspraxis. Auf der Ebene der Datenerhebung sind hier die methodische Ausgestaltung des explorativen Interviews (siehe Abschnitt A.2) sowie die entwickelten Fragebögen zu nennen (siehe Abschnitt A.3). Zudem befinden sich die für die Fragebögen verwendeten bzw. entwickelten Skalen im Anhang dieser Arbeit (siehe Anhang A.4). In Bezug auf das zugrundeliegende Forschungsdesign der *Design-Based Research* leistet die Arbeit insbesondere hinsichtlich der konkreten Entwicklung und didaktischen Ausgestaltung der *Design Principles* einen Beitrag, da dieser Aspekt in der einschlägigen Literatur oftmals nur umrissen und nicht im Detail geschildert wird.

## 20.3 Grenzen der Arbeit und Qualität der Erkenntnisse

Sowohl in Hinblick auf die Unterrichtssequenz als auch auf das Fortbildungskonzept übernahm die Autorin die Doppelrolle der Forscherin und Lehrerin bzw. Fortbildenden. Damit einher geht eine mögliche Beeinträchtigung der Objektivität, z. B. das Ausblenden negativer Aspekte der selbst entwickelten didaktischen Konzepte oder eine mögliche – wenn auch unbeabsichtigte – Verfälschung der erhobenen Daten. Um dem entgegenzuwirken, wurden verschiedene Maßnahmen ergriffen. Bei allen Durchführungen der Unterrichtssequenz im ersten, zweiten und dritten Erprobungszyklus, bei denen die Autorin die Rolle der Lehrerin einnahm, war eine zweite Person anwesend, mit der die jeweiligen Durchführungen im Anschluss besprochen und ausgewertet wurden. Darüber hinaus wurde die Unterrichtssequenz im zweiten und dritten Zyklus parallel mit einer weiteren Gruppe erprobt, die nicht von der Autorin unterrichtet wurde. Zusätzlich fand eine Bewertung und Weiterentwicklung der Unterrichtssequenz auf Grundlage der Analyse verschiedener Daten und des Einholens von regelmäßigem multiperspektivischem Feedback statt. Um dennoch eine objektivere Sichtweise auf die Unterrichtssequenz zu ermöglichen, wurde sie im vierten Zyklus von Grundschullehrkräften erprobt und bewertet. Auch bei der Erprobung des Fortbildungskonzepts war stets eine zweite Person anwesend, mit der die einzelnen Fortbildungsveranstaltungen nach der jeweiligen Durchführung besprochen und bewertet wurden. Zusätzlich wurde bereits bei der Entwicklung der didaktischen Konzeption multiperspektivisches Feedback eingeholt. Zur Bewertung des Fortbildungskonzepts wurden zudem verschiedene Daten erhoben, die qualitative und quantitative Auswertungen ermöglichten. Sowohl in Hinblick auf die Unterrichtssequenz als auch auf das Fortbildungskonzept wurden alle Phasen des Forschungsprozesses systematisch dokumentiert und sämtliche Zyklen umfassend beschrieben.

Auch hinsichtlich der empirischen Methodik muss die Rolle der Autorin betrachtet werden. So wurden z. B. sämtliche Interviews mit den Lehrkräften von der Autorin selbst geführt. Dies könnte dazu führen, dass die Lehrkräfte davon absehen, negative Aspekte der Unterrichtssequenz oder des Fortbildungskonzepts zu thematisieren. Um dies abzumildern, wurde im Vorfeld explizit thematisiert, dass man genau an diesen Punkten interessiert sei (siehe Anhang A.2). Zusätzlich ermöglichten die Fragebögen den Lehrkräften, ihre Meinung anonym zu äußern.

Die Auswahl der Schulen, deren Lehrkräfte an der Fortbildung teilnahmen und die Unterrichtssequenz im Rahmen dessen erprobten, erfolgte durch das Bayerische Staatsministerium für Unterricht und Kultus. In Bezug auf die Repräsentativität der Stichprobe können keine näheren Auskünfte gegeben werden, die Auswahl erfolgte laut Ministerium jedoch unter Berücksichtigung der für Schulversuche üblichen Repräsentativitätskriterien, wie ihrer Größe und Lage (einzügig-mehrzügig; Stadt-Land; Einbezug aller Regierungsbezirke, siehe Abbildung 3.1), sowie ihren Erfahrungen im Bereich der digitalen Bildung und ihrer digitalen Ausstattung. Auch darauf, welche Lehrkräfte die Schulen zur Teilnahme an der Fortbildung entsandten, hatte die Autorin keinen Einfluss. Allerdings war die Stichprobe der Lehrkräfte hinsichtlich ihrer Vorerfahrung, ihres Alters und Berufserfahrung sehr heterogen. Die Geschlechterverteilung war sehr unausgewogen, da nur zwei männliche Lehrkräfte teilnahmen, da dies jedoch der Tendenz in deutschen Grundschulen entspricht, wird es nicht als problematisch eingestuft. Eine Einschränkung hinsichtlich der Stichprobe ist jedoch, dass die Konzepte ausschließlich an bayerischen Grundschulen bzw. mit bayerischen Lehrkräften erprobt wurden.

Zur Qualität der empirischen Erkenntnisse der Teilstudien ist anzumerken, dass diese auf Selbsteinschätzungen und Selbstauskünften der Lehrkräfte beruhen. Aufgrund der Beschränkungen für die Forschung an Schulen durch das bayerische Staatsministerium für Unterricht und Kultus (siehe Abschnitt 8.4) war es nicht möglich, die Kompetenzen von Schülerinnen und Schüler sowie Lehrkräften direkt zu messen.

Hinsichtlich der Generalisierbarkeit der Ergebnisse dieser Arbeit ist deutlich zwischen den Konzepten der Unterrichtssequenz und der Fortbildung zu unterscheiden. Da das Fortbildungskonzept nur mit einer Gruppe von Lehrkräften erprobt wurde, gehen die daraus resultierenden Erkenntnisse nicht über eine *Local Theorie* bzw. einem *Proof of Principle* hinaus (siehe Abschnitte 6.3 und 7.4). Die Unterrichtssequenz wurde jedoch in verschiedenen Settings erprobt und die zugrundeliegenden *Design Principles* wurden von den Lehrkräften zum Teil unterschiedlich ausgestaltet. In diesem Fall sind die Erkenntnisse als *High-Level Theories* einzuordnen, die sich durch eine breite Anwendbarkeit auszeichnen.

In Bezug auf die entwickelte Unterrichtssequenz ist kritisch anzumerken, dass – auch wenn Aspekte des *Universal Design for Learning* (siehe Abschnitt 12.3.3) bei ihrer Entwicklung berücksichtigt wurden – die meisten Unterrichtsinhalte visualisiert werden. Der Großteil der Unterrichtsmaterialien und die visuelle Programmiersprache Scratch sind für Schülerinnen und Schüler mit Sehschädigung nicht geeignet, da man sie nur vollumfänglich verstehen kann, wenn man sie sieht (siehe Abschnitt 9.2.2). Die Unterrichtssequenz ist daher nicht inklusiv.

## 20.4 Anregungen für weitere Forschung

Die Ergebnisse dieser Dissertation bilden eine Grundlage für weitere Forschung im Bereich der informatischen Bildung im Primarbereich und der entsprechenden Fortbildung von Lehrkräften. Darüber hinaus ergeben sich konkrete Ansätze für weitere Forschungsarbeiten, die im Folgenden skizziert werden.

Im Rahmen dieser Arbeit konnten verschiedene Auswirkungen des Programmierens auf Schülerinnen und Schüler und Lehrkräfte festgestellt werden. Hinsichtlich der Veränderung der affektiven Merkmale der Lehrkräfte wäre von Interesse, ob diese Veränderung nachhaltig war. Denkbar wäre hier z. B. eine *Follow-Up* Befragung der Lehrkräfte, die an den Fortbildungen teilgenommen haben. Auch auf Ebene der Schülerinnen und Schüler wäre eine weiterführende Untersuchung denkbar, die untersucht, welchen Einfluss der Kontakt mit informatischen Themen in der Grundschule längerfristig auf das informatische Selbstbild und Interesse der Schülerinnen und Schüler hat oder welche Konsequenzen sich daraus für den Informatikunterricht in der Sekundarstufe ergeben. Da im Rahmen der Teilstudie zu den Auswirkungen des Programmierens auf Schülerinnen und Schüler nur Einschätzungen der Lehrkräfte untersucht wurden, stellt ein weiteres offenes Forschungsdesiderat die Entwicklung und Validierung eines Instruments dar, das eine tatsächliche Kompetenzmessung ermöglicht. In diesem Zusammenhang bleibt auch zu klären, ob das Programmieren in der Grundschule tatsächlich zur Ausbildung von *Computational Thinking* führt.

Da das entwickelte Fortbildungskonzept nur in einem Zyklus erprobt wurde, sind hier weitere Erprobungen denkbar. Da das Konzept in Zukunft für die Ausbildung von Multiplikatorinnen und Multiplikatoren genutzt und von diesen eingesetzt wird, bieten sich in diesem Rahmen weitere Untersuchungen an, die Aussagen über die Anwendbarkeit der Erkenntnisse dieser Arbeit ermöglichen würden.

Da die Ständige Wissenschaftliche Kommission der Kultusministerkonferenz empfiehlt, dass informatische Inhalte in Zukunft im Rahmen des Sachunterrichts behandelt werden, ist ein weiteres Forschungsdesiderat die Ausschärfung des entwickelten Unterrichtskonzepts für den Sachunterricht. Damit einhergehend sollte auch die erste Phase der Lehrerbildung in den Blick genommen werden. Hier gilt es zu untersuchen, wie Studierende des Grundschullehramts befähigt werden können, informatische Inhalte im Sachunterricht zu thematisieren.

# VERZEICHNISSE

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Literaturverzeichnis


Adams, Joel C. und Andrew R. Webster (2012). „What Do Students Learn About Programming from Game, Music Video, And Storytelling Projects?“ In: *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (SIGCSE)*. New York, NY: ACM, S. 643–648.

AG Medien & Digitalisierung der GDSU (2019). *Sachunterricht und Digitalisierung. Positionspapier der GDSU (Stand 29.10.2019)*. URL: https://gdsu.de/sites/default/files/PDF/GDSU_2021_Positionspapier_Sachunterricht_und_Digitalisierung_deutsch_de.pdf.

Aivaloglou, Efthimia und Felienne Hermans (2019). „Early Programming Education and Career Orientation. The Effects of Gender, Self-Efficacy, Motivation and Stereotypes“. In: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE)*. Hrsg. von Elizabeth Hawthorne u. a. New York, NY: ACM, S. 679–685. DOI: 10.1145/3287324.3287358.

Akao, Kensuke und Johannes Fischer (2021). „Zum Stand der Lehramtsausbildung für einen inklusiven Informatikunterricht“. In: *Informatik – Bildung von Lehrkräften in allen Phasen (INFOS)*. Hrsg. von Ludger Humbert. Bonn: Gesellschaft für Informatik, S. 291–294. DOI: 10.18420/infos2021_k217.

Akao, Kensuke und Johannes Fischer (2022). „Code-Puzzle für inklusiven Informatikunterricht. Alle Kinder lernen mit der für Förderschulen entwickelten Idee interaktiv!“ In: *Inklusion mit Informatik. 10. Münsteraner Workshop zur Schulinformatik*. Hrsg. von Marco Thomas und Michael Weigend. Münster: Books on Demand, S. 9–20.

Aldorf, Anna-Maria (2016). *Lehrerkooperation und die Effektivität von Lehrerfortbildung*. Wiesbaden: Springer. DOI: 10.1007/978-3-658-11677-4.

Altrichter, Herbert und Peter Posch (2007). *Lehrerinnen und Lehrer erforschen ihren Unterricht. Unterrichtsentwicklung und Unterrichtsevaluation durch Aktionsforschung*. 4., überarbeitete und erweiterte Auflage. Bad Heilbrunn: Julius Klinkhardt.

Amanullah, Kashif und Tim Bell (2019). „Analysis of Progression of Scratch Users based on their Use of Elementary Patterns“. In: *The 14th International Conference on Computer Science and Education (ICCSE)*. Piscataway, NJ: IEEE, S. 573–578. DOI: 10.1109/ICCSE.2019.8845495.

Anderson, Terry und Julie Shattuck (2012). „Design-Based Research: A Decade of Progress in Education Research?“ In: *Educational Researcher* 41.1, S. 16–25. DOI: 10.3102/0013189X11428813.

Andre, Thomas u. a. (1999). „Competency Beliefs, Positive Affect, and Gender Stereotypes of Elementary Students and their Parents about Science versus Other School Subjects“. In: *Journal of Research in Science Teaching* 36.6, S. 719–747.

Armoni, Michal (2013). „On Teaching Abstraction in Computer Science to Novices“. In: *Journal of Computers in Mathematics and Science Teaching* 32.3, S. 265–284.

Bagge, Philip und Shuchi Grover (2020). „Before you Program, Plan!“ In: *Computer Science in K-12: An A-to-Z Handbook on Teaching Programming*. Hrsg. von Shuchi Grover. Palo Alto, CA: Edfinity, S. 12–21.

Bak, Nayeon, Byeong-Mo Chang und Kwanghoon Choi (2018). „Smart Block: A Visual Programming Environment for SmartThings“. In: *Proceedings of the 42nd Annual Computer Software and Applications Conference* (Tokyo, Japan). Hrsg. von Sorel Reisman. Piscataway, NJ: IEEE, S. 32–37. DOI: 10.1109/COMPSAC.2018.10199.

Bakker, Arthur (2018). *Design Research in Education. A practical Guide for Early Career Researchers*. Abingdon, Oxon und New York, NY: Routledge.

Balanskat, Anja und Katja Engelhardt (2015). *Computing our Future. Computer Programming and Coding Priorities, School Curricula and Initiatives across Europe*. Brüssel: European Schoolnet.

Balzert, Helmut und Peter Liggesmeyer (2011). *Lehrbuch der Softwaretechnik: Entwurf, Implementierung, Installation und Betrieb*. 3. Aufl. Lehrbücher der Informatik. Heidelberg: Spektrum Akademischer Verlag. DOI: 10.1007/978-3-8274-2246-0.

Bandura, Albert (1997). *Self-Efficacy: The Exercise of Control*. New York: Worth Publishers.

Barcelos, Thiago Schumacher u. a. (2018). „Mathematics Learning through Computational Thinking Activities: A Systematic Literature Review“. In: *Journal of Universal Computer Science* 24.7, S. 815–845.

Bates, Richard (2002). „The Impact of Educational Research: Alternative Methodologies and Conclusions“. In: *Research Papers in Education* 17.4, S. 403–408. DOI: 10.1080/0267152022000031379.

Batur, Fatma und Jan Strobl (2019). „Discipline-Specific Language Learning in a Mainstream Computer Science Classroom“. In: *Proceedings of the 14th Workshop in Primary and Secondary Computing Education on (WiPSCE)*. Hrsg. von Quintin Cutts und Thorsten Brinda. New York, NY: ACM, S. 1–4. DOI: 10.1145/3361721.3362115.

Baum, Kevin u. a. (2019). „Informatikunterricht in der Grundschule? – Erprobung und Auswertung eines Unterrichtsmoduls mit Calliope mini“. In: *Informatik für alle. 18. GI-Fachtagung Informatik und Schule (INFOS)*. Hrsg. von Arno Pasternak. Bonn: Gesellschaft für Informatik, S. 49–58.

Baumert, Jürgen und Mareike Kunter (2006). „Stichwort: Professionelle Kompetenz von Lehrkräften“. In: *Zeitschrift für Erziehungswissenschaft* 9.4, S. 469–520.

Baumert, Jürgen und Mareike Kunter (2011). „Das mathematikspezifische Wissen von Lehrkräften, kognitive Aktivierung im Unterricht und Lernfortschritte von Schülerinnen und Schülern“. In: *Professionelle Kompetenz von Lehrkräften. Ergebnisse des Forschungsprogramms COACTIV*. Hrsg. von Mareike Kunter u. a. Münster: Waxmann, S. 163–192.

Baumert, Jürgen u. a. (2011). „Professionelle Kompetenz von Lehrkräften, kognitiv aktivierender Unterricht und die mathematische Kompetenz von Schülerinnen und Schülern (COACTIV) - Ein Forschungsprogramm“. In: *Professionelle Kompetenz von Lehrkräften. Ergebnisse des Forschungsprogramms COACTIV*. Hrsg. von Mareike Kunter u. a. Münster: Waxmann, S. 7–25.

Beauftragter der Bundesregierung für die Belange von Menschen mit Behinderungen, Hrsg. (2007). *Die UN-Behindertenrechtskonvention. Übereinkommen über die Rechte von Menschen*

*mit Behinderungen*. URL: https://www.behindertenbeauftragter.de/DE/AS/rechtliches/un-brk/un-brk-node.html.

Bell, Tim (2016). „Demystifying Coding for Schools. What are we actually trying to teach?“ In: *Bulletin of EATCS* 120.

Bell, Tim und Caitlin Duncan (2018). „Teaching Computing in Primary Schools“. In: *Computer Science Education. Perspectives on Teaching and Learning in School*. Hrsg. von Sue Sentance, Erik Barendsen und Carsten Schulte. London u. a.: Bloomsbury Academic, S. 131–150.

Bell, Tim, Caitlin Duncan und Austen Rainer (2018). „What is Coding?“ In: *Creating the Coding Generation in Primary Schools. A Practical Guide for Cross-Curricular Teaching*. Hrsg. von Steve Humble. London und New York: Routledge, S. 3–21.

Bell, Tim und Jan Vahrenhold (2014). „CS Unplugged - How Is It Used, and Does It Work?“ In: *Adventures Between Lower Bounds and Higher Altitudes*. Hrsg. von Hans-Joachim Böckenhauer, Dennis Komm und Walter Unger. Cham: Springer, S. 497–521. DOI: 10.1007/978-3-319-98355-4_29.

Bell, Tim u. a. (2009). „Computer Science Unplugged. School Students Doing Real Computing without Computers“. In: *New Zealand Journal of Applied Computing and Information Technology* 13.1, S. 20–29.

Bender, Elena, Niclas Schaper und Andreas Seifert (2018). „Professionelle Überzeugungen und motivationale Orientierungen von Informatiklehrkräften“. In: *Journal for Educational Research Online* 10.1. DOI: 10.25656/01:15414.

Bender, Elena u. a. (2015a). „Identifying and Formulating Teachers' Beliefs and Motivational Orientations for Computer Science Teacher Education“. In: *Studies in Higher Education* 41.11, S. 1958–1973. DOI: 10.1080/03075079.2015.1004233.

Bender, Elena u. a. (2015b). „Towards a Competency Model for Teaching Computer Science“. In: *Peabody Journal of Education* 90.4, S. 519–532. DOI: 10.1080/0161956X.2015.1068082.

Bergner, Nadine und Kathrin Müller (2018). „Fachempfehlung Informatiksysteme“. In: *Frühe informatische Bildung – Ziele und Gelingensbedingungen für den Elementar- und Primarbereich*. Hrsg. von Stiftung „Haus der kleinen Forscher“. Opladen, Berlin, Toronto: Verlag Barbara Budrich, S. 268–301.

Bergner, Nadine u. a. (2018). „Zieldimensionen informatischer Bildung im Elementar- und Primarbereich“. In: *Frühe informatische Bildung – Ziele und Gelingensbedingungen für den Elementar- und Primarbereich*. Hrsg. von Stiftung „Haus der kleinen Forscher“. Opladen, Berlin, Toronto: Verlag Barbara Budrich, S. 38–267.

Berry, Miles (2015). *QuickStart Primary Handbook*. Swindon: BCS.

Berry, Miles (2018). „"Computing" als neues Schulfach“. In: *LOG IN* 189/190, S. 20–26.

Bers, Marina Umaschi u. a. (2014). „Computational Thinking and Tinkering: Exploration of an Early Childhood Robotics Curriculum“. In: *Computers & Education* 72, S. 145–157. DOI: 10.1016/j.compedu.2013.10.020.

Best, Alexander (2019). „Bild der Informatik von Grundschullehrpersonen. Ergebnisse eines mehrjährigen Projekts zu informatikbezogenen Vorstellungen“. In: *Informatik für alle. 18. GI-Fachtagung Informatik und Schule (INFOS)*. Hrsg. von Arno Pasternak. Bonn: Gesellschaft für Informatik, S. 59–68.

Best, Alexander (2020). „Vorstellungen von Grundschullehrpersonen zur Informatik und zum Informatikunterricht“. Dissertation. Münster: Westfälische Wilhelms-Universität.

Beyer, Sylvia u. a. (2003). „Gender Differences in Computer Science Students“. In: *Proceedings of the 34th Technical Symposium on Computer Science Education (SIGCSE)*. New York, NY: ACM, S. 49–53. DOI: 10.1145/611892.611930.

Birman, Beatrice F. u. a. (2000). „Designing Professional Development That Works“. In: *Educational Leadership* 57.8, S. 28–33.

Bitkom Bundesverband Informationswirtschaft, Telekommunikation und Neue Medien e. V. (2019). *Kinder und Jugendliche in der digitalen Welt*. Berlin. URL: https://www.bitkom.org/sites/default/files/2019-05/bitkom_pk-charts_kinder_und_jugendliche_2019.pdf.

Black, Paul u. a. (2004). „Working inside the Black Box: Assessment for Learning in the Classroom“. In: *Phi Delta Kappan* 86.1, S. 8–21. DOI: 10.1177/003172170408600105.

Blumenfeld, Phyllis C. u. a. (1991). „Motivating Project-Based Learning: Sustaining the Doing, Supporting the Learning“. In: *Educational Psychologist* 26.3-4, S. 369–398. DOI: 10.1080/00461520.1991.9653139.

BMBF - Bundesministerium für Bildung und Forschung (2016). *Bildungsoffensive für die digitale Wissensgesellschaft. Strategie des Bundesministeriums für Bildung und Forschung*. Berlin. URL: https://www.bildung-forschung.digital/digitalezukunft/de/bildung/hochschule/bildungsoffensive-fuer-die-digitale-wissensgesellschaft/bildungsoffensive-fuer-die-digitale-wissensgesellschaft_node.html.

BMDV - Bundesministerium für Digitales und Verkehr (2022). *Digitalstrategie Deutschland. Gemeinsam digitale Werte schöpfen*. URL: https://digitalstrategie-deutschland.de/.

BMWi - Bundesministerium für Wirtschaft und Energie (2016). *Digitale Bildung. Der Schlüssel zu einer Welt im Wandel*. URL: https://www.bmwi.de/Redaktion/DE/Publikationen/Digitale-Welt/digitale-bildung-der-schluessel-zu-einer-welt-im-wandel.html.

Bocconi, Stefania u. a. (2016). *Developing Computational Thinking in Compulsory Education. Implications for Policy and Practice*. Luxembourg: Publications Office of the European Union. DOI: 10.2791/792158.

Boesch, Pascal, Katharina Geldreich und Lisa-Andrea Glatz (2020). *StarkeSeiten 5/6. Informatik*. Stuttgart und Leipzig: Ernst Klett Verlag.

Bogner, Alexander (2014). *Interviews mit Experten. Eine praxisorientierte Einführung*. Wiesbaden: Springer VS. 105 S. DOI: 10.1007/978-3-531-19416-5.

Böhm, Corrado und Giuseppe Jacopini (1966). „Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules“. In: *Communications of the ACM* 9.5, S. 366–371. DOI: 10.1145/355592.365646.

Bonar, Jeffrey G. und Blaise W. Liffick (1987). *A Visual Programming Language for Novices*. Fort Belvoir, VA.

Bortz, Jürgen und Nicola Döring (2002). *Forschungsmethoden und Evaluation*. 3., überarbeitete Auflage. Berlin, Heidelberg: Springer. DOI: 10.1007/978-3-662-07299-8.

Bortz, Jürgen und Christof Schuster (2010). *Statistik für Human- und Sozialwissenschaftler*. Berlin, Heidelberg: Springer. DOI: 10.1007/978-3-642-12770-0.

Boshernitsan, Marat und Michael S. Downes (2004). *Visual Programming Languages: A Survey*. Hrsg. von EECS Department. University of California, Berkeley. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2004/6201.html.

Brämer, Martin u. a. (2020). „Eine digitale Perspektive für den Sachunterricht – ein Vorschlag zur Diskussion". In: *GDSU-Journal* 10.

Braught, Grant, Tim Wahls und L. Marlin Eby (2011). „The Case for Pair Programming in the Computer Science Classroom". In: *ACM Transactions on Computing Education* 11.1, S. 1–21. DOI: 10.1145/1921607.1921609.

Breiter, Andreas u. a. (2020). *Informatische Bildung und Technik in der Grundschule. Abschlussbereicht im Auftrag des Niedersächsischen Landesinstituts für schulische Qualitätsentwicklung (NLQ)*. de. DOI: 10.13140/RG.2.2.34054.80967.

Brennan, Karen (2013). „Learning Computing through Creating and Connecting". In: *Computer* 46.9, S. 52–59. DOI: 10.1109/MC.2013.229.

Brennan, Robert L. und Dale J. Prediger (1981). „Coefficient Kappa: Some Uses, Misuses, and Alternatives". In: *Educational and Psychological Measurement* 41.3, S. 687–699. DOI: 10.1177/001316448104100307.

Brinda, Thorsten u. a. (2019). *Frankfurt-Dreieck zur Bildung in der digital vernetzten Welt. Ein interdisziplinäres Modell*. URL: https://www.keine-bildung-ohne-medien.de/frankfurter-dreieck/.

Brosnan, Mark Jeremy (1999). „A New Methodology, an Old Story? Gender Differences in the "Draw-a-Computer-User" Test". In: *European Journal of Psychology of Education* 14.3, S. 375–385. DOI: 10.1007/BF03173121.

Brown, Ann L. (1992). „Design Experiments: Theoretical and Methodological Challenges in Creating Complex Interventions in Classroom Settings". In: *The Journal of the Learning Sciences* 2.2, S. 141–178.

Brown, John L. und Grant P. Wiggins (2004). *Making the Most of Understanding by Design*. Alexandria, VA: Association for Supervision and Curriculum Development.

Brown, Neil Christopher Charles und Michael Kölling (2013). „A Tale of Three Sites: Resource and Knowledge Sharing Amongst Computer Science Educators". In: *Proceedings of the ninth Annual International ACM Conference on International Computing Education Research (ICER)*. Hrsg. von Beth Simon, Alison Clear und Quintin Cutts. New York, NY: ACM Press, S. 27–34. DOI: 10.1145/2493394.2493398.

Broy, Manfred (1998). *Informatik Eine grundlegende Einführung*. Berlin, Heidelberg: Springer Berlin Heidelberg. DOI: 10.1007/978-3-642-58722-1.

Broy, Manfred und Alexander Malkis (2019). *Logische und methodische Grundlagen der Programm- und Systementwicklung. Datenstrukturen, funktionale, sequenzielle und objektorientierte Programmierung*. Lehrbuch. Wiesbaden: Springer Fachmedien. DOI: 10.1007/978-3-658-26302-7.

Bruner, Jerome S. (1971). „Über kognitive Entwicklung". In: *Studien zur kognitiven Entwicklung. Eine kooperative Untersuchung am "Center for Cognitive Studies" der Harvard-Universität*. Hrsg. von Jerome S. Bruner, Rose R. Olver und Patricia Marks Greenfield. Stuttgart: Ernst Klett Verlag, S. 21–54.

Bruner, Jerome S. und Arnold Harttung (1973). *Der Prozeß der Erziehung*. 3. Auflage. Berlin: Berlin Verlag.

Buchholz, Malte, Mara Saeli und Carsten Schulte (2013). „PCK and Reflection in Computer Science Teacher Education". In: *Proceedings of the 8th Workshop in Primary and Secondary Computing Education (WiPSCE)*. Hrsg. von Michael E. Caspersen, Maria Knobelsdorf und Ralf Romeike. New York, NY: ACM, S. 8–16. DOI: 10.1145/2532748.2532752.

Buczynski, Sandy und Bobbi Hansen (2010). „Impact of Professional Development on Teacher Practice: Uncovering Connections". In: *Teaching and Teacher Education* 26.3, S. 599–607.

Burnett, Margaret M. (2001). „Visual Programming". In: *Wiley Encyclopedia of Electrical and Electronics Engineering*. Hrsg. von John G. Webster. New York, NY: John Wiley and Sons Ltd, S. 275–283. DOI: 10.1002/047134608X.W1707.

Byrne, J. R., L. Fisher und B. Tangney (2015). „Computer Science Teacher Reactions towards Raspberry Pi Continuing Professional Development (CPD) Workshops using the Bridge21 Model". In: *Proceedings of the 10th International Conference on Computer Science & Education (ICCSE)*. Piscataway, NJ: IEEE, S. 267–272. DOI: 10.1109/ICCSE.2015.7250254.

Camilleri, Patrick (2017). „Minding the Gap. Proposing a Teacher Learning-Training Framework for the Integration of Robotics in Primary Schools". In: *Informatics in Education* 16.2, S. 165–179. DOI: 10.15388/infedu.2017.09.

Capovilla, Dino J. (2015). *Inklusion in der Informatischen Bildung am Beispiel von Menschen mit Sehschädigung*. Dissertation. München: Technische Universität München.

Capovilla, Dino J. (2019). „Informatische Bildung und inklusive Pädagogik". In: *Informatik für alle. 18. GI-Fachtagung Informatik und Schule (INFOS)*. Hrsg. von Arno Pasternak. Bonn: Gesellschaft für Informatik, S. 35–46.

Caspersen, Michael E. (2018). „Teaching Programming". In: *Computer Science Education. Perspectives on Teaching and Learning in School*. Hrsg. von Sue Sentance, Erik Barendsen und Carsten Schulte. London u. a.: Bloomsbury Academic, S. 109–130.

Caspersen, Michael E. und Jens Bennedsen (2007). „Instructional Design of a Programming Course – A Learning Theoretic Approach". In: *Proceedings of the third International Workshop on Computing Education Research (ICER)*. New York, NY: ACM.

CDC - The Curriculum Development Council (2017a). *General Studies Curriculum Giude for Primary Schools. (Primary 1 - Primary 6)*. Hrsg. von The Education Bureau. Hong Kong. URL: https://www.edb.gov.hk/en/curriculum-development/cross-kla-studies/gs-primary/curriculum-documents.html.

CDC - The Curriculum Development Council (2017b). *Technology Education. Key Learning Area Curriculum Guide (Primary 1 - Secondary 6)*. Hrsg. von The Education Bureau. Hong Kong. URL: https://www.edb.gov.hk/en/curriculum-development/kla/technology-edu/curriculum-doc/index.html.

CDC - The Curriculum Development Council (2020). *Computational Thinking - Coding Education: Supplement to the Primary Curriculum*. Hrsg. von The Education Bureau. Hong Kong. URL: https://www.edb.gov.hk/en/curriculum-development/kla/technology-edu/curriculum-doc/index.html.

Chalmers, Christina (2018). „Robotics and Computational Thinking in Primary School". In: *International Journal of Child-Computer Interaction* 17, S. 93–100. DOI: 10.1016/j.ijcci.2018.06.005.

Chandler, Paul und John Sweller (1991). „Cognitive Load Theory and the Format of Instruction". In: *Cognition and Instruction* 8.4, S. 193–332.

Claus, Volker und Andreas Schwill (2006). *Duden Informatik A - Z. Fachlexikon für Studium, Ausbildung und Beruf*. 4. Aufl. Duden. Mannheim: Dudenverlag.

Cobb, Paul, Kara Jackson und Charlotte Dunlap (2015). „Design Research: An Analysis and Critique". In: *Handbook of International Research in Mathematics Education*. Hrsg. von Lyn D. English und David Kirshner. New York, NY und London: Routledge, S. 481–503.

Cobb, Paul u. a. (2003). „Design Experiments in Educational Research". In: *Educational Researcher* 32.1, S. 9–13. DOI: 10.3102/0013189X032001009.

Cohen, Jacob (1988). *Statistical Power Analysis for the Behavioral Sciences*. 2. Auflage. Hoboken: Taylor and Francis. DOI: 10.4324/9780203771587.

Collins, Allan (1992). „Towards a Design Science of Education". In: *New Directions in Educational Technology*. Hrsg. von Eileen Scanlon und Tim O'Shea. New York: Springer, S. 15–22.

Collins, Allan, Diana Joseph und Katerine Bielaczyc (2004). „Design Research: Theoretical and Methodological Issues". In: *The Journal of the Learning Sciences* 13.1, S. 15–42.

Condie, Rae und Bob Munro (2006). *The Impact of ICT in School - a Landscape Review*. Coventry: Becta.

CSTA - Computer Science Teachers Association (2017). *CSTA K-12 Computer Science Standards*. URL: http://www.csteachers.org/standards.

CSTA - Computer Science Teachers Association (2020). *CSTA Standards for Computer Science Teachers*. URL: https://csteachers.org/teacherstandards.

Curzon, Paul u. a. (2020). „Using Semantic Waves to Analyse the Effectiveness of Unplugged Computing Activities". In: *Proceedings of the 15th Workshop on Primary and Secondary Computing Education (WiPSCE)*. Hrsg. von Torsten Brinda und Michal Armoni. New York, NY: ACM. DOI: 10.1145/3421590.3421606.

D-EDK - Deutschschweizer Erziehungsdirektorenkonferenz (2016). „Medien und Informatik“. In: *Lehrplan 21. Bereinigte Fassung vom 29.02.2016*. Hrsg. von D-EDK.

Dagienė, Valentina, Tatjana Jevsikova und Gabrielė Stupurienė (2019). „Introducing Informatics in Primary Education: Curriculum and Teachers’ Perspectives“. In: *Informatics in Schools. New Ideas in School Informatics (ISSEP)*. Hrsg. von Sergei N. Pozdniakov und Valentina Dagienė. Cham: Springer International Publishing, S. 83–94.

Dalehefte, Inger Marie u. a. (2014). „Bilanz von neun Jahren SINUS an Grundschulen in Deutschland. Evaluation der mathematikbezogenen Daten im Rahmen von TIMSS 2011“. In: *Zeitschrift für Pädagogik* 60.2, S. 245–263.

Darling-Hammond, Lina u. a. (2017). *Effective Teacher Professional Development*. Palo Alto, CA: Learning Policy Institute.

Darsow, Annkathrin, Jennifer Paetsch und Anja Felbrich (2012). „Konzeption und Umsetzung der fachbezogenen Sprachförderung im BeFo-Projekt“. In: *Deutsch als Zweitsprache in Kindertageseinrichtungen und Schulen. Aneignung, Förderung, Unterricht*. Hrsg. von Stefan Jeuk. Stuttgart: Fillibach bei Klett, S. 215–234.

DBRC - The Design-Based Research Collective (2003). „Design-Based Research: An Emerging Paradigm for Educational Inquiry“. In: *Educational Researcher* 32.1, S. 5–8.

De Jong, Ton (2010). „Cognitive Load Theory, Educational Research, and Instructional Design: Some Food for Thought“. In: *Instructional Science* 38.2, S. 105–134. DOI: 10.1007/s11251-009-9110-0.

Deci, Edward L. und Richard M. Ryan (1993). „Die Selbstbestimmungstheorie der Motivation und ihre Bedeutung für die Pädagogik“. In: *Zeitschrift für Pädagogik* 39.2, S. 223–238.

Decristan, Jasmin u. a. (2014). „Heterogenität von Schülerleistungen in der Grundschule“. In: *Zeitschrift für Entwicklungspsychologie und Pädagogische Psychologie* 46.4, S. 181–190. DOI: 10.1026/0049-8637/a000115.

Dengel, Andreas und Ute Heuer (2018). „A Curriculum of Computational Thinking as a Central Idea of Information & Media Literacy“. In: *Proceedings of the 13th Workshop in Primary and Secondary Computing Education (WiPSCE)*. Hrsg. von Andreas Mühling und Quintin Cutts. New York, NY: ACM, S. 1–6. DOI: 10.1145/3265757.3265777.

Denham, Pearl (1993). „Nine- to Fourteen-Year-Old Children’s Conception of Computers Using Drawings“. In: *Behaviour & Information Technology* 12.6, S. 346–358. DOI: 10 . 1080 / 01449299308924399.

Denning, Peter J., Matti Tedre und Pat Yongpradit (2017). „Misconceptions About Computer Science“. In: *Communications of the ACM* 60.3, S. 31–33. DOI: 10.1145/3041047.

Diethelm, Ira und Juliana Goschler (2015). „Questions on Spoken Language and Terminology for Teaching Computer Science“. In: *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE)*. Hrsg. von Valentina Dagienė, Carsten Schulte und Tatjana Jevsikova. New York, NY: ACM, S. 21–26. DOI: 10.1145/2729094.2742600.

Diethelm, Ira, Juliana Goschler und Timo Lampe (2018). „Language und Computing“. In: *Computer Science Education. Perspectives on Teaching and Learning in School*. Hrsg. von Sue Sentance, Erik Barendsen und Carsten Schulte. London u. a.: Bloomsbury Academic, S. 207–219.

Diethelm, Ira und Melanie Schaumburg (2016). „IT2School – Development of Teaching Materials for CS Through Design Thinking“. In: *Informatics in Schools: Improvement of Informatics Knowledge and Perception (ISSEP)*. Hrsg. von Andrej Brodnik und Françoise Tort. Cham: Springer, S. 193–198. DOI: 10.1007/978-3-319-46747-4_16.

Diethelm, Ira u. a. (2011). „Die Didaktische Rekonstruktion für den Informatikunterricht“. In: *Informatik in Bildung und Beruf. 14. GI-Fachtagung Informatik und Schule (INFOS)*. Hrsg. von Marco Thomas. Bonn: Gesellschaft für Informatik, S. 77–86.

Diethelm, Ira u. a. (2020). „Investigation of the Informatics-Based Self-Concept of Primary School Children“. In: *Proceedings of the 15th Workshop on Primary and Secondary Computing Education (WiPSCE)*. Hrsg. von Torsten Brinda und Michal Armoni. New York, NY: ACM, S. 1–6. DOI: 10.1145/3421590.3421601.

DIVSI - Deutsches Institut für Vertrauen und Sicherheit im Internet (2015). *DIVSI U9-Studie: Kinder in der digitalen Welt*. Hamburg.

Döbeli Honegger, Beat (2017). *Mehr als 0 und 1. Schule in einer digitalisierten Welt*. 2., durchgesehene Auflage. Bern: hep der Bildungsverlag.

Döbeli Honegger, Beat und Michael Hielscher (2017). „Vom Lehrplan zur LehrerInnenbildung - Erste Erfahrungen mit obligatorischer Informatikdidaktik für angehende Schweizer PrimarlehrerInnen“. In: *Informatische Bildung zum Verstehen und Gestalten der digitalen Welt (INFOS)*. Hrsg. von Ira Diethelm. Bonn: Gesellschaft für Informatik, S. 97–107.

Du Boulay, Benedict (1986). „Some Difficulties of Learning to Program“. In: *Journal of Educational Computing Research* 2.1, S. 57–73. DOI: 10.2190/3LFX-9RRF-67T8-UVK9.

Duncan, Caitlin, Tim Bell und James Atlas (2017). „What do the Teachers Think? Introducing Computational Thinking in the Primary School Curriculum“. In: *Proceedings of the Nineteenth Australasian Computing Education Conference (ACE)*. New York, NY: Academic Press, S. 65–74. DOI: 10.1145/3013499.3013506.

Duncan, Caitlin, Tim Bell und Steve Tanimoto (2014). „Should Your 8-year-old Learn Coding?“ In: *Proceedings of the 9th Workshop in Primary and Secondary Computing Education (WiPSCE)*. New York, NY: ACM, S. 60–69. DOI: 10.1145/2670757.2670774.

Edelson, Daniel C. (2002). „Design Research: What We Learn When We Engage in Design“. In: *Journal of the Learning Sciences* 11.1, S. 105–121.

Ehlenz, Matthias, Birte Heinemann und Ulrik Schroeder (2022). „Heterogenität und Inklusion in der fachbezogenen Lehramtsausbildung. Reflexion & Praktische Erfahrungen aus fünf Iterationen eines Praktikums“. In: *Inklusion mit Informatik. 10. Münsteraner Workshop zur Schulinformatik*. Hrsg. von Marco Thomas und Michael Weigend. Münster: Books on Demand, S. 45–56.

Ehses, Erich (2012). „Algorithmen und Datenstrukturen“. In: *Taschenbuch der Informatik*. Hrsg. von Uwe Schneider. 7., neu bearbeitete Auflage. München: Fachbuchverlag Leipzig im Carl Hanser Verlag, S. 171–194.

Einsiedler, Wolfgang (2010). „Didaktische Entwicklungsforschung als Transferförderung“. In: *Zeitschrift für Erziehungswissenschaft* 13.1, S. 59–81. DOI: 10.1007/s11618-010-0106-y.

Einsiedler, Wolfgang (2011). „Was ist Didaktische Entwicklungsforschung?“ In: *Unterrichtsentwickung und Didaktische Entwicklungsforschung*. Hrsg. von Wolfgang Einsiedler. Bad Heilbrunn: Verlag Julius Klinkhardt, S. 41–70.

Einsiedler, Wolfgang (2014). „Lehr-Lernkonzepte für die Grundschule“. In: *Handbuch Grundschulpädagogik und Grundschuldidaktik*. Hrsg. von Wolfgang Einsiedler. 4. ergänzte und aktualisierte Auflage. Bad Heilbrunn: Verlag Julius Klinkhardt, S. 355–364.

Einsiedler, Wolfgang (2015). *Geschichte der Grundschulpädagogik. Entwicklungen in Westdeutschland und in der DDR*. Bad Heilbrunn: Verlag Julius Klinkhardt.

Eisinga, Rob, Manfred te Grotenhuis und Ben Pelzer (2013). „The Reliability of a Two-Item Scale: Pearson, Cronbach, or Spearman-Brown?“ In: *International Journal of Public Health* 58.4, S. 637–642. DOI: 10.1007/s00038-012-0416-3.

El-Hamamsy, Laila u. a. (2020). „A Computer Science and Robotics Integration Model for Primary School: Evaluation of a Large-Scale In-Service K-4 Teacher-Training Program“. eng. In: *Education and Information Technologies*, S. 1–31. DOI: 10.1007/s10639-020-10355-5.

Elliott, John (1991). *Action Research for Educational Change*. Developing teachers and teaching. Philadelphia: Open University Press.

Euler, Dieter (2014a). „Design Principles als Kristallisationspunkt für Praxisgestaltung“. In: *Design-Based Research*. Hrsg. von Dieter Euler und Peter F. E. Sloane. Zeitschrift für Berufs- und Wirtschaftspädagogik Beiheft 27. Stuttgart: Franz Steiner Verlag, S. 97–112.

Euler, Dieter (2014b). „Design Research. A Paradigm under Development“. In: *Design-Based Research*. Hrsg. von Dieter Euler und Peter F. E. Sloane. Zeitschrift für Berufs- und Wirtschaftspädagogik Beiheft 27. Stuttgart: Franz Steiner Verlag, S. 15–41.

European Commission/EACEA/Eurydice (2022). *Informatics Education at School in Europe. Eurydice Report*. Luxembourg: Publications Office of the European Union.

Fagerlund, Janne u. a. (2020). „Computational Thinking in Programming with Scratch in Primary Schools: A Systematic Review“. In: *Computer Applications in Engineering Education* 29.1, S. 12–28. DOI: 10.1002/cae.22255.

Falkner, Katrina, Rebecca Vivian und Nickolas Falkner (2014). „The Australian Digital Technologies Curriculum: Challenge and Opportunity“. In: *Proceedings of the Sixteenth Australasian Computing Education Conference (ACE)*. Hrsg. von Jacqueline Whalley und Daryl D’Souza. New York, NY: ACM, S. 3–12.

Fields, Deborah A., Yasmin B. Kafai und Michael T. Giang (2017). „Youth Computational Participation in the Wild“. In: *ACM Transactions on Computing Education* 17.3, S. 1–22. DOI: 10.1145/3123815.

Frädrich, Christoph u. a. (2020). „Common Bugs in Scratch Programs“. In: *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE)*. Hrsg. von Michail Giannakos u. a. New York, NY: ACM, S. 89–95. DOI: 10.1145/3341525.3387389.

Franklin, Diana u. a. (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 (SIGCSE)*. New York, NY: ACM, S. 552–557. DOI: 10.1145/2676723.2677288.

Freiberger, Ulli (2018). *Robot Karol 3.0*.

Funke, Alexandra und 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)*. Hrsg. von Erik Barendsen. New York, NY: ACM, S. 57–64. DOI: 10.1145/3137065.3137067.

Funke, Alexandra, Katharina Geldreich und 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)*. New York, NY, USA: ACM, S. 135–139. DOI: 10.1145/2999541.2999547.

Funke, Alexandra, Katharina Geldreich und 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)*. IEEE, S. 1229–1236.

Garet, Michael S. u. a. (2001). „What Makes Professional Development Effective? Results From a National Sample of Teachers“. In: *American Educational Research Journal* 38.4, S. 915–945. DOI: 10.3102/00028312038004915.

Gärtig-Daugs, Anja u. a. (2016). „Computer Science Experimenter's Kit for Use in Preschool and Primary School“. In: *Proceedings of the 11th Workshop in Primary and Secondary Computing Education (WiPSCE)*. Hrsg. von Jan Vahrenhold und Erik Barendsen. New York, NY: ACM, S. 66–71. DOI: 10.1145/2978249.2978258.

GDSU - Gesellschaft für Didaktik des Sachunterrichts, Hrsg. (2013). *Perspektivrahmen Sachunterricht. Vollständig überarbeitete und erweiterte Ausgabe*. Bad Heilbrunn: Klinkhardt.

Gebauer, Susanne (2019). „Praxisbezogene Beispiele vorschalten – den Theorie-Input nachschalten: Gestaltungsvarianten für Lehrer(innen)fortbildungen“. In: *Grundschulpädagogik zwischen Wissenschaft und Transfer*. Hrsg. von Christian Donie u. a. Wiesbaden: Springer Fachmedien Wiesbaden, S. 162–168.

Geldreich, Katharina, Alexandra Funke und Peter Hubwieser (2016). „A Programming Circus for Primary Schools“. In: *Informatics in Schools: Improvement of Informatics Knowledge and Perception*. Hrsg. von Andrej Brodnik und Françoise Tort. Cham: Springer, S. 46–47.

Geldreich, Katharina, Alexandra Funke und Peter Hubwieser (2017). „Willkommen im Programmierzirkus - Ein Programmierkurs für Grundschulen“. In: *Informatische Bildung zum Verstehen und Gestalten der digitalen Welt (INFOS)*. Hrsg. von Ira Diethelm. Bonn: Gesellschaft für Informatik, S. 327–334.

Geldreich, Katharina, Alexandra Simon und Elena Starke (2019). „Which Perceptions Do Primary School Children Have about Programming?“ In: *Proceedings of the 14th Workshop in Primary and Secondary Computing Education (WiPSCE)*. New York, NY, USA: ACM Press, S. 1–7. DOI: 10.1145/3361721.3361728.

George, Darren und Paul Mallery (2016). *IBM SPSS Statistics 23 Step by Step. A Simple Guide and Reference*. New York, NY: Routledge.

Gerstenmaier, Jochen und Heinz Mandl (1995). „Wissenserwerb unter konstruktivistischer Perspektive“. In: *Zeitschrift für Pädagogik* 41.6, S. 867–888.

GFD - Gesellschaft für Fachdidaktik e. V. (2018). *Fachliche Bildung in der digitalen Welt. Positionspapier der Gesellschaft für Fachdidaktik*. Münster. URL: https://www.fachdidaktik.org/wordpress/wp-content/uploads/2018/07/GFD-Positionspapier-Fachliche-Bildung-in-der-digitalen-Welt-2018-FINAL-HP-Version.pdf.

GI - Gesellschaft für Informatik e.V. (2008). „Grundsätze und Standards für die Informatik in der Schule. Bildungsstandards Informatik für die Sekundarstufe I“. In: *LOG IN* 28.150/151.

GI - Gesellschaft für Informatik e.V. (2016a). „Bildungsstandards Informatik für die Sekundarstufe II“. In: *LOG IN* 36.183/184.

GI - Gesellschaft für Informatik e.V. (2016b). *Dagstuhl-Erklärung. Bildung in der digitalen vernetzten Welt*. Berlin. URL: https://dagstuhl.gi.de/fileadmin/GI/Hauptseite/Aktuelles/Projekte/Dagstuhl/Dagstuhl-Erklaerung_2016-03-23.pdf.

GI - Gesellschaft für Informatik e.V. (2016c). *Empfehlungen für Bachelor- und Masterprogramme im Studienfach Informatik an Hochschulen*. Bonn: Gesellschaft für Informatik e.V.

GI - Gesellschaft für Informatik e.V. (2019). „Kompetenzen für informatische Bildung im Primarbereich. Empfehlungen der Gesellschaft für Informatik e.V“. In: *LOG IN* 39.191/192.

Glötzl, Herbert (2000). *Prinzipien effektiven Unterrichts. Handbuch für die Erziehungs- und Unterrichtspraxis*. Schulpädagogik. Stuttgart: Klett.

Goecke, Lennart und Jurik Stiller (2018). „Informatische Phänomene und Sachunterricht. Beispiele für vielperspektivischen Umgang mit einem Einplatinencomputer.“ In: *Informatik und Medien: 8. Münsteraner Workshop zur Schulinformatik*. Hrsg. von Marco Thomas und Michael Weigend. Münster: Books on Demand.

Gökçe, Semirhan und Arzu Aydoğan Yenmez (2022). „Ingenuity of Scratch Programming on Reflective Thinking Towards Problem Solving and Computational Thinking“. In: *Education and Information Technologies*. PII: 11385. DOI: 10.1007/s10639-022-11385-x.

Goll, Joachim (2011). *Methoden und Architekturen der Softwaretechnik*. Wiesbaden: Vieweg+Teubner Verlag. DOI: 10.1007/978-3-8348-8164-9.

Goode, Joanna, Jane Margolis und Gail Chapman (2014). „Curriculum is Not Enoughh: The Educational Theory and Research Foundation of the Exploring Computer Science Professional Development Model“. In: *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE)*. Hrsg. von J. D. Dougherty. New York, NY: ACM, S. 493–498. DOI: 10.1145/2538862.2538948.

Google Inc. und Gallup Inc. (2017). *Encouraging Students Toward Computer Science Learning. Results From the 2015-2016 Google-Gallup Study of Computer Science in U.S. K-12 Schools*. URL: https://goo.gl/iM5g3A.

Grafe, Silke (2008). „Förderung von Problemlösefähigkeit beim Lernen mit Computersimulationen". Dissertation. Paderborn: Universität Paderborn.

Gray, Simon u. a. (2007). „Suggestions for Graduated Exposure to Programming Concepts using Fading Worked Examples". In: *Proceedings of the third International Workshop on Computing Education Research (ICER)*. New York, NY: ACM, S. 99–110.

Greaves, Anna (2017). „What Training are Primary Teachers Receiving to Implement the New Computing Curriculum and are they Prepared to Succeed?" In: *Computing Conference*. Computing Conference (London). Piscataway, NJ: IEEE, S. 1426–1430.

Greifenstein, Luisa, Isabella Graßl und Gordon Fraser (2021). „Challenging but Full of Opportunities: Teachers' Perspectives on Programming in Primary Schools". In: *Proceedings of the 21st International Conference on Computing Education Research (Koli Calling)*. Hrsg. von Otto Seppälä und Andrew Petersen. New York, NY: ACM, S. 1–10. DOI: 10.1145/3488042.3488048.

Grover, Shuchi und Roy Pea (2018). „Computational Thinking: A Competency Whose Time Has Come". In: *Computer Science Education. Perspectives on Teaching and Learning in School*. Hrsg. von Sue Sentance, Erik Barendsen und Carsten Schulte. London u. a.: Bloomsbury Academic, S. 19–38.

Grover, Shuchi und Aman Yadav (2020). „Integrating Programming Into Other Subjects". In: *Computer Science in K-12: An A-to-Z Handbook on Teaching Programming*. Hrsg. von Shuchi Grover. Palo Alto, CA: Edfinity, S. 83–98.

Günther, Christine und Christin Nenner (2021). „Das I in MINT von Anfang an". In: *INFORMATIK 2021*. Bonn: Gesellschaft für Informatik. DOI: 10.18420/INFORMATIK2021-140.

Güting, Ralf Hartmut und Stefan Dieker (2018). *Datenstrukturen und Algorithmen*. 4., erweiterte und überarbeitete Auflage. Wiesbaden: Springer Vieweg. DOI: 10.1007/978-3-658-04676-7.

Guzdial, Mark (2010). *What CS Teachers know: PCK für CS Ed*. URL: https://computinged.wordpress.com/2010/02/05/what-cs-teachers-know-pck-for-cs-ed/.

Hansen, Alexandria K. u. a. (2017). „Assessing Children's Understanding of the Work of Computer Scientists". In: *Proceedings of the 2017 ACM Technical Symposium on Computer Science Education (SIGCSE)*. Hrsg. von Michael E. Caspersen und A. Special Interest Group on Computer ScienceC.M. Education. New York, NY: ACM, S. 279–284. DOI: 10.1145/3017680.3017769.

Hardy, Ilonca (2012). „Kognitive Strukturierung – Empirische Zugänge zu einem heterogenen Konstrukt der Unterrichtsforschung". In: *Bedingungen des Lehrens und Lernens in der Grundschule*. Hrsg. von Frank Hellmich, Sabrina Förster und Fabian Hoya. Wiesbaden: VS Verlag für Sozialwissenschaften, S. 51–62. DOI: 10.1007/978-3-531-19137-9_4.

Hardy, Ilonca u. a. (2011). „Adaptive Lerngelegenheiten in der Grundschule. Merkmale, methodisch-didaktische Schwerpunktsetzungen und erforderliche Lehrerkompetenzen". In: *Zeitschrift für Pädagogik* 57.6, S. 819–833.

Hartig, Johannes und Eckhard Klieme (2006). „Kompetenz und Kompetenzdiagnostik“. In: *Leistung und Leistungsdiagnostik*. Hrsg. von Karl Schweizer. Heidelberg: Springer. DOI: 10.1007/3-540-33020-8_9.

Hartinger, Andreas (2005). *Interessen von Mädchen und Jungen aufgreifen und entwickeln*. Kiel: IPN.

Hartinger, Andreas und Maria Fölling-Albers (2002). *Schüler motivieren und interessieren. Ergebnisse aus der Forschung, Anregungen für die Praxis*. Bad Heilbrunn: Klinkhardt.

Hartinger, Andreas, Maria Fölling-Albers und Dzenana Mörtl-Hafizovic (2005). „Die Bedeutung der Ambiguitätstoleranz für das Lernen in situierten Lernbedingungen“. In: *Psychologie in Erziehung und Unterricht* 52, S. 113–126.

Hartinger, Andreas und Katrin Lohrmann (2014). „Entdeckendes Lernen“. In: *Handbuch Grundschulpädagogik und Grundschuldidaktik*. Hrsg. von Wolfgang Einsiedler. 4. ergänzte und aktualisierte Auflage. Bad Heilbrunn: Verlag Julius Klinkhardt, S. 385–389.

Haselmeier, Kathrin (2019). „Informatik in der Grundschule – Stellschraube Lehrerbildung“. In: *Informatik für alle. 18. GI-Fachtagung Informatik und Schule (INFOS)*. Hrsg. von Arno Pasternak. Bonn: Gesellschaft für Informatik, S. 89–98.

Haselmeier, Kathrin u. a. (2016). „Informatikunterricht im Primarbereich – ohne qualifizierte Lehrkräfte geht es nicht“. In: *Informatik für Kinder. 7. Münsteraner Workshop zur Schulinformatik*. Hrsg. von Marco Thomas und Michael Weigend. Norderstedt: Books on Demand.

Hassinen, Marko und Hannu Mäyrä (2006). „Learning Programming by Programming: a Case Study“. In: *Proceedings KolliCalling*. Hrsg. von Anders Berglund und Mattias Wigbberg, S. 117–119.

Hattie, John (2003). *Teachers Make a Difference: What is the Research Evidence?* Paper presented at the Australian Council for Educational Research Annual Conference on Building Teacher Quality. URL: https://www.educationalleaders.govt.nz/Pedagogy-and-assessment/Building-effective-learning-environments/Teachers-make-a-difference-What-is-the-research-evidence.

Hattie, John (2009). *Visible learning. A synthesis of over 800 meta-analyses relating to achievement*. London: Routledge.

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

HdkF - Stiftung „Haus der kleinen Forscher“, Hrsg. (2018). *Frühe informatische Bildung – Ziele und Gelingensbedingungen für den Elementar- und Primarbereich*. Opladen, Berlin, Toronto: Verlag Barbara Budrich. DOI: 10.3224/84742107.

Heimann, Paul, Gunter Otto und Wolfgang Schulz (1965). *Unterricht. Analyse und Planung*. Hannover: Schroedel.

Heintz, Fredrik, Linda Mannila und Tommy Farnqvist (2016). „A Review of Models for Introducing Computational Thinking, Computer Science and Computing in K–12 Education“. In: *Proceedings of the 2016 IEEE Frontiers in Education Conference (FIE)*. Piscataway, NJ: IEEE, S. 1–9. DOI: 10.1109/FIE.2016.7757410.

Heintz, Fredrik u. a. (2017). „Introducing Programming and Digital Competence in Swedish K-9 Education“. In: *Informatics in Schools. Focus on Learning Programming (ISSEP)*. Hrsg. von Valentina Dagienė und Arto Hellas. 10696. Cham: Springer International Publishing, S. 117–128.

Heinzel, Friederike (2014). „Kindheit und Grundschule“. In: *Handbuch Grundschulpädagogik und Grundschuldidaktik*. Hrsg. von Wolfgang Einsiedler. 4. ergänzte und aktualisierte Auflage. Bad Heilbrunn: Verlag Julius Klinkhardt, S. 155–163.

Heinzel, Friederike und Annedore Prengel (2014). „Mädchen und Jungen in der Grundschule“. In: *Handbuch Grundschulpädagogik und Grundschuldidaktik*. Hrsg. von Wolfgang Einsiedler. 4. ergänzte und aktualisierte Auflage. Bad Heilbrunn: Verlag Julius Klinkhardt, S. 200–204.

Heller, Joan I. u. a. (2012). „Differential Effects of Three Professional Development Models on Teacher Knowledge and Student Achievement in Elementary Science“. In: *Journal of Research in Science Teaching* 49.3, S. 333–362. DOI: 10.1002/tea.21004.

Hellmich, Frank und Frederike Günther (2011). „Entwicklung von Selbstkonzepten bei Kindern im Grundschulalter - ein Überblick“. In: *Selbstkonzepte im Grundschulalter. Modelle, empirische Ergebnisse, pädagogische Konsequenzen*. Hrsg. von Frank Hellmich. Stuttgart: W. Kohlhammer Verlag, S. 19–46.

Hellmich, Frank und Hanna Kiper (2006). *Einführung in die Grundschuldidaktik*. Weinheim: Beltz.

Helmke, Andreas (2017). *Unterrichtsqualität und Lehrerprofessionalität. Diagnose, Evaluation und Verbesserung des Unterrichts*. 7. Auflage. Seelze-Velber: Klett/Kallmeyer.

Henze, Godehard, Uwe Sandfuchs und Clemens Zumhasch (2014). „Hochbegabte in der Grundschule“. In: *Handbuch Grundschulpädagogik und Grundschuldidaktik*. Hrsg. von Wolfgang Einsiedler. 4. ergänzte und aktualisierte Auflage. Bad Heilbrunn: Verlag Julius Klinkhardt, S. 217–224.

Hermans, Felienne und Efthimia Aivaloglou (2016). „Do Code Smells Hamper Novice Programming? A Controlled Experiment on Scratch Programs“. In: *Proceeding of 24th International Conference on Program Comprehension (ICPC)* (Austin, TX, USA). Piscataway, NJ: IEEE, S. 1–10. DOI: 10.1109/ICPC.2016.7503706.

Hermans, Felienne und Efthimia Aivaloglou (2017). „To Scratch or not to Scratch?“ In: *Proceedings of the 12th Workshop on Primary and Secondary Computing Education (WiPSCE)*. Hrsg. von Erik Barendsen und Peter Hubwieser. New York, NY: ACM, S. 49–56. DOI: 10.1145/3137065.3137072.

Hersen, Michel und David H. Barlow (1976). *Single-Case Experimental Designs. Strategies for Studying Behavior Change*. Oxford: Pergamon Press.

Hiller, Jan (2017). *Die Unternehmensfallstudie als Unterrichtsmethode für den Geographieunterricht*. Bd. 67. Geographiedidaktische Forschungen. Münster: Münsterscher Verlag für Wissenschaft.

HITSA - Information Technology Foundation for Education (2015). *ProgeTiger Programme 2015–2017*. URL: https://media.voog.com/0000/0034/3577/files/Programm%20ProgeTiiger%202015_2017eng.pdf.

Ho, Angela, David Watkins und Mavis Kelly (2001). „The Conceptual Change Approach to Improving Teaching and Learning: An Evaluation of a Hong Kong Staff Development Programme“. In: *Higher Education* 42.2, S. 143–169. DOI: 10.1023/A:1017546216800.

Hodges, Steve u. a. (2020). „Physical Computing: A Key Element of Modern Computer Science Education“. In: *Computer* 53.4, S. 20–30. DOI: 10.1109/MC.2019.2935058.

Hoffmann, Sabrina, Katharina Wendlandt und Matthias Wendlandt (2017). „Algorithmisieren im Grundschulalter“. In: *Informatische Bildung zum Verstehen und Gestalten der digitalen Welt (INFOS)*. Gesellschaft für Informatik, Bonn, S. 73–82.

Honer, Anne (2011). „Das explorative Interview. Zur Rekonstruktion der Relevanzen von Expertinnen und anderen Leuten“. In: *Kleine Leiblichkeiten. Erkundungen in Lebenswelten*. Hrsg. von Anne Honer und Ronald Hitzler. Wissen, Kommunikation und Gesellschaft. Wiesbaden: VS Verlag für Sozialwissenschaften / Springer Fachmedien Wiesbaden GmbH Wiesbaden, S. 41–58.

Hoppe, Heinz Ulrich und Wolfram J. Luther (1996). „Informatik und Schule – Ein Fach im Spiegel neuer Entwicklungen der Fachdidaktik“. In: *LOG IN* 16.1, S. 8–14.

Hoppe-Graff, Siegfried (2014). „Denkentwicklung aus dem Blickwinkel des strukturgenetischen Konstruktivismus“. In: *Theorien in der Entwicklungspsychologie*. Hrsg. von Lieselotte Ahnert. Berlin: Springer VS.

Hörmann, Isabel (2018). „Fehlvorstellungen zum Thema Ozon vermeiden - eine Design-based Research-Studie unter besonderer Berücksichtigung der methodischen Großform des Experiments und motivationaler Perspektiven“. Dissertation. Augsburg: Universität Augsburg.

Hsu, Hui-mei Justina (2014). „Gender Differences in Scratch Game Design“. In: *Proceedings of the 2014 International Conference on Information, Business and Education Technology (ICIBET)*. Paris. DOI: 10.2991/icibet-14.2014.28.

Hubbard, Aleata (2018). „Pedagogical Content Knowledge in Computing Education: A Review of the Research Literature“. In: *Computer Science Education* 28.2, S. 117–135. DOI: 10.1080/08993408.2018.1509580.

Hubwieser, Peter (2007). *Didaktik der Informatik*. Berlin, Heidelberg: Springer. DOI: 10.1007/978-3-540-72478-0.

Hubwieser, Peter, Elena Hubwieser und Dorothee Graswald (2016). „How to Attract the Girls: Gender-Specific Performance and Motivation in the Bebras Challenge“. In: *Informatics in Schools: Improvement of Informatics Knowledge and Perception (ISSEP)*. Hrsg. von Andrej Brodnik und Françoise Tort. Bd. 9973. Cham: Springer, S. 40–52. DOI: 10.1007/978-3-319-46747-4_4.

Hubwieser, Peter, Andreas Mühling und Gerd Aiglstorfer (2013). *Fundamente der Informatik. Funktionale, imperative und objektorientierte Sicht, Algorithmen und Datenstrukturen*. 2., vollständig überarbeitete Auflage. Berlin/Boston: De Gruyter.

Hubwieser, Peter u. a. (2013a). „Pedagogical Content Knowledge for Computer Science in German Teacher Education Curricula“. In: *Proceedings of the 8th Workshop in Primary and Secondary Computing Education (WiPSCE)*. Hrsg. von Michael E. Caspersen, Maria Knobelsdorf und Ralf Romeike. New York, NY: ACM, S. 95–103. DOI: 10.1145/2532748.2532753.

Hubwieser, Peter u. a. (2013b). „Towards a Conceptualization of Pedagogical Content Knowledge for Computer Science“. In: *Proceedings of the 2013 ACM Conference on International Computing*

*Education Research (ICER)*. Hrsg. von Beth Simon, Alison Clear und Quintin Cutts. New York, NY: ACM, S. 1–8. DOI: 10.1145/2493394.2493395.

Humbert, Ludger u. a. (2020). „Informatik – Kompetenzentwicklung bei Kindern". In: *Informatik Spektrum* 43.2, S. 85–93. DOI: 10.1007/s00287-020-01247-6.

Inckemann, Elke (2014). „Binnendifferenzierung - Individualisierung - adaptiver Unterricht". In: *Handbuch Grundschulpädagogik und Grundschuldidaktik*. Hrsg. von Wolfgang Einsiedler. 4. ergänzte und aktualisierte Auflage. Bad Heilbrunn: Verlag Julius Klinkhardt, S. 374–384.

ISB - Staatsinstitut für Schulqualität und Bildungsforschung München (2017). *Kompetenzrahmen zur Medienbildung an bayerischen Schulen*. URL: https : / / mebis . bycs . de / beitrag / kompetenzrahmen-zur-medienbildung.

Iskrenovic-Momcilovic, Olivera (2019). „Pair Programming With Scratch". In: *Education and Information Technologies* 24.5. PII: 9905, S. 2943–2952. DOI: 10.1007/s10639-019-09905-3.

Jäger, Reinhold S. und Rainer Bodensohn (2007). *Bericht zur Befragung von Mathematiklehrkräften: Die Situation der Lehrerfortbildung im Fach Mathematik aus der Sicht der Lehrkräfte*. Hrsg. von Deutsche Telekom Stiftung. URL: https : / / dzlm . de / files / uploads / 17 _ 01 _ 07 _ mathematiklehrerbefragung.pdf.

Jäger, Reinhold S. und Urban Lissmann (2000). *Von der Beobachtung zur Notengebung. Ein Lehrbuch. Diagnostik und Benotung in der Aus-, Fort- und Weiterbildung*. 3. Auflage. Landau: Verlag Empirische Pädagogik.

Jaglo, Maggie (2013). „Hardwarefreaks und Kellerkinder – Klischeevorstellungen über Informatik und die Auseinandersetzung der Studierenden damit". In: *Informatik Spektrum* 36.3, S. 274–277. DOI: 10.1007/s00287-013-0692-1.

Jahn, Dirk (2014). „Durch das praktische Gestalten von didaktischen Designs nützliche Erkenntnisse gewinnen: Eine Einführung in die Gestaltungsforschung". In: *W&E* 66.1, S. 3–15.

Jaipal-Jamani, Kamini und Charoula Angeli (2017). „Effect of Robotics on Elementary Preservice Teachers' Self-Efficacy, Science Learning, and Computational Thinking". In: *Journal of Science Education and Technology* 26.2. PII: 9663, S. 175–192. DOI: 10.1007/s10956-016-9663-z.

Jank, Werner und Hilbert Meyer (2021). *Didaktische Modelle*. 14. Auflage. Berlin: Cornelsen.

Jerusalem, Matthias u. a. (2009). *Förderung von Selbstwirksamkeit und Selbstbestimmung im Unterricht. Skalen zur Erfassung von Lehrer- und Schülermerkmalen*. Berlin: Humboldt-Universität zu Berlin.

Jürgens, Eiko und Werner Sacher (2008). *Leistungserziehung und pädagogische Diagnostik in der Schule. Grundlagen und Anregungen für die Praxis*. Stuttgart: Kohlhammer.

*K–12 Computer Science Framework* (2016). URL: http://www.k12cs.org..

Kabátová, Martina, Ivan Kalas und Monika Tomcsányiová (2016). „Programming in Slovak Primary Schools". In: *Olympiads in Informatics* 10.1, S. 125–159. DOI: 10.15388/ioi.2016.09.

Kahlert, Joachim (2007). „Was kommt nach der Erkenntnis? Zum schwierigen Verhältnis pädagogischer Disziplinen zu der Erwartung, sich nützlich zu machen". In: *Der Nutzen wird vertagt. Bildungswissenschaften im Spannungsfeld zwischen wissenschaftlicher Profilbildung und prak-*

*tischem Mehrwert*. Hrsg. von Gabi Reinmann und Joachim Kahlert. Lengerich: Pabst Science Publishers, S. 20–45.

Kanemune, Susumu, Shizuka Shirai und Seiichi Tani (2017). „Informatics and Programming Education at Primary and Secondary Schools in Japan“. In: *Olympiads in Informatics* 11, S. 143–150. DOI: 10.15388/ioi.2017.11.

Kastl, Petra und Ralf Romeike (2015). „Entwicklung eines agilen Frameworks für Projektunterricht mit Design-Based Research“. In: *Informatik allgemeinbildend begreifen*. Hrsg. von Jens Gallenbacher. Bonn: Gesellschaft für Informatik e.V, S. 201–210.

Kattmann, Ulrich u. a. (1997). „Das Modell der Didaktischen Rekonstruktion. Ein Rahmen für naturwissenschaftsdidaktische Forschung und Entwicklung“. In: *Zeitschrift für Didaktik der Naturwissenschaften* 3.3, S. 3–18.

Keller, John M. (1987). „Development and Use of the ARCS Model of Instructional Design“. In: *Journal of Instructional Development* 10.3, S. 2–10. DOI: 10.1007/BF02905780.

Kellow, Jan-Marie (2018). „Digital Technologies in the New Zealand Curriculum“. In: *Waikato Journal of Education* 23.2. DOI: 10.15663/wje.v23i2.656.

Kelly, Anthony (2004). „Design Research in Education: Yes, but is it Methodological?“ In: *Journal of the Learning Sciences* 13.1, S. 115–128. DOI: 10.1207/s15327809jls1301_6.

Kerres, Michael (2018). *Mediendidaktik*. Oldenburg: De Gruyter. DOI: 10.1515/9783110456837.

Kind, Amanda (2015). „Computing Attitudes: Will Teaching 2nd Grade Students Computer Science Improve their Self-Efficacy and Attitude and Eliminate Gender Gaps?“ In: *Rising Tide* 8, S. 1–34.

Klafki, Wolfgang (2007). *Neue Studien zur Bildungstheorie und Didaktik. Zeitgemäße Allgemeinbildung und kritisch-konstruktive Didaktik*. Weinheim: Beltz Verlagsgruppe.

Klauer, Karl Josef und Detlev Leutner (2007). *Lehren und Lernen. Einführung in die Instruktionspsychologie*. Weinheim: Beltz.

Klieme, Eckhard u. a. (2003). *Zur Entwicklung nationaler Bildungsstandards. Eine Expertise*. Berlin: Bundesministerium für Bildung und Forschung.

Kluczniok, Katharina, Christiane Große und Hans-Günther Roßbach (2014). „Heterogene Lerngruppen in der Grundschule“. In: *Handbuch Grundschulpädagogik und Grundschuldidaktik*. Hrsg. von Wolfgang Einsiedler. 4. ergänzte und aktualisierte Auflage. Bad Heilbrunn: Verlag Julius Klinkhardt, S. 194–200.

KMK - Kultusministerkonferenz (2015). *Empfehlungen zur Arbeit in der Grundschule. Beschluss der Kultusministerkonferenz vom 02.07.1970 i. d. F. vom 11.06.2015*. Hrsg. von Sekretariat der Kultusministerkonferenz. URL: https://www.kmk.org/fileadmin/Dateien/veroeffentlichungen_beschluesse/1970/1970_07_02_Empfehlungen_Grundschule.pdf.

KMK - Kultusministerkonferenz (2016). *Strategie der Kultusministerkonferenz zur Bildung in der digitalen Welt. Beschluss der Kultusministerkonferenz vom 08.12.2016*. Hrsg. von Sekretariat der Kultusministerkonferenz. URL: https://www.kmk.org/themen/bildung-in-der-digitalen-welt/strategie-bildung-in-der-digitalen-welt.html.

KMK - Kultusministerkonferenz (2019). *Ländergemeinsame inhaltliche Anforderungen für die Fachwissenschaften und Fachdidaktiken in der Lehrerbildung. Beschluss der Kultusministerkonferenz vom 16.10.2008 i. d. F. vom 16.05.2019*. Hrsg. von Sekretariat der Kultusministerkonferenz. URL: https://www.kmk.org/fileadmin/Dateien/veroeffentlichungen_beschluesse/2008/2008_10_16-Fachprofile-Lehrerbildung.pdf.

Knauf, Tassilo (2009). *Einführung in die Grundschuldidaktik. Lernen, Entwicklungsförderung und Erfahrungswelten in der Primarstufe*. 2. überarbeitete Auflage. Stuttgart: Verlag W. Kohlhammer.

Knight, Jim (2010). *Unmistakable Impact. A Partnership Approach for Dramatically Improving Instruction*. Thousand Oaks: SAGE Publications. DOI: 10.4135/9781452219721.

Knobelsdorf, Maria, Jonathan Otto und Sandra Sprenger (2017). „A Computing Education Approach for Geography Students in Context of GIS". In: *2017 IEEE Global Engineering Education Conference (EDUCON)* (Athens, Greece, 2017). IEEE, S. 1790–1796. DOI: 10.1109/EDUCON.2017.7943092.

Kong, Siu-Cheung und Andrew Chan-Chio Lao (2019). „Assessing In-service Teachers' Development of Computational Thinking Practices in Teacher Development Courses". In: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE)*. Hrsg. von Elizabeth Hawthorne u. a. New York, NY: ACM, S. 976–982. DOI: 10.1145/3287324.3287470.

Krapp, Andreas und Tina Hascher (2014a). „Die Erforschung menschlicher Motivation". In: *Theorien in der Entwicklungspsychologie*. Hrsg. von Lieselotte Ahnert. Berlin: Springer VS, S. 234–251.

Krapp, Andreas und Tina Hascher (2014b). „Theorien der Lern- und Leistungsmotivation". In: *Theorien in der Entwicklungspsychologie*. Hrsg. von Lieselotte Ahnert. Berlin: Springer VS, S. 252–281.

Kubitsykey, Mary Elizabeth (2006). „Extended Professional Development for Systemic Curriculum Reform". Dissertation. Ann Arbor: University of Michigan.

Kuckartz, Udo (2016). *Qualitative Inhaltsanalyse. Methoden, Praxis, Computerunterstützung*. 3., überarbeitete Auflage. Grundlagentexte Methoden. Weinheim und Basel: Beltz Juventa.

Kuhl, Maria (2008). *Studienkultur Informatik neu denken. Geschlechterkonstruktionen im Informatikstudium an der Universität Dortmund und der Carnegie Mellon University*. Soziologische Studien. Aachen: Shaker.

Kunnari, Irma, Liisa Ilomäki und Auli Toom Toom (2018). „Successful Teacher Teams in Change: The Role of Collective Efficacy and Resilience". In: *International Journal of Teaching and Learning in Higher Education* 30.1, S. 111–126.

Kunter, Mareike (2011). „Motivation als Teil der professionellen Kompetenz - Forschungsbefunde zum Enthusiasmus von Lehrkräften". In: *Professionelle Kompetenz von Lehrkräften. Ergebnisse des Forschungsprogramms COACTIV*. Hrsg. von Mareike Kunter u. a. Münster: Waxmann, S. 259–275.

Kunter, Mareike und Britta Pohlmann (2009). „Lehrer". In: *Pädagogische Psychologie*. Hrsg. von Elke Wild und Jens Möller. Berlin/Heidelberg: Springer, S. 261–282. DOI: 10.1007/978-3-540-88573-3_11.

Kunter, Mareike und Ulrich Trautwein (2013). *Psychologie des Unterrichts*. Paderborn: Schöningh.

Kunter, Mareike u. a., Hrsg. (2011). *Professionelle Kompetenz von Lehrkräften. Ergebnisse des Forschungsprogramms COACTIV*. Münster: Waxmann.

Kwon, Sei und Katri Schroderus (2017). *Coding in Schools: Comparing Integration of Programming into Basic Education Curricula of Finland and South Korea*. Helsinki: Finnish Society on Media Education.

Lamprou, Anna und Alexander Repenning (2018). „Teaching How to Teach Computational Thinking". In: *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE)*. Hrsg. von Irene Polycarpou und A. Special Interest Group on Computer ScienceC.M. Education. New York, NY: ACM, S. 69–74. DOI: 10.1145/3197091.3197120.

Landeshauptstadt München, Hrsg. (2014). *Die Bildungsregion Hasenbergl in Zahlen*. URL: https://www.pi-muenchen.de/profil/wir-ueber-uns/stabsstelle-kommunales-bildungsmanagement/kommunales-bildungsmonitoring/bildungsregion-hasenbergl-in-zahlen/.

Landis, J. Richard und Gary G. Koch (1977). „The Measurement of Observer Agreement for Categorical Data". In: *Biometrics* 33.1, S. 159. DOI: 10.2307/2529310.

Lee, Irene u. a. (2011). „Computational Thinking for Youth in Practice". In: *ACM Inroads* 2.1. DOI: 10.1145/1929887.1929902.

Leifheit, Luzia u. a. (2020). „SCAPA: Development of a Questionnaire Assessing Self-Concept and Attitudes Toward Programming". In: *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE)*. Hrsg. von Michail Giannakos und Guttorme Sindre. New York, NY: ACM, S. 138–144.

Leutner, Detlev (2001). „Instruktionspsychologie". In: *Handwörterbuch Pädagogische Psychologie*. Hrsg. von Detlef H. Rost. 2., überarb. und erw. Aufl. Schlüsselbegriffe. Weinheim, Basel und Berlin: Beltz, S. 267–276.

Linck, Barbara u. a. (2013). „Competence Model for Informatics Modelling and System Comprehension". In: *Proceedings of the IEEE Global Engineering Education Conference (EDUCON)*. Piscataway, NJ: IEEE, S. 85–93. DOI: 10.1109/EduCon.2013.6530090.

Lindner, Gertrud und Sandra Mayerhofer (2018). *Kompetenzorientierter guter Unterricht und bedarfsorientierte Lehrerfortbildung*. Münster und New York: Waxmann.

Linn, Marcia C. (1990). „Summary: Establishing a Science and Engineering of Science Education". In: *Toward a Scientific Practice of Science Education*. Hrsg. von Marjorie Gardner u. a. Hoboken: Taylor and Francis, S. 323–241.

Lipowsky, Frank (2006). „Auf den Lehrer kommt es an. Empirische Evidenzen für Zusammenhänge zwischen Lehrerkompetenzen, Lehrerhandeln und dem Lernen der Schüler". In: *Zeitschrift für Pädagogik* (51): *Kompetenz und Kompetenzentwicklung von Lehrerinnen und Lehrern*. Hrsg. von C. Allemann-Ghionda und E. Terhart, S. 47–70.

Lipowsky, Frank (2010). „Lernen im Beruf. Empirische Befunde zur Wirksamkeit von Lehrerfortbildung“. In: *Lehrerinnen und Lehrer lernen. Konzepte und Befunde der Lehrerfortbildung*. Hrsg. von F. H. Müller u. a. Münster: Waxmann, S. 51–70.

Lipowsky, Frank und Daniela Rzejak (2012). „Lehrerinnen und Lehrer als Lerner - Wann gelingt der Rollentausch? Merkmale und Wirkungen wirksamer Lehrerfortbildungen“. In: *Schulpädagogik heute* 5.3, S. 1–17.

Lockwood, James und Aidan Mooney (2017). *Computational Thinking in Education: Where does it fit? A Systematic Literary Review*. Maynooth: National University of Ireland Maynooth. URL: https://arxiv.org/pdf/1703.07659.

Lohaus, Arnold und Marc Vierhaus (2019). *Entwicklungspsychologie des Kindes- und Jugendalters für Bachelor*. 4. vollständig überarbeitete Auflage. Berlin: Springer. 359 S. DOI: 10.1007/978-3-662-59192-5.

Lohrmann, Katrin und Andreas Hartinger (2014). „Lernemotionen, Lernmotivation und Interesse“. In: *Handbuch Grundschulpädagogik und Grundschuldidaktik*. Hrsg. von Wolfgang Einsiedler. 4. ergänzte und aktualisierte Auflage. Bad Heilbrunn: Verlag Julius Klinkhardt, S. 275–279.

Lorenz, Ramona u. a., Hrsg. (2017). *Schule digital – der Länderindikator 2017. Schulische Medienbildung in der Sekundarstufe I mit besonderem Fokus auf MINT-Fächer im Bundesländervergleich und Trends von 2015 bis 2017*. Münster und New York: Waxmann.

Losch, Daniel und Ludger Humbert (2019). „Informatische Bildung für alle Lehramtsstudierenden“. de. In: *Informatik für alle. 18. GI-Fachtagung Informatik und Schule (INFOS)*. Hrsg. von Arno Pasternak. Bonn: Gesellschaft für Informatik, S. 119–128. DOI: 10.18420/INFOS2019-B8.

Lotz, Miriam und Frank Lipowsky (2015). „Die Hattie-Studie und ihre Bedeutung für den Unterricht – Ein Blick auf ausgewählte Aspekte der Lehrer-Schüler-Interaktion“. In: *Begabungen entwickeln & Kreativität fördern*. Hrsg. von Gerlinde Mehlhorn, Karola Schöppe und Frank Schulz. München: kopaed, S. 97–136.

Lowe, Tony A. (2018). „Misconceptions and the Notional Machine in Very Young Programming Learners“. In: *School of Engineering Education Graduate Student Series* Paper 76.

Luxton-Reilly, Andrew u. a. (2018). „Introductory Programming. A Systematic Literature Review“. In: *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE)*. Hrsg. von Guido Rößling. New York, NY: ACM, S. 55–106. DOI: 10.1145/3293881.3295779.

Magenheim, Johannes u. a. (2018a). „Das Projekt Informatik an Grundschulen“. In: *LOG IN Informatische Bildung und Computer in der Schule* 189/190, S. 57–66.

Magenheim, Johannes u. a. (2018b). „Evaluation of Learning Informatics in Primary Education. Views of Teachers and Students“. In: *Informatics in schools: fundamentals of computer science and software engineering (ISSEP)*. Hrsg. von Sergej N. Pozdnjakov und Valentina Dagienė. Cham: Springer, S. 339–353.

Makris, Dimosthenis u. a. (2013). „Could you Help me to Change the Variables? Comparing Instruction to Encouragement for Teaching Programming“. In: *Proceedings of the 8th Workshop*

*in Primary and Secondary Computing Education (WiPSCE)*. Hrsg. von Michael E. Caspersen, Maria Knobelsdorf und Ralf Romeike. New York, NY: ACM.

Malisu, Petri und Lucie Bryndova (2020). „The Usage of Visual Programming Languages among Primary School Teachers in the Czech Republic". In: *Proceedings of the 7th SWS International Scientific Conference on Social Sciences (ISCSS)*. SGEM Scientific eLibrary, S. 477–482.

Maloney, John u. a. (2004). „Scratch: A Sneak Preview". In: *Proceedings of the Second International Conference on Creating, Connecting and Collaborating through Computing (C5)*. Hrsg. von Yahiko Kambayashi. Los Alamitos, CA: IEEE Computer Society, S. 104–109. DOI: 10.1109/C5.2004.1314376.

Maloney, John u. a. (2010). „The Scratch Programming Language and Environment". In: *ACM Transactions on Computing Education* 10.4, S. 1–15. DOI: 10.1145/1868358.1868363.

Maras, Rainer und Josef Ametsbichler (2014). *Unterrichtsgestaltung in der Grundschule – ein Handbuch. Pädagogische und didaktische Grundlagen, methodische und praktische Anregungen, Strukturmodelle*. 3. Auflage. Donauwörth: Auer-Verlag.

Margolis, Jane und Allan Fisher (2002). *Unlocking the Clubhouse: Women in Computing*. Cambridge, Mass.: MIT Press.

Marsh, Herbert W. und Rhonda G. Craven (2006). „Reciprocal Effects of Self-Concept and Performance From a Multidimensional Perspective. Beyond Seductive Pleasure and Unidimensional Perspectives". In: *Perspectives on Psychological Science* 1.2, S. 133–163. DOI: 10.1111/j.1745-6916.2006.00010.x.

Marsh, Herbert W. und Andrew J. Martin (2011). „Academic Self-Concept and Academic Achievement. Relations and Causal Ordering". In: *The British journal of educational psychology* 81, S. 59–77. DOI: 10.1348/000709910X503501.

Martin, Christopher, Janet Hughes und John Richards (2017). „Learning Dimensions: Lessons from Field Studies". In: *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE)*. New York, NY: ACM, S. 299–304. DOI: 10.1145/3059009.3059046.

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.

Mayring, Philipp (2010). „Qualitative Inhaltsanalyse". In: *Handbuch Qualitative Forschung in der Psychologie*. Hrsg. von Günter Mey und Katja Mruck. Wiesbaden: VS Verlag, S. 601–613.

McIntyre, Donald (2005). „Bridging the Gap Between Research and Practice". In: *Cambridge Journal of Education* 35.3, S. 357–382. DOI: 10.1080/03057640500319065.

McKenney, Susan, Nienke Nieveen und Jan van den Akker (2006). „Design Research from a Curriculum Perspective". In: *Educational Design Research*. Hrsg. von Jan van den Akker u. a. New York: Routledge, S. 67–90.

McKenney, Susan und Thomas C. Reeves (2019). *Conducting Educational Design Research*. 2nd ed. Milton: Routledge.

Medienberatung NRW (2020). *Medienkompetenzrahmen NRW*. URL: https://medienkompetenzrahmen.nrw/fileadmin/pdf/LVR_ZMB_MKR_Broschuere.pdf.

Meerbaum-Salant, Orni, Michal Armoni und Mordechai Ben-Ari (2010). „Learning Computer Science Concepts with Scratch“. In: *Proceedings of the Sixth international workshop on Computing education research (ICER)*. Hrsg. von Michael E. Caspersen. New York, NY: ACM.

Merschmeyer-Brüwer und Wilhelm Schipper (2014). „Größen und Messen“. In: *Handbuch Grundschulpädagogik und Grundschuldidaktik*. Hrsg. von Wolfgang Einsiedler. 4. ergänzte und aktualisierte Auflage. Bd. 497-500. Bad Heilbrunn: Verlag Julius Klinkhardt.

Meyer, Hilbert (2008). *Unterrichtsmethoden*. 12. Auflage. Berlin und Frankfurt am Main: Cornelsen.

Michaeli, Tilman (2020). *Debugging im Informatikunterricht*. Dissertation. Freie Universität Berlin.

Michalak, Magdalena, Valerie Lemke und Marius Goeke (2015). *Sprache im Fachunterricht. Eine Einführung in Deutsch als Zweitsprache und sprachbewussten Unterricht*. Tübingen: Narr Francke Attempto.

Miller, Susanne und Katrin Velten (2015). *Kinderstärkende Pädagogik in der Grundschule*. 1. Aufl. Stuttgart: Kohlhammer Verlag.

Minitério da Educação e Ciência (2015). *Projeto “Iniciação à Programação no 1.º Ciclo do Ensino Básico”*. URL: https://erte.dge.mec.pt/iniciacao-programacao-no-1o-ciclo-do-ensino-basico.

Mittermeir, Roland (2010). „Informatikunterricht zur Vermittlung allgemeiner Bildungswerte“. In: *25 Jahre Schulinformatik. Zukunft mit Herkunft*. Hrsg. von Gerhard Brandhofer. Wien: Österreichische Computer Gesellschaft, S. 54–73.

Möller, Jens (2005). „Zur Entwicklung und Genese fachbezogener Selbstkonzepte: Effekte sozialer und dimensionaler Vergleiche“. In: *Bausteine einer Bildungsgangtheorie*. Hrsg. von Barbara Schenk. Wiesbaden: VS Verlag für Sozialwissenschaften, S. 223–238. DOI: 10.1007/978-3-322-80754-0_12.

Möller, Kornelia (2015). „Genetisches Lernen und Conceptual Change“. In: *Handbuch Didaktik des Sachunterrichts*. Hrsg. von Joachim Kahlert u. a. 2. aktualisierte und erweiterte Auflage. Bad Heilbrunn: Verlag Julius Klinkhardt, S. 243–249.

Monjelat, Natalia und Annika Lantz-Andersson (2020). „Teachers’ Narrative of Learning to Program in a Professional Development Effort and the Relation to the Rhetoric of Computational Thinking“. In: *Education and Information Technologies* 25.3, S. 2175–2200. DOI: 10.1007/s10639-019-10048-8.

Montada, Leo, Ulman Lindenberger und Wolfgang Schneider (2018). „Fragen, Konzepte, Perspektiven“. In: *Entwicklungspsychologie*. Hrsg. von Wolfgang Schneider und Ulman Lindenberger. 8. überarbeitete Auflage. Weinheim und Basel: Beltz, S. 27–60.

Monteiro, Ana Francisca u. a. (2019). „Curicular Integration of Computational Thinking, Programming and Robotics in Basic Education: A Proposal for Teacher Training“. In: *Proceedings of the 12th annual International Conference of Education, Research and Innovation (ICERI)*. Hrsg. von Luis Gómez Chova, Agustín López Martínez und Ignacio Candel Torres. IATED, S. 742–749. DOI: 10.21125/iceri.2019.0232.

Moreno-Leon, Jesus, Gregorio Robles und Marcos Roman-Gonzalez (2020). „Towards Data-Driven Learning Paths to Develop Computational Thinking with Scratch“. In: *IEEE Transactions on Emerging Topics in Computing* 8.1, S. 193–205. DOI: 10.1109/TETC.2017.2734818.

Moreno-Leon, Jesus, Marcos Roman-Gonzalez und Gregorio Robles (2018). „On Computational Thinking as a Universal Skill. A Review of the Latest Research on this Ability“. In: *Proceedings of 2018 IEEE Global Engineering Education Conference (EDUCON)*. Piscataway, NJ: IEEE, S. 1684–1689. DOI: 10.1109/EDUCON.2018.8363437.

MPFS - Medienpädagogischer Forschungsverbund Südwest (2019). *KIM-Studie 2018: Kindheit, Internet, Medien. Basisstudie zum Medienumgang 6- bis 13-Jähriger in Deutschland*. Stuttgart. URL: https://www.mpfs.de/studien/kim-studie/2018/.

MSB NRW - Ministerium für Schule und Bildung des Landes Nordrhein-Westfalen (2020). *Masterplan Grundschule. Qualität stärken - Lehrkräfte unterstützen*. URL: https://www.schulministerium.nrw.de/themen/schulpolitik/masterplan-grundschule.

Müller, Dorothee (2015). *Informatikunterricht und Informatikselbstkonzept*. URL: https://docplayer.org/10471394-Informatikunterricht-und-informatikselbstkonzept.html.

Müller, Kathrin, Carsten Schulte und Johannes Magenheim (2019). „Zur Relevanz eines Prozessbereiches Interaktion und Exploration im Kontext informatischer Bildung im Primarbereich“. In: *Informatik für alle. 18. GI-Fachtagung Informatik und Schule (INFOS)*. Hrsg. von Arno Pasternak. Bonn: Gesellschaft für Informatik, S. 139–148.

Murmann, Lydia u. a. (2018). *Calliope mini: Eine Explorationsstudie im pädagogisch-didaktischen Kontext – Abschlussbericht*. URL: https://media.suub.uni-bremen.de/bitstream/elib/3457/1/00106848-1.pdf.

Neber, Heinz und Birgit Neuhaus (2018). „Entdeckendes Lernen“. In: *Handwörterbuch Pädagogische Psychologie*. Hrsg. von Detlef H. Rost, Jörn R. Sparfeldt und Susanne Buch. 5. überarbeitete und erweiterte Auflage. Weinheim und Basel: Beltz, S. 119–127.

Nelson, Beryl (2014). „The Data on Diversity“. In: *Communications of the ACM* 57.11, S. 86–95. DOI: 10.1145/2597886.

Nenner, Christin und Nadine Bergner (2022). „Informatics Education in German Primary School Curricula“. In: *Informatics in Schools: A Step Beyond Digital Education (ISSEP)*. Hrsg. von Andreas Bollin und Gerald Futschek. Springer.

Nenner, Christin, Gregor Damnik und Nadine Bergner (2021). „Weil informatische Bildung nicht erst in der Sekundarstufe beginnen darf: Integration informatischer Bildung ins Grundschullehramtsstudium“. In: *Informatik – Bildung von Lehrkräften in allen Phasen (INFOS)*. Hrsg. von Ludger Humbert. Bonn: Gesellschaft für Informatik. DOI: 10.18420/INFOS2021_F226.

Nieveen, Nienke (2007). „Formative Evaluation in Educational Design Research“. In: *An Introduction to Educational Design Research*. Hrsg. von Tjeerd Plomp und Nienke Nieveen. Enschede: SLO, S. 89–101.

OECD - Organization for Economic Co-operation and Development (2013). *PISA 2012 Assessment and Analytical Framework. Mathematics, Reading, Science, Problem Solving and Financial Literacy*. Paris: OECD Publishing. DOI: 10.1787/9789264190511-en.

Oerter, Rolf und Michael Dreher (2002). „Entwicklung des Problemlösens". In: *Entwicklungspsychologie*. Hrsg. von Rolf Oerter und Leo Montada. 5. vollständig überarbeitete Auflage. Weinheim: Beltz PVU, S. 469–494.

Paivio, Allan (1986). *Mental Representations. A Dual-Coding Approach*. New York, NY: Oxford University Press.

Papadakis, Stamatios u. a. (2016). „Using Scratch and App Inventor for Teaching Introductory Programming in Secondary Education. A Case Study". In: *International Journal of Technology Enhanced Learning* 8.3/4, S. 217–233.

Papert, Seymour (1980). *Mindstorms. Children, computers, and powerful ideas*. New York, NY: Basic Books.

Papert, Seymour und Idit Harel, Hrsg. (1991). *Constructionism*. Westport, CT: Praeger Publishers.

Penuel, William R., Lawrence P. Gallagher und Savitha Moorthy (2011). „Preparing Teachers to Design Sequences of Instruction in Earth Systems Science". In: *American Educational Research Journal* 48.4, S. 996–1025. DOI: 10.3102/0002831211410864.

Penuel, William R. u. a. (2007). „What Makes Professional Development Effective? Strategies That Foster Curriculum Implementation". In: *American Educational Research Journal* 44.4, S. 921–958. DOI: 10.3102/0002831207308221.

Petko, Dominik, Beat Döbeli Honegger und Doreen Prasse (2018). „Digitale Transformation in Bildung und Schule: Facetten, Entwicklungslinien und Herausforderungen für die Lehrerinnen- und Lehrerbildung". In: *Beiträge zur Lehrerinnen- und Lehrerbildung* 36. DOI: 10.25656/01:17094.

Petre, Marian und Blaine Price (2004). „Using Robotics to Motivate 'Back Door' Learning". In: *Education and Information Technologies* 9.2, S. 147–158. DOI: 10.1023/B:EAIT.0000027927.78380.60.

Peyton Jones, Simon und Jöran Muuß-Meerholz (2014). „Schulfach "Computing" ab Klasse 1". In: *c't* 14, S. 110–111.

Piaget, Jean (2003). *Meine Theorie der geistigen Entwicklung*. Hrsg. von Reinhard Fatke. Weinheim und Basel: Beltz.

Pinto, Maribel Santos Miranda u. a. (2017). „Laboratory of Technologies and Learning of Programming and Robotics for Pre and Primary School". In: *Proceedings of the 10th annual International Conference of Education, Research and Innovation (ICERI)*. Hrsg. von Luis Gómez Chova, Agustín López Martínez und Ignacio Candel Torres. IATED, S. 1497–1502.

Plomp, Tjeerd (2007). „Educational Design Research: an Introduction". In: *An Introduction to Educational Design Research*. Hrsg. von Tjeerd Plomp und Nienke Nieveen. Enschede: SLO, S. 9–35.

Pomberger, Gustav (1999). „Prozedurorientierte Programmierung“. In: *Informatik-Handbuch*. Hrsg. von Peter Rechenberg und Gustav Pomberger. 2., aktualisierte und erweiterte Aufl. München: Hanser, S. 517–528.

Preußler, Annabell, Michael Kerres und Mandy Schiefner-Rohs (2014). „Gestaltungsorientierung in der Mediendidaktik: Methodologische Implikationen und Perspektiven“. In: *Jahrbuch Medienpädagogik 10*. Hrsg. von Anja Hartung u. a. Wiesbaden: Springer Fachmedien, S. 253–274. DOI: 10.1007/978-3-658-04718-4_13.

PRIMAS (2013). *Guide for Professional Development Providers*. URL: https://primas-project.eu/wp-content/uploads/sites/323/2017/11/FINAL_WP4_Guide_PD_providers_licence_150708.pdf.

Przybylla, Mareen (2018). „From Embedded Systems to Physical Computing : Challenges of the "Digital World" in Secondary Computer Science Education“. Dissertation. Potsdam: Universität Potsdam.

Quigley, Claire (2017). *Working Towards a Better Gender Balance in CoderDojo Scotland Clubs*. URL: http://coderdojoscotland.com/reports/scotgov-gender-balance-2017/Gender-Balance-Analysis-V1.0-Final.pdf.

Rank, Astrid und Anja Wildemann (2015). „Die Sachen versprachlichen“. In: *Handbuch Didaktik des Sachunterrichts*. Hrsg. von Joachim Kahlert u. a. 2. aktualisierte und erweiterte Auflage. Bad Heilbrunn: Verlag Julius Klinkhardt, S. 474–479.

Rank, Astrid u. a. (2012). „Situiertes Lernen in der Lehrerfortbildung“. In: *Lehrerbildung auf dem Prüfstand* 5.2, S. 180–199.

Rasch, Björn u. a. (2014a). *Quantitative Methoden 1*. Berlin, Heidelberg: Springer. DOI: 10.1007/978-3-662-43524-3.

Rasch, Björn u. a. (2014b). *Quantitative Methoden 2*. Berlin, Heidelberg: Springer. DOI: 10.1007/978-3-662-43548-9.

Reding, Tracie Evans und Brian Dorn (2017). „Understanding the "Teacher Experience" in Primary and Secondary CS Professional Development“. In: *Proceedings of the 2017 ACM Conference on International Computing Education Research (ICER)*. Hrsg. von Josh Tenenberg. New York, NY: ACM, S. 155–163. DOI: 10.1145/3105726.3106185.

Reese, Kerstin und Verena Wolf (2018). „Calliope mini in der Lehrerfortbildung“. In: *LOG IN* 189/190, S. 108–111.

Reeves, Thomas C. (2000). „Enhancing the Worth of Instructional Technology Research through ’Design Experiments’ and Other Development Research Strategies“. In: *Annual Meeting of the American Educational Research Association* (New Orleans, LA).

Rehle, Cornelia u. a. (2017). *Einführung in grundschulpädagogisches Denken*. 4. Auflage. Augsburg: Auer.

Reinmann, Gabi (2005). „Innovation ohne Forschung? Ein Plädoyer für den Design-Based Research-Ansatz in der Lehr-Lernforschung“. In: *Unterrichtswissenschaft* 33.1, S. 52–69.

Reinmann, Gabi (2014). „Entwicklungsfrage: Welchen Stellenwert hat die Entwicklung im Kontext von Design Research? Wie wird Entwicklung zu einem wissenschaftlichen Akt?“ In: *Design-*

*Based Research*. Hrsg. von Dieter Euler und Peter F. E. Sloane. Zeitschrift für Berufs- und Wirtschaftspädagogik Beiheft 27. Stuttgart: Franz Steiner Verlag, S. 63–78.

Reinmann, Gabi (2017). „Design-based Research". In: *Gestaltungsorientierte Forschung - Basis für soziale Innovationen. Erprobte Ansätze im Zusammenwirken von Wissenschaft und Praxis*. Hrsg. von Dorothea Schemme und Hermann Novak. Berichte zur beruflichen Bildung. Bielefeld: W. Bertelsmann Verlag, S. 49–61.

Reinmann, Gabi und Heinz Mandl (2006). „Unterrichten und Lernumgebungen gestalten". In: *Pädagogische Psychologie. Ein Lehrbuch*. Hrsg. von Andreas Krapp und Bernd Weidenmann. 5. vollständig überarbeitete Auflage. Weinheim: Beltz PVU, S. 613–658.

Reinmann, Gabi und Werner Sesink (2011). *Entwicklungsorientierte Bildungsforschung*. Diskussionspapier. URL: https://gabi-reinmann.de/wp-content/uploads/2011/11/Sesink-Reinmann_Entwicklungsforschung_v05_20_11_2011.pdf.

Resnick, Mitchel und Ken Robinson (2017). *Lifelong Kindergarten. Cultivating Creativity Through Projects, Passion, Peers, and Play*. Cambridge: MIT Press.

Resnick, Mitchel, Natalie Rusk und Stina Cooke (1998). „The Computer Clubhouse: Technological Fluency in the Inner City". In: *High technology and low-income communities. Prospects for the positive use of advanced information technology*. Hrsg. von Bishwapriya Sanyal, Donald A. Schön und William John Mitchell. Cambridge, Mass: MIT Press, S. 263–285.

Resnick, Mitchel u. a. (2009). „Scratch: Programming for All". In: *Communications of the ACM* 52.11, S. 60. DOI: 10.1145/1592761.1592779.

Rich, Kathryn M. und Carla Strickland (2020). „Testing and Debugging". In: *Computer Science in K-12: An A-to-Z Handbook on Teaching Programming*. Hrsg. von Shuchi Grover. Palo Alto, CA: Edfinity, S. 211–218.

Rich, Peter J. u. a. (2017a). *Coding in K-8: Computational Practices in Primary Education from Around the World*. URL: https://www.academia.edu/32395293/Coding_in_K_8_Computational_Practices_in_Primary_Education_from_Around_the_World.

Rich, Peter Jacob u. a. (2017b). „Computing and Engineering in Elementary School: The Effect of Yearlong Training on Elementary Teacher Self-efficacy and Beliefs About Teaching Computing and Engineering". In: *International Journal of Computer Science Education in Schools* 1.1. DOI: 10.21585/ijcses.v1i1.6.

Richter, Tobias, Johannes Naumann und Holger Horz (2010). „Eine revidierte Fassung des Inventars zur Computerbildung (INCOBI-R)". In: *Zeitschrift für Pädagogische Psychologie* 24.1, S. 23–37. DOI: 10.1024/1010-0652/a000002.

Riedl, Alfred (2010). *Grundlagen der Didaktik*. 2. überarbeitete Auflage. Stuttgart: Steiner.

Riedl, Alfred und Andreas Schelten (2013). *Grundbegriffe der Pädagogik und Didaktik beruflicher Bildung*. Stuttgart: Franz Steiner Verlag.

Riefling, Markus, Anatolij Fandrich und Ira Diethelm (2020). „Mit IT2School die digitale Welt analog und digital entdecken". In: *Digitalpakt – was nun?* Hrsg. von Anabel Ternès von Hattburg und Matthias Schäfer. Wiesbaden: Springer Fachmedien, S. 293–300.

Robins, Anthony V. (2019). „Novice Programmers and Introductory Programming“. In: *The Cambridge Handbook of Computing Education Research*. Hrsg. von Sally Fincher und Anthony W. Robins. New York, NY: Cambridge University Press, S. 327–376.

Röhner, Gerhard u. a. (2020). *Gemeinsamer Referenzrahmen Informatik (GeRRI) – Mindeststandards für die auf Informatik bezogene Bildung*. Hrsg. von Gesellschaft für Informatik e.V. Bonn.

Rohrbach-Lochner, Friederike (2019). *Design-Based Research zur Weiterentwicklung der chemiedidaktischen Lehrerausbildung zu Schülervorstellungen. Entwicklung und Evaluation eines an Forschendem Lernen orientierten Seminarkonzepts*. Berlin: Logos Verlag.

Romeike, Ralf und Dominik Reichert (2011). „PicoCrickets als Zugang zur Informatik in der Grundschule“. In: *Informatik in Bildung und Beruf. 14. GI-Fachtagung "Informatik und Schule (INFOS)*. Hrsg. von Marco Thomas. Bonn: Ges.ellschaft für Informatik, S. 177–186.

Rose, David H. und Anne Meyer (2002). *Teaching Every Student in the Digital Age. Universal Design for Learning*. Alexandria, VA: Association for Supervision and Curriculum Development.

Roth, Kathleen J. u. a. (2011). „Videobased Lesson Analysis. Effective Science PD for Teacher and Student Learning“. In: *Journal of Research in Science Teaching* 48.2, S. 117–148. DOI: 10.1002/tea.20408.

Rücker, Michael T. und Niels Pinkwart (2016). „Review and Discussion of Children’s Conceptions of Computers“. In: *Journal of Science Education and Technology* 25.2, S. 274–283. DOI: 10.1007/s10956-015-9592-2.

Rushkoff, Douglas (2010). *Program or Be Programmed*. Soft Skull Press. DOI: 10.2307/j.ctt207g7rj.

Sabitzer, Barbara, Peter K. Antonitsch und Stefan Pasterk (2014). „Informatics Concepts For Primary Education: Preparing Children For Computational Thinking“. In: *Proceedings of the 9th Workshop in Primary and Secondary Computing Education (WiPSCE)*. New York, NY: ACM, S. 108–111. DOI: 10.1145/2670757.2670778.

Saeli, Mara (2012). „Teaching Programming for Secondary School. A Pedagogical Content Knowledge Based Approach“. Dissertation. URL: 10.6100/IR724491.

Sajons, Christin Marie (2020). *Kognitive und motivationale Dynamik in Schülerlaboren. Kontextualisierung, Problemorientierung und Autonomieunterst"utzung der didaktischen Struktur analysieren und weiterentwickeln*. Berlin: Logos Verlag.

Sandfuchs, Uwe (2014). „Fördern und Förderunterricht“. In: *Handbuch Grundschulpädagogik und Grundschuldidaktik*. Hrsg. von Wolfgang Einsiedler. 4. ergänzte und aktualisierte Auflage. Bad Heilbrunn: Verlag Julius Klinkhardt, S. 324–328.

Sands, Philip (2019). „Addressing Cognitive Load in the Computer Science Classroom“. In: *ACM Inroads* 10.1, S. 44–51. DOI: 10.1145/3210577.

Scaradozzi, David u. a. (2015). „Teaching Robotics at the Primary School: An Innovative Approach“. In: *Procedia - Social and Behavioral Sciences* 174. PII: S1877042815011817, S. 3838–3846. DOI: 10.1016/j.sbspro.2015.01.1122.

Schaarschmidt, Uwe (2006). „AVEM - ein persönlichkeitsdiagnostisches Instrument für die berufsbezogene Rehabilitation“. In: *Psychologische Diagnostik - Weichenstellung für den Reha-Verlauf*.

Hrsg. von Arbeitskreis Klinische Psychologie in der Rehabilitation BDP. Bonn: Deutscher Psychologen Verlag, S. 59–82.

Schaper, Niclas (2008). „(Arbeits-)Psychologische Kompetenzforschung“. In: *Forschungsperspektiven in Facharbeit und Berufsbildung. Strategien und Methoden der Berufsbildungsforschung*. Hrsg. von Martin Fischer und Georg Spöttl. Frankfurt am Main u. a.: Lang, S. 91–115.

Schemme, Dorothea (2017). „Kritische Überlegungen zu theoretischen und methodologischen Fragestellungen einer gestaltungsorietierten Forschung und ihren Rahmensetzungen in Reformprogrammen“. In: *Gestaltungsorientierte Forschung - Basis für soziale Innovationen. Erprobte Ansätze im Zusammenwirken von Wissenschaft und Praxis*. Hrsg. von Dorothea Schemme und Hermann Novak. Berichte zur beruflichen Bildung. Bielefeld: W. Bertelsmann Verlag, S. 15–45.

Schiefele, Ulrich und Ellen Schaffner (2020). „Motivation“. In: *Pädagogische Psychologie*. Hrsg. von Elke Wild und Jens Möller. 3. Aufl. 2020. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 163–185. DOI: 10.1007/978-3-662-61403-7_7.

Schiffer, Stefan (1998). *Visuelle Programmierung. Grundlagen und Einsatzmöglichkeiten*. Bonn: Addison-Wesley.

Schiffer, Stefan (1999). „Visuelle Programmierung“. In: *Informatik-Handbuch*. Hrsg. von Peter Rechenberg und Gustav Pomberger. 2., aktualisierte und erweiterte Aufl. München: Hanser, S. 619–632.

Schmid, Ute und Anja Gärtig-Daugs (2018). „Notwendigkeit der Integration elementarinformatischer Lerneinheiten in den Vor- und Grundschulunterricht“. In: *MedienPädagogik: Zeitschrift für Theorie und Praxis der Medienbildung* 31, S. 78–106. DOI: 10.21240/mpaed/31/2018.03.29.X.

Schmidt, Christin Andrea (2020). *Formatives Assessment in der Grundschule. Konzept, Einschätzungen der Lehrkräfte und Zusammenhänge*. Wiesbaden: Springer VS. DOI: 10.1007/978-3-658-26921-0.

Schmidt-Lauff, Sabine (2010). „Zeitfragen und Temporalität in der Erwachsenenbildung“. In: *Handbuch Erwachsenenbildung/Weiterbildung*. Hrsg. von Rudolf Tippelt und Aiga von Hippel. 4. durchgesehene Auflage. Wiesbaden: VS Verlag für Sozialwissenschaften, S. 213–228. DOI: 10.1007/978-3-531-92016-0_13.

Schneekloth, Urlich und Monika Pupeter (2010). „Wohlbefinden, Wertschätzung, Selbstwirksamkeit: Was Kinder für ein gutes Leben brauchen“. In: *Kinder in Deutschland 2010. 2. World Vision Kinderstudie*. Hrsg. von World Vision Deutschland e. V. Frankfurt am Main: Fischer, S. 187–221.

Schneider, Uwe, Hrsg. (2012). *Taschenbuch der Informatik*. 7., neu bearbeitete Auflage. München: Fachbuchverlag Leipzig im Carl Hanser Verlag.

Schnotz, Wolfgang (2006). *Pädagogische Psychologie: Workbook*. Weinheim, Basel: Beltz, PVU.

Schögel, Marcus und Torsten Tomczak (2009). „Fallstudie“. In: *Empirische Mastertechniken. Eine anwendungsorientierte Einführung für die Marketing- und Managementforschung*. Hrsg. von Carsten Baumgarth, Martin Eisend und Heiner Evanschitzky. Wiesbaden: Gabler Verlag, S. 79–105. DOI: 10.1007/978-3-8349-8278-0_3.

Scholz, Holger und Martin Haussmann (2017). *Bikablo 2.0. Visuelles Wörterbuch : neue Bilder für Meeting, Training & Learning*. 11. Auflage. Eichenzell: Neuland.

Schorch, Günther (2007). *Studienbuch Grundschulpädagogik. Die Grundschule als Bildungsinstitution und pädagogisches Handlungsfeld*. 3. überarbeitete und erweiterte Auflage. Bad Heilbrunn: Verlag Julius Klinkhardt.

Schröder, Hartwig (2000). *Lernen - Lehren - Unterricht. Lernpsychologische und didaktische Grundlagen*. München: Oldenbourg.

Schubert, Sigrid (1991). „Fachdidaktische Fragen der Schulinformatik und (un)mögliche Anworten". In: *Informatik und Schule. Informatik-Fachberichte*. Hrsg. von Peter Gorny. Berlin und Heidelberg: Springer, S. 27–33. DOI: 10.1007/978-3-642-76982-5_3.

Schulmeister, Rolf (2012). „Vom Mythos der Digital Natives und der Net Generation". In: *Berufsbildung in Wissenschaft und Praxis* 41.3, S. 42–46.

Schulz, Sandra (2018). „Physical Computing als Mittel der wissenschaftlichen Erkenntnisgewinnung". Dissertation. Berlin: Humboldt-Universität.

Schumacher, Eva und Liselotte Denner (2017). *Grundschulpädagogik verstehen Grundschule gestalten*. Weinheim: Beltz.

Schwarz, Jürgen (2022). *Einfaktorielle Varianzanalyse (mit Messwiederholung)*. URL: https://www.methodenberatung.uzh.ch/de/datenanalyse_spss/unterschiede/zentral/evarianzmessw.html.

Schwarz, Richard, Lutz Hellmig und Steffen Friedrich (2022). *Informatik-Monitor*. Hrsg. von Gesellschaft für Informatik e.V. Version 3. Auflage. URL: https://informatik-monitor.de.

Schwarzer, Ralf und Matthias Jerusalem (2002). „Das Konzept der Selbstwirksamkeit". In: *Selbstwirksamkeit und Motivationsprozesse in Bildungsinstitutionen*. Hrsg. von Matthias Jerusalem und Diether Hopf. Zeitschrift für Pädagogik Beiheft 44. Weinheim: Beltz, S. 28–53.

Schwill, Andreas (1993). „Fundamentale Ideen der Informatik". In: *Zentralblatt für Didaktik der Mathematik* 25.1, S. 20–31.

Schwippert, Knut u. a. (2020). „TIMSS 2019: Wichtige Ergebnisse im Überblick". In: *TIMSS 2019. Mathematische und naturwissenschaftliche Kompetenzen von Grundschulkindern in Deutschland im internationalen Vergleich*. Hrsg. von Knut Schwippert u. a. Münster: Waxmann, S. 13–24.

Sedlmeier, Peter und Frank Renkewitz (2008). *Forschungsmethoden und Statistik in der Psychologie*. München: Pearson Studium.

Seegerer, Stefan (2021). „Informatik für alle – Beitrag und exemplarische Ausgestaltung informatischer Bildung als Grundlage für Bildung in der digitalen Transformation". Dissertation. Berlin: Freie Universität Berlin.

Seidel, Tina (2014). „Angebots-Nutzungs-Modelle in der Unterrichtspsychologie. Integration von Struktur- und Prozessparadigma". In: *Zeitschrift für Pädagogik* 60.6, S. 850–866.

Sentance, Sue und Katharine Childs (2020). „X-ing Boundaries With Physical Computing". In: *Computer Science in K-12: An A-to-Z Handbook on Teaching Programming*. Hrsg. von Shuchi Grover. Palo Alto, CA: Edfinity, S. 250–258.

Sentance, Sue und Andrew Csizmadia (2017). „Computing in the Curriculum. Challenges and Strategies from a Teacher's Perspective". In: *Education and Information Technologies* 22.2, S. 469–495. DOI: 10.1007/s10639-016-9482-0.

Sentance, Sue und Simon Humphreys (2015). „Online vs Face-To-Face Engagement of Computing Teachers for their Professional Development Needs". In: *Informatics in Schools. Curricula, Competences, and Competitions (ISSEP)*. Hrsg. von Andrej Brodnik und Jan Vahrenhold. Cham: Springer, S. 69–81.

Sentance, Sue und 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)*. Hrsg. von Erik Barendsen und Peter Hubwieser. New York, NY: ACM. DOI: 10.1145/3137065.3137084.

Sentance, Sue u. a. (2017a). „Teaching with Physical Computing Devices". In: *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE)*. New York, NY: ACM, S. 87–96. DOI: 10.1145/3137065.3137083.

Sentance, Sue u. a. (2017b). „"Creating Cool Stuff". Pupils' Experience of the BBC micro:bit". In: *Proceedings of the 2017 ACM Technical Symposium on Computer Science Education (SIGCSE)*. Hrsg. von Michael E. Caspersen und A. Special Interest Group on Computer ScienceC.M. Education. New York, NY: ACM, S. 531–536. DOI: 10.1145/3017680.3017749.

Seow, Peter u. a. (2019). „Educational Policy and Implementation of Computational Thinking and Programming: Case Study of Singapore". In: *Computational thinking education*. Hrsg. von Siu Cheung Kong und Harold Abelson. Singapore: Springer Open, S. 345–361. DOI: 10.1007/978-981-13-6528-7_19.

Seufert, Tina und Roland Brünken (2018). „Multi-Media". In: *Handwörterbuch Pädagogische Psychologie*. Hrsg. von Detlef H. Rost, Jörn R. Sparfeldt und Susanne Buch. 5. überarbeitete und erweiterte Auflage. Weinheim und Basel: Beltz, S. 575–582.

Shavelson, Richard J., Judith J. Hubner und George C. Stanton (1976). „Self-Concept. Validation of Construct Interpretations". In: *Review of Educational Research* 46.3, S. 407–441. DOI: 10.3102/00346543046003407.

Shavelson, Richard J. u. a. (2003). „On the Science of Education Design Studies". In: *Educational Researcher* 32.1, S. 25–28. DOI: 10.3102/0013189X032001025.

Sheehan, Robert (2003). „Children's Perception of Computer Programming as an Aid to Designing Programming Environments". In: *Proceedings of the 2003 Conference on Interaction Design and Children (IDC)*. Hrsg. von Stuart MacFarlane. New York, NY: ACM, S. 75–83. DOI: 10.1145/953536.953548.

Shulman, Lee S. (1986). „Those Who Understand: Knowledge Growth in Teaching". In: *Educational Researcher* 15.2, S. 4–14.

Shulman, Lee S. (1987). „Knowledge and Teaching. Foundations of the New Reform". In: *Harvard Educational Review* 57.1, S. 1–22.

Simon, Alexandra, Katharina Geldreich und Peter Hubwieser (2019). „How to Transform Programming Processes in Scratch to Graphical Visualizations“. In: *Proceedings of the 14th Workshop in Primary and Secondary Computing Education on (WiPSCE)*. Hrsg. von Quintin Cutts und Thorsten Brinda. New York, NY: ACM, S. 1–9. DOI: 10.1145/3361721.3361723.

Sloane, Peter F. E. (2014). „Wissensgenese in Design-Based Research Projekten“. In: *Design-Based Research*. Hrsg. von Dieter Euler und Peter F. E. Sloane. Zeitschrift für Berufs- und Wirtschaftspädagogik Beiheft 27. Stuttgart: Franz Steiner Verlag, S. 113–139.

Smith, Neil, Clare Sutcliffe und Linda Sandvik (2014). „Code Club: Bringing Programming to UK Primary Schools through Scratch“. In: *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE)*. Hrsg. von J. D. Dougherty. New York, NY: ACM, S. 517–522. DOI: 10.1145/2538862.2538919.

Smith, Neil u. a. (2015). „Master Teachers in Computing: What Have we Achieved?“ In: *Proceedings of the Workshop in Primary and Secondary Computing Education (WiPSCE)*. New York, NY: ACM, S. 21–24. DOI: 10.1145/2818314.2818332.

SMK Sachsen - Staatsministerium für Kultus Sachsen (2019). *Lehrplan Grundschule Werken*. URL: https://www.schulportal.sachsen.de/lplandb/index.php?lplanid=73&lplansc=5qcMpegw6pQpLsPppKnZ&token=22f66f7629064400dffe1906fa1a23f3&lplansearchfield=Grundschule+Werken#page73_3397.

Sorva, Juha (2020). „Naïve Conceptions of Novice Programmers“. In: *Computer Science in K-12: An A-to-Z Handbook on Teaching Programming*. Hrsg. von Shuchi Grover. Bd. 141-157. Palo Alto, CA: Edfinity.

Stamann, Christoph, Markus Janssen und Margrit Schreier (2016). „Qualitative Inhaltsanalyse – Versuch einer Begriffsbestimmung und Systematisierung“. In: *Forum Qualitative Sozialforschung* 17.3, Art. 16.

Standl, Bernhard und Nadine Schlomske-Bodenstein (2021). „Exploring Indicators to Promote Preservice Teachers' Self-Efficacy in Programming Tasks“. In: *Proceedings of the 21st International Conference on Computing Education Research (Koli Calling)*. Hrsg. von Otto Seppälä und Andrew Petersen. New York, NY: ACM, S. 1–3. DOI: 10.1145/3488042.3489965.

Stark, Robin (2004). „Eine integrative Forschungsstrategie zur anwendungsbezogenen Generierung relevanten wissenschaftlichen Wissens in der Lehr-Lern-Forschung“. In: *Unterrichtswissenschaft* 32.3, S. 257–273.

Statter, David und Michal Armoni (2020). „Teaching Abstraction in Computer Science to 7 th Grade Students“. In: *ACM Transactions on Computing Education* 20.1, S. 1–37. DOI: 10.1145/3372143.

StMBKWK - Bayerisches Staatsministerium für Bildung und Kultus, Wissenschaft und Kunst (2014). *LehrplanPLUS Grundschule. Lehrplan für die bayerische Grundschule*. München: StMBKWK.

Storte, Davide u. a. (2019). *Coding, Programming and the Changing Curriculum for Computing in Schools. Report of UNESCO/IFIP TC3 Meeting at OCCE*. URL: https://www.ifip-tc3.org/app/download/7193588451/UNESCO+meeting+at+OCCE+2018+report+final+edit+301019.pdf?t=1572465399.

Straube, Philipp u. a. (2018). „Eine digitale Perspektive für den Sachunterricht? Fachdidaktische Überlegungen und Implikationen“. In: *Widerstreit Sachunterricht* 24.

Strick, Heinz Klaus (2009). *Mohammed Al-Khwarizmi (780–850): »Vater der Algebra«*. Hrsg. von Spektrum der Wissenschaft. URL: https://www.spektrum.de/wissen/mohammed-al-khwarizmi-780-850/979324.

Suessenbach, Felix, Eike Schröder und Mathias Winde (2022). *Informatik für alle! Informatikunterricht zur gesellschaftlichen Teilhabe und Chancengleichheit*. Hrsg. von Stifterverband für die Deutsche Wissenschaft e.V.

Sweller, John, Jeroen J. G. van Merriënboer und Fred G. W. C. Paas (1998). „Cognitive Architecture and Instructional Design“. In: *Educational Psychology Review* 10.3, S. 251–296.

SWK - Ständige Wissenschaftliche Kommission der Kultusministerkonferenz (2022). *Digitalisierung im Bildungssystem: Handlungsempfehlungen von der Kita bis zur Hochschule. Gutachten der Ständigen Wissenschaftlichen Kommission der Kultusministerkonferenz (SWK)*. DOI: 10.25656/01:25273.

Sysło, Maciej M. und Anna Beate Kwiatkowska (2015). „Introducing a New Computer Science Curriculum for All School Levels in Poland“. In: *Informatics in Schools. Curricula, Competences, and Competitions (ISSEP)*. Hrsg. von Andrej Brodnik und Jan Vahrenhold. Cham: Springer, S. 141–154.

Tausch, Reinhard und Anne-Marie Tausch (1998). *Erziehungspsychologie. Begegnung von Person zu Person*. 11., korrigierte Aufl. Göttingen: Hogrefe Verlag für Psychologie.

Taylor, Merilyn, Ann Harlow und Michael Forret (2010). „Using a Computer Programming Environment and an Interactive Whiteboard to Investigate Some Mathematical Thinking“. In: *Procedia - Social and Behavioral Sciences* 8, S. 561–570. DOI: 10.1016/j.sbspro.2010.12.078.

Tent, Lothar und Ingeborg Stelzl (1993). *Pädagogisch-psychologische Diagnostik. Band 1: Theoretische und methodische Grundlagen*. Göttingen: Hogrefe Verlag für Psychologie.

Terhart, Ewald (2014). „Der Heilige Gral der Schul- und Unterrichtsforschung - gefunden?“ In: *Die Hattie-Studie in der Diskussion. Probleme sichtbar machen*. Hrsg. von Ewald Terhart. Seelze: Klett Kallmeyer, S. 10–23.

The Royal Society, Hrsg. (2017). *After the Reboot: Computing Education in UK Schools*. URL: https://royalsociety.org/computing-education.

Theodoropoulos, Anastasios u. a. (2018). „Computing in the Physical World Engages Students. Impact on their Attitudes and Self-Efficacy towards Computer Science through Robotic Activities“. In: *Proceedings of the 13th Workshop in Primary and Secondary Computing Education (WiPSCE)*. Hrsg. von Andreas Mühling und Quintin Cutts. New York, NY: ACM, S. 1–4. DOI: 10.1145/3265757.3265770.

Timperley, Helen u. a. (2007). *Teacher Professional Learning and Development. Best Evidence Synthesis Iteration [BES]*. Wellington: Ministry of Education New Zealand.

Trautner, Hanns Martin (1992). *Lehrbuch der Entwicklungspsychologie*. 2. Auflage. Göttingen: Hogrefe.

Tsai, Meng-Jung, Ching-Yeh Wang und Po-Fen Hsu (2017). „Developing the Computer Programming Self-Efficacy Scale for Computer Literacy Education“. In: *Journal of Educational Computing Research* 56.8, S. 1345–1360. DOI: 10.1177/0735633117746747.

Tsarava, Katerina, Korbinian Moeller und Manuel Ninaus (2018). „Training Computational Thinking through Board Games. The Case of Crabs & Turtles“. In: *International Journal of Serious Games* 5.2, S. 25–44. DOI: 10.17083/ijsg.v5i2.248.

Tulodziecki, Gerhard, Silke Grafe und Bardo Herzig (2013). *Gestaltungsorientierte Bildungsforschung und Didaktik. Theorie - Empirie - Praxis*. Bad Heilbrunn: Verlag Julius Klinkhardt.

Tulodziecki, Gerhard, Bardo Herzig und Sigrid Blömeke (2017). *Gestaltung von Unterricht. Eine Einführung in die Didaktik*. 3., überarbeitete und erweiterte Auflage. Bad Heilbrunn: Verlag Julius Klinkhardt. DOI: 10.36198/9783838547947.

van den Akker, Jan (1999). „Principles and Methods of Development Research“. In: *Design Approaches and Tools in Education and Training*. Netherlands: Kluwer Academic Publishers, S. 1–14.

Vanderlinde, Ruben und Johan van Braak (2010). „The Gap Between Educational Research and Practice: Views of Teachers, School Leaders, Intermediaries and Researchers“. In: *British Educational Research Journal* 36.2, S. 299–316. DOI: 10.1080/01411920902919257.

Vigerske, Stefanie (2017). *Transfer von Lehrerfortbildungsinhalten in die Praxis*. Wiesbaden: Springer Fachmedien Wiesbaden. DOI: 10.1007/978-3-658-17685-3.

Vivian, Rebecca, Katrina Falkner und Claudia Szabo (2014). „Can Everybody Learn to Code?“ In: *Proceedings of the 14th Koli Calling International Conference on Computing Education Research*. Hrsg. von Päivi Kinnunen und Simon. New York, NY: ACM, S. 41–50. DOI: 10.1145/2674683.2674695.

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

Wademan, Marc R. (2005). „Utilizing Development Research to Guide People Capability Maturity Model Adoption Considerations“. Dissertation. Syracuse, NY: Syracuse University.

Waite, Jane u. a. (2019). „Unplugged Computing and Semantic Waves“. In: *Proceedings of the 1st UK and Ireland Computing Education Research Conference*. Hrsg. von Janet Carter. New York, NY: ACM, S. 1–7. DOI: 10.1145/3351287.3351291.

Waite, Jane Lisa u. a. (2018). „Abstraction in Action. K-5 Teachers' Uses of Levels of Abstraction, articularly the Design Level, in Teaching Programming“. In: *International Journal of Computer Science Education in Schools* 2.1, S. 14–40. DOI: 10.21585/ijcses.v2i1.23.

Wang, Feng und Michael J. Hannafin (2005). „Design-Based Research and Technology-Enhanced Learning Environments“. In: *Educational Technology Research and Development* 53.4, S. 5–23. DOI: 10.1007/BF02504682.

Webb, Mary u. a. (2017). „Computer Science in K-12 School Curricula of the 2lst Century. Why, what and when?“ In: *Education and Information Technologies* 22.2, S. 445–468. DOI: 10.1007/s10639-016-9493-x.

Wedekind, Hartmut u. a. (1998). „Modellierung , Simulation, Visualisierung: Zu aktuellen Aufgaben der Informatik“. In: *Informatik Spektrum* 21.5, S. 265–272. DOI: 10.1007/s002870050104.

Weinert, Franz E. (2001a). „Concept of Competence: A Conceptual Clarification“. In: *Defining and Selecting Key Competencies*. Hrsg. von Dominique Simone Rychen und Laura Hersh Salganik. Seattle, Bern und Göttingen: Hogrefe & Huber, S. 45–65.

Weinert, Franz E. (2001b). „Vergleichende Leistungsmessung in Schulen - eine umstrittene Selbstverständlichkeit“. In: *Leistungsmessungen in Schulen*. Hrsg. von Franz E. Weinert. Weinheim: Beltz, S. 17–31.

Weintrop, David und Shuchi Grover (2020). „JavaScript, Python, Scratch, or Something Else? Navigating the Bustling World of Introductory Programming Languages“. In: *Computer Science in K-12: An A-to-Z Handbook on Teaching Programming*. Hrsg. von Shuchi Grover. Palo Alto, CA: Edfinity, S. 99–112.

Weitz, Katharina u. a. (2017). „Computer Science in Early Childhood Education: Pedagogical Beliefs and Perceived Self-Confidence in Preschool Teachers“. In: *Proceedings of the 12th Workshop on Primary and Secondary Computing Education (WiPSCE)*. Hrsg. von Erik Barendsen. New York, NY: ACM, S. 117–118. DOI: 10.1145/3137065.3144573.

Wenger, Etienne (1998). *Communities of Practice: Learning, Meaning and Identity*. Cambridge: Cambridge University Press.

Weskamp, Stephanie (2019). *Heterogene Lerngruppen im Mathematikunterricht der Grundschule. Design Research im Rahmen substanzieller Lernumgebungen*. Wiesbaden: Springer Fachmedien. DOI: 10.1007/978-3-658-25233-5.

Wiater, Werner (2018). *Unterrichtsprinzipien*. 7. Auflage. Augsburg: Auer.

Wiliam, Dylan (2013). „Assessment: The Bridge between Teaching and Learning“. In: *Voices from the Middle* 21.2, S. 15–20.

Wing, Jeannette M. (2006). „Computational thinking“. In: *Communications of the ACM* 49.3, S. 33. DOI: 10.1145/1118178.1118215.

Winslow, Leon E. (1996). „Programming Pedagogy – A Psychological Overview“. In: *ACM SIGCSE Bulletin* 28.3, S. 17–22. DOI: 10.1145/234867.234872.

Wirth, Niklaus (2008). „A Brief History of Software Engineering“. In: *IEEE Annals of the History of Computing* 30.3, S. 32–39. DOI: 10.1109/MAHC.2008.33.

Wolking, Maike und Ute Schmid (2017). „Mental Models, Career Aspirations, and the Acquirement of Basic Concepts of Computer Science in Elementary Education“. In: *Proceedings of the 12th Workshop on Primary and Secondary Computing Education (WiPSCE)*. Hrsg. von Erik Barendsen und Peter Hubwieser. New York, NY, USA: ACM, S. 119–120. DOI: 10.1145/3137065.3137076.

Wong, Gary K.W. und Shan Jiang (2018). „Computational Thinking Education for Children: Algorithmic Thinking and Debugging“. In: *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*. Piscataway, NJ: IEEE, S. 328–334. DOI: 10.1109/TALE.2018.8615232.

Wood, David (2003). „The Why? What? When? and How? of Tutoring: The Development of Helping and Tutoring Skills in Children“. In: *Literacy Teaching and Learning* 7.1-2, S. 1–30.

Wood, David, Heather Wood und David Middleton (1978). „An Experimental Evaluation of Four Face-to-Face Teaching Strategies“. In: *International Journal of Behavioral Development* 1.2, S. 131–147. DOI: 10.1177/016502547800100203.

Worth, Karen (1999). „The Power of Children's Thinking“. In: *Inquiry: Thoughts, Views, and Strategies for the K-5 classroom*. Hrsg. von National Research Council. National Science Foundation, S. 25–31.

Yadav, Aman und Marc Berges (2019). „Computer Science Pedagogical Content Knowledge: Characterizing Teacher Performance“. In: *ACM Transactions on Computing Education* 19.3, S. 1–24. DOI: 10.1145/3303770.

Yadav, Aman, Hai Hong und Chris Stephenson (2016). „Computational Thinking for All. Pedagogical Approaches to Embedding 21st Century Problem Solving in K-12 Classrooms“. In: *TechTrends* 60.6, S. 565–568. DOI: 10.1007/s11528-016-0087-7.

Yadav, Aman, Alex Lishinski und Phil Sands (2021). „Self-efficacy Profiles for Computer Science Teachers“. In: *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (SIGCSE)*. Hrsg. von Mark Sherriff u. a. New York, NY: ACM, S. 302–308. DOI: 10.1145/3408877.3432441.

Yadav, Aman u. a. (2014). „Computational Thinking in Elementary and Secondary Teacher Education“. In: *ACM Transactions on Computing Education* 14.1, S. 1–16. DOI: 10.1145/2576872.

Yadav, Aman u. a. (2016). „Expanding Computer Science Education in Schools. Understanding Teacher Experiences and Challenges“. In: *Computer Science Education* 26.4, S. 235–254. DOI: 10.1080/08993408.2016.1257418.

Yin, Robert K. (2014). *Case Study Research. Design and Methods*. 5. edition. Los Angeles u. a.: Sage.

Zech, Friedrich (2002). *Grundkurs Mathematikdidaktik. Theoretische und praktische Anleitungen für das Lehren und Lernen von Mathematik*. 10. Auflage. Beltz Pädagogik. Weinheim und Basel: Beltz Verlag.

Zhang, LeChen und Jalal Nouri (2019). „A Systematic Review of Learning Computational Thinking through Scratch in K-9“. In: *Computers & Education* 141, S. 103607. DOI: 10.1016/j.compedu.2019.103607. (Besucht am 11. 07. 2019).

Zumhasch, Clemens (2014). „Schulleistungsbeurteilung: Leistungen feststellen und bewerten“. In: *Handbuch Grundschulpädagogik und Grundschuldidaktik*. Hrsg. von Wolfgang Einsiedler. 4. ergänzte und aktualisierte Auflage. Bad Heilbrunn: Verlag Julius Klinkhardt, S. 302–310.

# ANHANG

# A Datenerhebung

## A.1 Übersichten

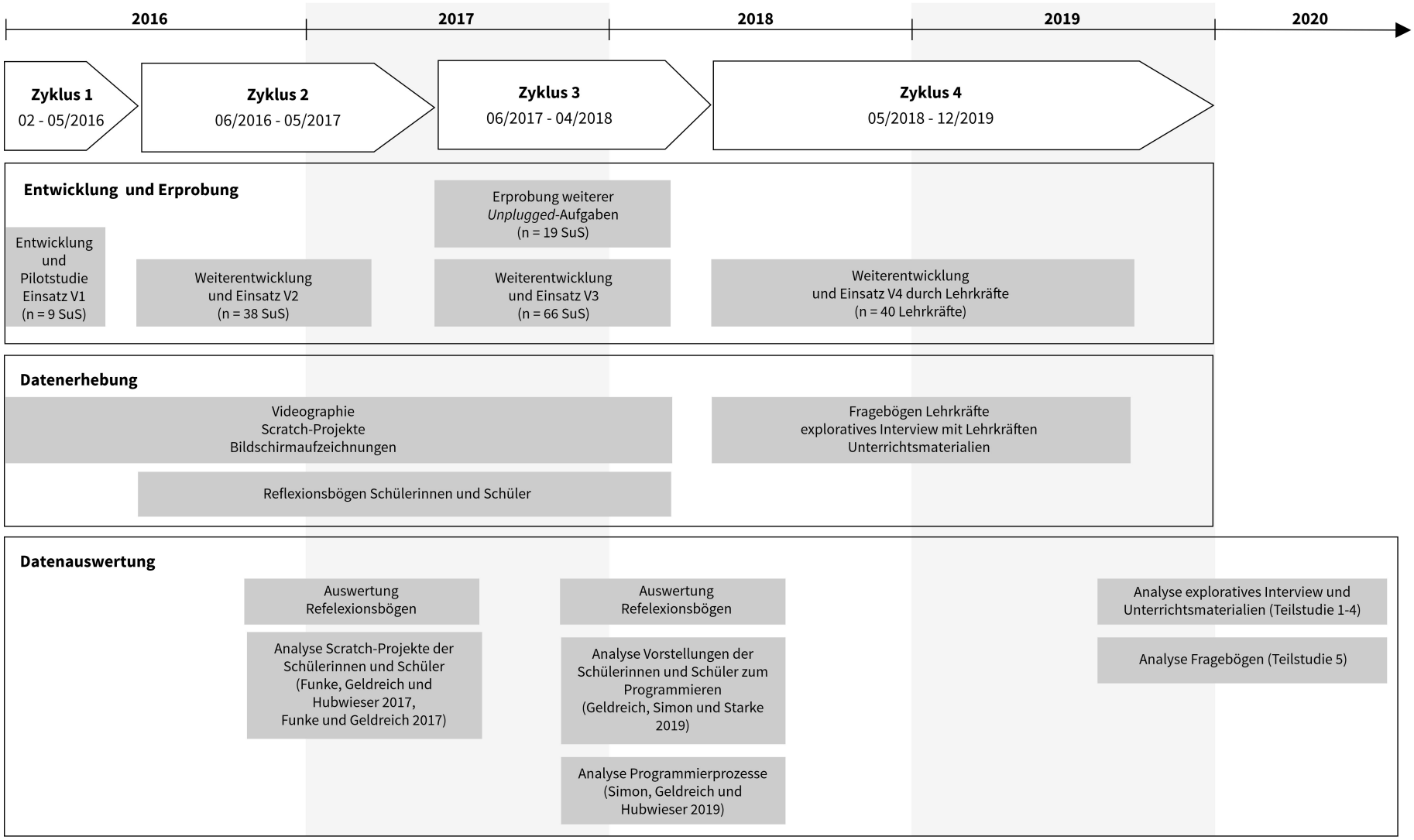



*Abbildung A.1 Übersicht über die einzelnen Zyklen des DBR-Prozesses zur Entwicklung der Unterrichtssequenz und die jeweilige Datenerhebung und -auswertung*

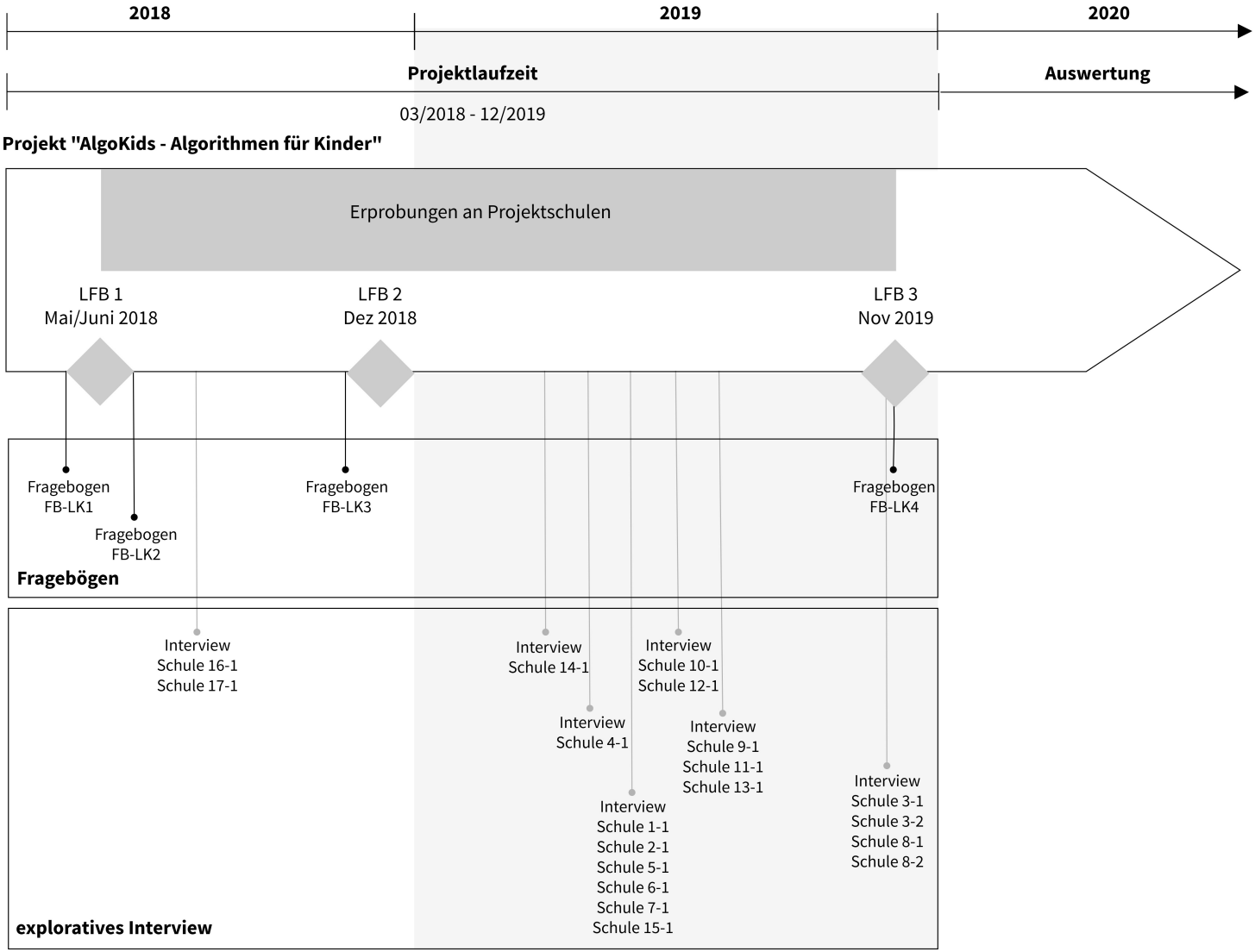



*Abbildung A.2 Übersicht über das Projekt „AlgoKids – Algorithmen für Kinder“ und die Datenerhebung*

## A.2 Ablauf des explorativen Interviews

Sämtliche Interviews (n=19), die im Rahmen des Projekt *AlgoKids – Algorithmen für Kinder* mit den Lehrkräften geführt wurden, folgten dem nachfolgenden Ablauf.

**Begrüßung:**
„Schön, dass ich dich/euch heute an der Schule besuchen darf. Vielen Dank, dass du/ihr euch die Zeit nehmt! Ich führe die Interviews, um einen Einblick in die Erfahrungen zu bekommen, die du/ihr im Laufe von *AlgoKids* gesammelt habt.

Das Gespräch hat keine festgelegte Dauer – wir reden einfach, bis uns die Themen ausgehen, oder du/ihr wegmusst/wegmüsst. Ich zeichne das Gespräch auf und für die Auswertung wird es dann transkribiert, also abgetippt. Alles, was du/ihr sagst/sagt, wird soweit anonymisiert, dass daraus nicht mehr auf dich/euch oder deine/eure Schule geschlossen werden kann. Also wenn ich später in meiner Arbeit aus den Gesprächen zitiere, wissen die Lesenden nicht, welche Lehrkraft das gesagt hat."

---

**Schaffen einer offenen Gesprächsatmosphäre:**
„Es ist kein klassisches Interview, in dem ich dir/euch feste Fragen stelle und du/ihr dann direkt antworten musst/müsst. Sondern ein relativ freies Gespräch, in dem du/ihr zuerst von deinen/euren Erfahrungen im *AlgoKids*-Projekt berichtest/berichtet und ich hake dann einfach an den Stellen ein, zu denen ich gerne noch etwas mehr wissen würde. Im Laufe des Gesprächs habe ich dann zu einigen Themen noch Gesprächsimpulse mitgebracht, die wir eventuell aufgreifen können. Mir ist ganz wichtig, dass du/ihr nichts beschönigt – also auch, wenn etwas nicht wie gewünscht lief oder Materialien sich im Einsatz nicht bewährt haben. Das sind mitunter die Punkte, die für mich besonders spannend sind."

---

[Beginn der Aufzeichnung]

**1. Phase der Interviewführung – quasi-normales Gespräch mit erzählungsgenerierender Frage:**
„Kannst du/könnt ihr mir zu Beginn einfach einmal von deinen/euren Erfahrung im *AlgoKids*-Projekt erzählen?"

*Die Lehrkräfte berichten von ihren Erfahrungen. Die Interviewerin fragt ggf. nach, merkt etwas an oder berichtet von ihren eigenen Erfahrungen, z. B. mit dem Programmieren oder den Materialien im Unterricht.*

---

**2. Phase der Interviewführung – situationsflexibel gehandhabtes Leitfadeninterview:**

„Ich habe verschiedene Kärtchen mitgebracht, auf denen unterschiedliche Themen abgebildet sind. Es wäre super, wenn du/ihr mir noch etwas zu denen Themen erzählen kannst/könnt, die euch ansprechen bzw. zu denen ihr etwas teilen könnt."

*Die Interviewerin legt verschiedene Karten mit Piktogrammen*[1] *als visuelle Impulse aus (siehe Abbildung A.3, A.4 und A.5). Diese müssen nicht alle thematisiert werden, sondern sind lediglich als Themenangebote zu verstehen. Die Lehrkraft/Lehrkräfte äußert/äußern sich zu ausgewählten Impulsen.*

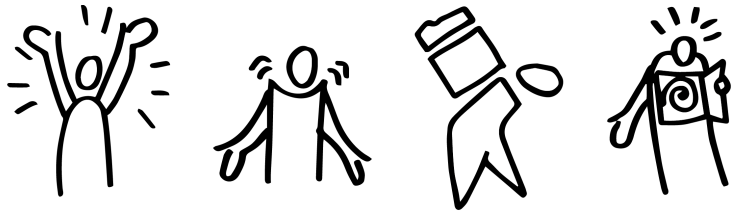


*Abbildung A.3 Bildkarten 1-4 (von links nach rechts): Erfolgserlebnisse, Ratlosigkeit, Belastungen, entdeckte Fähigkeiten*

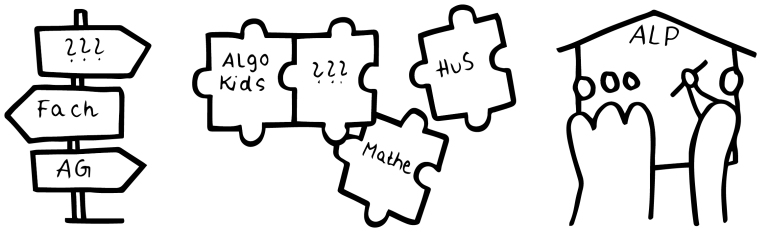



*Abbildung A.4 Bildkarten 5-7 (von links nach rechts): Verortung des Programmierens in der Grundschule, Anknüpfung an den Lehrplan, Fortbildungen an der ALP*

*Abbildung A.5 Bildkarten 8-10 (von links nach rechts): Computer, Werkzeuge und Materialien, Genderunterschiede*

---

**3. Phase der Interviewführung – Klärung offen gebliebener Fragen:**

*Die Interviewerin kommt ggf. auf Fragen zurück, die in den ersten beiden Phasen nicht (vollständig) geklärt werden konnten.*

---

**Abschluss:**

„Jetzt sind wir auch schon am Ende. Vielen Dank, dass du/ihr dir/euch die Zeit für das Interview genommen hast/habt!"

[Ende der Aufzeichnung]

[1] Die Abbildungen wurden von der Autorin in Anlehnung an ein visuelles Wörterbuch von Scholz und Haussmann (2017) gestaltet.

## A.3 Fragebögen

### A.3.1 Fragebogen zu Beginn der ersten Fortbildungsveranstaltung bzw. zum *AlgoKids*-Projektstart (FB-LK1)

Technische Universität München
TUM School of Education
Professur für Didaktik der Informatik
Prof. Dr. Peter Hubwieser, Katharina Geldreich

# Fragebogen für Lehrkräfte im Projekt
## AlgoKids – Algorithmen für Kinder

genehmigt am 12.11.2018, Aktenzeichen IV.8-BO7106/113/9

**Liebe Lehrerinnen und Lehrer,**

in diesem Fragebogen werden Informationen abgefragt und Daten erhoben, die wir für die Bewertung des weiteren Projektverlaufs und der Ergebnisse des Projekts benötigen. Selbstverständlich werden Ihre Angaben streng vertraulich behandelt und ausschließlich für wissenschaftliche und organisatorische Zwecke verwendet.

Das Ausfüllen des Fragebogens ist freiwillig. Sie haben außerdem die Möglichkeit, die Fragen nur teilweise zu beantworten. Für Rückfragen stehen wir Ihnen gerne zur Verfügung.

**Herzlichen Dank für Ihre Unterstützung!**

**Um die Angaben und Ergebnisse, die im Laufe des Projekts gesammelt werden, einander zuordnen zu können, verwenden wir einen persönlichen Code.** Damit Sie diesen nicht vergessen können, verwenden wir hierzu eine Kombination aus Buchstaben und Zahlen, die Sie sich jederzeit wieder herleiten können.

**Beispiel für einen Code:**

- Ihre Mutter heißt **Hi**lde
- Ihr Vater heißt **Fr**anz
- Ihr Geburtstag ist der **05**.08.1974

Als Personencode ergibt sich: **HI FR 05**

**Bitte geben Sie hier Ihren Personencode an:**

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| **1. und 2. Buchstabe des Vornamens Ihrer Mutter** | | **1. und 2. Buchstabe des Vornamens Ihres Vaters** | | **1. und 2. Ziffer Ihres Geburtstages** | |



## Angaben zu Ihrer Person

1. **Ihr Geschlecht:** ☐ weiblich ☐ männlich ☐ keine Angabe

2. **Wie alt sind Sie?**

   ☐ unter 30 Jahre ☐ 41-50 Jahre
   ☐ 30-40 Jahre ☐ Über 50 Jahre

3. **Ich bin Lehrer/-in seit _____ Jahren (inkl. Vorbereitungsdienst).**

4. **Haben Sie eine andere (Berufs-)Ausbildung (mit oder ohne Abschluss)?**

   ☐ ja ☐ nein

   Wenn ja, welche?

   _____

5. **Hatten Sie in Ihrer Schulzeit Unterricht in Informatik oder Informationstechnik?**

   ☐ nein, überhaupt nicht
   ☐ in einem Wahlfach über die Dauer von _____ Jahren

   **In einem Pflichtfach und zwar…**

   ☐ Natur und Technik in der Unterstufe des bayerischen G8
   ☐ Informatik in der Mittelfstufe des bayerischen G8
   ☐ Informatik in der Oberstufe des bayerischen G8
   ☐ Informationstechnik an einer bayerischen Realschule
   ☐ Informatik in der Mittelfstufe des bayerischen G8

   ☐ in einer anderen Form:

   Land/Bundesland: _____

   Schulart: _____

   Schulstufen: _____

   Bezeichnung des Faches: _____

6. **Zu Ihrem Hochschulabschluss:**

   - **Haben Sie das zweite Staatsexamen für das Lehramt an Grundschulen abgelegt?**

     ☐ nein

     ☐ ja, in Bayern

     ☐ ja, in folgendem anderen Bundesland: _____

     → **Wenn ja, in welchem Unterrichtsfach oder -fächern haben Sie Staatsexamen gemacht?**

     _____

     → **Wenn ja, in welchen Didaktikfächern haben Sie Staatsexamen gemacht?**

     _____

   - **Haben Sie zusätzlich oder stattdessen einen anderen Hochschulabschluss?**

     ☐ nein

     ☐ ja, und zwar (Art, Fächer, Bundesland):

     _____

7. **Auf wessen Initiative hin, nehmen Sie an dieser Fortbildung teil?**

   ☐ überwiegend auf meine eigene Initiative

   ☐ überwiegend auf Initiative der Schulleitung

   ☐ gleichmäßig verteilt auf meine eigene Initiative und die der Schulleitung



## Situation an Ihrer Schule

8. **In welcher Jahrgangsstufe unterrichten Sie in diesem Schuljahr hauptsächlich?**

   ☐ erste Klasse ☐ dritte Klasse

   ☐ zweite Klasse ☐ vierte Klasse

9. **Welche Fächer unterrichten Sie in diesem Schuljahr?**

   ☐ Deutsch ☐ Sport

   ☐ Mathematik ☐ Religionslehre/Ethik

   ☐ Heimat- und Sachunterricht ☐ Englisch

   ☐ Kunst ☐ Werken und Gestalten

   ☐ Musik ☐ sonstige: _____

10. **Zur technischen Ausstattung an Ihrer Schule:**

    Bitte geben Sie an, inwiefern Sie den folgenden Aussagen zustimmen:

| | stimme nicht zu | stimme eher nicht zu | stimme eher zu | stimme voll und ganz zu |
|---|---|---|---|---|
| Es ist eine ausreichende IT-Ausstattung vorhanden (z.B. Computer, Software). | ☐ | ☐ | ☐ | ☐ |
| Die Computer meiner Schule sind technisch auf dem aktuellen Stand. | ☐ | ☐ | ☐ | ☐ |
| Der Internetzugang ist ausreichend (z.B. Geschwindigkeit und Stabilität der Verbindung). | ☐ | ☐ | ☐ | ☐ |

## Sicherheit im Umgang mit Computern und Computeranwendungen

**Bitte geben Sie an, in welchem Ausmaß die Aussagen auf Sie zutreffen:**

| | stimme nicht zu | stimme eher nicht zu | neutral | stimme eher zu | stimme zu |
|---|---|---|---|---|---|
| 11. Im Umgang mit Computern fühle ich mich sicher. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 12. Wenn mein Computer abstürzt, gerate ich in Panik. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 13. Bei der Arbeit mit dem Computer lasse ich mich durch auftretende Schwierigkeiten leicht frustrieren. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 14. Das Arbeiten am Computer bereitet mir Unbehagen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 15. Beim Arbeiten mit dem Computer habe ich oft Angst, etwas kaputt zu machen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 16. Ich habe das Gefühl, dass ich meinen Computer im Griff habe. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 17. Wenn möglich, vermeide ich das Arbeiten am Computer. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 18. Auch bei auftretenden Computerproblemen bleibe ich ruhig. | ☐ | ☐ | ☐ | ☐ | ☐ |



## Vorerfahrungen im Bereich der Programmierung

**Bitte geben Sie an, welche Vorerfahrungen Sie haben:**

| | keine Vor-erfahrung | Auspro-bieren | Produktiver Einsatz eigener Programme durch mich selbst | Professioneller Einsatz meiner Programme durch andere | Einsatz meiner Programme im Unterricht mit Schülerinnen und Schülern |
|---|---|---|---|---|---|
| 19. Programmierung mit textuellen Programmier-sprachen, wie z.B. Java, C, C++, C#, Pascal, R, Python | ☐ | ☐ | ☐ | ☐ | ☐ |
| 20. Programmierung mit grafischen blockorientierten Programmiersystemen, wie z.B. Scratch, Blocks, Snap! | ☐ | ☐ | ☐ | ☐ | ☐ |
| 21. Programmierung von Makros, z.B. in Word, Excel etc. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 22. Erstellen von Webseiten in XML, HTML, CSS, etc. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 23. Reale Automatisierungs-systeme wie LEGO Mindstorms oder WeDo, Fischer Technik etc. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 24. Andere: _____ | ☐ | ☐ | ☐ | ☐ | ☐ |

## Arbeitsbezogenes Verhalten und Erleben

Wir bitten Sie, einige Ihrer üblichen Verhaltensweisen, Einstellungen und Gewohnheiten zu beschreiben, wobei vor allem auf Ihr Arbeitsleben Bezug genommen wird.

Bitte lesen Sie jeden der folgenden Sätze durch und entscheiden Sie, in welchem Maße er auf Sie persönlich zutrifft.

| | Die Aussage trifft… | überhaupt nicht zu | eher nicht zu | teil-weise zu | eher zu | voll und ganz zu |
|---|---|---|---|---|---|---|
| 25. | Meine Arbeit soll stets ohne Fehl und Tadel sein. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 26. | Wenn ich keinen Erfolg habe, resigniere ich schnell. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 27. | Für mich sind Schwierigkeiten dazu da, dass ich sie überwinde. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 28. | Mich bringt so leicht nichts aus der Ruhe. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 29. | Ich kontrolliere lieber noch dreimal nach, als dass ich fehlerhafte Arbeitsergebnisse abliefere. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 30. | Misserfolge kann ich nur schwer verkraften. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 31. | Wenn mir etwas nicht gelingt, sage ich mir: Jetzt erst recht! | ☐ | ☐ | ☐ | ☐ | ☐ |
| 32. | Ich bin ein ruheloser Mensch. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 33. | Bei meiner Arbeit habe ich den Ehrgeiz, keinerlei Fehler zu machen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 34. | Berufliche Fehlschläge können mich leicht entmutigen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 35. | Misserfolge werfen mich nicht um, sondern veranlassen mich zu noch stärkerer Anstrengung. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 36. | Ich glaube, dass ich ziemlich hektisch bin. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 37. | Was immer ich tue, es muss perfekt sein. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 38. | Wenn ich in der Arbeit erfolglos bin, deprimiert mich das sehr. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 39. | Ich bin mir sicher, dass ich auch die künftigen Anforderungen des Lebens gut bewältigen kann. | ☐ | ☐ | ☐ | ☐ | ☐ |



| | Die Aussage trifft… | überhaupt nicht zu | eher nicht zu | teil-weise zu | eher zu | voll und ganz zu |
|---|---|---|---|---|---|---|
| 40. | Ich glaube, ich bin ein ruhender Pol in meinem Umfeld. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 41. | Für mich ist die Arbeit erst dann getan, wenn ich rundum mit dem Ergebnis zufrieden bin. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 42. | Ich verliere leicht den Mut, wenn ich trotz Anstrengung keinen Erfolg habe. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 43. | Ein Misserfolg kann bei mir neue Kräfte wecken. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 44. | Ich kann mich in fast allen Situationen ruhig und bedächtig verhalten. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 45. | Es widerstrebt mir, wenn ich eine Arbeit abschließen muss, obwohl sie noch verbessert werden könnte. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 46. | Wenn ich irgendwo versagt habe, kann mich das ziemlich mutlos machen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 47. | Wenn mir etwas nicht gelingt, bleibe ich hartnäckig und strenge mich umso mehr an. | ☐ | ☐ | ☐ | ☐ | ☐ |
| 48. | Hektik und Aufregung um mich herum lassen mich kalt. | ☐ | ☐ | ☐ | ☐ | ☐ |

## Herzlichen Dank für Ihre Angaben!

### A.3.2 Fragebogen zum Ende der ersten Fortbildungsveranstaltung (FB-LK2)

Technische Universität München
TUM School of Education
Professur für Didaktik der Informatik
Prof. Dr. Peter Hubwieser, Katharina Geldreich

# Fragebogen für Lehrkräfte im Projekt
## AlgoKids – Algorithmen für Kinder

genehmigt am 12.11.2018, Aktenzeichen IV.8-BO7106/113/9

**Liebe Lehrerinnen und Lehrer,**

in diesem Fragebogen werden Informationen abgefragt und Daten erhoben, die wir für die Bewertung des Halbwochenlehrgangs sowie die Planung des weiteren Projektverlaufs benötigen. Selbstverständlich werden Ihre Angaben streng vertraulich behandelt und ausschließlich für wissenschaftliche und organisatorische Zwecke verwendet.

Das Ausfüllen des Fragebogens ist freiwillig. Sie haben außerdem die Möglichkeit, die Fragen nur teilweise zu beantworten. Für Rückfragen stehen wir Ihnen gerne zur Verfügung.

**Herzlichen Dank für Ihre Unterstützung!**

**Um die Angaben und Ergebnisse, die im Laufe des Projekts gesammelt werden, einander zuordnen zu können, verwenden wir einen persönlichen Code.** Damit Sie diesen nicht vergessen können, verwenden wir hierzu eine Kombination aus Buchstaben und Zahlen, die Sie sich jederzeit wieder herleiten können.

**Beispiel für einen Code:**

- Ihre Mutter heißt **Hi**lde
- Ihr Vater heißt **Fr**anz
- Ihr Geburtstag ist der **05**.08.1974

Als Personencode ergibt sich: **HI FR 05**

**Bitte geben Sie hier Ihren Personencode an:**

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| **1. und 2. Buchstabe des Vornamens Ihrer Mutter** | | **1. und 2. Buchstabe des Vornamens Ihres Vaters** | | **1. und 2. Ziffer Ihres Geburtstages** | |



## Zufriedenheit mit dem Halbwochenlehrgang

**1. Wie beurteilen Sie die Dauer der Veranstaltung?**

| viel zu kurz | zu kurz | angemessen | zu lang | viel zu lang |
|---|---|---|---|---|
| ☐ | ☐ | ☐ | ☐ | ☐ |

**2. Wie beurteilen Sie den Schwierigkeitsgrad der Fortbildungsveranstaltung?**

| viel zu leicht | eher zu leicht | angemessen | eher zu schwierig | viel zu schwierig |
|---|---|---|---|---|
| ☐ | ☐ | ☐ | ☐ | ☐ |

**3. Wurden Ihre Erwartungen hinsichtlich des Nutzens für die Umsetzung an der Schule erfüllt?**

| überhaupt nicht | kaum | teilweise | weitgehend | voll und ganz |
|---|---|---|---|---|
| ☐ | ☐ | ☐ | ☐ | ☐ |

**4. Die Unterlagen und Materialien…**

| | trifft überhaupt nicht zu | trifft eher nicht zu | teilweise | trifft eher zu | trifft voll und ganz zu |
|---|---|---|---|---|---|
| haben die Erarbeitung der Fortbildungsinhalte unterstützt. | ☐ | ☐ | ☐ | ☐ | ☐ |
| sind hilfreich für die Umsetzung der Fortbildungsinhalte an der Schule. | ☐ | ☐ | ☐ | ☐ | ☐ |

**5. Wie zufrieden sind Sie insgesamt mit der Fortbildung?**

| unzufrieden | eher unzufrieden | teilweise zufrieden | eher zufrieden | zufrieden |
|---|---|---|---|---|
| ☐ | ☐ | ☐ | ☐ | ☐ |

**6. Würden Sie diese Fortbildung an Kolleginnen/Kollegen weiterempfehlen?**

| nein | eher nein | teilweise | eher ja | ja |
|---|---|---|---|---|
| ☐ | ☐ | ☐ | ☐ | ☐ |

7. **Welche Inhalte oder Themen der Fortbildung halten Sie für überflüssig und würden diese eher weglassen?**

8. **Welche Inhalte oder Themen haben Sie in der Fortbildung vermisst?**



## Persönliche Reflexion der Fortbildungsinhalte

9. **Bitte geben Sie an, inwiefern die folgenden Aussagen zutreffen:**

| | trifft überhaupt nicht zu | trifft eher nicht zu | teil-weise | trifft eher zu | trifft voll und ganz zu |
|---|---|---|---|---|---|
| Ich kann einen Überblick über die Inhalte der Fortbildung geben. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich habe die Inhalte der Fortbildung verstanden. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich habe in der Fortbildung viel gelernt. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich nehme aus der Fortbildung ein erweitertes bzw. vertieftes Verständnis des Fortbildungs-themas mit. | ☐ | ☐ | ☐ | ☐ | ☐ |
| **Durch die Teilnahme an der Fortbildung ist es mir möglich…** | | | | | |
| die Fortbildungsinhalte an der Schule umzusetzen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| die Fortbildungsinhalte an die Anforderungen unterschiedlicher Situationen anzupassen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| die notwendigen Arbeitsschritte bei der Umsetzung des Gelernten systematisch zu planen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| geeignete Ziele bei der Umsetzung der Fortbildungsinhalte zu definieren. | ☐ | ☐ | ☐ | ☐ | ☐ |
| die Fortbildungsinhalte an Kolleginnen und Kollegen weiterzugeben. | ☐ | ☐ | ☐ | ☐ | ☐ |
| interessierte Kolleginnen und Kollegen bezüglich der Fortbildungsinhalte zu beraten. | ☐ | ☐ | ☐ | ☐ | ☐ |

## Selbsteinschätzung bezüglich der Umsetzung der Fortbildungsinhalte

**10. Bitte geben Sie an, inwiefern die folgenden Aussagen zutreffen:**

| | trifft überhaupt nicht zu | trifft eher nicht zu | teil-weise | trifft eher zu | trifft voll und ganz zu |
|---|---|---|---|---|---|
| Beim „unplugged"-Programmieren mit meinen Schülern fühle ich mich sicher. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich bin motiviert, die Fortbildungsinhalte auch bei Misserfolgen weiter zu verfolgen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich bin im Stande, eigene Unterrichtsszenarien zum Thema zu entwickeln. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich fühle mich kompetent, im Programmierunterricht auf die Fragen der Schüler einzugehen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Mit meinen Schülern am Computer zu programmieren, bereitet mir Unbehagen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich traue mir zu, meine Schüler für das Programmieren zu begeistern. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich weiß, dass ich es schaffe, selbst den problematischsten Schülern diesen Stoff zu vermitteln. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich bin mir sicher, dass ich auf die individuellen Probleme der Schüler beim Programmieren gut eingehen kann. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Selbst, wenn mein Programmier-unterricht gestört wird, bin ich mir sicher, die notwendige Gelassen-heit bewahren zu können. | ☐ | ☐ | ☐ | ☐ | ☐ |

**11. Gehen Sie davon aus, dass Sie bis zu den Sommerferien die Gelegenheit haben werden, die Fortbildungsinhalte anzuwenden?**

| ja | weiß nicht | nein |
|---|---|---|
| ☐ | ☐ | ☐ |

**12. Gehen Sie davon aus, dass Sie im nächsten Schuljahr die Gelegenheit haben werden, die Fortbildungsinhalte anzuwenden?**

| ja | weiß nicht | nein |
|---|---|---|
| ☐ | ☐ | ☐ |



## Selbsteinschätzung der Programmierfähigkeiten

**13. Bitte geben Sie an, inwiefern die folgenden Aussagen zutreffen:**

| | trifft überhaupt nicht zu | trifft eher nicht zu | teil-weise | trifft eher zu | trifft voll und ganz zu |
|---|---|---|---|---|---|
| Ich kann die grundlegende logische Struktur eines Programms verstehen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann eine bedingte Anweisung wie "falls…sonst..." verstehen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann das Endergebnis eines Programms mit bedingten Anweisungen vorhersagen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann eine Wiederholungs-anweisung wie „wiederhole…bis…" verstehen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann das Endergebnis eines Programms mit Wiederholungen vorhersagen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann das Ergebnis eines Programms vorhersagen, wenn mir die Eingabewerte vorliegen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich weiß, dass sich Programmierarbeiten in Teilaufgaben für verschiedene Menschen unterteilen lassen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann mit anderen zusammenarbeiten, wenn ich ein Programm schreibe. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann Programmieraufgaben unterteilen, um die Erstellung des Programms auf mehrere Programmierer zu verteilen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann Programmabläufe ohne Beispielablauf herausfinden. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich brauche keine Hilfe von anderen, um ein Programm zu erstellen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann Programmierung nutzen, um ein Problem zu lösen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann ein Programm in einer Programmierumgebung öffnen und speichern. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann ein Programm in einer Programmierumgebung bearbeiten und überarbeiten. | ☐ | ☐ | ☐ | ☐ | ☐ |

| | trifft überhaupt nicht zu | trifft eher nicht zu | teil-weise | trifft eher zu | trifft voll und ganz zu |
|---|---|---|---|---|---|
| Ich kann ein Programm in einer Programmierumgebung ausführen und testen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann die Ursache eines Fehlers beim Testen eines Programms finden. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann einen Fehler beim Testen eines Programms beheben. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann durch die Fehlersuche und -behebung besser Programmieren lernen. | ☐ | ☐ | ☐ | ☐ | ☐ |



## Mitwirkung an der Gestaltung des weiteren Projektverlaufs

In der Gestaltung des vierten regionalen Fortbildungstages und der weiterführenden Online-Materialien sind wir flexibel bzw. gibt es noch Freiräume. Gerne würden wir hier auf Ihre Bedürfnisse und Anregungen eingehen.

**14. In welchen Bereichen gibt es inhaltliche Unsicherheiten oder offene Fragen, auf die wir weiter eingehen sollen?**

## Herzlichen Dank für Ihre Angaben!

### A.3.3 Fragebogen zu Beginn der zweiten Fortbildungsveranstaltung (FB-LK3)

Technische Universität München
TUM School of Education
Professur für Didaktik der Informatik
Prof. Dr. Peter Hubwieser, Katharina Geldreich

# Fragebogen für Lehrkräfte im Projekt
## AlgoKids – Algorithmen für Kinder

genehmigt am 12.11.2018, Aktenzeichen IV.8-BO7106/113/9

**Liebe Lehrerinnen und Lehrer,**

in diesem Fragebogen werden Informationen abgefragt und Daten erhoben, die wir für die Bewertung des bisherigen Projektverlaufs sowie die weitere Planung von AlgoKids benötigen. Selbstverständlich werden Ihre Angaben streng vertraulich behandelt und ausschließlich für wissenschaftliche und organisatorische Zwecke verwendet.

Das Ausfüllen des Fragebogens ist freiwillig. Sie haben außerdem die Möglichkeit, die Fragen nur teilweise zu beantworten. Für Rückfragen stehen wir Ihnen gerne zur Verfügung.

**Herzlichen Dank für Ihre Unterstützung!**

**Um die Angaben und Ergebnisse, die im Laufe des Projekts gesammelt werden, einander zuordnen zu können, verwenden wir einen persönlichen Code.** Damit Sie diesen nicht vergessen können, verwenden wir hierzu eine Kombination aus Buchstaben und Zahlen, die Sie sich jederzeit wieder herleiten können.

**Beispiel für einen Code:**

- Ihre Mutter heißt **Hi**lde
- Ihr Vater heißt **Fr**anz
- Ihr Geburtstag ist der **05**.08.1974

Als Personencode ergibt sich: **HI FR 05**

**Bitte geben Sie hier Ihren Personencode an:**

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| **1. und 2. Buchstabe des Vornamens Ihrer Mutter** | | **1. und 2. Buchstabe des Vornamens Ihres Vaters** | | **1. und 2. Ziffer Ihres Geburtstages** | |



## Umsetzung im Unterricht

1. **Hatten Sie bereits die Gelegenheit, die Fortbildungsinhalte im Unterricht anzuwenden?**

   ja ☐ nein ☐

2. **Gehen Sie davon aus, dass Sie im weiteren Verlauf des Schuljahres die Gelegenheit haben werden, die Fortbildungsinhalte anzuwenden?**

   ja ☐ weiß nicht ☐ nein ☐

## Selbsteinschätzung bezüglich der Umsetzung der AlgoKids-Fortbildungsinhalte

**3. Bitte geben Sie an, inwiefern die folgenden Aussagen zutreffen:**

| | trifft überhaupt nicht zu | trifft eher nicht zu | teil-weise | trifft eher zu | trifft voll und ganz zu |
|---|---|---|---|---|---|
| Beim „unplugged"-Programmieren mit meinen Schülern fühle ich mich sicher. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich bin motiviert, die Fortbildungsinhalte auch bei Misserfolgen weiter zu verfolgen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich bin im Stande, eigene Unterrichtsszenarien zum Thema zu entwickeln. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich fühle mich kompetent, im Programmierunterricht auf die Fragen der Schüler einzugehen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Mit meinen Schülern am Computer zu programmieren, bereitet mir Unbehagen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich traue mir zu, meine Schüler für das Programmieren zu begeistern. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich weiß, dass ich es schaffe, selbst den problematischsten Schülern diesen Stoff zu vermitteln. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich bin mir sicher, dass ich auf die individuellen Probleme der Schüler beim Programmieren gut eingehen kann. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Selbst, wenn mein Programmier-unterricht gestört wird, bin ich mir sicher, die notwendige Gelassen-heit bewahren zu können. | ☐ | ☐ | ☐ | ☐ | ☐ |



## Selbsteinschätzung der Programmierfähigkeiten

**4. Bitte geben Sie an, inwiefern die folgenden Aussagen zutreffen:**

| | trifft überhaupt nicht zu | trifft eher nicht zu | teil-weise | trifft eher zu | trifft voll und ganz zu |
|---|---|---|---|---|---|
| Ich kann die grundlegende logische Struktur eines Programms verstehen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann eine bedingte Anweisung wie "falls…sonst..." verstehen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann das Endergebnis eines Programms mit bedingten Anweisungen vorhersagen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann eine Wiederholungs-anweisung wie „wiederhole…bis…" verstehen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann das Endergebnis eines Programms mit Wiederholungen vorhersagen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann das Ergebnis eines Programms vorhersagen, wenn mir die Eingabewerte vorliegen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich weiß, dass sich Programmierarbeiten in Teilaufgaben für verschiedene Menschen unterteilen lassen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann mit anderen zusammenarbeiten, wenn ich ein Programm schreibe. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann Programmieraufgaben unterteilen, um die Erstellung des Programms auf mehrere Programmierer zu verteilen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann Programmabläufe ohne Beispielablauf herausfinden. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich brauche keine Hilfe von anderen, um ein Programm zu erstellen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann Programmierung nutzen, um ein Problem zu lösen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann ein Programm in einer Programmierumgebung öffnen und speichern. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann ein Programm in einer Programmierumgebung bearbeiten und überarbeiten. | ☐ | ☐ | ☐ | ☐ | ☐ |

| | trifft überhaupt nicht zu | trifft eher nicht zu | teil-weise | trifft eher zu | trifft voll und ganz zu |
|---|---|---|---|---|---|
| Ich kann ein Programm in einer Programmierumgebung ausführen und testen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann die Ursache eines Fehlers beim Testen eines Programms finden. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann einen Fehler beim Testen eines Programms beheben. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann durch die Fehlersuche und -behebung besser Programmieren lernen. | ☐ | ☐ | ☐ | ☐ | ☐ |

## Herzlichen Dank für Ihre Angaben!

### A.3.4 Fragebogen zum Ende der dritten Fortbildungsveranstaltung bzw. zum *AlgoKids*-Projektende (FB-LK4)

Technische Universität München
TUM School of Education
Professur für Didaktik der Informatik
Prof. Dr. Peter Hubwieser, Katharina Geldreich

# Fragebogen für Lehrkräfte im Projekt
## AlgoKids – Algorithmen für Kinder

genehmigt am 12.11.2018, Aktenzeichen IV.8-BO7106/113/9

**Liebe Lehrerinnen und Lehrer,**

in diesem Fragebogen werden Informationen abgefragt und Daten erhoben, die wir für die Bewertung des Halbwochenlehrgangs sowie die Planung des weiteren Projektverlaufs benötigen. Selbstverständlich werden Ihre Angaben streng vertraulich behandelt und ausschließlich für wissenschaftliche und organisatorische Zwecke verwendet.

Das Ausfüllen des Fragebogens ist freiwillig. Sie haben außerdem die Möglichkeit, die Fragen nur teilweise zu beantworten. Für Rückfragen stehen wir Ihnen gerne zur Verfügung.

**Herzlichen Dank für Ihre Unterstützung!**

**Um die Angaben und Ergebnisse, die im Laufe des Projekts gesammelt werden, einander zuordnen zu können, verwenden wir einen persönlichen Code.** Damit Sie diesen nicht vergessen können, verwenden wir hierzu eine Kombination aus Buchstaben und Zahlen, die Sie sich jederzeit wieder herleiten können.

**Beispiel für einen Code:**

- Ihre Mutter heißt **Hi**lde
- Ihr Vater heißt **Fr**anz
- Ihr Geburtstag ist der **05**.08.1974

Als Personencode ergibt sich: **HI FR 05**

**Bitte geben Sie hier Ihren Personencode an:**

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| **1. und 2. Buchstabe des Vornamens Ihrer Mutter** | | **1. und 2. Buchstabe des Vornamens Ihres Vaters** | | **1. und 2. Ziffer Ihres Geburtstages** | |



## Selbsteinschätzung bezüglich der Umsetzung der Fortbildungsinhalte

**1. Bitte geben Sie an, inwiefern die folgenden Aussagen zutreffen:**

| | trifft überhaupt nicht zu | trifft eher nicht zu | teil-weise | trifft eher zu | trifft voll und ganz zu |
|---|---|---|---|---|---|
| Beim „unplugged"-Programmieren mit meinen Schülern fühle ich mich sicher. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich bin motiviert, die Projektinhale auch bei Misserfolgen weiter zu verfolgen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich bin im Stande, eigene Unterrichtsszenarien zum Thema *Algorithmik und Programmierung* zu entwickeln. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich fühle mich kompetent, im Programmierunterricht auf die Fragen der Schüler einzugehen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Mit meinen Schülern am Computer zu programmieren, bereitet mir Unbehagen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich traue mir zu, meine Schüler für das Programmieren zu begeistern. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich weiß, dass ich es schaffe, selbst den problematischsten Schülern diesen Stoff zu vermitteln. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich bin mir sicher, dass ich auf die individuellen Probleme der Schüler beim Programmieren gut eingehen kann. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Selbst, wenn mein Programmier-unterricht gestört wird, bin ich mir sicher, die notwendige Gelassen-heit bewahren zu können. | ☐ | ☐ | ☐ | ☐ | ☐ |

## Selbsteinschätzung der Programmierfähigkeiten

**2. Bitte geben Sie an, inwiefern die folgenden Aussagen zutreffen:**

| | trifft überhaupt nicht zu | trifft eher nicht zu | teil-weise | trifft eher zu | trifft voll und ganz zu |
|---|---|---|---|---|---|
| Ich kann die grundlegende logische Struktur eines Programms verstehen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann eine bedingte Anweisung wie "falls…sonst…" verstehen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann das Endergebnis eines Programms mit bedingten Anweisungen vorhersagen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann eine Wiederholungs-anweisung wie „wiederhole…bis…“ verstehen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann das Endergebnis eines Programms mit Wiederholungen vorhersagen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann das Ergebnis eines Programms vorhersagen, wenn mir die Eingabewerte vorliegen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich weiß, dass sich Programmierarbeiten in Teilaufgaben für verschiedene Menschen unterteilen lassen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann mit anderen zusammenarbeiten, wenn ich ein Programm schreibe. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann Programmieraufgaben unterteilen, um die Erstellung des Programms auf mehrere Programmierer zu verteilen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann Programmabläufe ohne Beispielablauf herausfinden. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich brauche keine Hilfe von anderen, um ein Programm zu erstellen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann Programmierung nutzen, um ein Problem zu lösen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann ein Programm in einer Programmierumgebung öffnen und speichern. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann ein Programm in einer Programmierumgebung bearbeiten und überarbeiten. | ☐ | ☐ | ☐ | ☐ | ☐ |



| | trifft überhaupt nicht zu | trifft eher nicht zu | teil-weise | trifft eher zu | trifft voll und ganz zu |
|---|---|---|---|---|---|
| Ich kann ein Programm in einer Programmierumgebung ausführen und testen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann die Ursache eines Fehlers beim Testen eines Programms finden. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann einen Fehler beim Testen eines Programms beheben. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich kann durch die Fehlersuche und -behebung besser Programmieren lernen. | ☐ | ☐ | ☐ | ☐ | ☐ |

## Umsetzung der Fortbildungsinhalte

**3. Bitte geben Sie an, inwiefern die folgenden Aussagen zutreffen:**

| | trifft überhaupt nicht zu | trifft eher nicht zu | teil-weise | trifft eher zu | trifft voll und ganz zu |
|---|---|---|---|---|---|
| **Ich habe die Fortbildungsinhalte in der Projektlaufzeit …** | | | | | |
| in mein schulisches Handeln integrieren können. | ☐ | ☐ | ☐ | ☐ | ☐ |
| als Ausgangspunkt zur Entwicklung eigener Ideen genutzt. | ☐ | ☐ | ☐ | ☐ | ☐ |
| für die Anwendung in meinem Schulalltag weiterentwickelt. | ☐ | ☐ | ☐ | ☐ | ☐ |
| an die Anforderungen unterschiedlicher Umsetzungs-situationen angepasst. | ☐ | ☐ | ☐ | ☐ | ☐ |
| in der Schule angewandt. | ☐ | ☐ | ☐ | ☐ | ☐ |
| ohne große Veränderungen anwenden können. | ☐ | ☐ | ☐ | ☐ | ☐ |
| **Die Fortbildungsinhalte…** | | | | | |
| sind ein fester Bestandteil meines Arbeitsalltags geworden. | ☐ | ☐ | ☐ | ☐ | ☐ |
| habe ich so oft wie möglich in meinem Arbeitsalltag umgesetzt. | ☐ | ☐ | ☐ | ☐ | ☐ |

**4. Bitte geben Sie an, inwieweit die Umsetzung der Fortbildungsinhalte Auswirkungen auf Ihre Arbeitsbelastung hatte:**

| deutlich geringere Arbeitsbelastung | etwas geringere Arbeitsbelastung | keine Auswirkungen | etwas höhere Arbeitsbelastung | deutlich höhere Arbeitsbelastung |
|---|---|---|---|---|
| ☐ | ☐ | ☐ | ☐ | ☐ |



**5. Ich habe die Materialien aus der Fortbildung…**

| | trifft überhaupt nicht zu | trifft eher nicht zu | teil-weise | trifft eher zu | trifft voll und ganz zu |
|---|---|---|---|---|---|
| ohne große Veränderungen im Unterricht anwenden können. | ☐ | ☐ | ☐ | ☐ | ☐ |
| in meiner schulischen Arbeit benutzt. | ☐ | ☐ | ☐ | ☐ | ☐ |
| methodisch weiterentwickelt. | ☐ | ☐ | ☐ | ☐ | ☐ |
| inhaltlich weiterentwickelt. | ☐ | ☐ | ☐ | ☐ | ☐ |

**6. Die Umsetzung der Fortbildungsinhalte wurde erschwert, da folgende Ressourcen nicht (ausreichend) zur Verfügung standen:**

| | ja | nein | nicht relevant |
|---|---|---|---|
| Zeit | ☐ | ☐ | ☐ |
| Ausstattung | ☐ | ☐ | ☐ |
| Lehr-Lern-Materialien | ☐ | ☐ | ☐ |
| Räumlichkeiten | ☐ | ☐ | ☐ |
| finanzielle Mittel | ☐ | ☐ | ☐ |

**Sonstiges (bitte benennen):** _____

**7. Bitte geben Sie an, inwiefern folgende Aussagen zutreffen:**

| | trifft überhaupt nicht zu | trifft eher nicht zu | teil-weise | trifft eher zu | trifft voll und ganz zu |
|---|---|---|---|---|---|
| Ich habe gemeinsam mit Kolleginnen/Kollegen die Fortbildungsinhalte umgesetzt. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich habe gemeinsam mit Kolleginnen/Kollegen die Anwendung der Fortbildungs-inhalte geplant. | ☐ | ☐ | ☐ | ☐ | ☐ |

## Persönliche Reflexion

**8. Bitte geben Sie an, inwiefern die folgenden Aussagen zutreffen:**

| | trifft überhaupt nicht zu | trifft eher nicht zu | teil-weise | trifft eher zu | trifft voll und ganz zu |
|---|---|---|---|---|---|
| Ich war motiviert, die Inhalte der Fortbildung umzusetzen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| **Welche Einstellungen haben Sie in der Zeit nach der Fortbildung zu Fortbildungsinhalten entwickelt?** | | | | | |
| Ich bin vom Nutzen der Fortbildungsinhalte überzeugt. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich denke, dass die Anwendung der Fortbildungsinhalte sinnvoll ist. | ☐ | ☐ | ☐ | ☐ | ☐ |
| **Die Umsetzung der Fortbildungsinhalte…** | | | | | |
| fiel mir leicht. | ☐ | ☐ | ☐ | ☐ | ☐ |
| machte mir Freude. | ☐ | ☐ | ☐ | ☐ | ☐ |
| war erfolgreich. | ☐ | ☐ | ☐ | ☐ | ☐ |



**9. Durch die Umsetzung der Fortbildungsinhalte konnte ich positive Auswirkungen feststellen in Bezug auf… (mehrere Kreuze möglich)**

| **Schüler/Klasse** | | **Ihre eigene Person** | |
|---|---|---|---|
| Kompetenzentwicklung der Schülerinnen und Schüler | ☐ | Ihre Professionalisierung | ☐ |
| Lernkultur der Klasse | ☐ | Ihre Motivation | ☐ |
| Soziales Klima in der Klasse | ☐ | Ihre Arbeitszufriedenheit | ☐ |
| **Unterricht** | | **Schule** | |
| Methodik | ☐ | Schulkonzept/Leitbild | ☐ |
| Didaktik | ☐ | Schulentwicklung | ☐ |
| (Fach)Inhalte | ☐ | Profil der Schule | ☐ |
| Lehrer-Schüler-Interaktion | ☐ | soziales Klima an der Schule | ☐ |
| individualisierte Lernprozesse | ☐ | | |

## Herzlichen Dank für Ihre Angaben!

## A.4 Verwendete Skalen[2]

*Sicherheit im Umgang mit Computern und Computeranwendungen (FB-LK1, FB-LK4)*

| Itemlabel | Text | Cronb. Alpha |
|---|---|---|
| sich_comp_01_I | Im Umgang mit Computern fühle ich mich sicher. | 0.88 |
| sich_comp_02 | Wenn mein Computer abstürzt, gerate ich in Panik. | |
| sich_comp_03 | Bei der Arbeit mit dem Computer lasse ich mich durch auftretende Schwierigkeiten leicht frustrieren. | |
| sich_comp_04 | Das Arbeiten am Computer bereitet mir Unbehagen. | |
| sich_comp_05 | Beim Arbeiten mit dem Computer habe ich oft Angst, etwas kaputt zu machen. | |
| sich_comp_06_I | Ich habe das Gefühl, dass ich meinen Computer im Griff habe. | |
| sich_comp_07 | Wenn möglich, vermeide ich das Arbeiten am Computer. | |
| sich_comp_08_I | Auch bei auftretenden Computerproblemen bleibe ich ruhig. | |
| Fünfstufiges Antwortformat:<br>stimme nicht zu (1) – stimme eher zu (2) – neutral (3) – stimme eher zu (4) – stimme zu (5) | | |
| Anmerkung:<br>Die Computerängstlichkeitsskala (COMA) wurde aus der revidierten Fassung des Inventars zur Computerbildung (INCOBI-R) von Richter, Naumann und Horz (2010) entnommen. Die subjektive Sicherheit im Umgang mit dem Computer und Computeranwendungen wird darin als Abwesenheit von Computerängstlichkeit definiert (ebd., S. 24). | | |

*Qualität von erhaltenen Unterlagen (FB-LK2)*

| Itemlabel | Text | Cronb. Alpha |
|---|---|---|
| | Die Unterlagen und Materialien … | 0.83 |
| mat_01 | haben die Erarbeitung der Fortbildungsinhalte unterstützt. | |
| mat_02 | sind hilfreich für die Umsetzung der Fortbildungsinhalte an der Schule. | |
| Fünfstufiges Antwortformat:<br>trifft überhaupt nicht zu (1) - trifft eher nicht zu (2) – teilweise (3) - trifft eher zu (4) - trifft voll und ganz zu (5) | | |
| Anmerkung:<br>Die Skala entstammt einem Instrument zur Evaluation von Fortbildungsveranstaltungen für Lehrkräfte von Vigerske (2017). | | |

[2] Es werden nur die Skalen angegeben, die für die vorliegende Arbeit relevant sind. Invers formulierte Item sind mit „_I“ gekennzeichnet. Für die Berechnung des Mittelwerts der summierten Items der Skala müssen diese invertiert werden.

*Zufriedenheit (FB-LK2)*

| Itemlabel | Text | Cronb. Alpha |
|---|---|---|
| zuf_01 | Wie zufrieden sind Sie insgesamt mit der Fortbildung? | 0.91 |
| Fünfstufiges Antwortformat:<br>unzufrieden (1) - eher unzufrieden (2) - teilweise unzufrieden (3) - eher zufrieden (4) - zufrieden (5) | | |
| zuf_02 | Würden Sie diese Fortbildung an Kolleginnen/Kollegen weiterempfehlen? | |
| Fünfstufiges Antwortformat:<br>nein (1) - eher nein (2) - teilweise (3) - eher ja (4) - ja (5) | | |
| Anmerkung:<br>Die Skala entstammt einem Instrument zur Evaluation von Fortbildungsveranstaltungen für Lehrkräfte von Vigerske (2017). | | |

*Wissenszuwachs (FB-LK2)*

| Itemlabel | Text | Cronb. Alpha |
|---|---|---|
| wz_01 | Ich kann einen Überblick über die Inhalte der Fortbildung geben. | 0.72 |
| wz_02 | Ich habe die Inhalte der Fortbildung verstanden. | |
| wz_03 | Ich habe in der Fortbildung viel gelernt. | |
| wz_04 | Ich nehme aus der Fortbildung ein erweitertes bzw. vertieftes Verständnis des Fortbildungsthemas mit. | |
| Fünfstufiges Antwortformat:<br>trifft überhaupt nicht zu (1) - trifft eher nicht zu (2) - teilweise (3) - trifft eher zu (4) - trifft voll und ganz zu (5) | | |
| Anmerkung:<br>Die Skala entstammt einem Instrument zur Evaluation von Fortbildungsveranstaltungen für Lehrkräfte von Vigerske (2017). | | |

*Befähigung zur Anwendung der Fortbildungsinhalte (FB-LK2)*

| Itemlabel | Text | Cronb. Alpha |
|---|---|---|
| | Durch die Teilnahme an der Fortbildung ist es mir möglich ... | 0.88 |
| bef_anw_01 | die Fortbildungsinhalte an der Schule umzusetzen. | |
| bef_anw_02 | die Fortbildungsinhalte an die Anforderungen unterschiedlicher Situationen anzupassen. | |
| bef_anw_03 | die notwendigen Arbeitsschritte bei der Umsetzung des Gelernten systematisch zu planen. | |
| bef_anw_04 | geeignete Ziele bei der Umsetzung der Fortbildungsinhalte zu definieren. | |
| Fünfstufiges Antwortformat:<br>trifft überhaupt nicht zu (1) - trifft eher nicht zu (2) - teilweise (3) - trifft eher zu (4) - trifft voll und ganz zu (5) | | |
| Anmerkung:<br>Die Skala entstammt einem Instrument zur Evaluation von Fortbildungsveranstaltungen für Lehrkräfte von Vigerske (2017). | | |

*Befähigung zur Weitergabe der Fortbildungsinhalte im Kollegium (FB-LK2)*

| Itemlabel | Text | Cronb. Alpha |
|---|---|---|
| | Durch die Teilnahme an der Fortbildung ist es mir möglich … | 0.83 |
| mat_01 | die Fortbildungsinhalte an Kolleginnen und Kollegen weiterzugeben. | |
| mat_02 | interessierte Kolleginnen und Kollegen bezüglich der Fortbildungsinhalte zu beraten. | |
| Fünfstufiges Antwortformat:<br>trifft überhaupt nicht zu (1) - trifft eher nicht zu (2) - teilweise (3) - trifft eher zu (4) - trifft voll und ganz zu (5) | | |
| Anmerkung:<br>Die Skala entstammt einem Instrument zur Evaluation von Fortbildungsveranstaltungen für Lehrkräfte von Vigerske (2017). | | |

*Selbsteinschätzung bzgl. der Umsetzung der Fortbildungsinhalte (FB-LK2, FB-LK3, FB-LK4)*

| Itemlabel | Text | Cronb. Alpha |
|---|---|---|
| sw_01 | Beim *unplugged*-Programmieren mit meinen Schülern fühle ich mich sicher. | MZP 1: 0.806<br>MZP 2: 0.873<br>MZP 3: 0.792 |
| sw_02 | Ich bin motiviert, die Fortbildungsinhalte auch bei Misserfolgen weiter zu verfolgen. | |
| sw_03 | Ich bin im Stande, eigene Unterrichtsszenarien zum Thema zu entwickeln. | |
| sw_04 | Ich fühle mich kompetent, im Programmierunterricht auf die Fragen der Schüler einzugehen. | |
| sw_05_I | Mit meinen Schülern am Computer zu programmieren, bereitet mir Unbehagen. | |
| sw_06 | Ich traue mir zu, meine Schüler für das Programmieren zu begeistern. | |
| sw_07 | Ich weiß, dass ich es schaffe, selbst den problematischsten Schülern diesen Stoff zu vermitteln. | |
| sw_08 | Ich bin mir sicher, dass ich auf die individuellen Probleme der Schüler beim Programmieren gut eingehen kann. | |
| sw_09 | Selbst, wenn mein Programmierunterricht gestört wird, bin ich mir sicher, die notwendige Gelassenheit bewahren zu können. | |
| Fünfstufiges Antwortformat:<br>stimme nicht zu (1) – stimme eher zu (2) – neutral (3) – stimme eher zu (4) – stimme zu (5) | | |
| Anmerkung:<br>Die Skala wurde in Anlehnung an die Skala zur Lehrer-Selbstwirksamkeitserwartung (SWLE) von Jerusalem u. a. (2009) konstruiert. | | |

*Selbsteinschätzung der Programmierfähigkeiten (FB-LK2, FB-LK3, FB-LK4)*

| Itemlabel | Text | Cronb. Alpha |
|---|---|---|
| | Subskala *Logical Thinking* | |
| ssch_prog_01 | Ich kann die grundlegende logische Struktur eines Programms verstehen. | MZP 1: 0.805 |
| ssch_prog_02 | Ich kann eine bedingte Anweisung wie „falls…sonst ...“ verstehen. | MZP 2: 0.811 |
| ssch_prog_03 | Ich kann das Endergebnis eines Programms mit bedingten Anweisungen vorhersagen. | MZP 3: 0.770 |
| ssch_prog_04 | Ich kann eine Wiederholungsanweisung wie „wiederhole ... bis ...“ verstehen. | |
| ssch_prog_05 | Ich kann das Endergebnis eines Programms mit Wiederholungen vorhersagen. | |
| ssch_prog_06 | Ich kann das Ergebnis eines Programms vorhersagen, wenn mir die Eingabewerte vorliegen. | |
| | Subskala *Cooperation* | |
| ssch_prog_07 | Ich weiß, dass sich Programmierarbeiten in Teilaufgaben für verschiedene Menschen unterteilen lassen. | MZP 1: 0.497 |
| ssch_prog_08 | Ich kann mit anderen zusammenarbeiten, wenn ich ein Programm schreibe. | |
| ssch_prog_09 | Ich kann Programmieraufgaben unterteilen, um die Erstellung des Programms auf mehrere Programmierer zu verteilen. | |
| | Subskala *Algorithm* | |
| ssch_prog_10 | Ich kann Programmabläufe ohne Beispielablauf herausfinden. | MZP 1: 0.742 |
| ssch_prog_11 | Ich brauche keine Hilfe von anderen, um ein Programm zu erstellen. | MZP 2: 0.805 |
| ssch_prog_12 | Ich kann Programmierung nutzen, um ein Problem zu lösen. | MZP 3: 0.844 |
| | Subskala *Control* | |
| ssch_prog_13 | Ich kann ein Programm in einer Programmierumgebung öffnen und speichern. | MZP 1: 0.818 |
| ssch_prog_14 | Ich kann ein Programm in einer Programmierumgebung bearbeiten und überarbeiten. | MZP 2: 0.805 |
| ssch_prog_15 | Ich kann ein Programm in einer Programmierumgebung ausführen und testen. | MZP 3: 0.747 |
| | Subskala *Debug* | Spearman-Brown-Koeffizient |
| ssch_prog_16 | Ich kann die Ursache eines Fehlers beim Testen eines Programms finden. | MZP 1: 0.830 |
| ssch_prog_17 | Ich kann einen Fehler beim Testen eines Programms beheben. | MZP 2: 0.890 |
| ssch_prog_18 | Ich kann durch die Fehlersuche und -behebung besser Programmieren lernen. | MZP 3: 0.890 |
| Fünfstufiges Antwortformat:<br>stimme nicht zu (1) – stimme eher zu (2) – neutral (3) – stimme eher zu (4) – stimme zu (5) | | |
| Anmerkung:<br>Die Skala zur Einschätzung der Programmierfähigkeit (CPSES: *Computer Programming Self-Efficacy Scale*) wurde von Tsai, Wang und Hsu (2017) entwickelt und vor dem Einsatz in dieser Arbeit ins Deutsche übersetzt. Es wurden zwei projektspezifische Items ergänzt (ssch_prog_04, ssch_prog_5). Da die Subskala *Cooperation* mit drei Items zum ersten Messzeitpunkt nur einen inakzeptablen Wert von 0.497 aufwies, wurde die Subskala mit den Items ssch_prog_07-09 für die Auswertungen nicht berücksichtigt. In der Subskala *Debug* konnte mit drei Items ein Wert von Cronbachs Alpha von 0.597 bestimmt werden. Daraufhin wurde Item ssch_prog_18 entfernt und die Reliabilität mithilfe des Spearman-Brown-Koeffizienten $\rho$ berechnet, der sich für die Berechnung der Reliabilität von Skalen mit nur zwei Items eignet. | | |

*Anwendung (FB-LK4)*

| Itemlabel | Text | Cronb. Alpha |
|---|---|---|
| | Ich habe die Fortbildungsinhalte in der Projektlaufzeit ... | 0.92 |
| anw_01 | in mein schulisches Handeln integrieren können. | |
| anw_02 | in der Schule angewandt. | |
| anw_03 | ohne große Veränderungen anwenden können. | |
| | Die Fortbildungsinhalte ... | |
| anw_04 | sind ein fester Bestandteil meines Arbeitsalltags geworden. | |
| anw_05 | habe ich so oft wie möglich in meinem Arbeitsalltag umgesetzt. | |
| Fünfstufiges Antwortformat:<br>trifft überhaupt nicht zu (1) - trifft eher nicht zu (2) - teilweise (3) - trifft eher zu (4) - trifft voll und ganz zu (5) | | |
| Anmerkung:<br>Die Skala entstammt einem Instrument zur Evaluation von Fortbildungsveranstaltungen für Lehrkräfte von Vigerske (2017). | | |

*Anpassung (FB-LK4)*

| Itemlabel | Text | Cronb. Alpha |
|---|---|---|
| | Ich habe die Fortbildungsinhalte in der Projektlaufzeit ... | 0.83 |
| anp_01 | als Ausgangspunkt zur Entwicklung eigener Ideen genutzt. | |
| anp_02 | für die Anwendung in meinem Schulalltag weiterentwickelt. | |
| anp_03 | an die Anforderungen unterschiedlicher Umsetzungssituationen angepasst. | |
| Fünfstufiges Antwortformat:<br>trifft überhaupt nicht zu (1) - trifft eher nicht zu (2) - teilweise (3) - trifft eher zu (4) - trifft voll und ganz zu (5) | | |
| Anmerkung:<br>Die Skala entstammt einem Instrument zur Evaluation von Fortbildungsveranstaltungen für Lehrkräfte von Vigerske (2017). | | |

*Verwendung der Fortbildungsmaterialien (FB-LK4)*

| Itemlabel | Text | Cronb. Alpha |
|---|---|---|
| | Ich habe die Materialien aus der Fortbildung… | |
| | Subskala Weiterentwicklung/Anpassung | |
| ma_anw_01 | ohne große Veränderungen im Unterricht anwenden können. | 0.88 |
| ma_anw_02 | in meiner schulischen Arbeit benutzt. | |
| | Subskala Anwendung | |
| ma_anp_01 | methodisch weiterentwickelt. | 0.87 |
| ma_anp_02 | inhaltlich weiterentwickelt. | |
| Fünfstufiges Antwortformat:<br>stimme nicht zu (1) – stimme eher zu (2) – neutral (3) – stimme eher zu (4) – stimme zu (5) | | |
| Anmerkung:<br>Die Skala entstammt einem Instrument zur Evaluation von Fortbildungsveranstaltungen für Lehrkräfte von Vigerske (2017). | | |

*Motivation und Überzeugung (FB-LK4)*

| Itemlabel | Text | Cronb. Alpha |
|---|---|---|
| mot_zeug_01 | Ich war motiviert, die Inhalte der Fortbildung umzusetzen. | 0.89 |
| mot_zeug_02 | Ich bin vom Nutzen der Fortbildungsinhalte überzeugt. | |
| mot_zeug_03 | Ich denke, dass die Anwendung der Fortbildungsinhalte sinnvoll ist. | |
| Fünfstufiges Antwortformat:<br>trifft überhaupt nicht zu (1) - trifft eher nicht zu (2) - teilweise (3) - trifft eher zu (4) - trifft voll und ganz zu (5) | | |
| Anmerkung:<br>Die Skala entstammt einem Instrument zur Evaluation von Fortbildungsveranstaltungen für Lehrkräfte von Vigerske (2017). | | |

*Empfinden der Umsetzung (FB-LK4)*

| Itemlabel | Text | Cronb. Alpha |
|---|---|---|
| | Die Umsetzung der Fortbildungsinhalte ... | 0.83 |
| empf_01 | fiel mir leicht. | |
| empf_02 | machte mir Freude. | |
| empf_03 | war erfolgreich. | |
| Fünfstufiges Antwortformat:<br>trifft überhaupt nicht zu (1) - trifft eher nicht zu (2) - teilweise (3) - trifft eher zu (4) - trifft voll und ganz zu (5) | | |
| Anmerkung:<br>Die Skala entstammt einem Instrument zur Evaluation von Fortbildungsveranstaltungen für Lehrkräfte von Vigerske (2017). | | |

# B Unterrichtssequenz

## B.1 Didaktische Konzeption

### B.1.1 Herleitung der unterrichtlichen Handlungslinie

*Tabelle B.1 Tabellarische Auflistung der einzelnen Unterrichtsschritte inklusive der Herleitung aus den Umsetzungsprinzipien und der Verbindung zu den Grobzielen der Unterrichtssequenz*

| Element der unterrichtlichen Handlungslinie | Begründung (aus den Umsetzungsprinzipien der *Design-Principles* mit Bezug zu Lernaktivitäten) | Grobziel |
|---|---|---|
| (1) Thematischer Einstieg | | |
| 1a) Aktivierung von Vorwissen der Schülerinnen und Schüler | DP 2: „In der Unterrichtssequenz wird an den Alltag und das Vorwissen der Schülerinnen und Schüler angeknüpft, …“ | |
| (2) Grundlagen von Algorithmen und Programmen | | |
| 2a) Formulieren von ersten Anweisungen und Verständigung über deren Eigenschaften | DP 1: „Zunächst sollen die Schülerinnen und Schüler dem Programmieren und den algorithmischen Grundstrukturen durch eigenständig ausgeführte Handlungen begegnen.“<br><br>DP 4: „Die Schülerinnen und Schüler nehmen von Beginn an eine aktive Rolle im Unterrichtsgeschehen ein.“ | Die Schülerinnen und Schüler führen Algorithmen in ihrer Lebenswelt aus<br><br>Die Schülerinnen und Schüler beschreiben Algorithmen in ihrer Alltagssprache |
| 2b) Herausarbeiten des Unterschieds zur natürlichen Sprache | DP 1: „Zunächst sollen die Schülerinnen und Schüler dem Programmieren und den algorithmischen Grundstrukturen durch eigenständig ausgeführte Handlungen begegnen. In einem nächsten Schritt durch graphische und bildliche Darstellungen und zuletzt durch Sprache und Zeichen.“ | |

| | | |
|---|---|---|
| (3) Programmieren *unplugged* | | |
| 3a) Bearbeiten von Programmieraufgaben mit haptischen Programmierbefehlen | DP 1: „In einem nächsten Schritt durch graphische und bildliche Darstellungen und zuletzt durch Sprache und Zeichen.“<br><br>DP 3: „Um die kognitive Belastung der Schülerinnen und Schüler während der Arbeit in der blockbasierten Programmierumgebung zu reduzieren, werden die charakteristischen Blöcke bzw. die algorithmischen Strukturen bereits während der Arbeit mit den *Unplugged*-Materialien eingeführt.“ | Die Schülerinnen und Schüler führen Algorithmen in ihrer Lebenswelt aus<br><br>Die Schülerinnen und Schüler entwerfen, realisieren und testen Algorithmen mit den algorithmischen Grundbausteinen Anweisung, Sequenz, Wiederholung und Verzweigung |
| 3b) Kontrolle der Lösungen im Klassenraum | DP 1: „Das Visualisieren von programmierten Handlungen durch das Ausführen von Handlungen im Klassenraum kann zur Überprüfung von Lösungen und als Hilfestellung eingesetzt werden.“ | |
| (4) Programmieren in Scratch | DP 1: „Um den Schülerinnen und Schülern im tätigen Umgang mit der Programmierung Lernerfahrungen zu ermöglichen, wird eine kindgerechte blockbasierte Programmierumgebung auf einem Informatiksystem genutzt.“ | |
| 4a) Präsentation der Oberfläche von Scratch | DP 3: „Die Programmierumgebung wird zudem kurz vorgestellt, bevor die Schülerinnen und Schüler diese eigenständig nutzen.“ | |

| | | |
|---|---|---|
| 4b) Angeleitetes Programmieren in Scratch und Bearbeiten von Transferaufgaben | DP 3: „Um die kognitive Belastung in Bezug auf die Programmieraufgaben zu reduzieren, wird zunächst mit vorgegebenen Programmen gearbeitet, die in einem nächsten Schritt angepasst oder erweitert werden, bis schließlich eigene Programmideen umgesetzt werden."<br><br>DP 4: „Die Aufgaben werden so gestaltet, dass sie Erfolgserlebnisse seitens der Lernenden ermöglichen, personalisiert werden können und – wo möglich – an ihre Alltagserfahrungen anknüpfen."<br><br>DP 5: „Einige Aufgaben der Unterrichtssequenz werden so konzipiert, dass sie durch Schlussfolgern gelöst werden können. Zudem werden die Schülerinnen und Schüler mit Aufgaben konfrontiert, für die keine offensichtliche Lösungsroutine abgerufen werden kann." | Die Schülerinnen und Schüler programmieren ein altersentsprechendes Informatiksystem<br><br>Die Schülerinnen und Schüler entwerfen, realisieren und testen Algorithmen mit den algorithmischen Grundbausteinen Anweisung, Sequenz, Wiederholung und Verzweigung |
| (5) Eigene Programmierprojekte | | |
| 5a) Planen eigener Programmideen | DP 5: „Im Rahmen einer offenen Aufgabenstellungwerden die Schülerinnen und Schüler an die Planung von Programmen herangeführt." | Die Schülerinnen und Schüler entwerfen, realisieren und testen Algorithmen mit den algorithmischen Grundbausteinen Anweisung, Sequenz, Wiederholung und Verzweigung |

| | | |
|---|---|---|
| 5b) Umsetzen der Ideen in Scratch | DP 4: „Auch offen gestaltete Programmieraufgaben werden eingesetzt“<br><br>DP 5: „Zudem werden die Schülerinnen und Schüler mit Aufgaben konfrontiert, für die keine offensichtliche Lösungsroutine abgerufen werden kann.“ | Die Schülerinnen und Schüler entwerfen, realisieren und testen Algorithmen mit den algorithmischen Grundbausteinen Anweisung, Sequenz, Wiederholung und Verzweigung<br><br>Die Schülerinnen und Schüler programmieren ein altersentsprechendes Informatiksystem |
| 5c) Präsentation der Ergebnisse | DP 4: „Die Aufgaben werden so gestaltet, dass sie Erfolgserlebnisse seitens der Lernenden ermöglichen, personalisiert werden können …“ | |

### B.1.2 Darstellung des überarbeiteten Unterrichtskonzepts

**(1) Thematischer Einstieg**

a) Aktivierung von Vorwissen der Schülerinnen und Schüler (DP 2)
b) Herstellen eines Bezugs zum Alltag der Schülerinnen und Schüler (DP 2)

Umsetzungsmöglichkeiten:

- Brainstorming:
  Zu Beginn der Unterrichtssequenz wird in Form eines Brainstormings gesammelt, was die Schülerinnen und Schüler bereits über das Programmieren wissen. Um zu vermeiden, dass nur diejenigen etwas sagen können, die konkrete Vorkenntnisse mitbringen, wird folgende Frage gestellt: „An was denkt ihr, wenn ihr das Wort ‚programmieren‘ hört?“

- Programmierung im Alltag:
  Im Rahmen des thematischen Einstiegs wird nach dem Brainstorming anhand verschiedener Bilder oder eines Wimmelbilds thematisiert, wo sich Programme in der Umgebung der Schülerinnen und Schüler verstecken. Gemeinsam stellen sie Vermutungen an, was auf den Bildern bzw. dem Bild programmiert sein könnte.

**(2) Grundlagen von Algorithmen und Programmen**

a) Formulieren von ersten Anweisungen und Verständigung über deren Eigenschaften (DP 1, DP 4)
b) Herausarbeiten des Unterschieds zur natürlichen Sprache (DP 1)

Umsetzungsmöglichkeiten:

- Lehrer-Roboter:
  Die Schülerinnen und Schüler programmieren die Lehrkraft mittels mündlicher Anweisungen. Die Lehrkraft spielt einen Roboter, der kleine Aufgaben im Klassenzimmer erfüllen soll, zum Beispiel das Fenster öffnen. Es wird nicht erklärt, wie man ihn steuert oder welche Anweisungen er kennt. Da er sich bei zu allgemeinen Anweisungen nicht rührt und nur präzise Befehle befolgt, wird jedoch schnell deutlich, dass die Anweisungen verständlich, genau und eindeutig formuliert werden müssen. Größere Vorgänge müssen dabei in Teilschritte zerlegt werden, die der Roboter nacheinander ausführt. Der Roboter führt die Anweisungen zudem wörtlich aus: z. B. bei der Anweisung „drehe dich nach rechts“ dreht sich der Roboter endlos nach rechts um die eigene Achse. Im Anschluss spielen Schülerinnen und Schüler den Roboter – der Rest der Klasse gibt mündliche Anweisungen und überprüft, ob der Roboter diese korrekt ausführt. Die Anweisungen für den Roboter können nicht nur mündlich erfolgen, sondern auch in Kleingruppen aufgeschrieben werden.

- Vom Bild zur Sprache:
  Es wird herausgearbeitet, wo die Schülerinnen und Schüler Algorithmen in ihrem Alltag begegnen, zum Beispiel in Form von Bastelanleitungen oder Rezepten. Es wird geübt, Abläufe in natürlicher Sprache zu beschreiben. Dabei kann die Lehrkraft auf die Textform der Vorgangsbeschreibung verweisen – falls diese bereits bekannt ist. Die Schülerinnen und Schüler verfassen zu einer bildlichen Anleitung sprachliche Anweisungen ohne dass ein Detaillevel vorgegeben wird (siehe Abschnitt B.2.1). Dabei können sie die Anleitung auch durch zusätzlich Bilder ergänzen. Die sprachlichen Anweisungen werden befolgt, um zu prüfen, ob sie präzise genug formuliert wurden. Nachdem die Schülerinnen und Schüler die Anweisungen formuliert haben, wird im Plenum besprochen, ob ihnen die Aufgabe leicht/schwergefallen ist. Zudem wird verglichen, wie viele Anweisungen jeweils von den Gruppen formuliert wurden. Die Lehrkraft verdeutlicht, dass Computer ganz genaue Anweisungen brauchen und Programmiersprachen deshalb – im Gegensatz zur natürlichen Sprache – eindeutig sind.

- Bilddiktat:
  Als Einstieg in die Thematik *Algorithmus* wird eine Variante der *Stillen Post* gespielt. Die Schülerinnen und Schüler malen oben auf ein leeres Blatt Papier ein einfaches Bild und geben es nach rechts an die Nachbarin bzw. den Nachbar weiter. Diese bzw. dieser beschreibt unter dem Bild, was sie oder er darauf sieht und knickt das gemalte Bild nach hinten um. Danach wird das Blatt erneut nach rechts weitergegeben, sodass nun jeder die Beschreibung eines Bildes

vor sich sieht. Zu dieser Beschreibung wird nun wieder ein Bild gemalt. Im Anschluss falten die Schülerinnen und Schüler die Blätter auf und vergleichen, wie ähnlich sich die beiden Bilder sehen oder worin sie sich unterscheiden. Falls die Bilder sehr unterschiedlich aussehen, können sie untersuchen, an was es liegt – wurde nicht genau beschrieben oder nicht genau gelesen? Anhand der Übung wird der Unterschied von Alltagsalgorithmen zu einem Algorithmus im eigentlichen Sinn herausgestellt.

- Bienenroboter:
  Die Schülerinnen und Schüler programmieren *BeeBots* oder *BlueBots*, dass diese bestimmte Wege abfahren. Die Programme können zunächst mit Pfeilkarten erstellt werden. Es ist außerdem zu empfehlen, dass die Schülerinnen und Schüler ihre Programme zunächst mithilfe eines *FakeBots*, einer ausgeschnittenen Abbildung des Bienenroboters, ausführen und auf Fehler untersuchen, bevor sie die Roboter programmieren. Auf diesem Weg wird eine *Trial and Error*-Herangehensweise verhindert.

**(3) Programmieren *unplugged***

a) Bearbeiten von Programmieraufgaben mit haptischen Programmierbefehlen (DP 1, DP 3)
b) Kontrolle der Lösungen im Klassenraum (DP 1)

Umsetzungsmöglichkeiten:

- Haptische Scratch Blöcke:
  Um die Schülerinnen und Schüler für das Programmieren in Scratch vorzuentlasten, werden die Befehle der Programmiersprache zunächst *unplugged* verwendet. Es werden haptische Programmierblöcke angefertigt, die optisch den Programmierbefehlen in Scratch entsprechen. Diese werden mit Magneten und Klettverschlüssen ausgestattet, sodass die Schülerinnen und Schüler mit ihnen sowohl an der Tafel als auch auf Filzbahnen programmieren können.

- Programmieren im Parcours:
  Um das Darstellen von Algorithmen zu vertiefen, programmieren die Schülerinnen und Schüler sich gegenseitig, um verschiedene Aufgaben in einem Parcours zu lösen, z. B. muss der Weg einer Person/eines Tieres zum Erreichen eines bestimmten Ziels beschrieben werden (siehe Abschnitt B.2.2). In Kleingruppen einigen sie sich zunächst auf einen Lösungsweg, zeichnen diesen auf den Arbeitsblättern ein und erstellen im Anschluss ein Programm mithilfe der haptischen Scratch-Blöcke. Gemäß der Aufgabenstellung bestimmen sie jeweils ein Kind, das das Programm vorliest und abläuft. Gruppenweise laufen die Schülerinnen und Schüler ihre Lösungen in einem aufgebauten Parcours ab. Der Rest der Gruppe kontrolliert, ob das Programm richtig ausgeführt wird. Im Plenum werden die ggf. unterschiedliche Lösungen miteinander verglichen. Falls es sich anbietet, kann die Lehrkraft zum Gebrauch algorithmischer Strukturelemente überleiten, indem sie z. B. fragt, wie man ein Programm möglichst kurz formulieren kann.

- Kontrollmöglichkeiten:
  Je nach verfügbarem Platz können die Lösungen der Parcoursaufgaben unterschiedlich kontrolliert werden. Im Klassenraum kann der Parcours z. B. mithilfe von Teppichfliesen gelegt werden oder die Umrisse mittels Malerkrepp auf den Boden geklebt werden. In dieser Variante muss Bedacht werden, dass die Kleingruppen ihre jeweilige Lösung nacheinander – nicht gleichzeitig – kontrollieren. Alternativ kann der Parcours auch im Pausenhof mit Kreide aufgemalt werden. Ist der zur Verfügung stehende Platz beschränkt, kann die Lehrkraft die Lösungen auch mit einer Spielfigur auf dem Arbeitsblatt ablaufen und dies über die Dokumentenkamera zeigen. Oder die Schülerinnen und Schüler laufen die Lösungen mit kleinen Spielfiguren auf den Arbeitsblättern ab – auf diesem Weg ist es möglich, dass die Kleingruppen ihre Lösungen parallel kontrollieren.

**(4) Programmieren in Scratch (DP 3)**

a) Präsentation der Oberfläche von Scratch (DP 3)
b) Angeleitetes Programmieren in Scratch und Bearbeiten von Transferaufgaben (DP 3, DP 4, DP 5)

Umsetzungsmöglichkeiten:

- Orientierung in Scratch:
  Um den Schülerinnen und Schülern den Einstieg in Scratch zu erleichtern, wird die Programmierumgebung gemeinsam betrachtet, z. B. über den Beamer. Hier können verschiedene Fragen gestellt werden, z. B. „Wer sieht etwas, das er bereits kennt?“, „Hat jemand eine Idee, wie man das Programm starten kann?“. Denkbar ist außerdem, den ersten Kontakt mit Scratch mittels eines Arbeitsblatts vorzustrukturieren, auf dem die verschiedenen Bereiche von Scratch benannt werden müssen.

- Bekannte Aufgaben:
  Damit die Schülerinnen und Schüler sich ganz auf die Bedienung der Programmierumgebung konzentrieren können, bearbeiten sie darin zunächst einige Aufgaben, die bereits *unplugged* im Rahmen der Parcours-Aufgaben gelöst wurden. Der Lösungsweg bzw. das Programm für diese Aufgaben ist bereits klar und das Zurechtfinden innerhalb der Programmierumgebung steht im Vordergrund.

- Angeleitetes Programmieren:
  In Scratch bearbeiten die Schülerinnen und Schüler einen Lernzirkel, in dem die Grundfunktionen von Scratch nacheinander thematisiert werden (siehe Abschnitt B.2.3). Ausgehend von Fragen der Bedienung führen die Zirkelkarten über einfache Sequenzen hin zu Kontrollstrukturen wie Wiederholungen und bedingte Anweisungen. An jeder Station wird die entsprechende Funktion zunächst schrittweise erklärt und die Schülerinnen und Schüler wiederholen jeden Schritt an ihrem eigenen Rechner. Im Anschluss daran bearbeiten sie eine dazu passende Aufgabe, bei der sie die eingeführte Funktion bzw. Struktur in einem anderen Kontext oder leicht abgewandelt verwenden müssen. Zu ausgewählten Aufgaben werden Tippkarten angefertigt, die von den

Schülerinnen und Schülern eigenständig verwendet werden können. Die Programmieraufgaben werden zunehmend anspruchsvoller, dass die Schülerinnen und Schüler relativ schnell erste Erfolgserlebnisse haben. Leistungsschwächere Schülerinnen und Schüler können zunächst vorgegebene Programme untersuchen, bevor sie selbst programmieren. Die Lehrkraft unterstützt durch verbale Hilfestellungen und verweist ggf. auf die Tippkarten. Als Kontext für die Programmieraufgaben kann bspw. das Thema Zirkus gewählt werden. Zum einen knüpft das Thema an die Alltagserfahrungen der Schülerinnen und Schüler an, zum anderen ergeben sich daraus viele Programmieranlässe, die sich auf konkrete Handlungen beziehen, z. B. die Begrüßung des Zirkusdirektors, die Bewegung eines Tigers in der Manege, das Erzählen von Witzen eines Clowns. Innerhalb der Programmieraufgaben haben die Schülerinnen und Schüler immer wieder die Möglichkeit, ihre Programme zu personalisieren.

- Zusätzliche Arbeitsaufträge:
  Die Aufgaben aus dem Lernzirkel können durch weitere Programmieraufgaben ergänzt werden, für deren Lösung die algorithmischen Grundstrukturen freier eingesetzt und kombiniert werden müssen. Hier kann auf Programmieraufträge aus den Fortbildungsveranstaltungen zurückgegriffen werden.

- Hilfestellungen beim Programmieren:
  Auch beim Programmieren nehmen die Lehrkräfte ihre Verantwortung für die Gestaltung der Unterrichtsprozesse wahr, indem sie die Kinder durch verschiedene Hilfestellungen in ihrem Lernprozess unterstützen. Bestimmte Merkmale der Programmiersprache Scratch können losgelöst vom Computer thematisiert werden, zum Beispiel die verschiedenen Blockkategorien und zugehörigen Farben, die das Auffinden von Blöcken in der Programmierumgebung erleichtern. Zudem gibt die Lehrkraft Hilfestellungen bei konkreten Programmieraufgaben. Beispielsweise kann sie den Lösungsweg einer Aufgabe vor der Implementierung in Scratch *unplugged* veranschaulichen oder die Lösung einer Aufgabe im Plenum erarbeiten, bevor die Schülerinnen und Schüler sie in Scratch umsetzen, zum Beispiel durch das Skizzieren des Programms in Pseudocode an der Tafel. Darüber hinaus kann die Lehrkraft beim Auffinden von Fehlern im Programm helfen, indem sie die Schülerlösung Anweisung für Anweisung im Raum ausführt und direkt aufzeigt, an welcher Stelle das Programm nicht das tut, was gewünscht ist. Diese Hilfestellung eignet sich für Scratch, da die Schülerinnen und Schüler in den meisten Fällen die Handlungen von einzelnen Figuren programmieren.

- Problemlöseprozess anleiten:
  Im Laufe der Unterrichtssequenz wird im Plenum thematisiert, wie man beim Problemlösen in Scratch vorgeht, z. B. anhand von Symbolen. Diese können im Klassenraum aufgehängt werden und als Orientierung für die Schülerinnen und Schüler dienen, wenn sie eine Aufgabe bearbeiten.

- Programme lesen:
  Zum Schreiben von Programmcode gehört auch das Lesen von Programmcode. Dabei muss

man die einzelnen Wörter bzw. Anweisungen entziffern und zusammenfügen, um daraus die Bedeutung des Programms zu erschließen. Mit Aufgaben, bei denen das Lesen von Code im Vordergrund steht, kann man das genaue Lesen und nachverfolgen von Code üben. Auf diesem Weg können außerdem häufige Fehler oder Fehlvorstellungen zum Ablauf von Programmen thematisiert werden. Zudem ist es möglich, den Fokus gezielt auf einzelne algorithmische Strukturen zu lenken.

- Gemeinsam programmieren:
  Während des Programmierens in Scratch können die Schülerinnen und Schüler in Paaren zusammenarbeiten. Hier ist wichtig, dass regelmäßig gewechselt wird, wer z. B. die Maus oder Tastatur bedient. Dieser Vorgang kann von der Lehrkraft angeleitet werden, indem die Rollen des „Fahrers“ und des „Navigators/Steuermanns“ eingeführt werden. Ersterer bedient Tastatur und Maus und somit die Programmierumgebung; letzterer gibt entsprechende Anweisungen und behält den Überblick über die jeweilige Aufgabenstellung oder das Projekt. Nachdem die beiden Rollen eingeführt wurden, kann die Lehrkraft während des Programmierens regelmäßig zum Rollentausch aufrufen.

- Austausch ermöglichen:
  Bei besonders schwierigen oder offenen Aufgaben, kann das Problemlösen in einem Unterrichtsgespräch thematisiert werden. Bei schwierigen Aufgaben können einzelne Schülerinnen und Schüler über den Beamer zeigen, wie sie die Aufgabe gelöst haben und beschreiben, wie sie dabei vorgegangen sind. Im Rahmen offener Aufgaben können verschiedene Ergebnisse von den Schülerinnen und Schülern vorgestellt werden. Dabei wird die Frage „welche besonders harten Nüsse geknackt wurden - und wie“ in den Fragenkanon aufgenommen.

**(5) Eigene Programmierprojekte**

a) Planen eigener Programmideen (DP 5)
b) Umsetzen der Ideen in Scratch (DP 4, DP 5)
c) Präsentation der Ergebnisse (DP 4)

Umsetzungsmöglichkeiten:

- Planungsphasen:
  Da die Komplexität der Programme der Schülerinnen und Schüler in der Regel überschaubar ist, können sie ihre Programmideen zunächst in natürlicher Sprache in Sätzen oder Stichpunkten notieren. Einfache Handlungen können sie außerdem in Form von Bildern oder kurzen Storyboards festhalten. Für den Planungsprozess kann die Lehrkraft eine Vorlage zur Verfügung stellen (siehe Abschnitt B.2.4). Um die Schülerinnen und Schüler an dieser Stelle nicht zu überfordern können inhaltliche Vorgaben (z. B. „Jeder programmiert seine eigene Zirkusaufführung!“) oder strukturelle Vorgaben (z. B. „Jedes Programm muss mindestens eine Wiederholung enthalten!“) erfolgen.

- Umgang mit Fragen und Problemen:
  Auch wenn die Lehrkraft Fehler im Programm sieht, gibt sie den Schülerinnen und Schülern die Chance, sie bei der Programmausführung selbst zu erkennen. Bei Fragen fordert die Lehrkraft die Schülerinnen und Schüler auf, zuerst ihre Mitschülerinnen und Mitschüler um Rat zu fragen. Komplexe Fragestellungen/Probleme werden gemeinsam mit den Schülerinnen und Schülern gelöst – dabei können konkrete Schritte der Fehlersuche eingeführt werden sowie verbale Hilfestellungen gegeben werden. Falls die Lehrkraft Fehler nicht sofort lösen kann, kommuniziert sie dies offen und sucht gemeinsam mit den Schülerinnen und Schülern nach Lösungen).

- Ergebnispräsentation:
  Die Ergebnisse der Schülerinnen und Schüler werden (freiwillig) über den Beamer vorgestellt. Dabei erläutern sie, welche Schwierigkeiten sie bei der Umsetzung gemeistert haben. Jedes Programm wird mit Applaus wertgeschätzt. Es wird zudem thematisiert, inwieweit mögliche Vorgaben an die Programme eingehalten wurden.

- Motivierende Programmieranlässe:
  Ohne Vorgaben fällt es manchen Kindern schwer, eigene Ideen zu entwickeln. Zudem kommt es häufig vor, dass sich Ideen der Schülerinnen und Schüler nur schwer in Scratch implementieren lassen. Darüber hinaus ist es eine große Herausforderung für die Lehrkraft, jedem Kind individuelle Hilfestellungen für das jeweilige Projekt zu geben. Zugleich scheinen aber offene Aufgaben, bei denen die Schülerinnen und Schüler ihr Programm individuell gestalten können, für sie besonders motivierend zu sein. Um dem zu begegnen, kann die Lehrkraft gewisse Vorgaben für das Programm geben, die jedoch gleichzeitig Raum für Kreativität zulassen, z. B. (1) ein Hase lässt sich mit den Pfeiltasten steuern; (2) er läuft durch ein Labyrinth zu seinem Ziel (einer Karotte); (3) wenn er die Mauern berührt, fällt er zurück auf den Startpunkt. Der Code der Kinder wird in diesem Fall zwar sehr ähnlich aussehen, trotzdem können sie ihre eigenen Programme gestalten und diese beliebig erweitern.

- Klassenprojekt:
  Wenn die Klasse schon Erfahrungen im Programmieren in Scratch gesammelt hat, können Projekte durchgeführt werden, die arbeitsteilig bearbeitet und zum Schluss von der Lehrkraft in einem Scratch-Programm zusammengefügt werden. Ein Beispiel hierfür ist ein virtueller Rundgang durch die Heimatstadt. In Kleingruppen recherchieren die Schülerinnen und Schüler jeweils zu einem Ort und suchen Fotos und Informationen heraus. Im Anschluss fügen sie die Fotos in ein Scratch-Programm und programmieren eine Figur, welche die Informationen an bestimmten Zeitpunkten vorträgt.

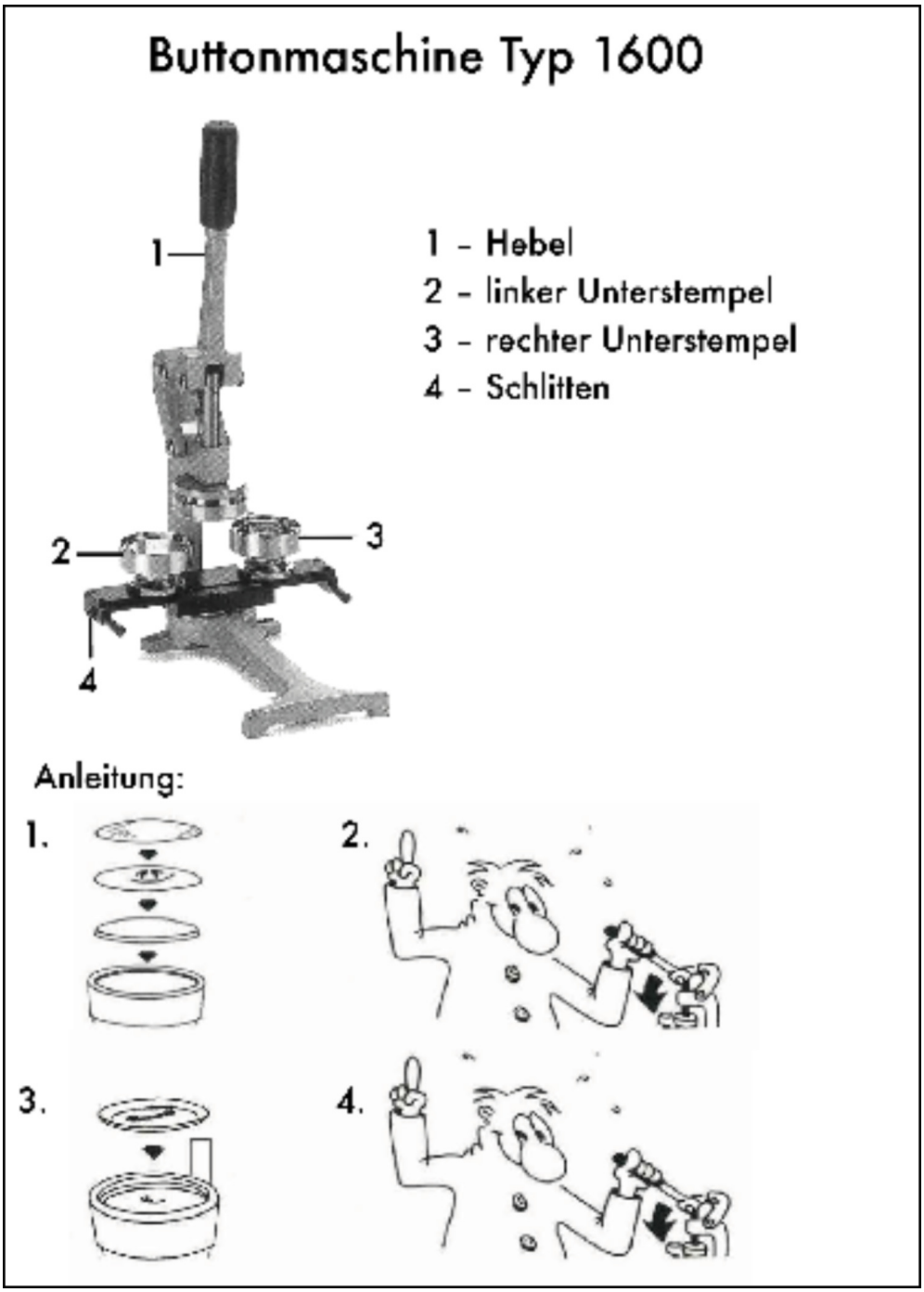
- Zusätzliche Systeme:
  Sobald die Schülerinnen und Schüler mit den algorithmischen Grundstrukturen vertraut sind, können Microcontroller, wie z. B. der *Calliope mini*, oder Roboter, z. B. *Lego WeDo*, programmiert werden.

## B.2 Unterrichtsmaterialien

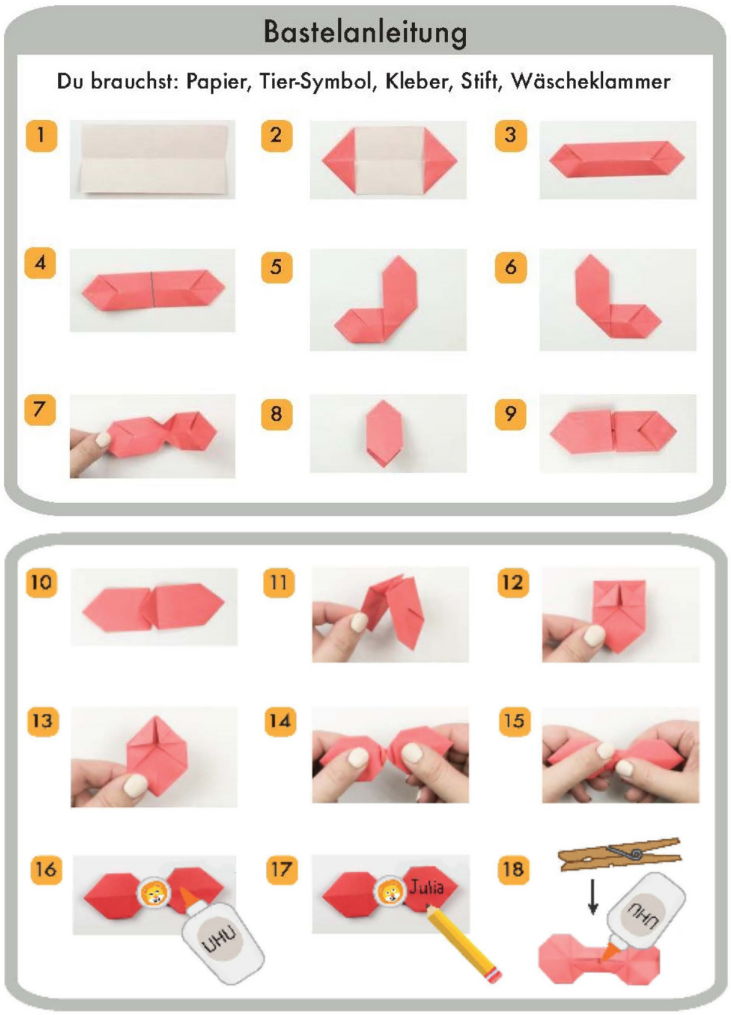
Im Folgenden werden ausgewählte Unterrichtsmaterialien dargestellt, die im Rahmen dieser Arbeit entwickelt, erprobt und überarbeitet wurden. Die Einsicht in weitere Materialien, die an dieser Stelle nicht oder nicht vollständig dargestellt sind (z. B. Scratch-Dateien), kann per E-Mail direkt bei der Autorin angefragt werden (katharina.geldreich@tum.de).

### B.2.1 Bildliche Anleitungen

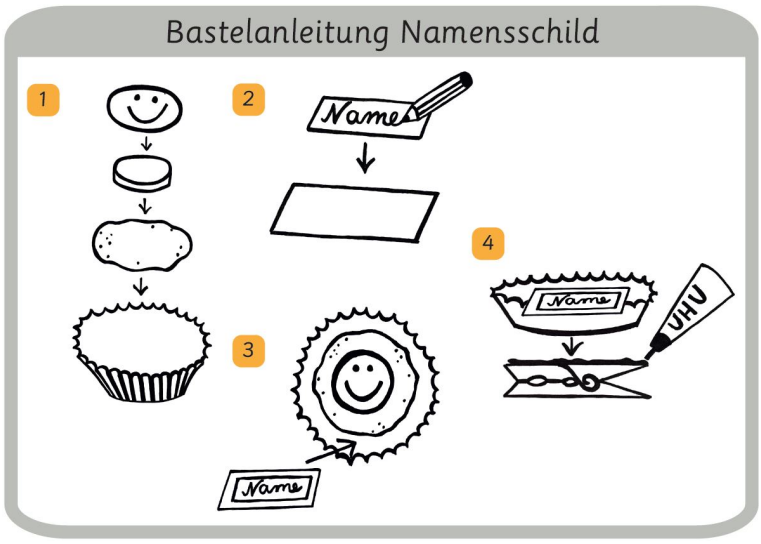
#### B.2.1.1 Zyklus 1 (Pilotstudie)



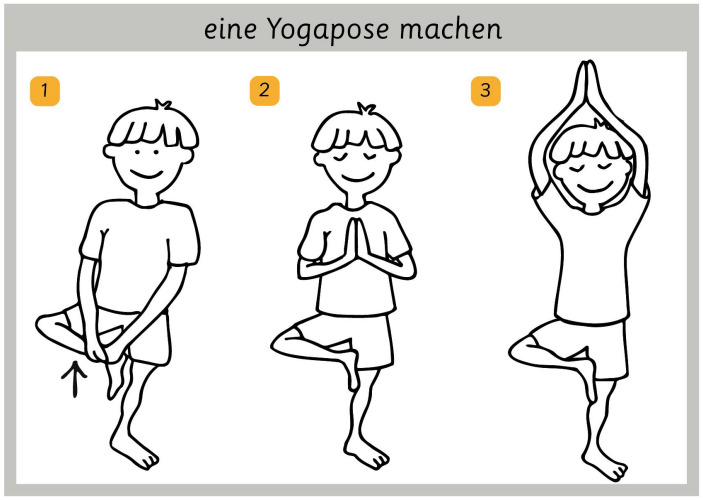
### B.2.1.2 Zyklus 2

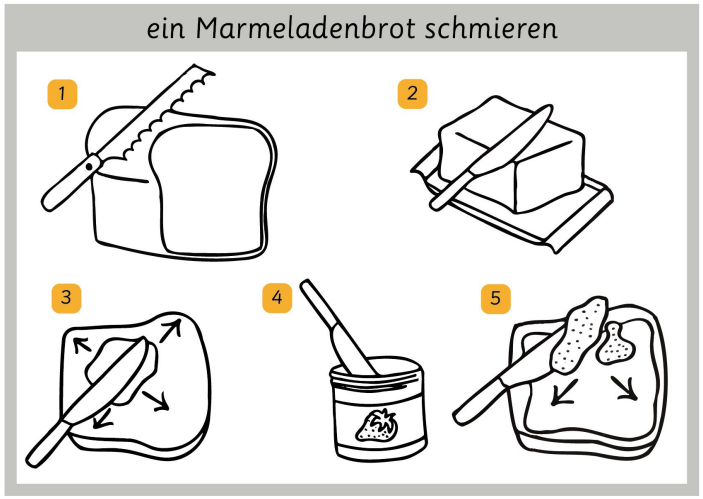



### B.2.1.3 Zyklus 3

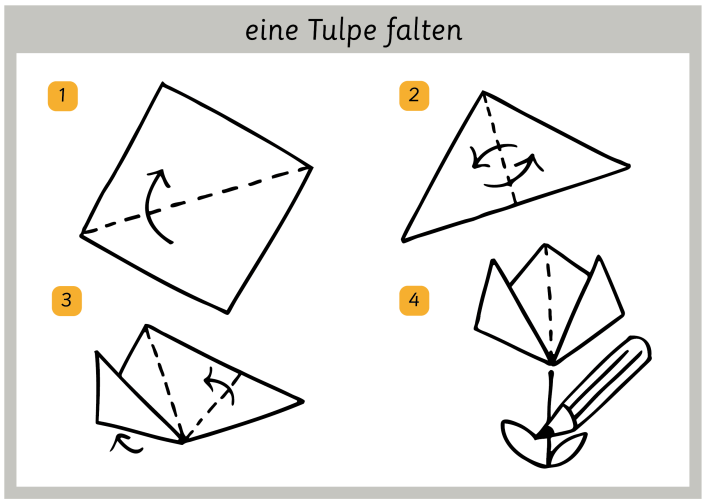


### B.2.1.4 Zyklus 4



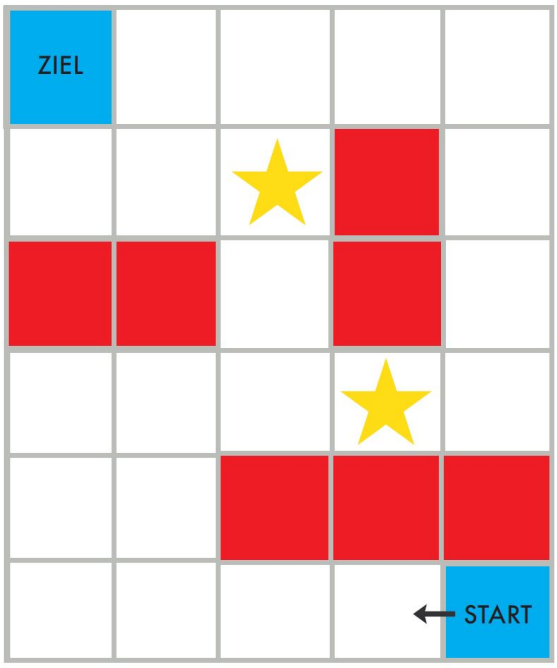




### B.2.1.5 Ergänzende Anleitung nach Zyklus 4



## B.2.2 Parcours-Aufgaben

### B.2.2.1 Zyklus 1 (Pilotstudie)




INFO

Die roten Felder sind Mauern, durch die du nicht durchlaufen kannst.

AUFGABE

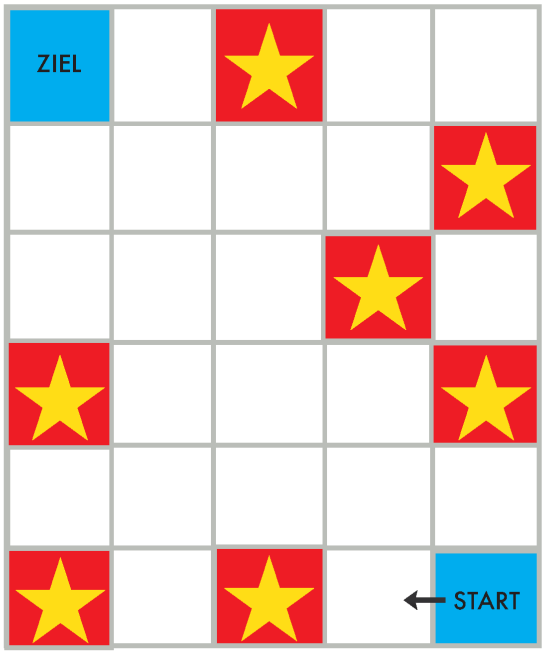
1. Überlege dir einen Weg vom Start zum Ziel auf dem du die Gegenstände (★) einsammelst.Zeichne ihn in das Feld links ein.

2. Besprecht eure Wege in der Gruppe und einigt euch auf einen Weg, den ihr zusammen programmieren wollt.

   Programmiert den Weg mit den vorhandenen Befehlen. Wenn es einen Befehl nicht gibt, könnt ihr einen neuen Befehl erfinden.

3. Bestimmt eine Person die
   – das Programm abläuft
   – das Programm vorliest

   Die anderen überprüfen, ob die Befehle richtig ausgeführt werden.




INFO

Unter jedem roten Feld verbirgt sich ein Gegenstand. Wenn du vor einem roten Feld stehst, kannst du mit dem Befehl „decke auf" darunter schauen und den Gegenstand einsammeln.

AUFGABE

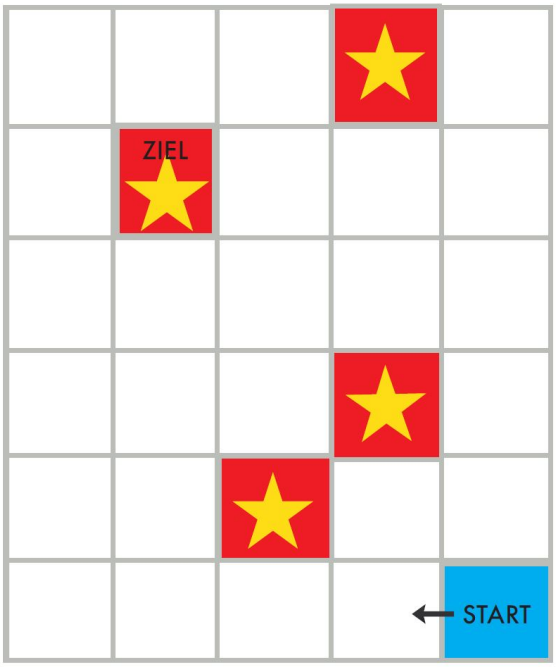
1. Überlegt euch in der Gruppe ein Programm, mit dem ihr alle Gegenstände einsammeln könnt. Versucht so wenige Befehle wie möglich zu verwenden.

2. Programmiert den Weg mit den vorhandenen Befehlen. Wenn es einen Befehl nicht gibt, könnt ihr einen neuen Befehl erfinden.

3. Bestimmt eine Person die
   – das Programm abläuft
   – das Programm vorliest

   Die anderen überprüfen, ob die Befehle richtig ausgeführt werden.

### B.2.2.2 Zyklus 2




INFO

Unter jedem roten Feld verbirgt sich ein Gegenstand. Wenn du vor einem roten Feld stehst, kannst du mit dem Befehl „decke auf" darunter schauen und den Gegenstand einsammeln.

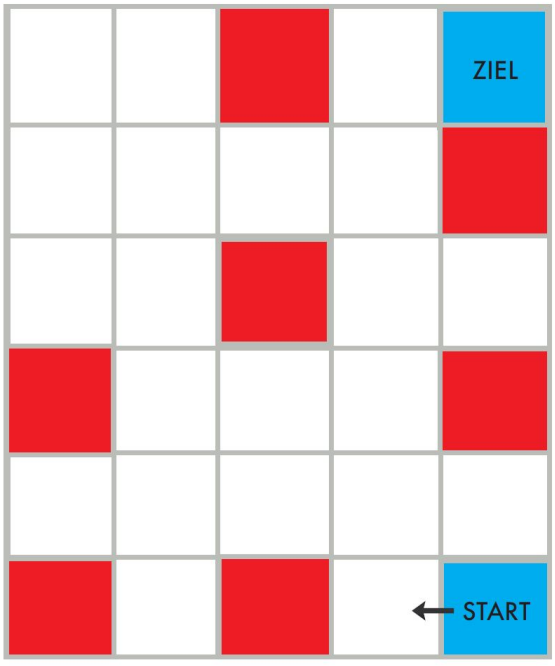
AUFGABE

1. Überlegt euch in der Gruppe ein Programm, mit dem ihr alle Gegenstände einsammeln könnt. Versucht so wenige Befehle wie möglich zu verwenden. Zeichnet den Weg ein!

2. Programmiert den Weg mit den vorhandenen Befehlen. Wenn es einen Befehl nicht gibt, könnt ihr einen neuen Befehl erfinden.

3. Bestimmt eine Person die
   – das Programm abläuft
   – das Programm vorliest

   Die anderen überprüfen, ob die Befehle richtig ausgeführt werden.




INFO

Unter einigen roten Feldern verbergen sich Gegenstände.

AUFGABE

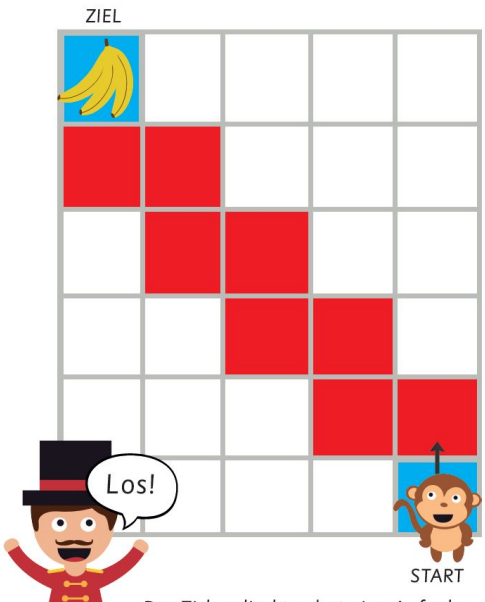
1. Überlegt euch in der Gruppe ein Programm, mit dem ihr alle Gegenstände einsammeln könnt. Versucht zu programmieren, dass der Läufer immer unter das rote Feld schaut, wenn er vor einem steht.

2. Programmiert den Weg mit den vorhandenen Befehlen. Wenn es einen Befehl nicht gibt, könnt ihr einen neuen Befehl erfinden.

3. Bestimmt eine Person die
   – das Programm abläuft
   – das Programm vorliest

   Die anderen überprüfen, ob die Befehle richtig ausgeführt werden.

### B.2.2.3 Zyklus 3 und Zyklus 4

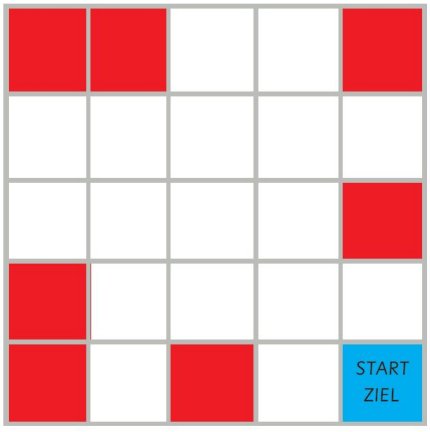



Der Zirkusdirektor hat eine Aufgabe für den Affen:

Wenn er „Los!“ ruft, soll er entlang der roten Felder zur Banane laufen.

AUFGABE

1. Überlegt euch gemeinsam ein Programm, das den Affen an das Ziel bringt.
2. Programmiert den Weg mit den Befehls-Blöcken.
3. Bestimmt eine Person, die
   - das Programm abläuft
   - das Programm vorliest

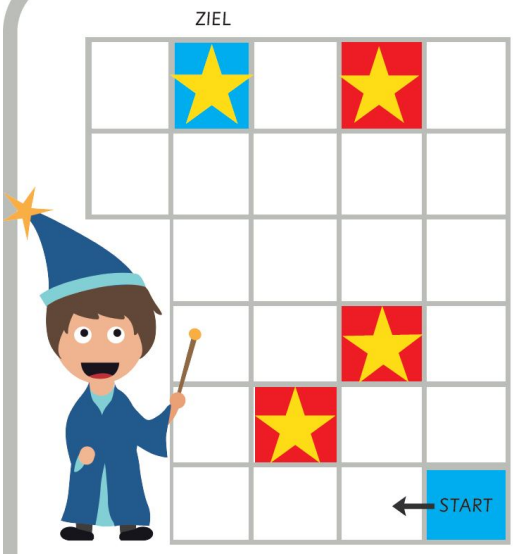



Der Clown weiß nicht mehr, wo er seine Sachen versteckt hat. Er erinnert sich aber, dass er sie unter rote Felder gelegt hat.

Was kann er tun, um alles zu finden?

AUFGABE

1. Überlegt euch gemeinsam ein Programm, mit dem der Clown seine verlorenen Sachen wiederfinden kann.
2. Zeichnet den Weg des Clowns in das Feld auf der linken Seite ein.
3. Programmiert den Weg mit den Befehls-Blöcken.
4. Bestimmt eine Person, die
   - das Programm abläuft
   - das Programm vorliest




Um alle Gegenstände zu bekommen, muss der Zauberer die roten Felder ablaufen.

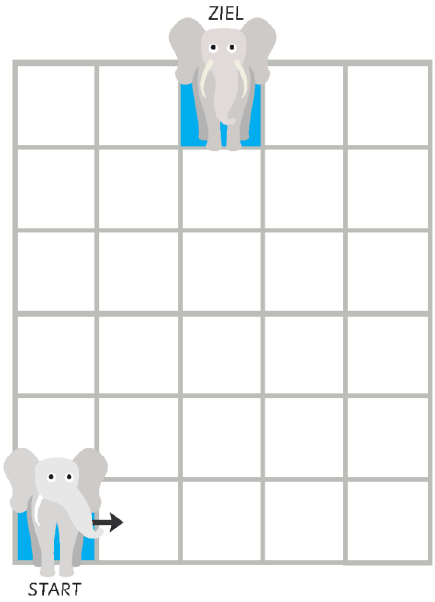
Das Spielfeld ist verhext. Man bekommt die Gegenstände nur, wenn das Programm aus maximal acht Blöcken besteht.

Welches Programm kann ihm helfen?

AUFGABE

1. Überlegt euch gemeinsam ein Programm, mit dem der Zauberer alle Gegenstände einsammeln kann.
2. Zeichnet den Weg des Zauberers in das Feld auf der linken Seite ein.
3. Programmiert den Weg mit den Befehls-Blöcken.
4. Bestimmt eine Person, die
   - das Programm abläuft
   - das Programm vorliest

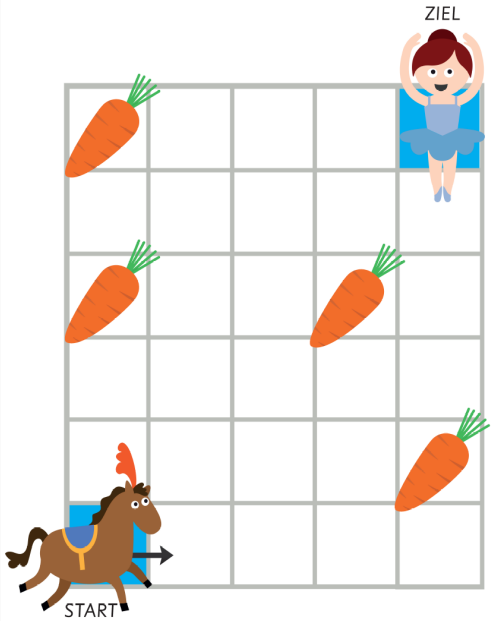
### B.2.2.4 Ergänzende Aufgaben nach Zyklus 4




Der Elefant möchte so schnell wie möglich zu seinem besten Freund laufen.

AUFGABE

1. Überlegt euch gemeinsam einen möglichst kurzen Weg für den Elefanten.
2. Zeichnet den Weg in das Feld auf der linken Seite ein.
3. Programmiert den Weg mit den Befehls-Blöcken.
4. Bestimmt eine Person, die
   - das Programm vorliest
   - das Programm abläuft

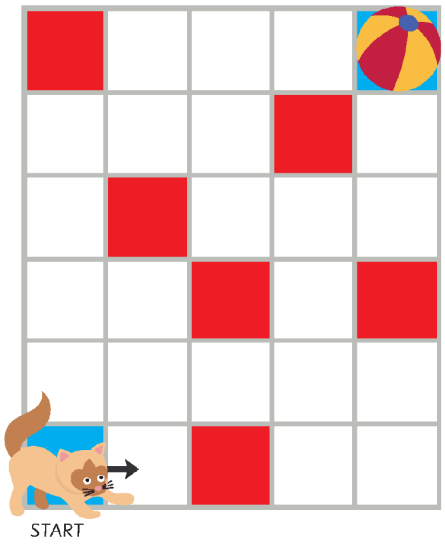



Das Pferd soll zur Seiltänzerin galoppieren.

Es wurde noch nicht gefüttert und braucht mindestens zwei Karotten um satt zu werden.

AUFGABE

1. Überlegt euch gemeinsam einen Weg für das Pferd.
2. Zeichnet den Weg in das Feld auf der linken Seite ein.
3. Programmiert den Weg mit den Befehls-Blöcken.
4. Bestimmt eine Person, die
   - das Programm vorliest
   - das Programm abläuft




Die Katze möchte sich den Ball schnappen.

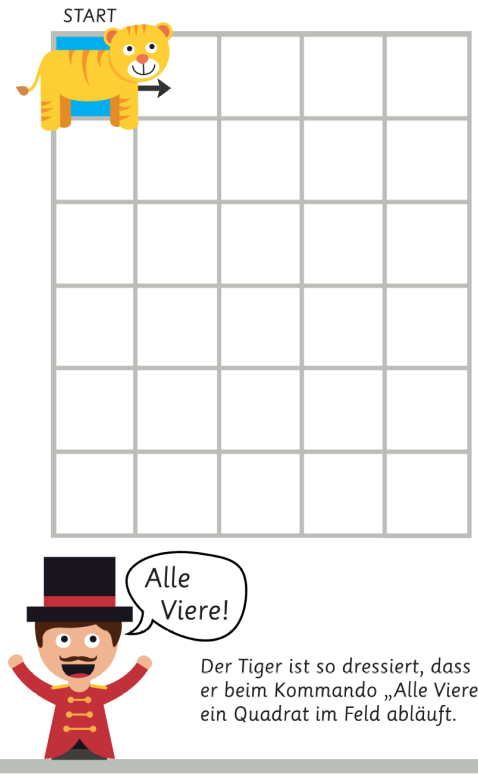
Auf ihrem Weg darf sie jedoch keine roten Felder berühren.

AUFGABE

1. Überlegt euch gemeinsam einen Weg für die Katze.
2. Zeichnet den Weg in das Feld auf der linken Seite ein.
3. Programmiert den Weg mit den Befehls-Blöcken.
4. Bestimmt eine Person, die
   - das Programm vorliest
   - das Programm abläuft

ZUSATZFRAGE

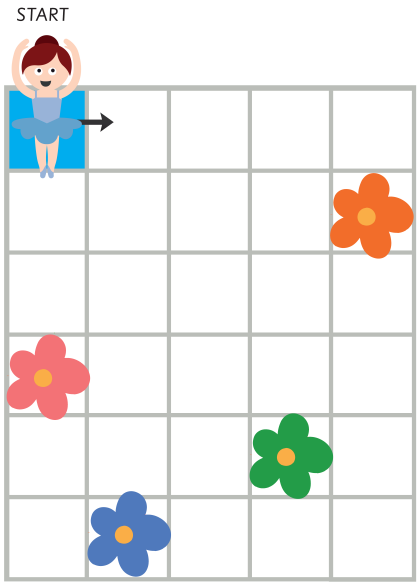
Wie viele mögliche Wege gibt es zum Ball?




Der Tiger ist so dressiert, dass er beim Kommando „Alle Viere!“ ein Quadrat im Feld abläuft.

AUFGABE

1. Überlegt euch gemeinsam ein Programm, für den Tiger.
2. Zeichnet den Weg in das Feld auf der linken Seite ein.
3. Programmiert den Weg mit den Befehls-Blöcken.
4. Bestimmt eine Person, die
   - das Programm vorliest
   - das Programm abläuft




Die Seiltänzerin möchte für ihre Garderobe eine Blume pflücken.

AUFGABE

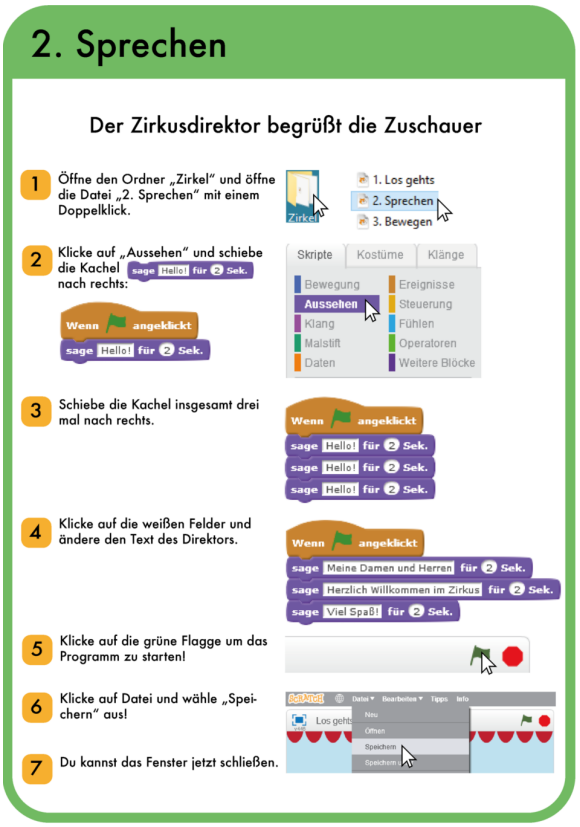
1. Überlegt euch gemeinsam einen Weg für die Seiltänzerin zu einer Blume.
2. Zeichnet den Weg in das Feld auf der linken Seite ein.
3. Programmiert den Weg der Seiltänzerin mit den Befehls-Blöcken.

   Falls sie am Ende des Programms eine Blume berührt, dann soll die Seiltänzerin sie pflücken (schreibt einen neuen Block: pflücken).
4. Bestimmt eine Person, die
   - das Programm vorliest
   - das Programm abläuft

## B.2.3 Scratch-Lernzirkel

### B.2.3.1 Zyklus 1 (Pilotstudie) und Zyklus 2

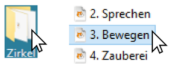





# 3. Bewegen

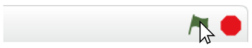
## Der Tiger durchquert die Manege

1. Öffne den Ordner „Zirkel“ und öffne die Datei „Bewegen“ mit einem Doppelklick.




2. Klicke auf „Bewegung“ und schiebe die Kachel gehe 10 er-Schritt nach rechts.


3. Klicke auf die grüne Flagge um das Programm zu starten! Schaue was passiert, wenn du die Zahlen verändert!



4. Klicke auf „Steuerung“ und schiebe die Kachel warte 1 Sek. nach rechts.


5. Setze mehrere der Blöcke hintereinander.


6. Baue so viele Blöcke aneinander bis der Tiger von ganz links bis an die andere Seite der Manege läuft.

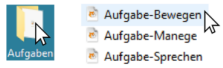
7. Klicke auf Datei und wähle „Speichern“ aus!

8. Du kannst das Fenster jetzt schließen.

# Aufgabe

## Der Tiger soll in der Manege hin und her laufen!

1. Öffne den Ordner „Aufgaben“ und öffne die Datei „Aufgabe-Bewegen“ mit einem Doppelklick.




2. Lasse den Tiger wie gerade eben an das rechte Ende der Manege laufen. Baue zwischen jedem Befehl den warte 1 Sek.-Block ein!

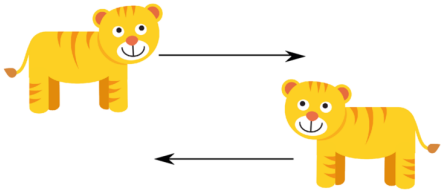
3. Der Tiger muss sich jetzt umdrehen. Überlege, welchen Bewegungs-Block du verwenden musst.

? Wenn du Hilfe brauchst, gibt es einen Tipp hinter der Tafel.

4. Lasse den Tiger wieder nach links laufen.

5. Klicke auf Datei und wähle „Speichern“ aus!

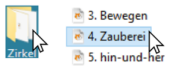
6. Du kannst das Fenster jetzt schließen.



# 4. Zauberei

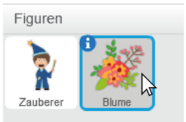
## Der Zauberer hat einen neuen Trick

1. Öffne den Ordner „Zirkel“ und öffne die Datei „4. Zauberei“ mit einem Doppelklick.



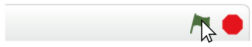

2. Bei mehreren Figuren musst du zuerst auswählen, für welche Figur du ein Programm schreiben möchtest.

   Klicke in der Figurenliste auf die Blumen. An dem blauen Rand siehst du, dass sie ausgewählt wurden.

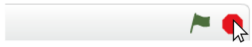



3. Klicke auf „Aussehen“ und benutze den Farbwechsel-Block


4. Klicke auf die grüne Flagge um das Programm zu starten!



5. Wenn du auf den roten Knopf klickst, bekommen die Blumen ihre Ausgangsfarbe zurück. Probiere verschiedene Zahlen aus!



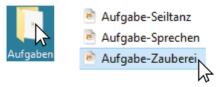
6. Klicke auf Datei und wähle „Speichern“ aus!

7. Du kannst das Fenster jetzt schließen.

# Aufgabe

## Welche neuen Tricks kannst du dir ausdenken?

1. Öffne den Ordner „Aufgaben“ und öffne die Datei „Aufgabe-Zauberei“ mit einem Doppelklick.




2. Schaue dir die Aussehen-Blöcke genau an. Hast du eine Idee für einen neuen Trick?

? Wenn du Hilfe brauchst, gibt es ein paar Hinweise hinter der Tafel.

3. Baue das Programm für den neuen Trick.

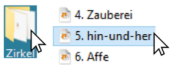
4. Klicke auf Datei und wähle „Speichern“ aus!

5. Du kannst das Fenster jetzt schließen.

## 5. Ewig hin und her

Um den Tiger hin und herlaufen zu lassen, waren viele Blöcke notwendig. Es geht auch schneller!

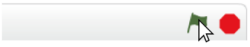
1. Öffne den Ordner „Zirkel“ und öffne die Datei „5. hin-und-her“ mit einem Doppelklick.




2. Versuche diesen Block nachzubauen:


? Wenn du Hilfe brauchst, gibt es einen Tipp hinter der Tafel.

3. Klicke auf die grüne Flagge um das Programm zu starten!



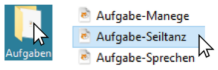
4. Klicke auf Datei und wähle „Speichern“ aus!

5. Du kannst das Fenster jetzt schließen.

## Aufgabe

Der Auftritt der Seiltänzerin

1. Öffne den Ordner „Aufgaben“ und öffne die Datei „Aufgabe-Seiltanz“ mit einem Doppelklick.




2. Die Seiltänzerin balanciert von dem einen Ende des Seils zum anderen. Baue für sie den gleichen Block wie für den Tiger.

3. Versuche, die Tänzerin ganz langsam hin und herbalancieren zu lassen.

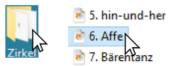
4. Klicke auf Datei und wähle „Speichern“ aus!

5. Du kannst das Fenster jetzt schließen.

## 6. Fang den Affen!

Oh Schreck – der Affe ist weg! Der Direktor versucht ihn mit Bananen zu ködern

1. Öffne den Ordner „Zirkel“ und öffne die Datei „6. Affe“ mit einem Doppelklick.

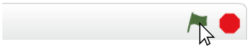



2. Wähle in der Figurenliste den Affen aus und versuche den Block nachzubauen:

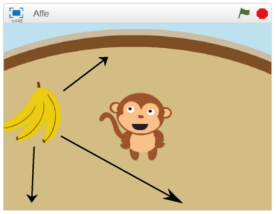


? Wenn du Hilfe brauchst, gibt es einen Tipp hinter der Tafel.

3. Klicke auf die grüne Flagge um das Programm zu starten!



4. Probiere, die Bananen auf der Bühne zu versetzen. Melde dich, wenn du Hilfe brauchst!



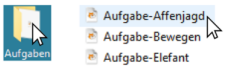

5. Klicke auf Datei und wähle „Speichern“ aus!

6. Du kannst das Fenster jetzt schließen.

## Aufgabe

Das Ködern dauert dem Direktor zu lange – er versucht ihn selbst einzufangen

1. Öffne den Ordner „Aufgaben“ und öffne die Datei „Aufgabe-Affenjagd“ mit einem Doppelklick.




2. Versuche einen Block für den Zirkusdirektor zu bauen, mit dem er den Affen schnell einfangen kann.

? Wenn du Hilfe brauchst, gibt es einen Tipp hinter der Tafel.

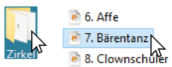
3. Klicke auf Datei und wähle „Speichern“ aus!

4. Du kannst das Fenster jetzt schließen.

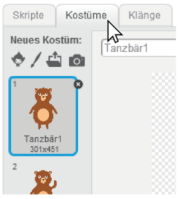
## 7. Bärentanz

Um den Bären tanzen zu lassen, muss er seine Kostüme wechseln. Er zieht aber keine anderen Kleider an, sondern nimmt eine andere Haltung ein. So sieht es aus, als würde er tanzen.

1. Öffne den Ordner „Zirkel“ und öffne die Datei „7. Bärentanz“ mit einem Doppelklick.




2. Klicke auf „Kostüme“ um dir die verschiedenen Tanzfiguren anzuschauen




3. Versuche den Block nachzubauen:


4. Klicke auf die grüne Flagge um das Programm zu starten!

5. Benutze auch die anderen Kostüme! Baue wie oben immer einen Warte-Block zwischen die Kostümwechsel.



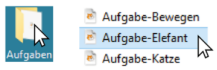
6. Klicke auf Datei und wähle „Speichern“ aus!

7. Du kannst das Fenster jetzt schließen.

## Aufgabe

Auch Elefanten können tanzen!

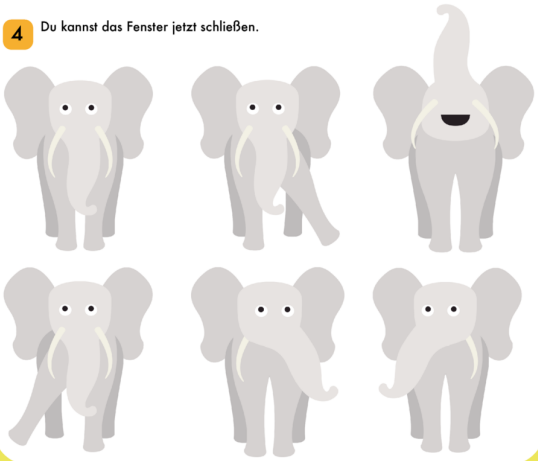
1. Öffne den Ordner „Aufgaben“ und öffne die Datei „Aufgabe-Elefant“ mit einem Doppelklick.




2. Schau dir die Kostüme des Elefanten an und baue sie zu einem Tanz zusammen.

3. Klicke auf Datei und wähle „Speichern“ aus!

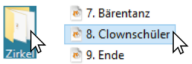
4. Du kannst das Fenster jetzt schließen.



## 8. Clownschüler

Unser Clown hat einen Clownschüler. Er ist noch unsicher und fragt lieber nochmal nach, wann er seinen Witz erzählen soll. Nur wenn du ihm „ja“ als antwort gibst, fängt er damit an.

1. Öffne den Ordner „Zirkel“ und öffne die Datei „8. Clownschüler“ mit einem Doppelklick.

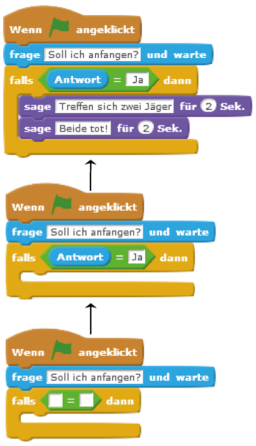



2. Versuche den Block nachzubauen:

? Der Block ist sehr kompliziert! Hier sind alle Schritte nacheinander zu sehen:








3. Klicke auf die grüne Flagge um das Programm zu starten!

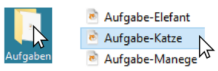
4. Klicke auf Datei und wähle „Speichern“ aus!

5. Du kannst das Fenster jetzt schließen.

## Aufgabe

In einer Pause schleicht sich eine Katze in die Manege und spielt mit einem Ball. Wenn die den Ball berührt, sagt sie „Hab ich dich!“

1. Öffne den Ordner „Aufgaben“ und öffne die Datei „Aufgabe-Katze“ mit einem Doppelklick.

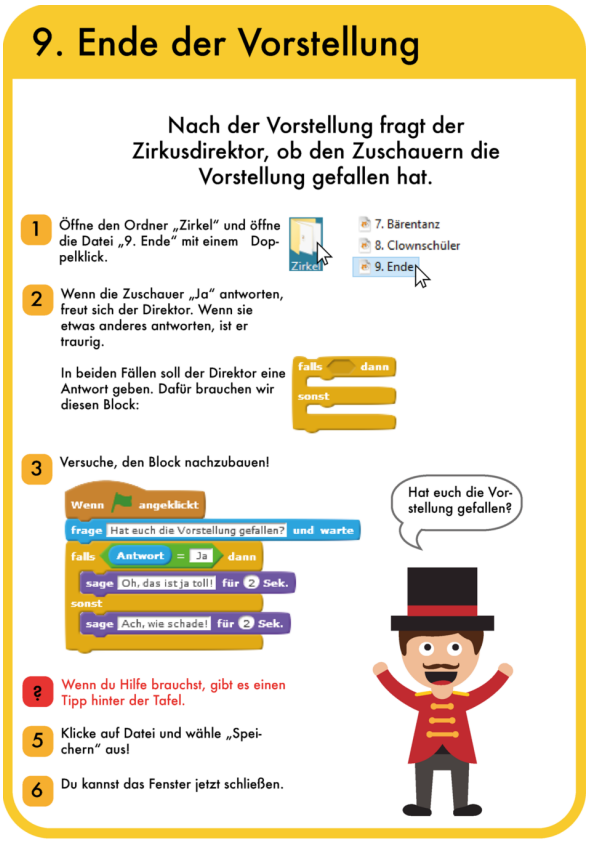



2. Versuche, die Situation nachzubauen. Benutze die folgenden Blöcke:

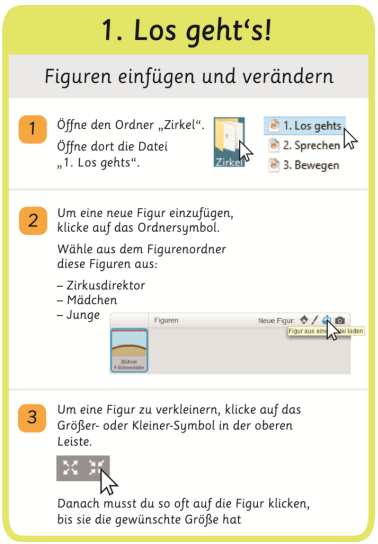

3. Um zu testen, ob das Programm funktioniert kannst du die Figuren so auf der Bühne verschieben, dass sie sich berühren.

4. Klicke auf Datei und wähle „Speichern“ aus!

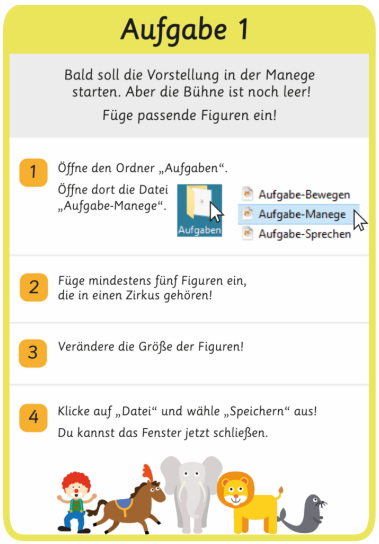
5. Du kannst das Fenster jetzt schließen.




### B.2.3.2 Zyklus 3 und Zyklus 4



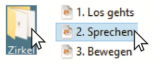




## 2. Sprechen

### Der Zirkusdirektor sagt etwas

1 Öffne den Ordner „Zirkel“.
Öffne dort die Datei „2. Sprechen“.




2 Klicke auf Aussehen!


3 Schiebe die Kachel sage Hello! für 2 Sek. insgesamt drei mal nach rechts.

So soll es aussehen:


4 Klicke auf die weißen Felder und ändere den Text des Direktors:


5 Klicke auf die grüne Flagge um das Programm zu starten!

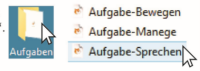
6 Klicke auf „Datei“ und wähle „Speichern“ aus!


7 Du kannst das Fenster jetzt schließen.

## Aufgabe 2

Der Clown soll einen Witz erzählen.
Schreibe ein passendes Programm!

1 Öffne den Ordner „Aufgaben“.
Öffne dort die Datei „Aufgabe-Sprechen“.



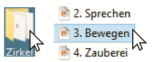

2 Lass den Clown einen kurzen Witz erzählen!

3 Klicke auf „Datei“ und wähle „Speichern“ aus!
Du kannst das Fenster jetzt schließen.


## 3. Bewegen

### Der Tiger soll sich bewegen

1 Öffne den Ordner „Zirkel“.
Öffne dort die Datei „3. Bewegen“.

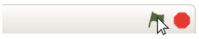



2 Klicke auf Bewegung!


3 Schiebe die Kachel gehe 10 er-Schritt nach rechts.

So soll es aussehen:


4 Klicke auf die grüne Flagge um das Programm zu starten!
Schaue was passiert, wenn du die Zahlen veränderst!



5 Klicke auf „Steuerung“!

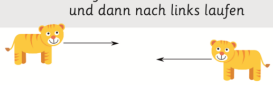

6 Schiebe die Kachel warte 1 Sek. nach rechts.

7 Baue so viele Blöcke aneinander, dass der Tiger von ganz links nach ganz rechts läuft!

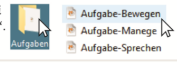

8 Klicke auf „Datei“ und wähle „Speichern“ aus!
Du kannst das Fenster jetzt schließen.

## Aufgabe 3

Der Tiger soll zuerst nach rechts und dann nach links laufen



1 Öffne den Ordner „Aufgaben“.
Öffne dort die Datei „Aufgabe-Bewegen“.




2 Lasse den Tiger einen Schritt nach rechts gehen!

3 Der Tiger muss sich jetzt umdrehen.
Überlege, welchen Bewegungs-Block du verwenden musst!

? Wenn du Hilfe brauchst, gibt es einen Tipp hinter der Tafel.

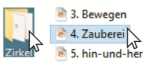
4 Lasse den Tiger einen Schritt nach links gehen!

5 Klicke auf „Datei“ und wähle „Speichern“ aus!
Du kannst das Fenster jetzt schließen.

## 4. Zauberei

### Der Zauberer hat einen neuen Trick

1 Öffne den Ordner „Zirkel“.
Öffne dort die Datei „4. Zauberei“.




2 Bei mehreren Figuren musst du zuerst auswählen, für welche Figur du ein Programm schreiben möchtest.

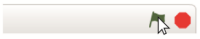
Klicke in der Figurenliste auf die Blumen!


3 Klicke auf „Aussehen“!


4 Benutze den Farbwechsel-Block!

So soll es aussehen:


5 Klicke auf die grüne Flagge um das Programm zu starten!



6 Wenn du auf den roten Knopf klickst, bekommen die Blumen ihre Ausgangsfarbe zurück.

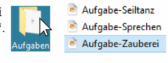
7 Probiere verschiedene Zahlen aus!

8 Klicke auf „Datei“ und wähle „Speichern“ aus!
Du kannst das Fenster jetzt schließen.

## Aufgabe 4

Überlege dir einen neuen Zaubertrick!

1 Öffne den Ordner „Aufgaben“.
Öffne dort die Datei „Aufgabe-Zauberei“.




2 Schaue dir die Aussehen-Blöcke genau an!
Hast du eine Idee für einen neuen Trick?

? Wenn du Hilfe brauchst, gibt es einen Tipp hinter der Tafel.

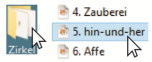
3 Baue das Programm für den neuen Trick!

4 Klicke auf „Datei“ und wähle „Speichern“ aus!
Du kannst das Fenster jetzt schließen.

## 5. Ewig hin und her

Mit einer Wiederholung läuft der Tiger ewig von links nach rechts

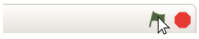
1 Öffne den Ordner „Zirkel“.
Öffne dort die Datei „5. hin-und-her“.




2 Baue diesen Block nach:


? Wenn du Hilfe brauchst, gibt es einen Tipp hinter der Tafel.

3 Klicke auf die grüne Flagge um das Programm zu starten!

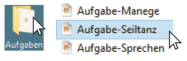


4 Klicke auf „Datei“ und wähle „Speichern“ aus!
Du kannst das Fenster jetzt schließen.

## Aufgabe 5

Lasse die Seiltänzerin balancieren!

1 Öffne den Ordner „Aufgaben“.
Öffne dort die Datei „Aufgabe-Seiltanz“.




2 Baue ein Programm, das die Tänzerin ewig von einem Ende des Seils zu anderen Ende laufen lässt!

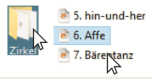
3 Versuche, die Tänzerin ganz langsam gehen zu lassen!

4 Klicke auf „Datei“ und wähle „Speichern“ aus!
Du kannst das Fenster jetzt schließen.

## 6. Fang den Affen

Mit der Hilfe einer Wiederholung können wir den Affen mit einer Banane ködern

1 Öffne den Ordner „Zirkel“.
Öffne dort die Datei „6. Affe“.

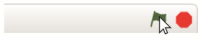



2 Wähle in der Figurenliste den Affen aus:

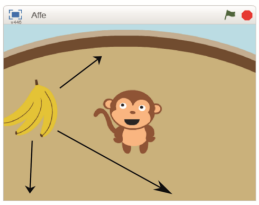

3 Baue diesen Block nach:


? Wenn du Hilfe brauchst, gibt es einen Tipp hinter der Tafel.

4 Klicke auf die grüne Flagge um das Programm zu starten!



5 Verschiebe die Banane auf der Bühne!
Melde dich, wenn du Hilfe brauchst!

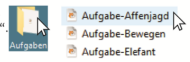



6 Klicke auf „Datei“ und wähle „Speichern“ aus!
Du kannst das Fenster jetzt schließen.

## Aufgabe 6

Schreibe ein Programm für den Zirkusdirektor!
Er soll automatisch zum Affen laufen.

1 Öffne den Ordner „Aufgaben“.
Öffne dort die Datei „Aufgabe-Affenjagd“.




2 Baue ein Programm, mit dem der Zirkusdirektor von alleine zum Affen läuft.

? Wenn du Hilfe brauchst, gibt es einen Tipp hinter der Tafel.

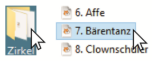
3 Klicke auf „Datei“ und wähle „Speichern“ aus!
Du kannst das Fenster jetzt schließen.



## 7. Bärentanz

Wenn wir den Bären seine „Kostüme“ wechseln lassen, sieht es aus, als würde er tanzen

1 Öffne den Ordner „Zirkel“.
Öffne dort die Datei „7. Bärentanz“.




2 Klicke auf „Kostüme“ um dir die verschiedenen Tanzfiguren des Bären anzuschauen!


3 Baue diesen Block nach:


4 Klicke auf die grüne Flagge um das Programm zu starten!

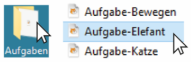
5 Benutze auch die anderen Kostüme!
Baue wie oben immer einen Warte-Block zwischen die Kostümwechsel.

6 Klicke auf „Datei“ und wähle „Speichern“ aus!
Du kannst das Fenster jetzt schließen.

## Aufgabe 7

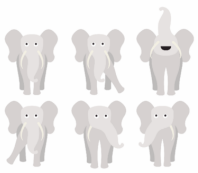
Auch der Elefant soll tanzen.
Schreibe ein passendes Programm!

1 Öffne den Ordner „Aufgaben“.
Öffne dort die Datei „Aufgabe-Elefant“.




2 Schaue dir die Kostüme des Elefanten an und baue sie zu einem Tanz zusammen.

3 Klicke auf „Datei“ und wähle „Speichern“ aus!
Du kannst das Fenster jetzt schließen.

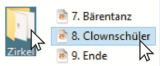


## 8. Clownschüler

Mit einer Bedingten Anweisung können wir den Clownschüler fragen lassen, ob er einen Witz erzählen soll.

Nur wenn man ihm „ja“ als Antwort gibt, fängt er damit an

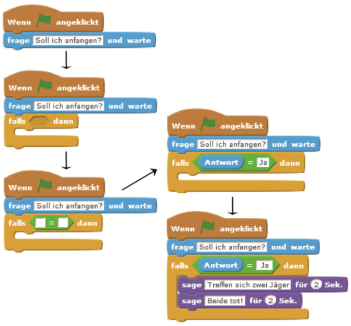
1 Öffne den Ordner „Zirkel“.
Öffne dort die Datei „8. Clownschüler“.




2 Baue diesen Block nach:


? Hilfe gibt es bei Punkt 3 auf der Rückseite!

3 Hier sind alle Schritte nacheinander:



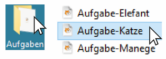
4 Klicke auf die grüne Flagge um das Programm zu starten!

5 Klicke auf „Datei“ und wähle „Speichern“ aus!
Du kannst das Fenster jetzt schließen.

## Aufgabe 8

Schreibe ein Programm für die Katze!
Immer wenn sie den Ball berührt, soll sie „Hab ich dich!“ sagen.

1 Öffne den Ordner „Aufgaben“.
Öffne dort die Datei „Aufgabe-Katze“.




2 Schreibe ein passendes Programm für die Katze!
Benutze diese Blöcke:

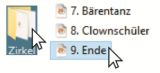

3 Verschiebe die Figuren auf der Bühne und teste, ob das Programm funktioniert!

4 Klicke auf „Datei“ und wähle „Speichern“ aus!
Du kannst das Fenster jetzt schließen.

## 9. Ende der Vorstellung

Nach der Vorstellung fragt der Direktor die Zuschauer, ob der Zirkus ihnen gefallen hat.

1 Öffne den Ordner „Zirkel“.
Öffne dort die Datei „9. Ende“.




2 Wenn die Zuschauer „Ja“ antworten, freut sich der Direktor.
Wenn sie etwas anderes antworten, ist er traurig.
In beiden Fällen soll der Direktor eine Antwort geben. Dafür brauchen wir diesen Block:


3 Baue diesen Block nach:


? Wenn du Hilfe brauchst, gibt es einen Tipp hinter der Tafel.

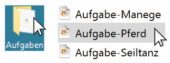
4 Klicke auf die grüne Flagge um das Programm zu starten!
Probiere verschiedene Antworten aus!

5 Klicke auf „Datei“ und wähle „Speichern“ aus!
Du kannst das Fenster jetzt schließen.

## Aufgabe 9

Zum Abschluss zeigt das Pferd noch einen seiner besten Tricks:
Es wechselt die Farbe und macht einen Salto.

1 Öffne den Ordner „Aufgaben“.
Öffne dort die Datei „Aufgabe-Pferd“.




2 Baue ein Programm für das Pferd!
Wenn man es mit dem Mauszeiger berührt, soll es dich drehen.
Wenn man es nicht berührt, soll es die Farbe wechseln.

? Tipp:
Du brauchst dafür diesen Block:

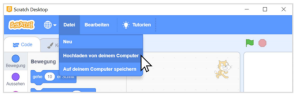

3 Klicke auf „Datei“ und wähle „Speichern“ aus!
Du kannst das Fenster jetzt schließen.

### B.2.3.3 Nach Zyklus 4

## 1. Einlass

Figuren einfügen und verändern.

1 Öffne das Programm Scratch auf deinem Computer.
Klicke auf „Datei“ und wähle „Hochladen von deinem Computer“ aus.



2 Öffne den Ordner „Zirkel“.
Öffne dort die Datei „1_Einlass“.


3 Um eine neue Figur einzufügen, gehe mit der Maus zu dem Button „Figur wählen“ und klicke auf „Figur hochladen“.


4 Füge diese drei Figuren aus dem Figurenordner hinzu:
– Zirkusdirektor
– Mädchen
– Junge

5 Hier kannst du die Größe der Figuren ändern:

Probiere es gleich aus!
Zahlen größer als 100: Figur wird größer
Zahlen kleiner als 100: Figur wird kleiner

6 Wenn du mit der rechten Maustaste auf die Figur klickst, öffnet sich ein Menü.
Hier kannst du Figuren löschen oder duplizieren (verdoppeln).
Verdopple die Kinder, so dass viele Kinder in der Manege stehen.


7 Klicke auf „Datei“ und wähle „Auf deinem Computer speichern“ aus.

## Aufgabe 1: Manege

Bald soll die Vorstellung in der Manege starten. Aber die Bühne ist noch leer!

1 Öffne in Scratch die Datei „1_Aufgabe_Manege“ aus dem Ordner „Aufgaben“.


2 Füge mindestens fünf Figuren ein, die in einen Zirkus gehören!

3 Verändere die Größe der Figuren!

4 Speichere dein Programm!

## 2. Begrüßung

Der Zirkusdirektor sagt etwas.

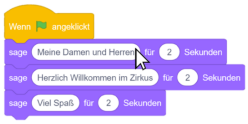
1 Öffne in Scratch die Datei „2_Begrüßung“ aus dem Ordner „Zirkel“.


2 Klicke auf den Blockbereich „Aussehen“.

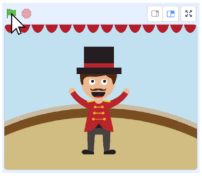
3 Schiebe den Block  insgesamt drei mal nach rechts.

So soll es aussehen:


4 Klicke auf die weißen Felder und ändere den Text des Direktors:




5 Klicke auf die grüne Flagge um das Programm zu starten!



6 Klicke auf „Datei“ und wähle „Auf deinem Computer speichern“ aus.

## Aufgabe 2: Witz

Der Clown soll einen Witz erzählen.

1 Öffne in Scratch die Datei „2_Aufgabe_Witz“ aus dem Ordner „Aufgaben“.


2 Lass den Clown einen kurzen Witz erzählen!

3 Speichere dein Programm!


## 3. Tigerauftritt

Der Tiger soll sich bewegen.

1 Öffne in Scratch die Datei „3_Tigerauftritt“ aus dem Ordner „Zirkel“.


2 Klicke auf den Blockbereich „Bewegung“.

3 Schiebe den Block  nach rechts.

So soll es aussehen:


4 Klicke auf die grüne Flagge um das Programm zu starten!

Schaue was passiert, wenn du die Zahlen veränderst.

5 Klicke auf den Blockbereich „Steuerung“.

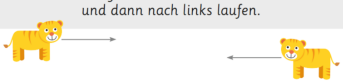

6 Schiebe den Block  nach rechts.

7 Baue so viele Blöcke aneinander, dass der Tiger von ganz links nach ganz rechts läuft!


8 Klicke auf „Datei“ und wähle „Auf deinem Computer speichern“ aus.

## Aufgabe 3: Tigergang

Der Tiger soll zuerst nach rechts und dann nach links laufen.



1 Öffne in Scratch die Datei „3_Aufgabe_Tigergang“ aus dem Ordner „Aufgaben“.


2 Lass den Tiger nach rechts gehen!

3 Der Tiger muss sich jetzt umdrehen.

Welchen Bewegungs-Block musst du verwenden?

4 Lass den Tiger zurück nach links gehen!

5 Speichere dein Programm!

## 4. Zauberei

Der Zauberer hat einen neuen Trick.

1 Öffne in Scratch die Datei „4_Zauberei“ aus dem Ordner „Zirkel“.


2 Bei mehreren Figuren musst du auswählen, welche Figur du programmieren möchtest.

Klicke in der Figurenleiste auf die Blumen:


3 Klicke auf den Blockbereich „Aussehen“.

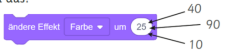
4 Benutze den Farbwechsel-Block!

So soll es aussehen:


5 Klicke auf die grüne Flagge um das Programm zu starten!

6 Wenn du auf den roten Knopf klickst, bekommen die Blumen ihre Ausgangsfarbe zurück.

7 Probiere im Farbwechsel-Block verschiedene Zahlen aus!




8 Klicke auf „Datei“ und wähle „Auf deinem Computer speichern“ aus.

## Aufgabe 4: Zaubertrick

Überlege dir einen neuen Zaubertrick!

1 Öffne in Scratch die Datei „4_Aufgabe_Zaubertrick“ aus dem Ordner „Aufgaben“.


2 Schaue dir die Aussehen-Blöcke genau an!

Hast du eine Idee für einen neuen Trick?

3 Baue das Programm für den neuen Zaubertrick zusammen!

4 Speichere dein Programm!

## 5. Galopp

Mit einer Wiederholung galoppiert das Pferd von links nach rechts.

1 Öffne in Scratch die Datei „5_Galopp“ aus dem Ordner „Zirkel“.


2 Baue dieses Programm nach:


Fange mit dem orangenen Wiederholungsblock an und füge dann die blauen Bewegungsblöcke ein!

3 Klicke auf die grüne Flagge um das Programm zu starten!

4 Klicke auf „Datei“ und wähle „Auf deinem Computer speichern“ aus.

## Aufgabe 5: Seiltanz

Lass die Seiltänzerin balancieren!

1 Öffne in Scratch die Datei „5_Aufgabe_Seiltanz“ aus dem Ordner „Aufgaben“.


2 Baue ein Programm, das die Tänzerin ewig von einem Ende des Seils zum anderen Ende laufen lässt!

3 Versuche, die Tänzerin ganz langsam gehen zu lassen!

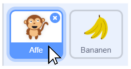
4 Speichere dein Programm!

## 6. Affenköder

Mit der Hilfe einer Wiederholung können wir den Affen mit einer Banane ködern.

1 Öffne in Scratch die Datei „6_Affenköder“ aus dem Ordner „Zirkel“.


2 Wähle in der Figurenleiste den Affen aus:

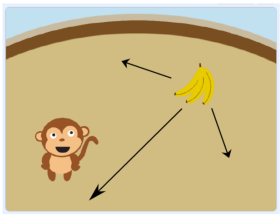



3 Baue dieses Programm nach:

Klicke im drehe-Block auf den kleinen Pfeil, um die Banane auszuwählen:


4 Klicke auf die grüne Flagge um das Programm zu starten!

5 Verschiebe die Banane auf der Bühne und schaue, was passiert!
Melde dich, wenn du Hilfe brauchst!



6 Klicke auf „Datei“ und wähle „Auf deinem Computer speichern“ aus.

## Aufgabe 6: Affenverfolgung

Schreibe ein Programm für den Zirkusdirektor! Er soll automatisch zum Affen laufen.

1 Öffne in Scratch die Datei „6_Aufgabe_Affenverfolgung“ aus dem Ordner „Aufgaben“.

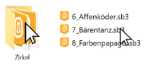

2 Baue ein Programm für den Zirkusdirektor, mit dem er von alleine zum Affen läuft.

3 Speichere dein Programm!


## 7. Bärentanz

Wenn der Bär seine „Kostüme“ wechselt, sieht es aus, als würde er tanzen.

1 Öffne in Scratch die Datei „7_Bärentanz“ aus dem Ordner „Zirkel“.




2 Klicke auf „Kostüme“ um dir die verschiedenen Tanzposen des Bären anzuschauen:


3 Baue dieses Programm nach:


4 Klicke auf die grüne Flagge um das Programm zu starten!

5 Benutze auch die anderen Kostüme!
Baue wie oben immer einen Warte-Block zwischen die Kostümwechsel.


6 Klicke auf „Datei“ und wähle „Auf deinem Computer speichern“ aus.

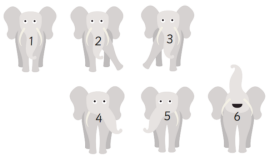
## Aufgabe 7: Elefantentanz

Auch der Elefant soll tanzen. Schreibe ein passendes Programm!

1 Öffne in Scratch die Datei „7_Aufgabe_Elefantentanz“ aus dem Ordner „Aufgaben“.


2 Schaue dir die Kostüme des Elefanten an. Programmiere ihn so, dass er tanzt!

3 Speichere dein Programm!



## 8. Farbenpapagei

Wenn der Papagei einen Luftballon berührt, sagt er, welche Farbe dieser hat.

1 Öffne in Scratch die Datei „8_Farbenpapagei“ aus dem Ordner „Zirkel“.


2 Programmiere den Papagei:



3 Um das Blau des Ballons auszuwählen, musst du in den Farbkreis (1) klicken.

Es öffnet sich ein Menü. Wähle die Pipette (2) ganz unten aus. Wenn du jetzt auf den Ballon klickst, übernimmt es dessen Farbe.

4 Ergänze die Blöcke für den gelben und den roten Ballon:

5 Klicke auf die grüne Flagge um das Programm zu starten!

Versetze den Papagei auf der Bühne, dass er immer einen anderen Luftballon berührt.

6 Klicke auf „Datei“ und wähle „Auf deinem Computer speichern“ aus.

## Aufgabe 8: Ballspiel

Immer wenn die Katze den Ball berührt, soll sie „Hab ich dich!“ sagen.

1 Öffne in Scratch die Datei „8_Aufgabe_Ballspiel“ aus dem Ordner „Aufgaben“.


2 Schreibe ein passendes Programm für die Katze! Benutze diese Blöcke:


3 Speichere dein Programm!


## 9. Hasenzauber

Der Hase führt einen Zaubertrick auf. Er verschwindet und taucht wieder auf.

1 Öffne in Scratch die Datei „9_Hasenzauber“ aus dem Ordner „Zirkel“.


2 Klicke auf den Blockbereich „Ereignisse“.


3 Schiebe diese drei Startereignisse nach rechts:


4 Ergänze die Startereignisse:


5 Klicke auf die grüne Flagge, dass der Hase verschwindet.

Tippe auf die Leertaste, dass er wieder auftaucht.

Klicke den Hasen auf der Bühne an, dass er seinen Satz sagt.

6 Klicke auf „Datei“ und wähle „Auf deinem Computer speichern“ aus.

## Aufgabe 9: Muskelauftritt

Wenn man die Pfeiltasten auf der Tastatur tippt, bewegt sich der Muskelmann.

1 Öffne in Scratch die Datei „9_Aufgabe_Muskelauftritt“ aus dem Ordner „Aufgaben“.


2 Programmiere den Muskelmann.

Benutze vier mal das Startereignis  und wähle darin die verschiedenen Pfeiltasten aus.

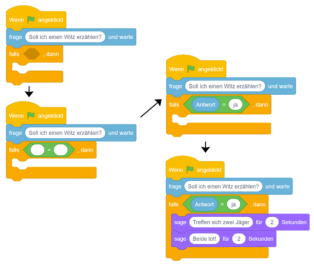

3 Speichere dein Programm!

## 10. Clownschüler

Der Clownschüler fragt, ob er einen Witz erzählen soll. Nur wenn man ihm „ja“ als Antwort gibt, fängt er an.

1 Öffne in Scratch die Datei „10_Clownschüler“ aus dem Ordner „Zirkel“.


2 Programmiere den Clown:



3 Klicke auf die grüne Flagge um das Programm zu starten!

Probiere aus, was passiert, wenn du „ja“ und „nein“ in das Antwortfeld eintippst.

Bestätige die Antwort mit einem Klick auf den Haken.


4 Klicke auf „Datei“ und wähle „Auf deinem Computer speichern“ aus.

## Aufgabe 10: Ende

Nach der Vorstellung fragt der Zirkusdirektor, ob der Zirkus dir gefallen hat.

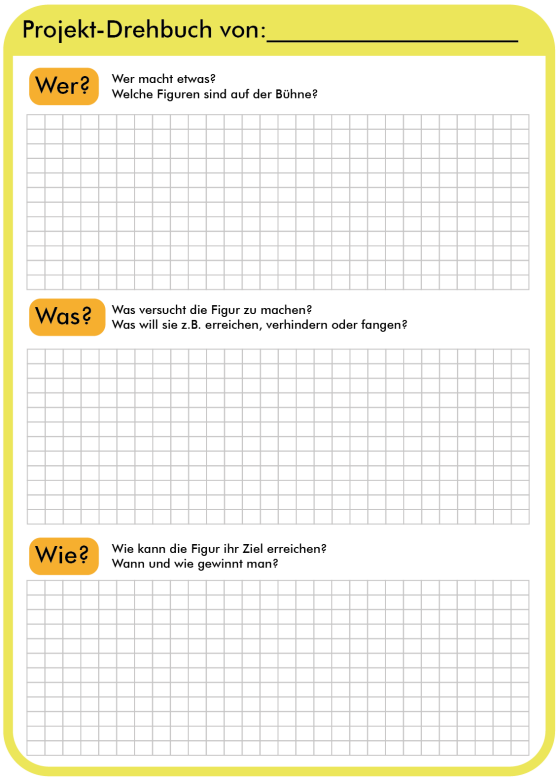
1 Öffne in Scratch die Datei „10_Aufgabe_Ende“ aus dem Ordner „Aufgaben“.


2 Wenn du „ja“ antwortest, freut sich der Direktor. Schreibe ein passendes Programm!

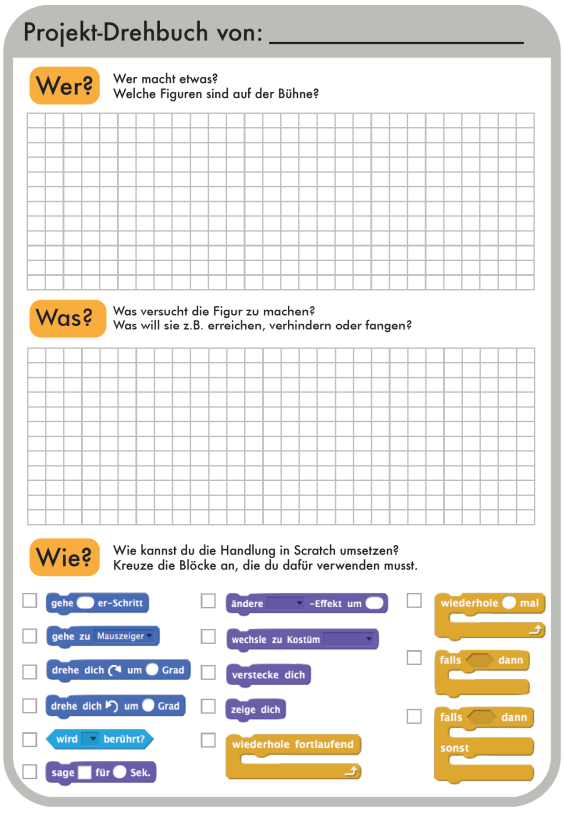

3 Speichere dein Programm!

## B.2.4 Planungsvorlage für eigene Programmierprojekte

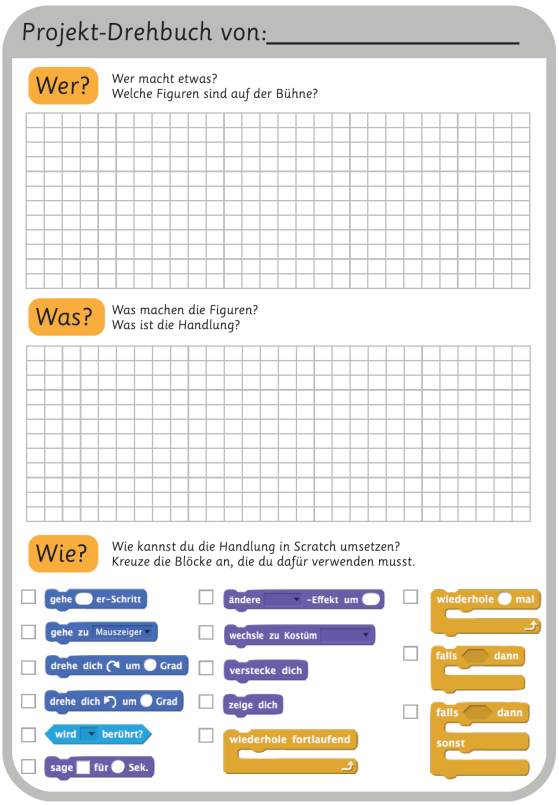
### B.2.4.1 Zyklus 1 (Pilotstudie)



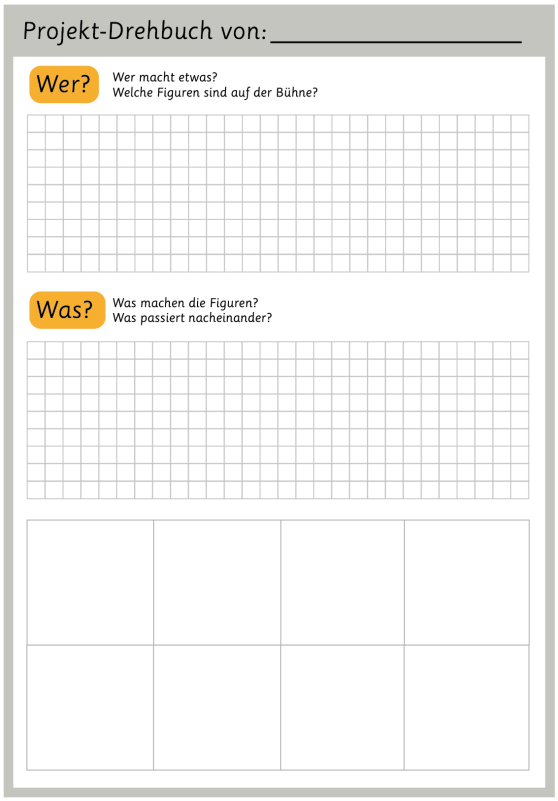

### B.2.4.2 Zyklus 2



### B.2.4.3 Zyklus 3 und Zyklus 4




### B.2.4.4 Nach Zyklus 4



# C Fortbildungskonzept

## C.1 Didaktische Konzeption

### C.1.1 Herleitung des Grobentwurfs

*Tabelle C.1 Tabellarische Auflistung der einzelnen Fortbildungsschritte inklusive der Herleitung aus den Umsetzungsprinzipien und der Verbindung zu den Zielen des Fortbildungskonzepts*

| Element des Grobentwurfs | Begründung (aus den Umsetzungsprinzipien der *Design-Principles* mit Bezug zu Lernaktivitäten) | Informatische Kompetenzen | Informatikdidaktische Kompetenzen |
|---|---|---|---|
| (1) Thematischer Einstieg | | | |
| 1a) Aktivierung von Vorwissen der Lehrkräfte | | | |
| 1b) Kurzvorstellung der Unterrichtssequenz und deren didaktischer Relevanz | | | |

| (2) Programmieren in der Grundschule | | | |
|---|---|---|---|
| 2a) Bearbeiten von ausgewählten Aufgaben der Unterrichtssequenz | LFB-DP 3: „Die Lehrkräfte erhalten im Rahmen der Fortbildung die Möglichkeit, dem Lernpfad der Schülerinnen und Schüler zu folgen …“<br><br>LFB-DP 6: „In der Fortbildung werden Möglichkeiten zur Anknüpfung an andere Fächer oder auf didaktischer Ebene zur Grundschuldidaktik bzw. fachdidaktischen Prinzipien bestehender Grundschulfächer aufgezeigt. Im Rahmen der Fortbildung gibt es immer wieder Phasen, in denen *unplugged* gearbeitet wird.“ | Die Lehrkräfte führen Algorithmen in ihrer Lebenswelt bzw. der Lebenswelt der Schülerinnen und Schüler aus<br><br>Die Lehrkräfte beschreiben Algorithmen in Alltagssprache<br><br>Die Lehrkräfte entwerfen, realisieren und testen Algorithmen mit den algorithmischen Grundbausteinen Anweisung, Sequenz, Wiederholung und Verzweigung<br><br>Die Lehrkräfte programmieren ein für die Grundschule altersentsprechendes Informatiksystem | Die Lehrkräfte sind in der Lage, Konzepte für das Erlernen einer einfachen Programmiersprache mit einem spielerischen, auch *Unplugged*-Ansatz, praktisch umzusetzen<br><br>Die Lehrkräfte sind in der Lage, Schülerinnen und Schüler für das Programmierenlernen zu motivieren sowie motivierende Aktivierungsmethoden und informatiktypische Sozialformen kompetenzfördernd einzusetzen<br><br>Die Lehrkräfte sind in der Lage, Aufgabenstellungen und Lerninhalte des Programmierens kindgemäß und auf unterschiedliche Weise vereinfacht, aber korrekt darzustellen sowie abstrakte informatische Konzepte durch verschiedene Beispiele und mittels spielerischer und explorativer Erkundungsanlässe für die Schülerinnen und Schüler erfahrbar zu machen<br><br>Die Lehrkräfte regen die Schülerinnen und Schüler zum selbstständigen Lernen und zur eigenständigen Begutachtung ihrer Lernergebnisse an und fördern so deren Selbstwirksamkeit<br><br>Die Lehrkräfte fördern durch Gruppengespräche die Reflexion der Schülerinnen und Schüler über ihr eigenes Handeln und ermöglichen so positives Feedback über die erzielten Ergebnisse |

| | | | |
|---|---|---|---|
| 2b) Reflexion des eigenen Lernprozesses und Sichtung der Ergebnisse von Schülerinnen und Schülern | LFB-DP 3: „Zusätzlich werden den Lehrkräften beispielhafte Arbeitsergebnisse von Schülerinnen und Schülern zur Verfügung gestellt."<br><br>LFB-DP 5: „Innerhalb der Fortbildungsveranstaltungen werden Reflexionsphasen fest eingeplant." | | Die Lehrkräfte sind in der Lage, typische Präkonzepte, Verstehens- und Lernbarrieren der Schülerinnen und Schüler in Bezug auf das Programmieren zu identifizieren und entsprechende Formalisierungen, Abstraktionen und Interventionsmöglichkeiten einzusetzen<br><br>Die Lehrkräfte sind in der Lage, individuelle Leistungsstände der Schülerinnen und Schüler zu identifizieren und darauf durch situationsangemessenes Handeln zu reagieren |
| (3) Fachliche Inhalte | | | |
| 3a) Vorstellung des Algorithmusbegriffs | LFB-DP 6: „In der Fortbildung werden Möglichkeiten zur Anknüpfung an andere Fächer oder auf didaktischer Ebene zur Grundschuldidaktik bzw. fachdidaktischen Prinzipien bestehender Grundschulfächer aufgezeigt. Im Rahmen der Fortbildung gibt es immer wieder Phasen, in denen *unplugged* gearbeitet wird." | Die Lehrkräfte entwerfen, realisieren und testen Algorithmen mit den algorithmischen Grundbausteinen Anweisung, Sequenz, Wiederholung und Verzweigung | |
| 3b) Einführung der algorithmischen Grundstrukturen sowie des Variablenkonzepts und Bearbeiten passender Programmieraufgaben in Scratch | LFB-DP 4: „Um eine gewisse Sicherheit im Programmieren auszubilden, erhalten die Lehrkräfte mehrfach die Möglichkeit, selbst in Scratch zu programmieren. Dabei werden einerseits die algorithmischen Grund-strukturen thematisiert, andererseits werden daran fachdidaktische Prinzipien aufgezeigt." | Die Lehrkräfte programmieren ein für die Grundschule altersentsprechendes Informatiksystem<br><br>Die Lehrkräfte verwenden Variablen und Wertzuweisungen<br><br>Die Lehrkräfte ordnen Bestandteile eines Informatiksystems der Eingabe, der Verarbeitung und der Ausgabe zu | Die Lehrkräfte können auch in Gruppenarbeits-situationen mit Informatiksystemen auf Lernschwierigkeiten einzelner Schülerinnen und Schüler eingehen und gezielte individuelle Hilfestellungen geben |

| | | | |
|---|---|---|---|
| (4) Reflexion | | | |
| 4a) Sammeln von Möglichkeiten und Maßnahmen zur Implementierung an den Schulen der Lehrkräfte und Thematisieren von Herausforderungen | LFB-DP 5: „Innerhalb der Fortbildungsveranstaltungen werden Reflexionsphasen fest eingeplant.“ | | |
| (5) Erfahrungs-austausch | | | |
| 5a) Austausch von Erfahrungen in der Umsetzung der Unterrichtssequenz | LFB-DP 5: „Zudem erhalten die Lehrkräfte die Möglichkeit, sich über die Erprobungen der Inhalte in ihrem Unterricht auszutauschen …“ | | Die Lehrkräfte sind in der Lage, typische Präkonzepte, Verstehens- und Lernbarrieren der Schülerinnen und Schüler in Bezug auf das Programmieren zu identifizieren und entsprechende Formalisierungen, Abstraktionen und Interventionsmöglichkeiten einzusetzen |
| 5b) Präsentation von Best-Practice-Beispielen | | | |

| | | | |
|---|---|---|---|
| (6) Fachdidaktische Grundlagen | | | |
| 6a) Kennenlernen von Aufgabenformaten in Scratch | LFB-DP 4: „Um eine gewisse Sicherheit im Programmieren auszubilden, erhalten die Lehrkräfte mehrfach die Möglichkeit, selbst in Scratch zu programmieren. Dabei werden einerseits die algorithmischen Grundstrukturen thematisiert, andererseits werden daran fachdidaktische Prinzipien aufgezeigt.“ | | Die Lehrkräfte sind in der Lage, Aufgabenstellungen und Lerninhalte des Programmierens kindgemäß und auf unterschiedliche Weise vereinfacht, aber korrekt darzustellen sowie abstrakte informatische Konzepte durch verschiedene Beispiele und mittels spielerischer und explorativer Erkundungsanlässe für die Schülerinnen und Schüler erfahrbar zu machen |
| 6b) Kennenlernen von Modellen über den Verlauf des Programmierenlernen | | | Die Lehrkräfte sind in der Lage, typische Präkonzepte, Verstehens- und Lernbarrieren der Schülerinnen und Schüler in Bezug auf das Programmieren zu identifizieren und entsprechende Formalisierungen, Abstraktionen und Interventionsmöglichkeiten einzusetzen<br><br>Die Lehrkräfte sind in der Lage, individuelle Leistungsstände der Schülerinnen und Schüler zu identifizieren und darauf durch situationsangemessenes Handeln zu reagieren |
| 6c) Kennenlernen weiterer kindgemäßer Software und altersgemäßer Informatiksysteme zum Programmieren | | | Die Lehrkräfte sind in der Lage, kindgemäße Software und einfache altersgemäße Informatiksysteme für das Programmieren im Primarbereich auszuwählen und sinnvoll in Lernszenarien zu integrieren |
| (7) Erfahrungsaustausch | | | |
| 7a) Präsentation der von den Lehrkräften entwickelten Lehr-Lernaktivitäten | LFB-DP 5: „Zudem erhalten die Lehrkräfte die Möglichkeit, sich über die Erprobungen der Inhalte in ihrem Unterricht auszutauschen …“ | | |

| | | | |
|---|---|---|---|
| (8) Fachdidaktische Vertiefung | | | |
| 8a) Kennenlernen von Programmierkontexten aus bestehenden Fächern der Grundschule | LFB-DP 4: „Um eine gewisse Sicherheit im Programmieren auszubilden, erhalten die Lehrkräfte mehrfach die Möglichkeit, selbst in Scratch zu programmieren. Dabei werden einerseits die algorithmischen Grundstrukturen thematisiert, andererseits werden daran fachdidaktische Prinzipien aufgezeigt.“<br><br>LFB-DP 6: „In der Fortbildung werden Möglichkeiten zur Anknüpfung an andere Fächer oder auf didaktischer Ebene zur Grundschuldidaktik bzw. fachdidaktischen Prinzipien bestehender Grundschulfächer aufgezeigt.“ | | Die Lehrkräfte sind in der Lage, Aufgabenstellungen und Lerninhalte des Programmierens kindgemäß und auf unterschiedliche Weise vereinfacht, aber korrekt darzustellen sowie abstrakte informatische Konzepte durch verschiedene Beispiele und mittels spielerischer und explorativer Erkundungsanlässe für die Schülerinnen und Schüler erfahrbar zu machen |
| 8b) Austausch über gute Programmierpraktiken | LFB-DP 5: „Innerhalb der Fortbildungsveranstaltungen werden Reflexionsphasen fest eingeplant.“ | | Die Lehrkräfte sind in der Lage, individuell Leistungsstände der Schülerinnen und Schüler zu identifizieren und darauf durch situationsangemessenes Handeln zu reagieren |
| 8c) Auseinandersetzen mit Aspekten des *Computational Thinking* | LFB-DP 6: „In der Fortbildung werden Möglichkeiten zur Anknüpfung an andere Fächer oder auf didaktischer Ebene zur Grundschuldidaktik bzw. fachdidaktischen Prinzipien bestehender Grundschulfächer aufgezeigt. Im Rahmen der Fortbildung gibt es immer wieder Phasen, in denen *unplugged* gearbeitet wird.“ | Die Lehrkräfte wenden Aspekte des *Computational Thinking* an, z. B. *Decomposition*, *Abstraction*, oder *Patterns/ Generalization* | |

## C.1.2 Darstellung des überarbeiteten Fortbildungskonzepts

### C.1.2.1 Variante mit zwei Praxisphasen

**(1) Thematischer Einstieg**

a) Aktivierung von Vorwissen der Lehrkräfte
b) Kurzvorstellung der Unterrichtssequenz und deren didaktischer Relevanz

Umsetzung:

- Begriffsassoziationen:
  Die Lehrkräfte ergänzen den Satz „Programmieren ist für mich ...“ auf Papierstreifen. Im Anschluss lesen sie diese vor. Die Kursleitung greift die Äußerungen der Lehrkräfte auf und präsentiert Assoziationen von Schülerinnen und Schülern mit dem Begriff *Programmieren*.
- Didaktische Relevanz:
  Die Kursleitung stellt den grundlegenden Aufbau der Unterrichtssequenz vor. Sie gibt einen Einblick in die Potenziale und Chancen der Programmierung in der Grundschule.

**(2) Programmieren in der Grundschule**

a) Bearbeiten von ausgewählten Aufgaben der Unterrichtssequenz (LFB-DP 3, LFB-DP 6)
b) Reflexion des eigenen Lernprozesses und Sichtung der Ergebnisse von Schülerinnen und Schülern (LFB-DP 3, LFB-DP 5)

Umsetzung:

- Unterrichtsmaterialien erproben:
  Die Lehrkräfte bekommen die Möglichkeit, die Perspektive der Schülerinnen und Schüler einzunehmen, indem sie ausgewählte Unterrichtsmaterialien selbst bearbeiten. Die Kursleitung zeigt die Aufgaben in einer Präsentation und erläutert die Aufgabenstellung sowie mögliche Anknüpfungspunkte an den Lehrplan der Grundschule. Die Kursleitung stellt zudem die Oberfläche von Scratch vor. Sie leistet während der ganzen Fortbildungsphase Hilfestellung, so wie sie auch Schülerinnen und Schülern helfen würde. Im Anschluss wird im Plenum thematisiert, welche Schwierigkeiten dabei auftraten und welche Erkenntnisse gewonnen wurden.
- Realistische Eindrücke:
  Wo es sich anbietet, werden kurze Videosequenzen aus dem Unterrichtsgeschehen gezeigt, die illustrieren, wie sich die Schülerinnen und Schüler während einer Übung verhalten bzw. wie die Lehrkraft die Übung durchführt. Hierfür eignet sich z. B. die Übung, in der die Schülerinnen und Schüler den *Lehrer-Roboter* programmieren. Zusätzlich werden im Laufe der Fortbildung Arbeitsergebnisse von Schülerinnen und Schülern gezeigt – aus den *Unplugged*- sowie den Scratch-Phasen. So werden bspw. Ergebnisse offener Programmieraufgaben präsentiert, um den Lehrkräften eine Vorstellung davon zu vermitteln, was Schülerinnen und Schüler leisten können.

**(3) Fachliche Inhalte**

a) Vorstellung des Algorithmusbegriffs (LFB-DP 6)
b) Einführung der algorithmischen Grundstrukturen sowie des Variablenkonzepts und Bearbeiten passender Programmieraufgaben in Scratch (LFB-DP 4)

Umsetzung:

- Bilddiktat:
  Als Einstieg in die Thematik *Algorithmus* wird eine Variante der *Stillen Post* gespielt. Jede Lehrkraft malt oben auf ein leeres Blatt Papier ein einfaches Bild und gibt es nach rechts an die Nachbarin bzw. den Nachbar weiter. Dieser beschreibt unter dem Bild, was er darauf sieht und knickt das gemalte Bild nach hinten um. Danach wird das Blatt erneut nach rechts weitergegeben, sodass nun jeder die Beschreibung eines Bildes vor sich sieht. Zu dieser Beschreibung wird nun wieder ein Bild gemalt. Im Anschluss falten die Lehrkräfte die Blätter auf und vergleichen, wie ähnlich sich die beiden Bilder sehen oder worin sie sich unterscheiden. Falls die Bilder sehr unterschiedlich aussehen, können sie untersuchen, an was es liegt – wurde nicht genau beschrieben oder nicht genau gelesen? Anhand der Übung wird der Unterschied von Alltagsalgorithmen zu einem Algorithmus im eigentlichen Sinn herausgestellt. Eine Definition des Begriffs wird präsentiert und die einzelnen Elemente besprochen.

- Fachinhalte:
  Die algorithmischen Grundstrukturen – Anweisung, Sequenz, Wiederholung und bedingte Anweisung – werden zunächst jeweils in Pseudocode sowie Blöcken der Programmiersprache Scratch beschrieben und danach in Programmieraufgaben unterschiedlicher Schwierigkeit vertieft (siehe Abschnitt C.2.1). Zusätzlich zu den algorithmischen Grundstrukturen wird das Variablenkonzept behandelt. Der theoretische Input wird dabei möglichst kurzgehalten – der Schwerpunkt liegt auf dem eigenständigen Bearbeiten der Aufgaben. Diese können zum Großteil auch im Unterricht eingesetzt werden.

- Fehlvorstellungen vermeiden:
  Um Fehlvorstellungen zum Ablauf einzelner algorithmischer Strukturen früh abzufangen, werden Codebeispiele in Scratch gemeinsam schrittweise gelesen und nachvollzogen. Hierbei wird bspw. thematisiert, was passiert, wenn das Programm unter verschiedenen Voraussetzungen, z. B. Positionen einer Figur auf der Bühne, gestartet wird.

**(4) Reflexion**

a) Sammeln von Möglichkeiten und Maßnahmen zur Implementierung an den Schulen der Lehrkräfte und Thematisieren von Herausforderungen (LFB-DP 5)

Umsetzung:

- Reflexionsphasen initiieren:
  Anhand von Plakatwänden wird thematisiert, wie sich die Lehrkräfte die Umsetzung der Themen Algorithmen und Programmieren in ihrem eigenen Unterricht vorstellen können. Diese werden im Raum verteilt und die Lehrkräfte können darauf Ideen, Inspirationen und Gedanken zu verschiedenen Themen sammeln, z. B. zu möglichen kurz-, mittel- und längerfristige Umsetzungen, Herausforderungen oder dem Bild des Programmierens und wie sich dieses bereits verändert hat. Die Kursleitung moderiert die Sichtung und Zusammenschau der Ergebnisse.

---

*1. Praxisphase:*
*Erprobungen der Unterrichtssequenz oder von ausgewählten Teilen der Unterrichtssequenz*

Zwischen dem ersten und zweiten Fortbildungsblock liegen mehrere Wochen. Die Lehrkräfte sind aufgefordert, mindestens ein Element der Unterrichtssequenz in ihrem Unterricht auszuprobieren. Druckvorlagen der Unterrichtsmaterialien, Scratch-Dateien sowie ein Satz haptischer Scratch-Blöcke und eine Programmierunterlage aus Filz werden den Lehrkräften zur Verfügung gestellt. Bei Fragen oder Problemen können sie sich per Mail oder Telefon an die Kursleitung wenden.

---

**(5) Erfahrungsaustausch**

a) Austausch von Erfahrungen in der Umsetzung der Unterrichtssequenz (LFB-DP 5)
b) Präsentation von *Best-Practice*-Beispielen (LFB-DP 5)

Umsetzung:

- Erfahrungsaustausch:
  In der Fortbildungsveranstaltung, die sich an die eingeforderte Erprobungsphase anschließt, wird ein Erfahrungsaustausch zwischen den Lehrkräften eingeplant. Auf verschiedenen Plakatwänden zu den einzelnen Elementen der Unterrichtssequenz sammeln die Lehrkräfte stichpunktartig, in welchem fachlichen Kontext sie etwas an ihrer Schule umsetzen konnten, welche Anpassungen sie vornahmen, wie die Reaktionen der Kinder ausfielen und welche Probleme auftraten. Die Ergebnisse sowie *Best-Practice*-Beispiele aus den Schulen werden im Anschluss im Plenum besprochen.

**(6) Fachdidaktische Grundlagen**

a) Kennenlernen von Aufgabenformaten in Scratch (LFB-DP 4)
b) Kennenlernen von Modellen über den Verlauf des Programmierenlernens (LFB-DP 4)
c) Kennenlernen weiterer kindgemäßer Software und altersgemäßer Informatiksysteme zum Programmieren (LFB-DP 4)

Umsetzung:

- Aufgabenformate:
  Die Übungsphase ist im Sinne eines *Didaktischen Doppeldeckers* gestaltet: die Lehrkräfte erfahren Unterrichtspraktiken und -methoden als Lernende und reflektieren anschließend aus der Sicht der Lehrenden. Auf diese Art und Weise werden unterschiedliche Formate für Programmieraufgaben thematisiert. Die Lehrkräfte bearbeiten Programmieraufgaben in Scratch, die in Anlehnung an die *Use-Modify-Create Learning Progression* von Lee u. a. (2011) gestaltet wurden – diese wird im Anschluss vorgestellt. Um die Theorie mit der Praxis zu verknüpfen, ordnen die Lehrkräfte die Aufgaben, die sie bereits bearbeitet haben, in die Kategorien *use*, *modify* und *create* ein. Hier arbeiteten sie zu zweit mit einem Satz Kärtchen, auf denen die Programmieraufgaben in Miniatur abgebildet sind und die sie entsprechend gruppieren und ordnen können.

- Zugänge zum Programmieren:
  An verschiedenen Stationen sammeln die Lehrkräfte Programmierpraxis und lernen gleichzeitig weitere Systeme zum Programmieren im Unterricht kennen (siehe Abschnitt C.2.3): den Einplatinencomputer *Calliope mini*, den *Makey Makey*, den mit Pfeiltasten programmierbaren Bienenroboter *Bluebot*, die App *ScratchJr* sowie die programmierbaren Bausätze *LEGO WeDo* und *LEGO EV3*. An jeder Station erhalten die Lehrkräfte grundlegende Informationen zu dem jeweiligen Produkt und bearbeiten eine einführende Aufgabe. Anschließend werden im Plenum die Vor- und Nachteile der einzelnen Produkte sowie mögliche Anknüpfungspunkte zur vorgestellten Unterrichtssequenz gesammelt und diskutiert.

---

*2. Praxisphase:*
*Weitere Erprobungen der Unterrichtssequenz und ggf. Entwicklung sowie Erprobung eigener Lehr-Lernaktivitäten zum Programmieren*

Zwischen dem zweiten und dritten Fortbildungsblock liegen mehrere Wochen. Die Lehrkräfte sind aufgefordert, die Unterrichtssequenz in ihrem Unterricht zu erproben und eigene Ideen für Programmierprojekte zu erarbeiten, die in bestehenden Fächern verortet werden können. Bei Fragen oder Problemen können sie sich per Mail oder Telefon an die Kursleitung wenden.

---

**(7) Erfahrungsaustausch**

a) Präsentation der von den Lehrkräften entwickelten Lehr-Lernaktivitäten (LFB-DP 5)

Umsetzung:

- Erfahrungsaustausch:
  Die Lehrkräfte präsentieren ihre entwickelten Lehr-Lernaktivitäten.

**(8) Fachdidaktische Vertiefung**

a) Kennenlernen von Programmierkontexten aus bestehenden Fächern der Grundschule (LFB-DP 4, LFB-DP 6)
b) Austausch über gute Programmierpraktiken (LFB-DP 5)
c) Auseinandersetzen mit Aspekten des *Computational Thinking* (LFB-DP 6)

Umsetzung:

- Bestehende Fächer:
  Die Lehrkräfte bearbeiten verschiedene Programmieraufgaben, die im Kontext bestehender Grundschulfächer eingesetzt werden können (siehe Abschnitt C.2.4). Zum Beispiel animieren sie ein kurzes Gedicht in Scratch oder programmieren geometrische Figuren. Die Kursleitung moderiert die Ergebnissicherung und holt Feedback der Lehrkräfte zu einem möglichen Einsatz der Aufgaben im Unterricht ein.

- Gute Programmierpraktiken:
  Den Lehrkräften werden zwei Lösungen derselben Aufgabe zur Verfügung gestellt, bei denen sie Unterschiede in der Umsetzung finden sollen (siehe Abschnitt C.2.5). Anhand der Übung werden verschiedene positive Programmierpraktiken herausgearbeitet, wie z. B. sinnvolles Benennen von Variablen, Kommentieren von Code, Entfernen unnötiger Blöcke.

- *Computational Thinking:*
  Innerhalb der Fortbildung werden einzelne Aspekte des *Computational Thinking* thematisiert. In Kleingruppen bearbeiteten die Lehrkräfte verschiedene Stationen und präsentierten diese anschließend im Plenum (siehe Abschnitt C.2.6). Dabei wird auf den Einsatz einer Programmiersprache verzichtet: die Lehrkräfte stellen z. B. einen Tanz-Algorithmus aus Symbolkarten mit unterschiedlichen Tanzposen auf oder erstellen einen Algorithmus für das Zusammenbauen einer Holzfigur mithilfe einer Digitalkamera.

### C.1.2.2 Variante mit einer Praxisphase

**(1) Thematischer Einstieg**

a) Aktivierung von Vorwissen der Lehrkräfte
b) Kurzvorstellung der Unterrichtssequenz und deren didaktischer Relevanz

Umsetzung:

- Begriffsassoziationen:
  Die Lehrkräfte ergänzen den Satz „Programmieren ist für mich ...“ auf Papierstreifen. Im Anschluss lesen sie diese vor. Die Kursleitung greift die Äußerungen der Lehrkräfte auf und präsentiert Assoziationen von Schülerinnen und Schülern mit dem Begriff *Programmieren*.
- Didaktische Relevanz:
  Die Kursleitung stellt den grundlegenden Aufbau der Unterrichtssequenz vor. Sie gibt einen Einblick in die Potenziale und Chancen der Programmierung in der Grundschule.

**(2) Programmieren in der Grundschule**

a) Bearbeiten von ausgewählten Aufgaben der Unterrichtssequenz (LFB-DP 3, LFB-DP 6)
b) Reflexion des eigenen Lernprozesses und Sichtung der Ergebnisse von Schülerinnen und Schülern (LFB-DP 3, LFB-DP 5)

Umsetzung:

- Unterrichtsmaterialien erproben:
  Die Lehrkräfte bekommen die Möglichkeit, die Perspektive der Schülerinnen und Schüler einzunehmen, indem sie ausgewählte Unterrichtsmaterialien selbst bearbeiten. Die Kursleitung zeigt die Aufgaben in einer Präsentation und erläutert die Aufgabenstellung sowie mögliche Anknüpfungspunkte an den Lehrplan der Grundschule. Die Kursleitung stellt zudem die Oberfläche von Scratch vor. Sie leistet während der ganzen Fortbildungsphase Hilfestellung, so wie sie auch Schülerinnen und Schülern helfen würde. Im Anschluss wird im Plenum thematisiert, welche Schwierigkeiten dabei auftraten und welche Erkenntnisse gewonnen wurden.
- Realistische Eindrücke:
  Wo es sich anbietet, werden kurze Videosequenzen aus dem Unterrichtsgeschehen gezeigt, die illustrieren, wie sich die Schülerinnen und Schüler während einer Übung verhalten bzw. wie die Lehrkraft die Übung durchführt. Hierfür eignet sich z. B. die Übung, in der die Schülerinnen und Schüler den *Lehrer-Roboter* programmieren. Zusätzlich werden im Laufe der Fortbildung Arbeitsergebnisse von Schülerinnen und Schülern gezeigt – aus den *Unplugged*- sowie den Scratch-Phasen. So werden bspw. Ergebnisse offener Programmieraufgaben präsentiert, um den Lehrkräften eine Vorstellung davon zu vermitteln, was Schülerinnen und Schüler leisten können.

**(3) Fachliche Inhalte und Fachdidaktische Grundlagen**

a) Vorstellung des Algorithmusbegriffs (LFB-DP 6)
b) Einführung der algorithmischen Grundstrukturen sowie des Variablenkonzepts und Bearbeiten passender Programmieraufgaben in Scratch (LFB-DP 4)
c) Kennenlernen von Aufgabenformaten in Scratch (LFB-DP 4)
d) Kennenlernen von Modellen über den Verlauf des Programmierenlernens (LFB-DP 4)

Umsetzung:

- Bilddiktat:
  Als Einstieg in die Thematik *Algorithmus* wird eine Variante der *Stillen Post* gespielt. Jede Lehrkraft malt oben auf ein leeres Blatt Papier ein einfaches Bild und gibt es nach rechts an die Nachbarin bzw. den Nachbar weiter. Dieser beschreibt unter dem Bild, was er darauf sieht und knickt das gemalte Bild nach hinten um. Danach wird das Blatt erneut nach rechts weitergegeben, sodass nun jeder die Beschreibung eines Bildes vor sich sieht. Zu dieser Beschreibung wird nun wieder ein Bild gemalt. Im Anschluss falten die Lehrkräfte die Blätter auf und vergleichen, wie ähnlich sich die beiden Bilder sehen oder worin sie sich unterscheiden. Falls die Bilder sehr unterschiedlich aussehen, können sie untersuchen, an was es liegt – wurde nicht genau beschrieben oder nicht genau gelesen? Anhand der Übung wird der Unterschied von Alltagsalgorithmen zu einem Algorithmus im eigentlichen Sinn herausgestellt. Eine Definition des Begriffs wird präsentiert und die einzelnen Elemente besprochen.
- Fachinhalte:
  Die algorithmischen Grundstrukturen – Anweisung, Sequenz, Wiederholung und bedingte Anweisung – werden zunächst jeweils in Pseudocode sowie Blöcken der Programmiersprache Scratch beschrieben und danach in Programmieraufgaben unterschiedlicher Schwierigkeit vertieft (siehe Abschnitt C.2.1). Zusätzlich zu den algorithmischen Grundstrukturen wird das Variablenkonzept behandelt. Der theoretische Input wird dabei möglichst kurzgehalten – der Schwerpunkt liegt auf dem eigenständigen Bearbeiten der Aufgaben. Diese können zum Großteil auch im Unterricht eingesetzt werden.
- Fehlvorstellungen vermeiden:
  Um Fehlvorstellungen zum Ablauf einzelner algorithmischer Strukturen früh abzufangen, werden Codebeispiele in Scratch gemeinsam schrittweise gelesen und nachvollzogen. Hierbei wird bspw. thematisiert, was passiert, wenn das Programm unter verschiedenen Voraussetzungen, z. B. Positionen einer Figur auf der Bühne, gestartet wird.
- Aufgabenformate:
  Die Programmieraufgaben zu den algorithmischen Grundstrukturen sind in Anlehnung an die *Use-Modify-Create Learning Progression* von Lee u. a. (2011) gestaltet – diese wird im Anschluss vorgestellt. Um die Theorie mit der Praxis zu verknüpfen, ordnen die Lehrkräfte die Aufgaben, die sie bereits bearbeitet haben, in die Kategorien *use*, *modify* und *create* ein. Hier arbeiteten sie

zu zweit mit einem Satz Kärtchen, auf denen die Programmieraufgaben in Miniatur abgebildet sind und die sie entsprechend gruppieren und ordnen können.

**(4) Reflexion**

a) Sammeln von Möglichkeiten und Maßnahmen zur Implementierung an den Schulen der Lehrkräfte und Thematisieren von Herausforderungen (LFB-DP 5)

Umsetzung:

- Reflexionsphasen initiieren:
  Anhand von Plakatwänden wird thematisiert, wie sich die Lehrkräfte die Umsetzung der Themen Algorithmen und Programmieren in ihrem eigenen Unterricht vorstellen können. Diese werden im Raum verteilt und die Lehrkräfte können darauf Ideen, Inspirationen und Gedanken zu verschiedenen Themen sammeln, z. B. zu möglichen kurz-, mittel- und längerfristige Umsetzungen, Herausforderungen oder dem Bild des Programmierens und wie sich dieses bereits verändert hat. Die Kursleitung moderiert die Sichtung und Zusammenschau der Ergebnisse.

---

*1. Praxisphase:*
*Erprobungen der Unterrichtssequenz*

Zwischen dem ersten und zweiten Fortbildungsblock liegen mehrere Wochen. Die Lehrkräfte sind aufgefordert, die Unterrichtssequenz in ihrem Unterricht auszuprobieren. Druckvorlagen der Unterrichtsmaterialien, Scratch-Dateien sowie ein Satz haptischer Scratch-Blöcke und eine Programmierunterlage aus Filz werden den Lehrkräften zur Verfügung gestellt. Bei Fragen oder Problemen können sie sich per Mail oder Telefon an die Kursleitung wenden.

---

**(5) Erfahrungsaustausch**

a) Austausch von Erfahrungen in der Umsetzung der Unterrichtssequenz (LFB-DP 5)
b) Präsentation von *Best-Practice*-Beispielen (LFB-DP 5)

Umsetzung:

- Erfahrungsaustausch:
  In der Fortbildungsveranstaltung, die sich an die eingeforderte Erprobungsphase anschließt, wird ein Erfahrungsaustausch zwischen den Lehrkräften eingeplant. Auf verschiedenen Plakatwänden zu den einzelnen Elementen der Unterrichtssequenz sammeln die Lehrkräfte stichpunktartig, in welchem fachlichen Kontext sie etwas an ihrer Schule umsetzen konnten, welche Anpassungen sie vornahmen, wie die Reaktionen der Kinder ausfielen und welche Probleme auftraten. Die Ergebnisse sowie *Best-Practice*-Beispiele aus den Schulen werden im Anschluss im Plenum besprochen.

**(6) Fachdidaktische Vertiefung**

a) Kennenlernen weiterer kindgemäßer Software und altersgemäßer Informatiksysteme zum Programmieren (LFB-DP 4)
b) Kennenlernen von Programmierkontexten aus bestehenden Fächern der Grundschule
c) Austausch über gute Programmierpraktiken (LFB-DP 5)
d) Auseinandersetzen mit Aspekten des *Computational Thinking* (LFB-DP 6)

Umsetzung:

- Zugänge zum Programmieren:
  An verschiedenen Stationen sammeln die Lehrkräfte Programmierpraxis und lernen gleichzeitig weitere Systeme zum Programmieren im Unterricht kennen (siehe Abschnitt C.2.3): den Einplatinencomputer *Calliope mini*, den *Makey Makey*, den mit Pfeiltasten programmierbaren Bienenroboter *Bluebot*, die App *ScratchJr* sowie die programmierbaren Bausätze *LEGO WeDo* und *LEGO EV3*. An jeder Station erhalten die Lehrkräfte grundlegende Informationen zu dem jeweiligen Produkt und bearbeiten eine einführende Aufgabe. Anschließend werden im Plenum die Vor- und Nachteile der einzelnen Produkte sowie mögliche Anknüpfungspunkte zur vorgestellten Unterrichtssequenz gesammelt und diskutiert.

- Bestehende Fächer:
  Die Lehrkräfte bearbeiten verschiedene Programmieraufgaben, die im Kontext bestehender Grundschulfächer eingesetzt werden können (siehe Abschnitt C.2.4). Zum Beispiel animieren sie ein kurzes Gedicht in Scratch oder programmieren geometrische Figuren. Die Kursleitung moderiert die Ergebnissicherung und holt Feedback der Lehrkräfte zu einem möglichen Einsatz der Aufgaben im Unterricht ein.

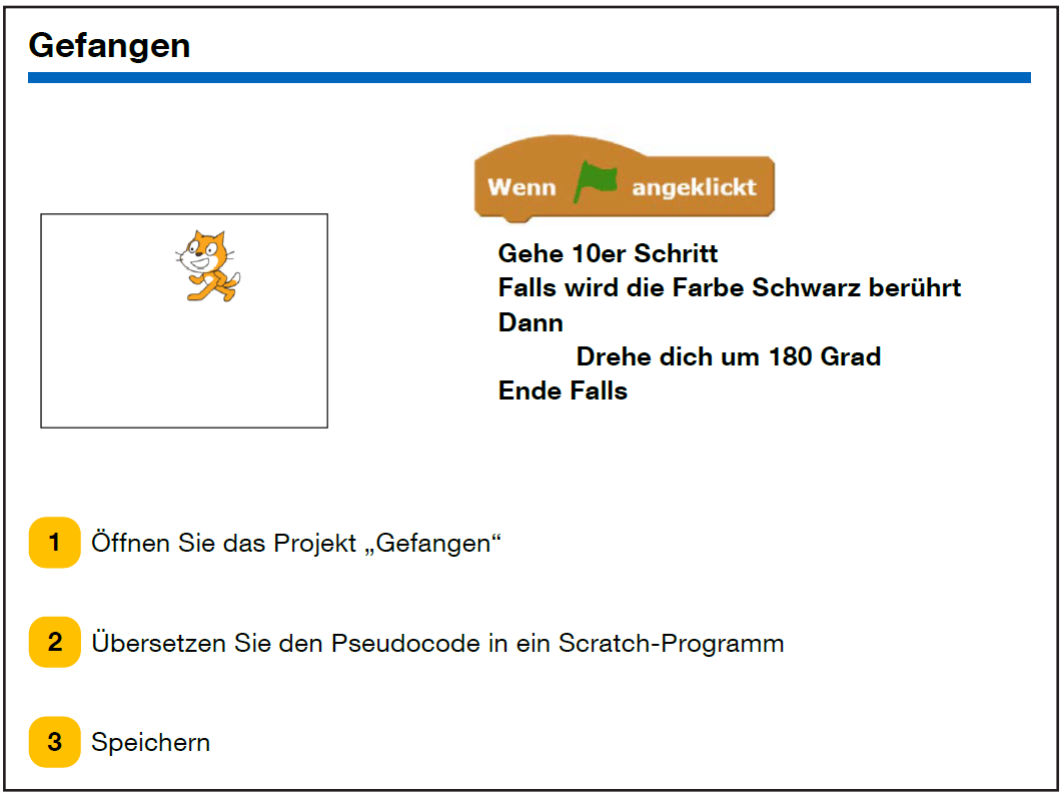
- Gute Programmierpraktiken:
  Den Lehrkräften werden zwei Lösungen derselben Aufgabe zur Verfügung gestellt, bei denen sie Unterschiede in der Umsetzung finden sollen (siehe Abschnitt C.2.5). Anhand der Übung werden verschiedene positive Programmierpraktiken herausgearbeitet, wie z. B. sinnvolles Benennen von Variablen, Kommentieren von Code, Entfernen unnötiger Blöcke.

- *Computational Thinking*:
  Innerhalb der Fortbildung werden einzelne Aspekte des *Computational Thinking* thematisiert. In Kleingruppen bearbeiteten die Lehrkräfte verschiedene Stationen und präsentierten diese anschließend im Plenum (siehe Abschnitt C.2.6). Dabei wird auf den Einsatz einer Programmiersprache verzichtet: die Lehrkräfte stellen z. B. einen Tanz-Algorithmus aus Symbolkarten mit unterschiedlichen Tanzposen auf oder erstellen einen Algorithmus für das Zusammenbauen einer Holzfigur mithilfe einer Digitalkamera.

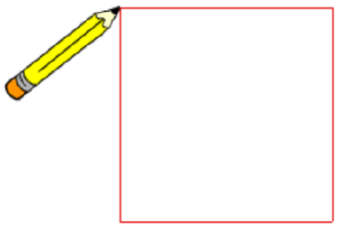
## C.2 Fortbildungsmaterialien

Im Folgenden werden ausgewählte Fortbildungsmaterialien dargestellt, die im Rahmen dieser Arbeit entwickelt und erprobt wurden. Die Einsicht in weitere Materialien, die an dieser Stelle nicht oder nicht vollständig dargestellt sind (z. B. Scratch-Dateien und Präsentationen), kann per E-Mail direkt bei der Autorin angefragt werden (katharina.geldreich@tum.de).

### C.2.1 Aufgaben zur Einführung der algorithmischen Grundstrukturen sowie des Variablenkonzepts (Auswahl)



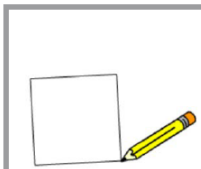
## Quadrat

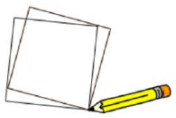


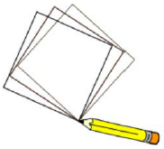

1. Öffnen Sie das Projekt „Quadrat“
2. Zeichnen Sie mit Scratch ein Quadrat
3. Speichern

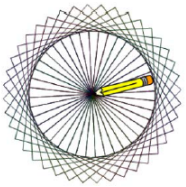
## Die quadratische Blume

Rosie möchte eine Blume aus 40 Quadraten mit Scratch zeichnen. Die folgende Abbildung zeigt die ersten 3 Schritte und das Resultat:


Schritt 1


Schritt 2

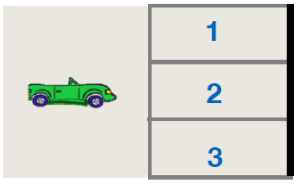

Schritt 3


Ergebnis

1. Öffnen Sie das Projekt „Die quadratische Blume“
2. Schreiben Sie mittels Pseudocode einen Algorithmus auf, welcher diese Blume zeichnet
3. Erstellen Sie das entsprechende Programm mit Scratch
4. Speichern

## Einparken

Das Auto soll automatisch in die Parklücke 2 einparken.




1. Öffnen Sie das Projekt „Einparken“
2. Lassen Sie das Auto vorwärts in die Parklücke 2 einfahren
   TIPP: Farbe
3. Speichern

## Pingu fängt Herzen

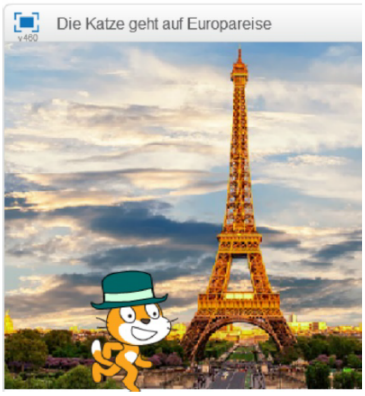
Pingu versucht, so viele Herzen wie möglich zu fangen. Die Anzahl der gefangenen Herzen soll mittels einer Variablen gezählt werden.


1. Öffnen Sie das Projekt „Pingu fängt Herzen“
2. Steuern Sie Pingu über die Pfeiltasten (rechts, links, oben, unten)


3. Definieren Sie eine Variable „Gefundene Herzen“ (Kategorie: Daten)
   **Jedes Mal, wenn Pingu ein Herz fängt, soll:**
   - die Variable „Gefundene Herzen“ um 1 erhöht werden
   - Das Herz auf eine zufällige neue Position im Spielfeld gesetzt werden
4. Speichern

## Die Katze geht auf Europareise

Die Katze bereist die Hauptstädte Europas.




1. Öffnen Sie das Projekt „Die Katze geht auf Europareise“
2. Schauen Sie sich das Register „Bühnenbilder“ sowie „Kostüme“ kurz an
3. Lassen Sie die Katze durch drücken der Pfeiltaste nach rechts durch Europa wandern

   **Dabei soll:**

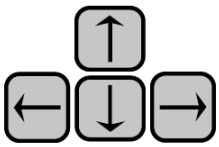
   - das Kostüm der Katze immer wieder gewechselt werden
   - das nächste Bühnenbild gezeigt werden, falls die Katze am rechten Rand angekommen ist. Die Katze beginnt dann wieder von links weg zu laufen

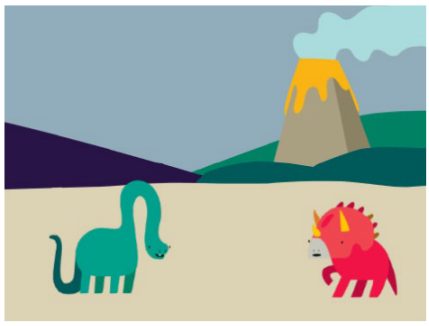
   Programmieren Sie das gleiche noch für die Pfeiltaste nach links.
4. Speichern

### C.2.2 Aufgaben zur Einführung verschiedener Aufgabenformate (Auswahl)

## Dinosaurier (Code + Text → Code)

Im Projekt "Dinosaurier" gibt es zwei urzeitliche Figuren. Den grünen Dinosaurier kann man bereits mit den Pfeiltasten steuern.





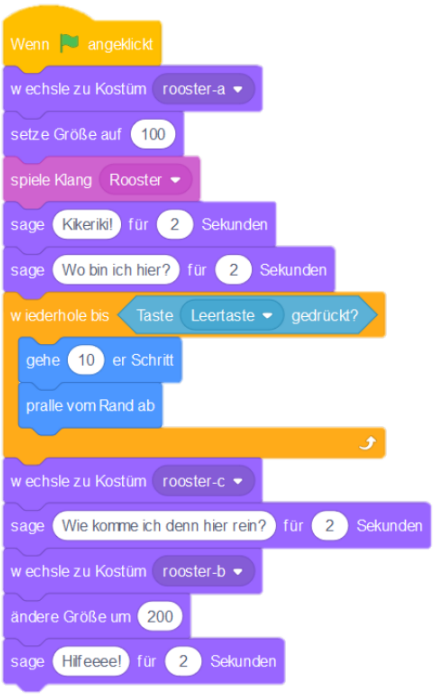
1. Öffnen Sie das Projekt „Dinosaurier“
2. Erweitern Sie das Programm!

   Der grüne Saurier soll
   - sich weh tun, wenn er die Lava berührt
   - eine Unterhaltung beginnen, wenn er den roten Dinosaurier berührt
3. Speichern

## Hahn im Haus (Copy Code)

**1** Öffnen Sie das Projekt „Hahn im Haus“

**2** Implementieren Sie für den Hahn folgendes Programm:




**3** Speichern

**4** Markieren Sie besonders knifflige Stellen im gedruckten Programm!
Wo muss man besonders sorgfältig sein?

## Fledermaus (Text → Code)

**1** Öffnen Sie das Projekt „Fledermaus“

**2** Animieren Sie die Flugbewegung der Fledermaus!

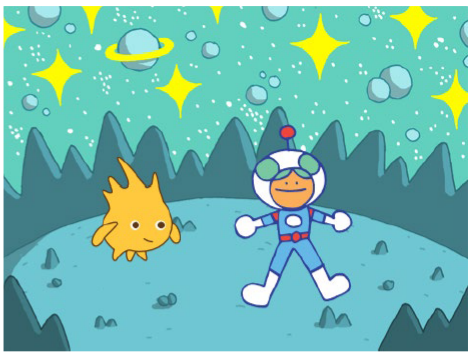
**3** Speichern

**Notizen:**

## Englisch im Weltall (Befehle erkunden)

Das Alien Gobo spricht kein Deutsch.

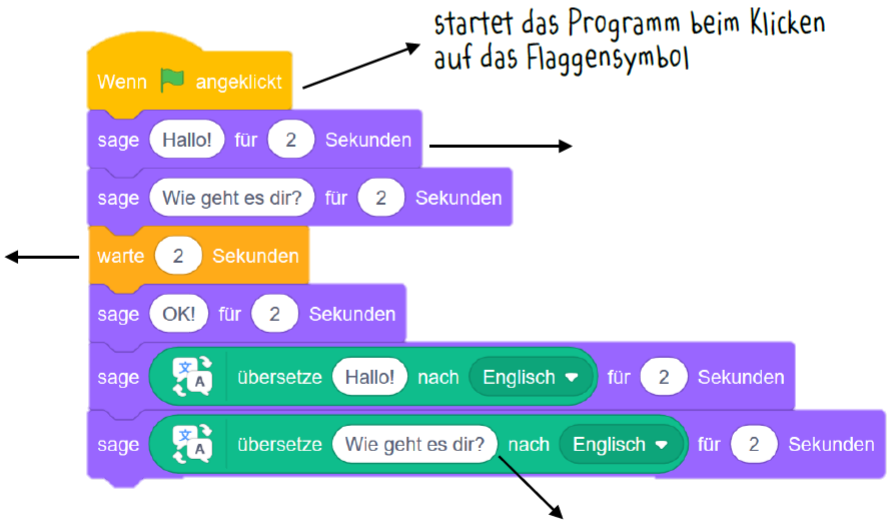
Es spricht und versteht nur Englisch!



**1** Öffnen Sie das Projekt „Weltall“

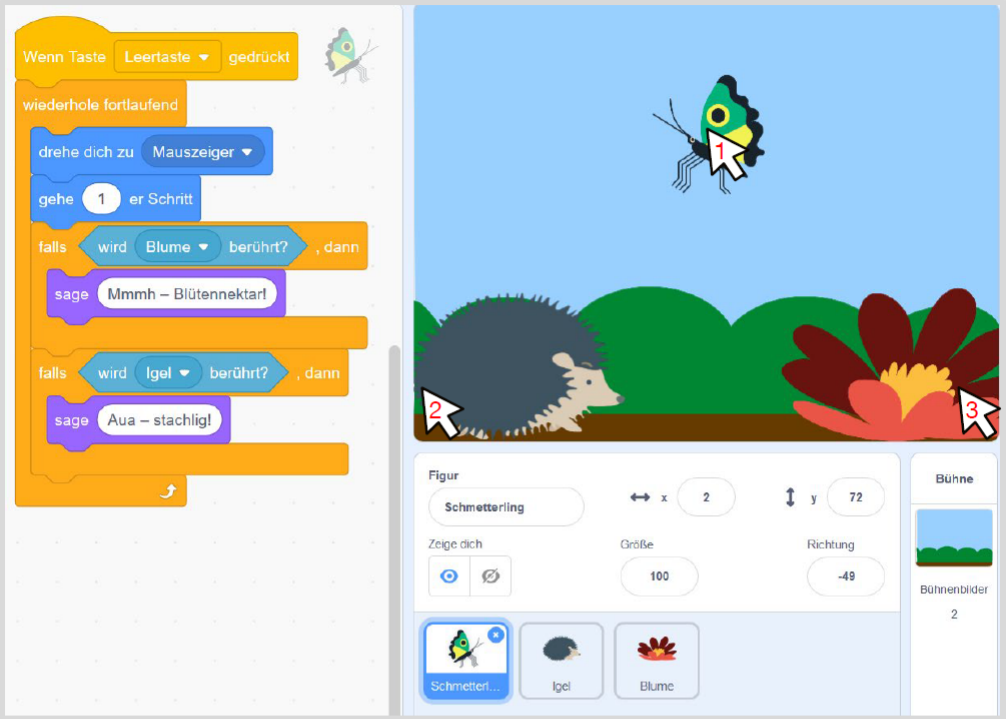
**2** Starten Sie das Programm mit dem Klicken auf 

**3** Welche Funktionen haben die einzelnen Blöcke?




**4** Bauen Sie das Programm um – für das Gespräch sollen "Nachrichten" eingesetzt werden!

## Schmetterling (Code → Text)




**1** Was passiert beim Klicken auf die Leertaste, wenn…

- der Mauszeiger auf Position 1 steht:

_____

- der Mauszeiger auf Position 2 steht:

_____

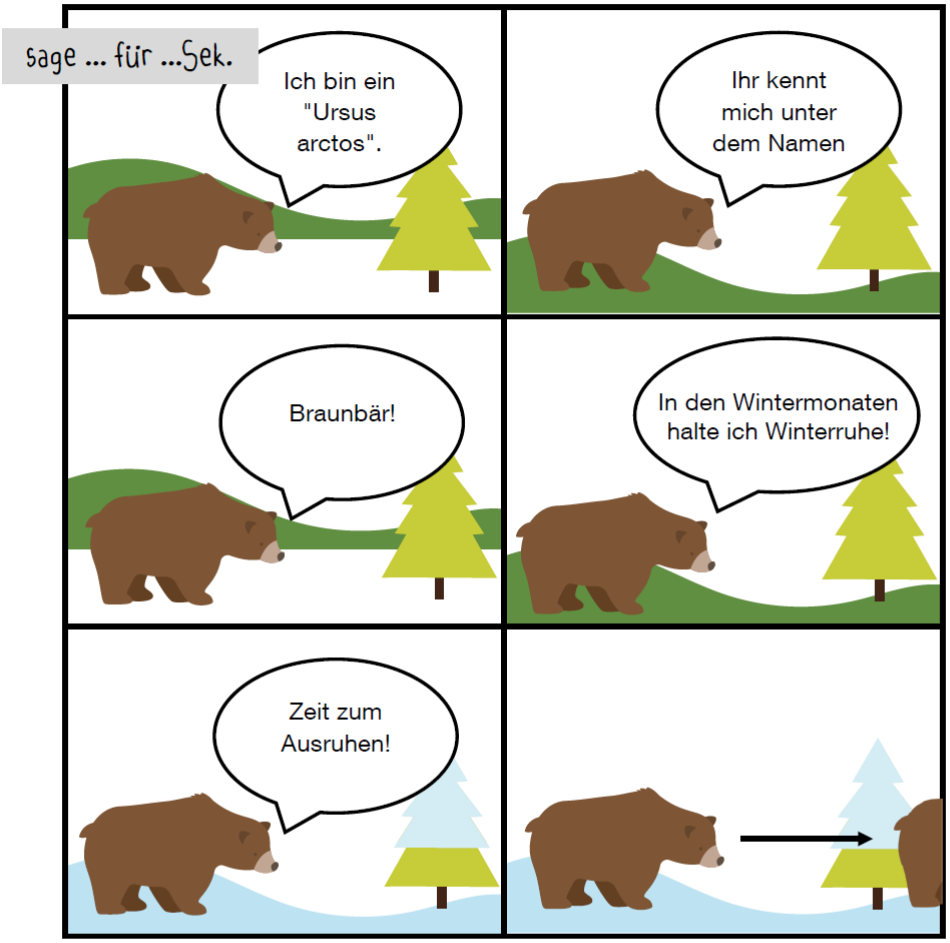
- der Mauszeiger auf Position 3 steht:

_____

**2** Öffnen Sie das Projekt "Schmetterling" und überprüfen Sie Ihre Lösungen!

## Braunbär (Bild → Code)

Die Geschichte des Bären soll als Scratch-Programm implementiert werden:




1. Machen Sie sich zunächst Notizen an den jeweiligen Bildrand, welche Scratch-Befehle sie zum brauchen könnten (siehe Beispiel grauer Kasten in Bild 1)!
2. Öffnen Sie das Projekt „Braunbär“
3. Programmieren Sie die vorgegebene Bildergeschichte in Scratch!
4. Speichern

### C.2.3 Stationen zum Kennenlernen weiterer kindgemäßer Software und altersgemäßer Informatiksysteme zum Programmieren (Auswahl)

## Lego WeDo

**ca. 150 €**

### Steckbrief

| | |
|---|---|
| **Altersgruppe**: | 6-9 Jahre |
| **Preis**: | ca. 150€ |
| **Stromversorgung**: | USB-Kabel |
| **System**: | Windows, iOS |

### Das bietet Lego WeDo

- Bauen von verschiedenen Figuren mit Lego-Bausteinen
- Bewegung durch Motoren
- Bewegungs- und Kippsensor
- Einfache blockbasierte Programmierung

**1** Lassen Sie das Krokodil sein Maul öffnen. Das Krokodil soll danach zubeißen, sobald etwas in seinem Mund ist. Nutzen Sie die folgenden Blöcke.


TIPP: Setzen Sie einen Soundblock nachdem Sie den Motor gestartet haben. Darauf folgt der Motor-Aus Block

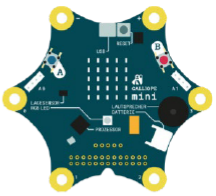
**2** Lassen Sie den Löwen mittels verschiedener Tasteneingaben aufstehen und sich hinlegen.

TIPP: 

## Calliope mini

**ca. 35 €**

### Steckbrief



| | |
|---|---|
| **Altersgruppe**: | ab 8 Jahre |
| **Preis**: | ab 35 € |
| **Stromversorgung**: | USB, Batterie |
| **System**: | Windows, iOS, iPad, iPhone |

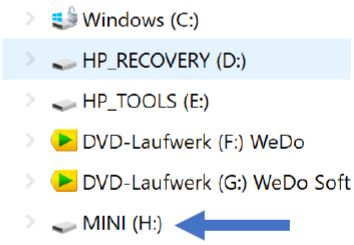
### Das bietet der Calliope mini

- Eingaben über Knöpfe, Ausgaben über direkte Ansteuerung von Lämpchen.
- Bewegungs-, Beschleunigungssensor
- Weitere Sensoren können über die Pins angeschlossen und angesteuert werden.
- Einfache blockbasierte Programmierung

***Hinweis***: es gibt verschiedene blockbasierte Editoren, um für den Calliope Programme zu schreiben

https://makecode.calliope.cc/
https://miniedit.calliope.cc/
https://lab.open-roberta.org

Sie können das Programm auf der Webseite zusammenstellen und danach auf Ihren Calliope herunterladen. Speichern Sie dazu die heruntergeladene Datei auf dem Laufwerk des Calliope Mini.




**1** Lassen Sie den Calliope „Hallo“ über die Lämpchen ausgeben, wenn der Knopf A gedrückt wird.

**2** Geben Sie einen Pfeifton aus und schreiben Sie ein X über die Lämpchen, wenn der Calliope geschüttelt wird.

## BlueBot

**ca. 100 €**

### Steckbrief

| | |
|---|---|
| **Altersgruppe**: | 5-8 Jahre |
| **Preis**: | ca. 100€ |
| **Stromversorgung**: | Akku |
| **System**: | Windows-PC, iPad |

### Das bietet der Blue Bot

- BlueBot lässt sich mit Tasten am Gerät programmieren oder per App
- Abfahren von programmierten Befehlsketten (links, rechts, geradeaus, rückwärts) →bis zu 200 Einzelbefehle können eingegeben werden

**1** Lassen Sie den BlueBot ein Viereck fahren.

**2** Wie könnten Sie den BlueBot die Ziffer **3** abfahren lassen?

**2** Legen Sie sich passende Aufgaben auf dem Spielfeld und fahren Sie diese mit dem Bluebot ab:

TIPP: Wo soll der BlueBot starten?
Wo soll der BlueBot ankommen?
Was soll er vermeiden?

## ScratchJr

### Steckbrief

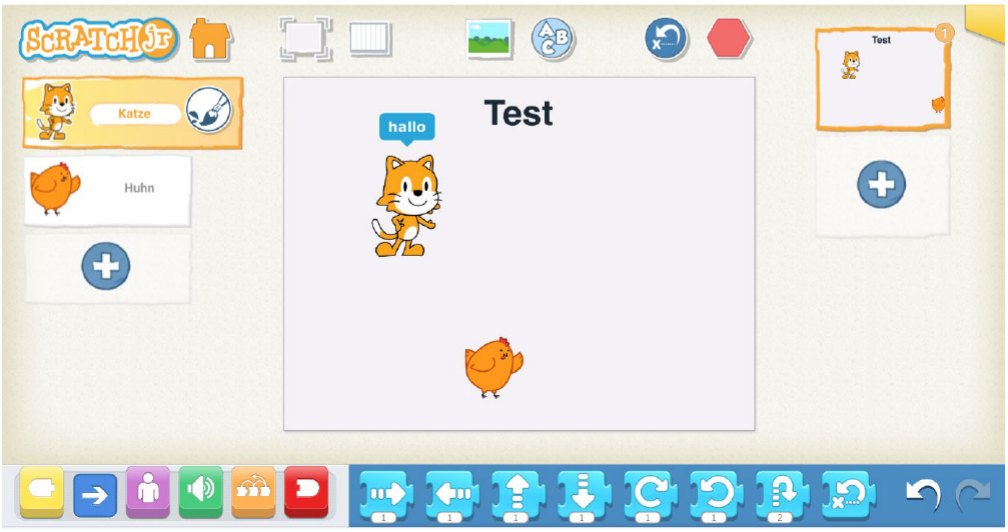
| | |
|---|---|
| **Altersgruppe**: | 5-8 Jahre |
| **Preis**: | frei |
| **System**: | iPad, iPhone, Android |

### Was bietet ScratchJr

- Einfachere Scratch-Umgebung mit deutlich weniger Befehlen
- Wiederholung (mit Anzahl)
- Abspielen von Tönen, Zugriff auf das Mikrofon

**1** Lassen Sie die Katze „Hallo“ sagen

**2** Das Huhn soll wiederholend durch das Feld gehen und dabei springen.

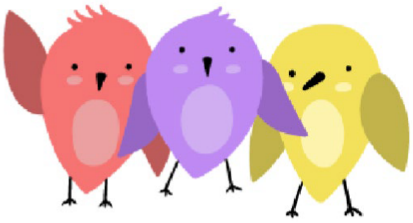


### C.2.4 Aufgaben zur Anknüpfung an bestehende Fächer der Grundschule

## Die drei Spatzen

In einem leeren Haselstrauch
Da sitzen drei Spatzen, Bauch an Bauch.

Der Erich rechts und links der Franz
Und mitten drin der freche Hans.



Sie haben die Augen zu, ganz zu,
Und obendrüber da schneit es, hu!

Sie rücken zusammen dicht an dicht.
So warm wie der Hans hats niemand nicht.

Sie hören alle drei ihrer Herzlein Gepoch.
Und wenn sie nicht weg sind, so sitzen sie noch.

*Christian Morgenstern*

---

Das Gedicht von Christian Morgenstern soll in Scratch animiert werden.

1. Öffnen Sie das Projekt „Spatzen“.
   Schauen Sie sich die bereits eingefügten Figuren an.

2. Schreiben Sie für den Erzähler ein Programm – er soll das Gedicht Zeile für Zeile vortragen.

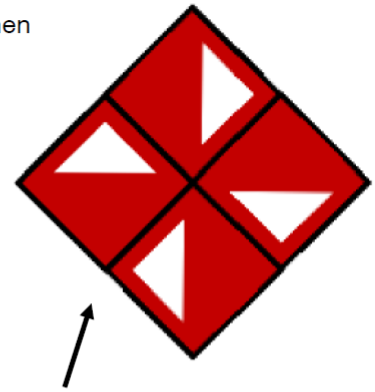
3. Markieren Sie oben im Gedicht Stellen, zu denen Sie die Figuren in Scratch animieren können und notieren Sie erste Ideen.
   Setzen Sie einige Ideen in Scratch um!

4. Speichern

## Kachelmuster

Der Block 

führt dazu, dass eine Figur einen Abdruck an der Stelle auf der Bühne hinterlässt, an der sie sich gerade befindet.



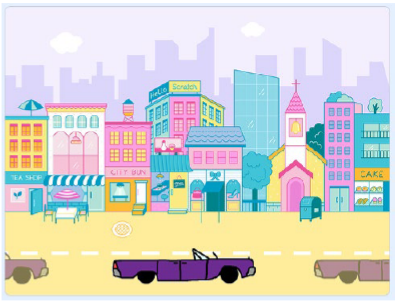
1. Öffnen Sie das Projekt „Kachelmuster“
2. Programmieren Sie ein Programm, das zu dem **Muster** führt.
   Nehmen Sie die Papierplättchen zur Hilfe!
3. Speichern

### Hilfe:

- Falls der Blockbereich „Malstift“ nicht angezeigt wird, finden Sie ihn unter „Erweiterungen“
- Der Drehpunkt der Kachel ist hier:

### C.2.5 Aufgaben zum Thematisieren weiterer fachdidaktischer Themen (Auswahl)

## Stadtrundfahrt



1 **Arbeiten Sie zu zweit.** Öffnen Sie auf einem Rechner das Projekt „Rundfahrt1“, auf dem anderen „Rundfahrt2.

2 Beide Programme sollten folgende Vorgaben erfüllen:

- Auto soll von links nach rechts fahren.
- Wenn es an der rechten Seite ankommt, soll es links wieder „auftauchen“.
- Sobald das Auto vom Mauszeiger berührt wird, soll es anhalten.

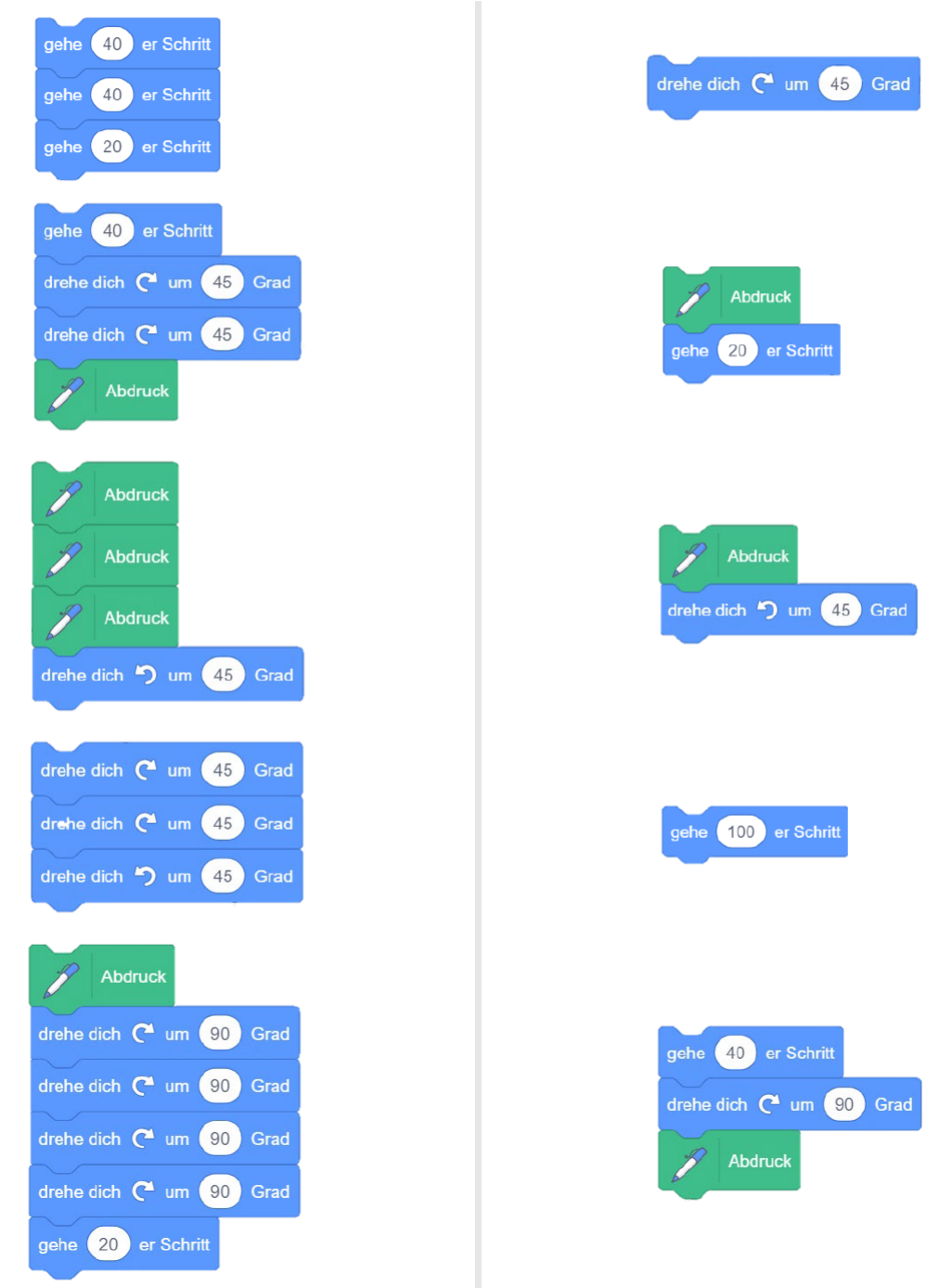
Suchen Sie nach Unterschieden in der Umsetzung.

3 Welche Kriterien könnte man zur Bewertung der Programme heranziehen?

## Kein „smelly“ Code!

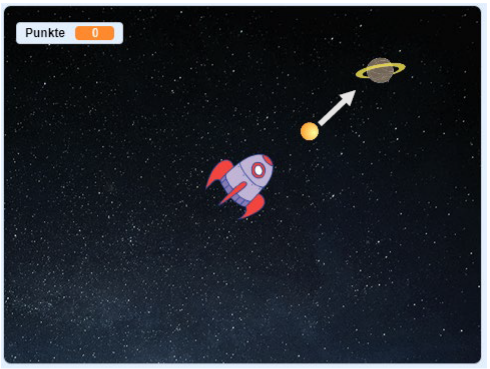
Welche Skripte führen jeweils zum gleichen Ergebnis?

Zeichnen Sie Verbindungslinien!



## Debugging-Odyssee im Weltraum

Das Raumschiff im Projekt „Weltraum“ lässt sich mit den Pfeiltasten nach links und rechts drehen. Alle 10 Sekunden erscheint an einer zufälligen Position ein Planet. Drückt man die Leertaste, kann man die Planeten abschießen. Bei jedem Treffer gibt es einen Punkt.




**1** Öffnen Sie das Projekt „Weltraum“

**2** Testen Sie das Programm aus – welche Fehler („Bugs“) gibt es?

**3** „Debuggen“ Sie das Programm – beheben Sie die Fehler!

**4** Speichern

### C.2.6 Aufgaben zur Auseinandersetzung mit Aspekten des *Computational Thinking* (Auswahl)

## Tanz-Algorithmus

1. Schauen Sie sich die verschiedenen Karten mit den Tanzschritten an und führen Sie jeden Tanzschritt einmal aus. Einigen Sie sich auf eine Art der „Ausführung“.
2. Teilen Sie die Gruppe auf:
   Jede Gruppe stellt einen Tanz-Algorithmus zusammen. Dieser wird von der anderen ausgeführt.
   Was passiert, wenn man Karten miteinander austauscht?
3. Teilen Sie die Gruppe auf:
   Jede Gruppe erstellt ein Video mit einem kurzen Tanz (mit den Bewegungen auf den Karten).Die andere Gruppe stellt einen dazu passenden Algorithmus auf.
4. Überlegen Sie sich mögliche Erweiterungen!

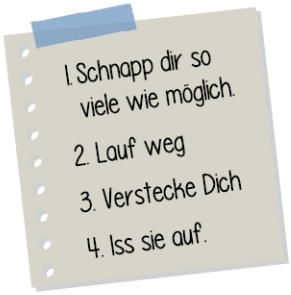
## Zahlen-Bot

1. Machen Sie sich mit dem BlueBot vertraut!
2. Wie könnte man den BlueBot die Form von Ziffern abfahren lassen?
   Notieren Sie für jede Ziffer den Algorithmus mit Pfeilsymbolen.
3. Testen Sie Ihren Algorithmus mit dem „FakeBot“ auf den ausgedruckten Ziffern.
4. Befestigen Sie mit dem Gummi einen Filzstift am Bluebot und testen Sie Ihre Algorithmen aus. Zeichnet er die Ziffern korrekt?
5. Überlegen Sie sich mögliche Erweiterungen!

## Süßes für alle!

Süßigkeiten sollen untereinander geteilt werden. Es soll ein Algorithmus aufgestellt werden, der den Naschkram **gerecht** aufteilt.

**1**




Wie wäre es mit diesem Algorithmus?

**2** Stellen Sie gemeinsam einen Algorithmus auf, der **10 Bonbons** gerecht zwischen **zwei Menschen** aufteilt. Testen Sie ihn mit den Glasperlen.

**3** Stellen Sie gemeinsam einen Algorithmus auf, der **20 Bonbons** gerecht zwischen **vier Menschen** aufteilt. Testen Sie ihn mit den Glasperlen.

**4** Stellen Sie gemeinsam einen Algorithmus auf, der **15 Bonbons** gerecht zwischen **vier Menschen** aufteilt. Testen Sie ihn mit den Glasperlen.

**5** Überlegen Sie sich mögliche Erweiterungen!

## Kuh-Bau-Algorithmus

**1** Bauen Sie die Holz-Kuh zusammen – Vorsicht, zerbrechlich!

**2** Die Kuh wird nun Schritt für Schritt wieder auseinander gebaut! Fotografieren Sie jeden Schritt mit der Surface-Kamera, bis die Kuh wieder komplett in ihre Einzelteile zerlegt ist.

**3** Benutzen Sie Ihre Fotos, um einen Algorithmus aufzustellen, der den „Kuh-Bau“ beschreibt.

**4** Überlegen Sie sich mögliche Erweiterungen!