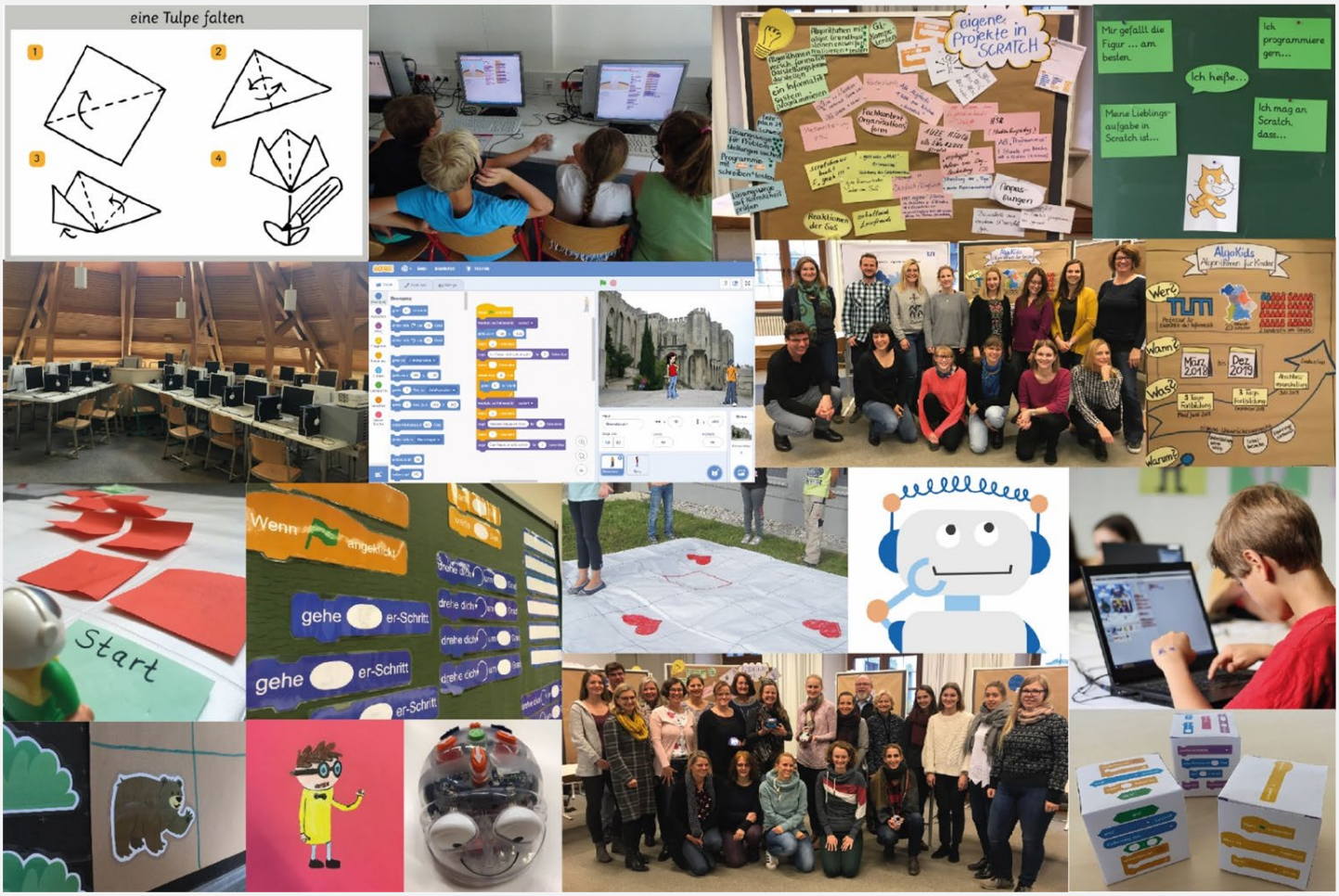
Prof. Dr. Peter Hubwieser, Katharina Geldreich
Professur für Didaktik der Informatik
TUM School of Education
Technische Universität München

Algorithmen für Kinder
(AlgoKids)



Abschlussbericht eines Kooperationsprojekts der Technischen Universität München und des Bayerischen Staatsministeriums für Unterricht und Kultus

Inhalt

1. Einleitung

Sowohl auf nationaler als auch auf internationaler Ebene setzt sich immer mehr die Überzeugung durch, dass die jeweiligen Schulsysteme auf die fortschreitende und umfassende Digitalisierung unserer Lebenswelt mit einer angemessenen Informatikausbildung reagieren müssen, um den Schülerinnen und Schülern auch in Zukunft eine kompetente, selbstbestimmte Teilhabe an dieser medienzentrierten Welt zu ermöglichen. Zudem kommen immer mehr Regierungen zu der Überzeugung, dass zu einer solchen Ausbildung eine gewisse Grundausbildung im Programmieren gehört.

Der Freistaat Bayern ist in dieser Richtung schon früh aktiv geworden, indem er bereits 2003 ein Pflichtfach Informatik an seinen Gymnasien und kurz darauf (als Informationstechnologie) auch an seinen Realschulen eingeführt hat. Im Rahmen der jüngsten bayerischen Digitalisierungsoffensive wurde dieses Unterrichtsangebot noch weiter ausgebaut sowie ein durchgehendes Fach Informatik auch an den Mittelschulen eingeführt. Damit verfügen alle weiterführenden Schulen in Bayern über verpflichtenden Informatikunterricht.

Weil aber auch für Grundschulkinder Computer, Tablets und Smartphones mittlerweile zum Alltag gehören, stellt sich die Frage, ob nicht bereits an den Grundschulen mit ersten Schritten im Programmieren begonnen werden muss. Schließlich wachsen die Kinder in einer durch die Digitalisierung geprägten Welt auf und kommen immer früher mit neuen Technologien und Medien in Kontakt. Bekanntermaßen bilden sich geschlechterspezifische Stereotype bzw. Fehlvorstellungen zu technischen Fächern wie der Informatik bereits in der frühen Pubertät aus, wodurch trotz durchaus vorhandener Begabung eine Wahl von Ausbildungsrichtungen oder Berufen in Richtung der MINT-Fächer be- oder sogar verhindert werden könnte. Zahlreiche internationale Initiativen zeigen, dass viele Länder bereits an einer kindgemäßen Informatikausbildung in der Primarstufe arbeiten.

Andererseits kann man Grundschulkinder natürlich nicht beliebig mit zusätzlichen Lerninhalten belasten. Es mehren sich zudem Stimmen, die ein Nachlassen der elementaren Kulturtechniken Lesen, Schreiben und Rechnen beklagen. Man muss also sorgfältig überlegen, ob, in welchem Umfang und mit welcher Zielsetzung man zusätzliche Lerninhalte wie Programmieren in die Grundschule bringen kann und will. Ein vorsichtiger erster Schritt könnte darin bestehen, zunächst das Interesse der Kinder für die Programmierung zu wecken und ihnen ihre eigenen Talente in dieser Richtung bewusst zu machen.

Vor diesem Hintergrund hat sich das Team der Professur für Didaktik der Informatik an der TUM School of Education (im Folgenden kurz TUMDDI) im Jahr 2015 entschlossen, ein Forschungsprojekt zum Programmieren in der Grundschule aufzusetzen. Wir wollten damit erforschen, inwieweit Kinder im Alter von 9-10 Jahren grundlegende algorithmische Konzepte verstehen und einfache Algorithmen implementieren können. Weiter interessierten wir uns für die Auswirkungen eines Programmierkurses auf Motivation und Selbstwirksamkeitsempfinden der Kinder.

Den Mittelpunkt des Projekts bildete eine sehr sorgfältig konzipierte dreitägige Intervention für Grundschulkinder. Als Kontext wählten wir die Zirkuswelt (*Programmierzirkus*), weil wir uns davon einerseits einen gewissen Bekanntheitsgrad in dieser Altersgruppe und andererseits eine relativ genderneutrale Motivationswirkung versprachen. Die Methodik des Kurses ist konsequent auf Minimierung des Einflusses der Lehrperson ausgelegt, was dem aktuellen Stand der Lernpsychologie entspricht und zudem eine gewisse Übertragbarkeit der Ergebnisse sichern soll. Nach Abschluss des Kurses sollen die Kinder erklären können, wie ein Computerprogramm grundsätzlich aufgebaut ist und mit einer kindgerechten Programmierumgebung (wie z.B. *Scratch*) einfache Algorithmen selbst implementieren können.

Der Kurs wurde in den Jahren 2016 und 2017 mit ca. 150 Grundschulkindern im Alter von 8-10 Jahren an der TUM School of Education und am Schülerforschungszentrum Berchtesgadener Land erprobt. Diese Probeläufe wurden ausführlich dokumentiert und ausgewertet. Die ersten Ergebnisse zeigten, dass die meisten Kinder dabei überraschend gute Lernerfolge erzielen.

Zunächst hatten wir geplant, die Einsatzmöglichkeiten des Programmierzirkus über das Schülerforschungszentrum Berchtesgadener Land flächendeckend an den Grundschulen im Berchtesgadener Land zu erproben. Um die Möglichkeiten einer finanziellen Förderung dafür zu besprechen, präsentierten wir das Vorhaben im Bayerischen Staatsministerium für Unterricht und Kultus (im Folgenden „Staatsministerium“ genannt). Dabei zeigte sich insbesondere der damalige Staatssekretär Georg Eisenreich sehr angetan von diesem Vorhaben. Er bedauerte allerdings, dass die Erprobung des Konzepts auf einen Landkreis beschränkt werden sollte. So entstand die Idee für das Projekt „Algorithmen für Kinder“ (im Folgenden kurz „AlgoKids“) als offiziellen Schulversuch mit organisatorischer Unterstützung und finanzieller Förderung des Staatsministeriums. Mit diesem Projekt sollte im Detail erforscht werden, in welcher Form eine derartige Intervention zum Programmieren Lernen an den bayerischen Grundschulen dauerhaft implementiert werden könnte und wie man die Lehrerinnen und Lehrer in die Lage versetzen könnte, die Unterrichtssequenz erfolgreich zu unterrichten.

Nach einer entsprechenden Ausschreibung wählte das Staatsministerium aus den Bewerbungen 20 über ganz Bayern verteilte Grundschulen aus, von denen dann jeweils zwei Lehrkräfte innerhalb der Projektlaufzeit fortgebildet und bei der Erprobung der Unterrichtssequenz begleitet wurden. Dieser Abschlussbericht schildert die Planung, den Verlauf und die Auswertung dieses Projekts.

Zunächst wird der theoretische Rahmen zur Informatik in der Grundschule dargelegt, wobei Potenziale und Chancen, die Situation im In- und Ausland sowie Befunde zur Lehrerbildung thematisiert werden. Nach einem Überblick zum Projekt folgen dessen fachliche und fachdidaktische Grundlagen sowie mögliche Bezüge zum LehrplanPLUS der Grundschule. Kapitel 5 und 6 geben einen Einblick in die Unterrichtssequenz zum Programmieren, die im Laufe des Projekts erprobt wurde sowie die Lehrerfortbildungen, die konzipiert und durchgeführt wurden. In Kapitel 7 und 8 wird das Evaluationskonzept von AlgoKids beschrieben und die Ergebnisse der Evaluation im Detail präsentiert. Daran schließt sich eine Zusammenfassung der Ergebnisse des Projekts sowie Empfehlungen sowohl zu kurz- und mittelfristigen als auch zu langfristigen Maßnahmen bezüglich dem Programmieren in der Grundschule. Der Bericht schließt mit der Dokumentation des Projekts, einer Danksagung sowie Angaben zu den verwendeten Abbildungen und zur Literatur.

2. Informatik in der Grundschule

Heutzutage gehören Computer, Tablets und Smartphones auch für Grundschulkinder zum Alltag. Sie wachsen in einer durch die Digitalisierung geprägten Welt auf und kommen so immer früher mit neuen Technologien und Medien in Berührung. Leider beschränkt sich der Einsatz digitaler Medien im Unterricht der Grundschule in erster Linie auf das Schreiben von Texten, das Recherchieren im Internet sowie das Nutzen von Lernprogrammen (MFS 2019, S. 51). Die Dagstuhl-Erklärung zur Bildung in der digitalen vernetzten Welt (GI 2016a) 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. 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.

2.1 Potenziale und Chancen

Obgleich Informatik inzwischen Pflichtfach an allen weiterführenden Schulen in Bayern ist, gibt es verschiedene Gründe, warum Kinder bereits im Grundschulalter mit informatischen Inhalten in Kontakt kommen sollten.

Teil der Allgemeinbildung

Nach Klafki (2007) besteht die Aufgabe der allgemeinen Bildung darin, Lernende in ihrer Entwicklung zu mündigen Bürgerinnen und Bürgern so zu begleiten, damit sie ihre eigene Zukunft verantwortungsvoll gestalten können. Informatiksysteme[1] sind nicht nur Teil der Lebenswirklichkeit von Kindern, derart viele Bereiche des Lebens hängen von ihnen ab, dass jeder Mensch zwangsläufig ein Verständnis für die Informatik entwickeln sollte, um an der Gesellschaft teilzuhaben und sie mitzugestalten (Brinda et al. 2019; Webb et al. 2017). Gleichzeitig zeigt sich, dass sowohl der Zugang zu digitalen Medien als auch die Qualität der Nutzungserfahrungen maßgeblich vom Elternhaus der Kinder abhängt (Chaudron 2015). 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).

Computational Thinking

Nach allgemeiner Auffassung gehört zu den basalen informatischen Fertigkeiten eine gewisse Grundfertigkeit im informatischen Denken (Grover und Pea 2018) – im Englischen *Computational Thinking*. 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 (Wing 2006). Berry (2015) konkretisiert dies weiter und benennt einzelne Kernaspekte des Problemlösens: *logical reasoning* (vorhersagen und analysieren), *algorithms* (Schritte und Regeln aufstellen), *decomposition* (etwas in einzelne Schritte zerlegen), *abstraction* (auf das Wesentliche konzentrieren), *patterns/generalization* (Muster erkennen und verwenden) und *evaluation* (Bewertungen vornehmen). Man geht davon aus, dass die Denkprozesse, die in der Informatik im Fokus stehen, hilfreich in anderen Fächern oder sogar im Alltag sind (Moreno-Leon et al. 2018).

[1] Als Informatiksystem bezeichnet man eine Zusammenstellung von Hardware, Software und Netzverbindungen.

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. Ob sich Interesse zu einem bestimmten Thema, einem Objekt oder einer Handlung entwickelt, hängt maßgeblich davon ab, welche Anreize sich den Kindern bieten und welchen Interessen sie überhaupt nachgehen können. Die Interessen erfüllen besonders in der frühen Kindheit eine zentrale Funktion, da sie die Aufmerksamkeit der Kinder auf verschiedene Umweltreize lenken und somit deren Erfahrungen beeinflussen können (Miller und Velten 2015, S. 33).

In verschiedenen Fallstudien wurde bereits gezeigt, dass informatische Inhalte bei Kindern im Grundschulalter auf große Begeisterung stoßen (Wolking und Schmid 2017; Romeike und Reichert 2011; Geldreich et al. 2017). 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 (Google Inc. & Gallip Inc. 2017). Diese Ergebnisse legen nahe, dass die Grundschulzeit ein besonders vielversprechendes Zeitfenster ist, 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 (Nelson 2014). Laut Jaglo (2013) wird der „typische Informatiker“ von vornherein als männlich sowie als unattraktiv und unsozial beschrieben. Diese Rollenbilder festigen sich in der Pubertät (Margolis und Fisher 2002; Beyer et al. 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 et al. (1999) stellten fest, dass bereits Kinder im Kindergartenalter Berufe einem bestimmten Geschlecht zuordnen. Berufe, die sich auf Mathematik und Naturwissenschaften beziehen, wurden dabei von den fünfjährigen Jungen und Mädchen als Männerberufe angesehen. Ein möglichst früher Erwerb von informatischen Kompetenzen könnte einerseits das Vorurteil zerstreuen, dass Informatik nichts für Mädchen ist, und andererseits allgemein ein realistischeres Bild der Informatik und der damit verbundenen Arbeitsweisen vermitteln.

Informatisches Selbstkonzept

Der Begriff *Selbstkonzept* bezeichnet nach Shavelson et al. (1976) 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 und Köller 2004). Laut Hellmich (2011) werden diese 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 (Müller 2015). 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 (Marsh und Martin 2011; Marsh

und Craven 2006). Aus diesem Grund 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) die subjektive Gewissheit einer Person, schwierige oder neue Anforderungen mit Hilfe 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 (1997) kann sich Selbstwirksamkeit zum Beispiel durch positives Feedback, das Erfahren von Erfolgserlebnissen, 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). Dabei wird herausgestellt, dass Kinder, die von Armut betroffen oder anderweitig sozial benachteiligt sind, 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. Pajares 2005). Die Studie von Kind (2015) zeigte ähnliche Ergebnisse bei Zweitklässlern in Bezug auf die Informatik und konnte darüber hinaus aufzeigen, dass eine neunwöchige Intervention im Programmieren die Geschlechterunterschiede in der Selbstwirksamkeit verringerte.

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 Perspektive

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 (Bell und Duncan 2018; Peyton Jones und Muuß-Meerholz 2014). Dieser Trend zeigt sich zum einen auch 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.

Informatik als eigenes Fach

In mehreren Ländern wird Informatik in der Grundschule als eigenes Fach unterrichtet, zum Beispiel in Australien (Falkner et al. 2014), Israel (Storte et al. 2019), Neuseeland (Kellow 2018), Polen (Sysło und Kwiatkowska 2015), Großbritannien (Berry 2018), der Schweiz (D-EDK 2016) und der Slowakei (Kabátová et al. 2016). Die konkreten Bezeichnungen des Fachs unterscheiden sich dabei erheblich. So wird das Fach teilweise eindeutig als Informatik bezeichnet, z.B. als *Computing* in Großbritannien oder *Informatika* in der Slowakei. 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 ebenfalls die Themen Daten, Informatiksysteme, Programmierung sowie algorithmisches Denken unterrichtet. Das Fach, welches zum größeren Lernbereich *technologies* gehört, wird dort als *digital technologies* bezeichnet (Falkner et al. 2014).

In Neuseeland wird Informatik und Medienkompetenz in der Grundschule eng verknüpft. Das neuseeländische Fach *technology*, das sich bisher 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 (Kellow 2018). Auch in den deutschsprachigen Kantonen der Schweiz unterrichten Lehrkräfte seit 2016 informatische und medienbezogene Inhalte in dem Fach *Medien und Informatik*. Im Kompetenzbereich Informatik werden die Themen Datenstrukturen, Algorithmen und Informatiksysteme aufgeführt (D-EDK 2016).

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 (Sysło und Kwiatkowska 2015), wird der Fokus nun stärker auf das Programmieren und *Computational Thinking* gelegt (van Blommenstein 2016). Auch in Israel wird derzeit schrittweise ein neues Curriculum für die Grundschule eingeführt, das den Schwerpunkt auf *Computational Thinking* legt (Storte et al. 2019).

In den USA wurde bereits für alle Altersstufen ein Informatikcurriculum entwickelt (K–12 Computer Science Framework 2016). Das amerikanische Bildungssystem ist jedoch stark dezentralisiert, sodass dieses bisher wenn überhaupt sehr unterschiedlich umgesetzt wird (Heintz et al. 2016).

Integration von informatischen Inhalten

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 (Dagienė et al. 2019). Dieses soll ab dem Schuljahr 2020/21 in Mathematik und anderen Fächern integriert unterrichtet werden (Xuequan 2018).

Auch in Schweden wird Informatik integriert in mehrere Fächer unterrichtet (Heintz et al. 2017). 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.

In Südkorea wird ebenfalls Problemlösen, Algorithmen und Programmieren unterrichtet. Diese 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).

In Finnland wird ein besonderer Fokus auf das Programmieren gelegt (ebd.). 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*[2] *Competence*, sodass die Lehrkräfte auch in anderen Fächern mit den Kindern programmieren können.

Informatik als Querschnittsaufgabe

Während die informatischen Inhalte in einigen Ländern konkret einzelnen Fächern zugeordnet werden, finden sich auch freiere Umsetzungen. In Italien, Frankreich und der Türkei wurde *Computational Thinking* und Programmieren laut Bocconi et al. (2016, S. 28) in der Grundschule als Querschnittsaufgabe eingeführt. Im Grundschulcurriculum in Estland findet man den Bereich *Technologie und Innovation* als Querschnittaufgabe, die alle Lehrkräfte umsetzen müssen. Darüber hinaus gibt es dort verschiedene fakultative Inhalte, wie zum Beispiel Programmierung, Robotik oder 3D-Grafik, die die

[2] *ICT* steht für *information and communications technology* (auf Deutsch *Informations- und Kommunikationstechnik*)

Schulen in ihr Programm aufnehmen können (HITSA 2020). Unterstützt werden sie dabei vom Projekt *ProgeTiger*, das 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).

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 konzentrieren sich dabei hauptsächlich auf das Programmieren. So zum Beispiel die kostenfrei nutzbaren Programmierwebseiten *Code.org*[3], *Scratch*[4] oder *Blockly*[5] sowie Apps, wie *ScratchJr*[6] oder *codeSpark Academy*[7]. 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 selbst verwenden kann (Brennan 2013).

Das gemeinsame Arbeiten steht auch bei verschiedenen Clubs im Fokus, zum Beispiel dem *Clubhouse Network*[8], das in den USA gegründet wurde. Diese vermitteln zwar auch informatische Inhalte, im Vordergrund steht jedoch, dass die Kinder und Jugendlichen lernen, sich mit neuen Technologien auszudrücken (Resnick et al. 1998). Auch bei den *CoderDojos*[9] können Kinder und Jugendliche in mittlerweile über 69 Ländern informatische Inhalte lernen – der Schwerpunkt liegt dabei vor allem auf dem Programmieren (Quigley 2017). Die *CoderDojo Girls Initiative*[10] 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 gibt es die Initiative *Code Club*[11], 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 et al. 2014).

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.

Bundesweite Initiativen

In dem Positionspapier zur digitalen Bildung zählt das Bundesministerium für Wirtschaft und Energie (2016) das Einführen von Grundkenntnissen in Informatik, Programmieren und Algorithmen als Pflichtbestandteil der Lehrpläne für die Grundschule zu den besonders wichtigen Zielen. Das Erkennen und Formulieren von algorithmischen Strukturen schule das analytische Denken, Kreativität sowie die Fähigkeit des Problemlösens. Auch die Kultusministerkonferenz zählt in ihrem Strategiepapier zur Bildung in der digitalen Welt (KMK 2016) das Erkennen und Formulieren von Algorithmen zum Kompetenzbereich *Problemlösen und Handeln*. Das Strategiepapier dient als Grundlage für die Überarbeitung der Lehrpläne aller Schularten.

Die Gesellschaft für Informatik geht noch einen Schritt weiter und formuliert in ihren Empfehlungen zur informatischen Bildung im Primarbereich Kompetenzerwartungen zu fünf

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

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 Empfehlung legt sich nicht fest, in welcher Organisationsform die informatische Bildung im Primarbereich stattfinden sollte, sie 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. 13).

Auf Initiative der *Stiftung Haus der kleinen Forscher* bildete sich eine Arbeitsgruppe, die ebenfalls Zieldimensionen für eine frühe informatische Bildung formulierte (HdkF 2018). Diese beziehen sich nicht nur auf die Grundschule, sondern auch auf den Kindergarten. Wie schon die Arbeitsgruppe der Gesellschaft für Informatik greifen sie dabei auf bereits bestehende Standards für die Sekundarstufen zurück (GI 2008, 2016b). Sie erweitern diese jedoch um den Prozessbereich *Interagieren und Explorieren* (vgl. Bergner et al. 2017; Müller et al. 2019).

Straube et al. (2018) stellen ebenfalls fest, dass Kompetenzen im Bereich der Digitalisierung, die für eine gesellschaftliche Teilhabe als mündige Bürger notwendig sind, durch die Curricula der Primarstufe nicht abgedeckt werden. Sie schlagen eine digitale Perspektive für den Sachunterricht vor, der als sechste Perspektive in den Perspektivrahmen Sachunterricht aufgenommen werden sollte (Brämer et al. 2020). Darin stellen sie drei perspektivbezogene Themenbereiche vor: *Robotik und Softwaregestaltung*, *Computer, Smartphone und co.* sowie *Alltag in einer Medienwelt*. 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 C*omputational Thinking* (z.B. „die Schülerinnen und Schüler können informatische Konzepte und Algorithmen für die Optimierung der eigenen Vorgehensweise beim Problemlösen nutzen“ (ebd., S. 4) als auch auf konkrete informatische Inhalte (z.B. „die Schülerinnen und Schüler können die Übertragung von Daten innerhalb und zwischen Informatiksystemen verstehen“ (ebd., S. 4).

Die Roberta-Initiative[12], 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 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.

Auch beim Informatik-Biber[13] 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.

Situation in einzelnen Bundesländern

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

[12] https://www.roberta-home.de/

[13] https://bwinf.de/biber/

dessen Beschreibung zum Beispiel das Erkennen von Algorithmen sowie das Modellieren und Programmieren gefordert werden. In Bezug auf die Grundschule wird u.a. 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., S.22).

In Sachsen trat im August 2019 ein überarbeiteter Lehrplan für die Grundschule in Kraft, der im Fach Werken den Lernbereich *Begegnung mit Robotern und Automaten* einführt (SMK Sachsen 2019). Dieser soll in der Klassenstufe 4 im Umfang von sechs Unterrichtsstunden unterrichtet werden. 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 (ebd., S.14f.).

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* sowie Grundkonzepte der informatischen Bildung in der Primarstufe thematisierten (Reese und Wolf 2018). 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 laufen mittlerweile in fast allen Bundesländern (Calliope gGmbH 2020).

Forschungsprojekte

Der Einsatz des *Calliope mini* wurde in verschiedenen Studien wissenschaftlich untersucht (z.B. Murmann et al. 2018; Baum et al. 2019; Goecke und Stiller 2018). Auch darüber hinaus werden auf Seiten der Forschung verschiedene Anstrengungen unternommen, Kindern die Möglichkeit zu geben, grundlegende Kenntnisse im Bereich Informatik zu erwerben und zu untersuchen, welche Unterrichtsmethoden und -inhalte sich für Grundschulen in Deutschland eignen. Im Projekt *IT2School – Gemeinsam IT entdecken* wurden in Zusammenarbeit mit der *Wissensfabrik*[14] Unterrichtsmodule und passende Materialien entwickelt, die Einblicke in Themen wie Kommunikation, Daten, Programmiersprache und das Zusammenspiel von Hard- und Software geben (Diethelm und Schaumburg 2016). Ausgewählte Module wurden vom niedersächsischen Kultusministerium aufgegriffen und im Projekt *Informatische Bildung und Technik in der Grundschule*[15] verwendet. In diesem Projekt des Ministeriums wird die Vermittlung von informatischen Grundlagen an 31 Pilotschulen erprobt und vom Institut für innovative Bildung (ifib) wissenschaftlich evaluiert (Riefling et al. 2020).

Die Forschungsgruppe Elementarinformatik (FELI) 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 (Gärtig-Daugs et al. 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).

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 et al. 2018). 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,

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

[15] https://www.infgsnds.de/doku. php

behandeln die Themen *Informationen und Daten*, *Kryptologie* und *Programmierung*. Die Evaluation des Projekts beschäftigt sich neben dem Lernfortschritt der Kinder mit deren Motivation und Interesse bezüglich Informatik.

Am Leibniz-Institut für Wissensmedien 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 entwickelt wurden (Tsarava et al. 2018). Die Arbeitsgruppe entwickelte außerdem einen Fragebogen, der das Selbstkonzept sowie die Einstellungen von Kindern in Bezug auf das Programmieren erfasst (Leifheit et al. 2020).

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*[16]. Zudem wurden Unternehmen gegründet, die Informatik-Kurse für Kinder und Jugendliche anbieten, z.B. die *Hacker School*[17] oder die *HABA Digitalwerkstätten*[18].

In vielen deutschen Städten bieten Schülerlabore Kurse sowohl für interessierte Kinder und Jugendliche als auch für ganze Klassen an und ergänzen damit den Schulunterricht (Euler et al. 2015). Einige Schülerlabore widmen sich ausschließlich der Vermittlung von informatischen Inhalten, zum Beispiel das *Schülerlabor Informatik InfoSphere*[19] der RWTH Aachen oder das *Schülerrechenzentrum*[20] der TU Dresden.

Auch online stehen viele 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!*[21], *Programmieren mir der Maus*[22] oder Ronjas Roboter[23]. Letztere ist auch als App verfügbar und bietet zusätzlich Tipps zur Lernbegleitung sowie Vertiefung der Inhalte an.

2.4 Befunde zur Lehrerbildung

Kindgerechte Materialien und Programmierumgebungen reichen jedoch nicht aus, um Informatik an Grundschulen zu unterrichten. 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 (vgl. Brown et al. 2013).

Notwendigkeit einer adäquaten Ausbildung

Goode et al. (2014) sowie Meerbaum-Salant et al. (2010) fanden heraus, dass die blockbasierte Programmiersprache *Scratch* zwar zum Unterrichten von grundlegenden Programmierkonzepten geeignet ist – jedoch nur in dem Fall, dass man die Lehrkräfte in der didaktischen Umsetzung anleitet. Yadav et al. (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. 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 (Sentance und Csizmadia 2017; Reding und Dorn 2017).

Selbstsicherheit als Ziel

Erwachsene, vor allem Lehrkräfte, werden von Kindern als Verhaltensmodelle betrachtet

[16] https://www.coderdojo-deutschland.de/
[17] https://hacker-school.de/
[18] https://www.digitalwerkstatt.de/
[19] https://schuelerlabor.informatik.rwth-aachen.de/
[20] https://www.srz.tu-dresden.de/
[21] https://code-it-studio.de/
[22] https://programmieren.wdrmaus.de/
[23] https://www.meine-forscherwelt.de/spiel/ronjas-roboter

und haben somit zwangsläufig Einfluss auf deren Selbstkonzept (Tausch und Tausch 1998, S. 19ff.). Makris et al. (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. Bereits elf Jahre zuvor kommen Condie und Munro (2006, S. 63) in ihrer Metastudie zu dem Ergebnis, dass es nicht ausreicht, Lehrkräfte nur darin fortzubilden, neue Technologien zu verwenden. Sie weisen darauf hin, dass die fehlende Vermittlung von Fachdidaktik sich negativ auf deren Selbstsicherheit auswirkt.

Klassenleiterprinzip als Chance

Während Informatik an den weiterführenden Schulen 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 über kein oder sehr wenig Vorwissen verfügen und nur bedingt Unterrichtserfahrungen in der eigenen Schulzeit sammeln konnten (Best 2019; Döbeli Honegger 2015).

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 et al. (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 et al. (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 den Informatikkonzepten aufzeigen.

Lockwood und Mooney (2017, S. 16ff.) tragen verschiedene Studien zusammen, in denen *Computational Thinking* 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.

Gestaltung von Fortbildungsprogrammen

Webb et al. (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. Die Weiterbildung von Lehrkräften, die bereits im Lehrerberuf tätig sind, ist daher unerlässlich. International gibt es dazu unterschiedliche Ansätze.

In Großbritannien gibt es die Initiative *Computing at School (CAS)*[24], die Lehrkräfte befähigen möchte, den britischen Lehrplan mit Selbstvertrauen und Begeisterung zu unterrichten, indem sie regionale *Communities of Practice* aufbaut (Sentance und Humphreys 2015). 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 et al. 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).

In Australien gibt es viele Grundschullehrkräfte in sehr abgelegenen Gebieten, für die es schwer ist, an Fortbildungsmaßnahmen teilzunehmen. Um dem zu entgegnen und gleichzeitig ein Angebot für viele Lehrkräfte zur Verfügung zu stellen, entwickelten Vivian et al. (2014a) einen *MOOC*[25], der fachwissenschaftliche und fachdidaktische Inhalte der Informatik thematisiert. Zudem wurden soziale Medien eingebunden, um den Teilnehmenden einen Austausch zu ermöglichen. In Deutschland entwickelte die Universität Bamberg in Zusammenarbeit mit der Ludwig-Maximilians-Universität München einen MOOC zum *Verstehen der digitalen Welt*[26], in dem sie in sieben Kapiteln *Computational Thinking* sowie grundlegende Informatikkonzepte thematisieren.

Informatik in der ersten Phase der Grundschullehrerbildung

Spätestens, wenn informatische Inhalte im Curriculum der Grundschule gefordert werden, muss sich auch die erste Phase der Lehrerausbildung – 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 et al. 2017).

An der PH Schwyz wurde die Informatik-Lehrveranstaltung *Grundlagen der Informatik* eingeführt, die alle angehenden Grundschullehrkräfte im ersten Semester belegen müssen. Döbeli Honegger et al. (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.

An der PH der Fachhochschule Nordwestschweiz entschied man sich für einen Kurs über zwei Semester, der für alle angehenden Grundschullehrkräfte obligatorisch ist. 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). Die wöchentlichen Sitzungen beinhalten einen Theorieteil, in denen

[24] https://www.computingatschool.org.uk/
[25] Ein Massive Open Online Course (MOOC) ist ein in der Regel kostenfreier Online-Kurs.
[26] https://t1p.de/ejv5

grundlegende informatische Themen, wie zum Beispiel Daten, Algorithmen, Programmieren oder das Internet, behandelt werden, und einen Praxisteil. In diesem arbeiten die Studierenden an eigenen Projekten mit dem Programmiertool *AgentCubes*[27].

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 (Haselmeier 2019). 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, 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.

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

3. Das Projekt im Überblick

Als relativ junge Disziplin weist die Informatik im Gegensatz zu den anderen Schulfächern immer noch ein großes Defizit an empirisch fundierter Unterrichtsforschung auf. Dies trifft insbesondere auf die frühkindliche Bildung sowie die Bildung in der Primarstufe zu. Zwar gibt es vielversprechende Lernumgebungen und didaktische Ansätze zur Gestaltung von informatischer Bildung für Kinder im Grundschulalter, jedoch wenig fundierte Erkenntnisse über die Umsetzung in Grundschulen oder darüber, wie eine angemessene Lehrerausbildung aussehen könnte. Das Projekt *AlgoKids – Algorithmen für Kinder* leistet einen Beitrag dazu, diesem Defizit entgegenzuwirken.

Digitale Bildung als gemeinsames Anliegen

Laut Bildungs- und Erziehungsauftrag schafft die Grundschule eine verlässliche Grundlage sowohl für die weitere schulische Bildung als auch die kulturelle und gesellschaftliche Teilhabe (StMBKWK 2014, S. 19). In den schulart- und fächerübergreifenden Zielen heißt es weiter, dass Schülerinnen und Schüler Kenntnisse und Fertigkeiten erwerben, um sachgerecht, selbstbestimmt und verantwortungsvoll in einer multimedial geprägten Welt zu agieren (ebd., S. 35). Digitale Bildung ist nicht nur eine verbindliche Aufgabe des LehrplanPLUS Grundschule, sondern gleichzeitig auch ein Anliegen der Technischen Universität München (TUM). Da Informatik die fundamentale Bezugswissenschaft der Digitalisierung darstellt, sollten informatische Grundkenntnisse und Fertigen auch in der Grundschule berücksichtigt werden.

Vorarbeiten an der TUM

Um die Möglichkeiten und Grenzen der Programmierung mit Grundschulkindern auszuloten, entwickelte die Professur für Didaktik der Informatik der TU München (TUMDDI) den *Programmierzirkus*, einen dreitägigen Programmierkurs speziell für Grundschulkinder der dritten und vierten Klasse (Geldreich et al. 2019a; Geldreich et al. 2017, 2016). Das Dreitagesformat (Abb. 1) erfüllt einerseits gerade noch die zeitlichen Mindestanforderungen, die durch die Inhalte vorgegeben werden, andererseits kann ein Dreitageskurs unter Umständen in den regulären Stundenplan eingeschoben werden, zum Beispiel in Form von Projekttagen.

Tag 1 – Arbeiten ohne Computer

- Programmieren der Lehrkraft
- Umwandeln von bildhaften Anleitungen in sprachliche Befehle
- Programmieren von Wegen in einem Parcours mit haptischen Programmierblöcken

Ziel: Einblick in grundlegende Funktionswesen von Programmen

↓

Tag 2 – *Scratch*-Lernzirkel

- Lernzirkel zu Funktionsweisen von *Scratch* und verschiedenen algorithmischen Strukturen

Ziel: Kennenlernen von Sequenzen, Wiederholungen und bedingten Anweisungen

↓

Tag 3 – eigene Projekte in *Scratch*

- Drehbuch zu einer kurzen Handlung erstellen
- Implementieren des Drehbuchs in *Scratch*
- Präsentation der Ergebnisse

Ziel: Eigenständiges Anwenden der Programmierkonzepte in einem offenen Projekt

Abb. 1 Ablauf des dreitägigen Programmierkurses

Nach Abschluss der Unterrichtssequenz sollen die Kinder wissen, wie ein Computerprogramm grundsätzlich aufgebaut ist, mit einer kindgerechten Programmierumgebung arbeiten und damit kleine multimediale Projekte

programmieren können. Die Sequenz wurde bereits mit ca. 150 Grundschulkindern im Alter von 8-10 Jahren in München und im Berchtesgadener Land außerhalb des Unterrichts erprobt. Diese Probeläufe wurden ausführlich dokumentiert und ausgewertet (vgl. Funke et al. 2017; Funke und Geldreich 2017; Geldreich et al. 2019b; Simon et al. 2019). Die ersten Ergebnisse zeigen, dass die meisten Kinder dabei überraschend gute Lernerfolge erzielen.

Auf dieser Grundlage wurde das Projekt Algorithmen für Kinder (AlgoKids) als Kooperationsprojekt zwischen der Professur für Didaktik der Informatik und dem Bayerischen Staatsministerium für Unterricht und Kultus gegründet. Die TUMDDI sollte das Projekt konzipieren, durchführen und wissenschaftlich auswerten. Das Staatsministerium übernahm die Finanzierung des wissenschaftlichen Personals und dessen Reisekosten.

Konzeption

In dem Projekt sollte untersucht werden, wie man die von der TUMDDI entwickelte und pilotierte Unterrichtsintervention im regulären Unterricht der bayerischen Grundschulen implementieren könnte. Dazu wollten wir die Einsatzmöglichkeiten der Intervention systematisch in den Jahrgangsstufen 3 und 4 an ca. 20 Grundschulen erproben. Der *Programmierzirkus* sollte an den Projektschulen erprobt und an ihre jeweiligen schulischen Anforderungen angepasst bzw. erweitert werden. In Schulbesuchen und Reflexionsgesprächen wollten wir beobachten, welche Inhalte und Methoden sich dabei bewährten. Insbesondere wollten wir dabei auch geeignete Anknüpfungspunkte zum LehrplanPLUS finden.

Zweitens wollten wir herausfinden, wie man Grundschullehrkräfte ohne informatische Vorbildung so fortbilden kann, dass sie die Themen Algorithmen und Programmierung kompetent unterrichten können. Im Rahmen des Projekts sollte dazu ein Fortbildungskonzept entwickelt und evaluiert werden, auf dessen Grundlage man später alle bayerischen Grundschulen in die Lage versetzen könnte, eine erste Einführung in das Programmieren dauerhaft an ihren Schulen zu etablieren.

Das Projekt war ursprünglich auf 24 Monate ausgelegt. Aufgrund einer Verzögerung des Projektstarts wurde die Projektlaufzeit auf den Zeitraum von 1. März 2018 bis 31. Dezember 2019 verkürzt. Ab Februar 2018 konnten sich alle staatlichen Grundschulen auf der Basis eines durch das Ministeriums initiierten bayernweiten Ausschreibungsverfahren bewerben.

Auswahl der Projektschulen

Nach der Ausschreibung durch das Staatsministerium im Februar 2018 konnten sich alle staatlichen Grundschulen in Bayern bewerben.

Die Auswahl der Schulen erfolgte durch das Staatsministerium unter Berücksichtigung der für Schulversuche üblichen Repräsentativitätskriterien, wie dem Einbezug aller Regierungsbezirke (Abb. 2), der Schulgröße, Erfahrungen im Bereich der digitalen Bildung sowie der digitalen Ausstattung. Die Relevanz des Anliegens zeigte sich an den zahlreichen Bewerbungen, die nach Angaben des Ministeriums das zur Verfügung stehende Platzkontingent von 20 Projektschulen deutlich überstiegen.

Abb. 2 Übersicht über Verteilung der Projektschulen

Die ausgewählten Schulen wurden im April 2018 über ihre Zulassung informiert und aufgefordert, jeweils zwei Lehrkräfte für die erste Fortbildungsveranstaltung anzumelden. Insgesamt 40 Lehrkräfte wurden somit von der TUM 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* (Tabelle 1). 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.

Tabelle 1 Altersverteilung der teilnehmenden Lehrkräfte

Alter	Anzahl Lehrkräfte
unter 30 Jahre	8
30-40 Jahre	17
41-50 Jahre	8
über 50 Jahre	8

Fortbildungsmaßnahmen

Für die Fortbildung der beteiligten Lehrkräfte hatte die TUMDDI ein spezielles Konzept entwickelt, das in Zusammenarbeit mit der *Akademie für Lehrerfortbildung und Personalführung* (ALP) realisiert wurde. Die ALP übernahm dabei die Organisation der Kurse, die TUMDDI war für die Inhalte und Methoden verantwortlich (siehe Kapitel 6). Um eine bestmögliche Betreuung sicherzustellen, wurden die Lehrkräfte in zwei Gruppen aufgeteilt. Für die Fachinhalte waren Referentinnen und Referenten der TUMDDI zuständig, die von einer Fachreferentin für die Grundschule sowie einem Fachreferenten für E-Learning der ALP unterstützt wurden. Am Anfang standen für jede Gruppe je eine Fortbildungshalbwoche an der ALP im Mai bzw. Juni 2018. Der Fokus lag dabei auf dem Kennenlernen und Ausprobieren des Unterrichtskonzepts der TUM und den darin behandelten informatischen Inhalten.

Die zweite Fortbildungsserie im Umfang je einer Halbwoche für jede Gruppe wurde im Dezember 2018 ebenfalls an der ALP durchgeführt. Hier lag der Schwerpunkt auf einem Erfahrungsaustausch bezüglich der ersten Erprobungen der Unterrichtssequenz sowie auf der Vertiefung von fachdidaktischen Themen.

Die abschließende dritte Fortbildung fand für beide Gruppen gemeinsam im November 2019 in den Räumlichkeiten der TUM School of Education in München statt. Hier standen fachdidaktische Themen und die Diskussion der Ergebnisse im Vordergrund. Das gemeinsame Format wurde gewählt, um die Erkenntnisse aus dem Projekt unter allen beteiligten Lehrerinnen und Lehrern diskutieren zu können.

Erprobung an den Projektschulen

Bereits nach der ersten Fortbildung erprobten die Lehrkräfte das Unterrichtskonzept erstmals mit ihren Schülerinnen und Schülern. Wie bereits erwähnt, waren sie dabei nicht auf ein bestimmtes Umsetzungsformat beschränkt. Die Lehrkräfte konnten sich auch jederzeit telefonisch oder per Mail an die Projektleitung wenden, um offene Fragen zu klären oder Feedback zu ihren eigenen Unterrichtsideen einzuholen.

Für alle in der ursprünglichen Unterrichtssequenz benötigten Materialien wurden von der TUM Druckvorlagen zur Verfügung gestellt. Darüber hinaus erhielten alle Lehrkräfte einen Satz haptische Programmierbefehle sowie eine Filzmatte, auf der sie mit den Befehlen programmieren können. Die TUM stellte zudem einen mobilen Satz von 20 Tablet-Computern zur Verfügung, den die Projektschulen bei Bedarf ausleihen konnten.

Unterrichtsbesuche

Während der Erprobung der Unterrichtssequenz sollten alle teilnehmenden Schulen mindestens einmal von der Projektleitung besucht werden. Bis auf zwei Schulen, bei denen ein Besuch aufgrund von organisatorischen

Hindernissen seitens der Schule nicht zustande kam, wurden alle Schulen zwischen einem und drei Mal besucht. Die Schulbesuche erfolgten nach Abschluss der ersten Fortbildungsserie im Juni 2018 bis zum Ende des Schuljahres 2018/19. Im Rahmen der Besuche wurden insgesamt neunzehn Reflexionsgespräche mit den Lehrkräften geführt. Diese fanden überwiegend gleichzeitig mit beiden Lehrkräften der jeweiligen Projektschule statt.

Ausblick

Auf der Grundlage der Ergebnisse und Empfehlungen dieses Abschlussberichts entscheidet das Staatsministerium über die weitere Implementierung des Konzepts. Das Team der TUM-DDI steht für weitere Fortbildungen sowie fachliche Beratung auch in Zukunft gerne zu Verfügung.

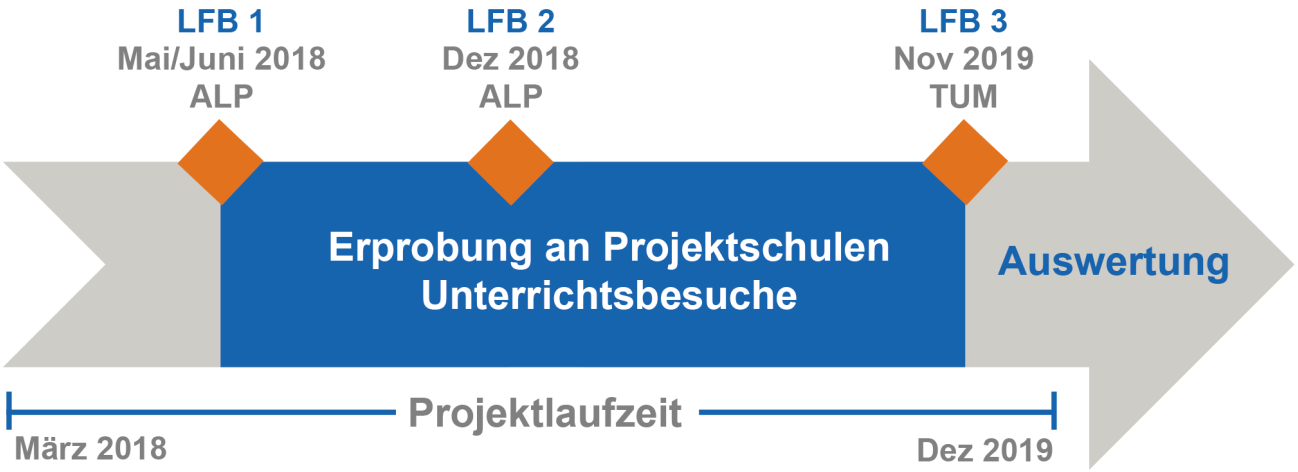


Abb. 3 Übersicht über den Projektverlauf

4. Fachliche und fachdidaktische Grundlagen

Die Unterrichtssequenz zur Algorithmik und Programmierung in der Grundschule wurde nach dem Ansatz der *Design-Based Research* entwickelt (vgl. Geldreich et al. 2019a). Dieser Forschungsansatz ist besonders geeignet, nachhaltige Innovationen für den Unterricht zu entwickeln (Reinmann 2005). Im Folgenden werden zunächst die fachlichen Inhalte erläutert, die im Laufe der Unterrichtssequenz explizit oder implizit thematisiert werden. Anschließend wird ein Auszug der fachdidaktischen Prinzipien und Konzepte beschrieben, die für das Unterrichtskonzept von zentraler Bedeutung sind. Das Kapitel schließt mit möglichen Bezügen der Unterrichtssequenz zum LehrplanPLUS.

4.1 Fachliche Grundlagen

Algorithmus

Algorithmen sind eine der ältesten (abstrakten) Beschreibungstechniken für Abläufe. Damit sie im gewünschten Sinne ausgeführt werden, müssen sie präzise in einer Sprache formuliert werden, die der Ausführende (z.B. ein Informatiksystem oder ein Mensch) versteht. Im Alltag begegnet man Algorithmen zum Beispiel in Form von Kochrezepten, Wegbeschreibungen oder Bauanleitungen.

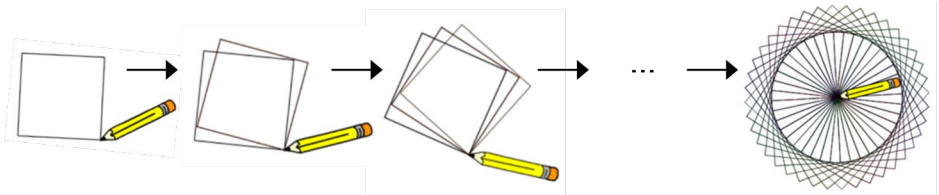
Broy (1998, S. 31) definiert einen Algorithmus als ein Verfahren mit einer präzisen (d.h. in einer genau festgelegten Sprache abgefassten) endlichen Beschreibung unter Verwendung effektiver (das heißt tatsächlich ausführbarer) elementarer (Verarbeitungs-)Schritte.

Computerprogramm

Ein Computerprogramm ist die Darstellung eines bestimmten Algorithmus in einer Sprache, die von einem Rechner automatisch interpretiert werden kann. Solche Sprachen, wie z.B. *Scratch* oder *Java*, heißen Programmiersprachen. Erst, wenn ein Algorithmus mittels einer Programmiersprache beschrieben wird, kann ihn ein Informatiksystem ausführen.

Beispiel: Ein Algorithmus in der Programmiersprache *Scratch*

Im Alltag werden Abläufe oftmals bildlich dargestellt, wie zum Beispiel der Ablauf des Zeichnens einer geometrischen Figur:



Der gleiche Ablauf lässt sich in der Programmiersprache *Scratch* beschreiben und kann auf diese Weise von einem Computer automatisch ausgeführt werden:

→ *Scratch ist eine präzise festgelegte Sprache.*
Die Beschreibung eines Programms (die Anzahl der Blöcke) ist endlich und jeder einzelne Block-Befehl ist ausführbar.

Programmiersprache

Mithilfe von Programmiersprachen kann man Algorithmen so präzise darstellen, dass ein Computer sie ausführen kann. Es gibt viele verschiedene Programmiersprachen, die sich in ihrer Syntax unterscheiden. Man findet jedoch in den meisten Programmiersprachen die gleichen algorithmischen Strukturelemente.

Scratch

Scratch ist eine visuelle Programmiersprache samt Entwicklungsumgebung, die 2007 am Massachusetts Institute of Technology (MIT) entwickelt wurde. *Scratch* ist so konzipiert, dass Programmieranfänger – insbesondere Kinder und Jugendliche – sich mit den Grundkonzepten der Programmierung vertraut machen können, ohne viel tippen zu müssen oder frustrierende Fehlermeldungen zu erhalten.

Algorithmische Strukturelemente

Alle Algorithmen und somit auch Programme lassen sich aus wenigen strukturellen Bausteinen und deren Kombinationen zusammensetzen:

Anweisung: Eine Anweisung ist ein elementarer, unteilbarer Verarbeitungsschritt, in *Scratch* ein einzelner Befehlsblock.

Sequenz: Mehrere Anweisungen, die hintereinander ausgeführt werden, bilden zusammen eine Sequenz. Die einzelnen Anweisungen werden innerhalb der Sequenz genau in er angegebenen Reihenfolge abgearbeitet.

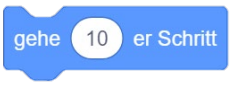
Wiederholung mit fester Anzahl: Oft müssen eine oder mehrere Anweisungen (Sequenz) mehrfach wiederholt werden. Wenn die Anzahl der Wiederholungen feststeht, nutzt man eine Wiederholung mit fester Anzahl.

Bedingte Wiederholung: In anderen Fällen soll eine Sequenz von Anweisungen so oft wiederholt werden, bis eine Bedingung erfüllt ist. Ist die Bedingung von Anfang an erfüllt, wird die Sequenz überhaupt nicht ausgeführt.

Bedingte Anweisung (Verzweigung): Manche Sequenzen sollen nur unter einer bestimmten Bedingung ausgeführt werden. Oft will man zusätzlich einen alternativen Verarbeitungsweg (sonst-Teil) angeben, der auszuführen ist, falls die genannte Bedingung nicht erfüllt ist.

Beispiel: Strukturelemente in *Scratch*

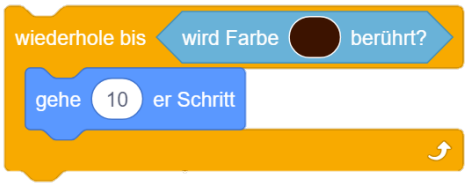
Anweisung



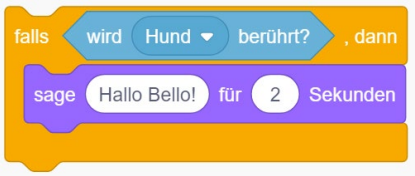
Sequenz

Wiederholung mit fester Anzahl

Bedingte Wiederholung



Bedingte Anweisung (Verzweigung)



Computational Thinking

Der Begriff *Computational Thinking* beschreibt den Ansatz, Probleme mithilfe informatischer Konzepte und Herangehensweisen zu bearbeiten. Er umfasst alle Denkprozesse, die daran beteiligt sind, ein Problem zu identifizieren und so aufzubereiten, dass ein Mensch oder Computer diese ausführen kann (Wing 2006).

Aspekte von Computational Thinking nach Berry (2015)[28]

Logisches Schlussfolgern

Vorhandenes Wissen über ein System nutzen, um zuverlässige Vorhersagen über dessen zukünftiges Verhalten zu treffen.

Algorithmen

Schritte oder Regeln aufstellen, um ein Problem oder eine Aufgabe zu bearbeiten.

Zerlegung

Ein Problem in kleinere, überschaubarere Teile zerlegen, um auf diese Weise auch komplexere Probleme zu bewältigen.

Abstraktion

Unterscheiden, was zur Lösung eines Problems erforderlich ist und was weggelassen werden kann.

Generalisierung

Ähnlichkeiten und Muster erkennen und bei der Lösung von Problemen verwenden.

Evaluation

Sicherstellen, dass die gefundene Lösung eines Problems auch ihren Zweck erfüllt.

Programmieren

Unter Programmieren versteht man die Aufstellung eines Algorithmus zur Lösung eines Problems (vgl. *Computational Thinking*) 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 (Duncan et al. 2017). Landläufig hat sich der Begriff *Coding* als Synonym für das Programmieren verbreitet. Der Nachteil dieser Formulierung ist, dass sie suggeriert, es würde sich dabei nur um den Vorgang des Codeschreibens handeln (Duncan et al. 2014).

EVA-Prinzip

Die Abkürzung *EVA* steht für Eingabe, Verarbeitung und Ausgabe. Viele Informatiksysteme funktionieren nach diesem Prinzip: zuerst werden Daten eingegeben, dann verarbeitet und schließlich wird wieder etwas ausgegeben. Die Eingabe erfolgt über sogenannte Eingabegeräte, zum Beispiel der Maus, der Tastatur oder einem Mikrofon. Der Computer verarbeitet dann die Daten – hier sind zum Beispiel der Prozessor, der Arbeitsspeicher und die Grafikkarte beteiligt. Nach der Verarbeitung werden die Daten wieder an den Nutzer ausgegeben. Das geschieht in den meisten Fällen über den Monitor, eine Ausgabe ist aber auch über andere Ausgabegeräte möglich, zum Beispiel über Lautsprecher oder Drucker.

4.2 Fachdidaktische Grundlagen

Informatik auch 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*

[28] Die Symbole sind den Visualisierungen von *Barefoot* nachempfunden: https://www.barefootcomputing.org

oder konkrete Fachinhalte verdeutlicht (vgl. HdkF 2018, S. 86; Bell et al. 2015).

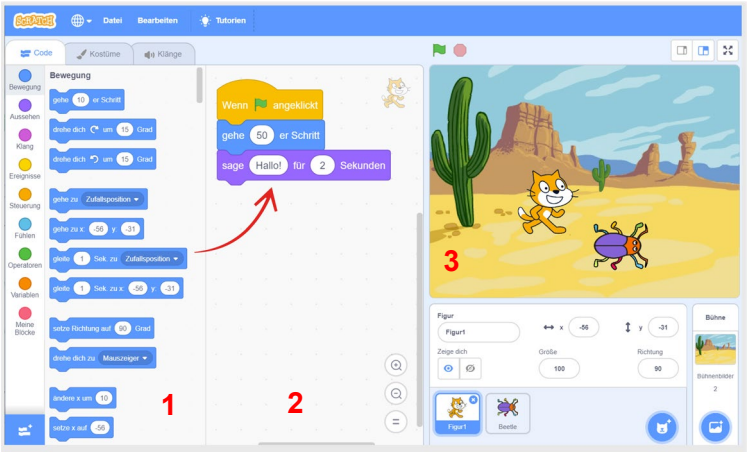
Algorithmische Grundstrukturen

Laut Hubwieser (2007, S. 82f.) 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) 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 Informatiksystem notwendig sind. Er führt weiter aus, dass man beim Unterrichten der Programmierung auch allgemeine Herangehensweisen zur Lösung eines Problems vermitteln sollte.

Visuelle Programmiersprachen

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). 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 (Papert und Harel 1991). Ein Ansatz, Schülerinnen und Schüler die Grundlagen der Programmierung näher zu bringen und ihnen gleichzeitig kreatives Arbeiten zu ermöglichen, ist die Programmierumgebung *Scratch*[29] (Resnick et al. 2009). *Scratch* bietet eine visuelle Programmiersprache, in der man den Programmcode aus Blöcken per *Drag and Drop* zusammenbaut. Es sind keine Vorkenntnisse der Programmiersyntax erforderlich, was bedeutet, dass die Lernenden schnell erste Ergebnisse erzielen, ohne entmutigende Fehlermeldungen zu erhalten (Vivian et al. 2014b). Zudem lassen sich in *Scratch* viele unterschiedliche Projekttypen nach den Vorstellungen der Schülerinnen und Schüler umsetzen, zum Beispiel Spiele, Geschichten oder Animationen.

Die Programmierumgebung *Scratch*



Die Oberfläche von *Scratch* besteht aus verschiedenen Bereichen. In der Blockpalette (1) findet man sämtliche Programmblöcke. Diese kann man nach rechts in den Programmierbereich (2) schieben und dort miteinander kombinieren. Rechts befindet sich die Bühne (3). Hier führen die Figuren das aus, was man für sie programmiert hat.

Abb. 4 Die Oberfläche von *Scratch*

[29] https://scratch.mit.edu/

Alltagstheorien zum Programmieren

Bereits vor dem Schuleintritt entwickeln Kinder die verschiedensten 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). Zwangsläufig entwickeln Kinder auch Vorstellungen darüber, wie Informatiksysteme funktionieren. Die Vorstellung, dass Computer auf magische Weise funktionieren oder ein eigenes Bewusstsein besitzen, ist unbedingt zu vermeiden.

Notional Machine

Eng verknüpft mit den subjektiven Theorien zum Programmieren ist die *notional machine*. Vereinfacht gesagt versteht man darunter die abstrakte Vorstellung davon, was passiert, wenn ein Programm ausgeführt wird (Du Boulay et al. 1999). Im Bereich der Grundschule gibt es nur wenig Befunde über die *notional machine*. Lowe (2018) zeigt jedoch, dass Zweitklässler bereits in der Lage sind zu verstehen, dass Programmierbefehle zu bestimmten Handlungen führen.

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 (Hassinen und Mäyrä 2006). Wie der Prozess des Programmierenlernens gestaltet werden sollte, ist jedoch nicht abschließend geklärt. Sowohl Lee et al. (2011) als auch Sentance und Waite (2017) beschreiben didaktische Ansätze, in denen die Lernenden auf dem Weg zum eigenständigen Programmieren begleitet werden können ohne sie zu überfordern. Die Schülerinnen und Schüler arbeiten zunächst mit bereits bestehenden Programmen, nehmen kleinere und größere Veränderungen daran vor und erstellen schließlich komplette Programme eigenständig. 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.

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. Waite et al. (2018) beschreiben vier Abstraktionsebenen, auf denen Planungsprozesse beim Programmieren stattfinden, die im Informatikunterricht berücksichtigt werden sollten. 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) 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 tut, was es soll, müssen Fehler im Programm erkannt und in einem nächsten Schritt korrigiert werden (Rich und Strickland 2020). Das Beheben von Fehlern stellt besonders für Programmieranfänger eine Herausforderung dar (Lahtinen et al. 2005) 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).

Programmieren als gemeinsame Tätigkeit

Ein Ansatz im Informatikunterricht, der den Fokus auf Zusammenarbeit und Absprache legt, ist das *Pair Programming* (Braught et al. 2011). 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 Navigators 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 et al. 2018, S. 70), und vor allem leistungsschwächere Lerner im Lernprozess unterstützen kann (Sands 2019, S. 47).

4.3 Bezüge zum LehrplanPLUS

Die Themen Algorithmen und Programmierung können mit verschiedenen bestehenden Kompetenzbereichen des LehrplanPLUS verknüpft werden und diese vertiefen oder erweitern.

Mathematik

Im Fachprofil der Grundschule für das Fach Mathematik sind im LehrplanPLUS die prozessbezogenen Kompetenzen *Modellieren*, *Probleme lösen* und *Darstellungen verwende*n beschrieben, die Schülerinnen und Schüler in Verbindung mit den inhaltsbezogenen Kompetenzen erwerben sollen (StMBKWK 2014, S. 106ff.). 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 Zahlen und Operatoren, TB Sachsituationen und Mathematik in Beziehung setzen[30])
- entnehmen relevante Daten aus verschiedenen Darstellungsformen und übertragen die Daten in geeignete andere Darstellungsformen.
  (LB Daten und Zufall, TB Daten erfassen und strukturiert darstellen)

Zusätzlich wird bei der Verwendung der algorithmischen Grundstrukturen beim Programmieren der Gegenstandsbereich *Muster und Strukturen*, der in allen Lernbereichen integriert werden soll, angesprochen.

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

[30] *LB* steht für Lernbereich, *TB* steht für Teilbereich

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 Sprechen und Zuhören, TB Über Lernen sprechen)

- veranschaulichen Abfolgen und Zusammenhänge im Text durch einfache Darstellungen.
(LB Lesen – mit Texten und weiteren Medien umgehen, TB 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 Lesen – mit Texten und weiteren Medien umgehen, TB 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 Schreiben, TB Texte planen und schreiben)
- nehmen zentrale Anregungen für die Überarbeitung auf und setzen sich dazu jeweils ein konkretes Überarbeitungsziel.
(LB Schreiben, TB 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 Sprachgebrauch und Sprache untersuchen und reflektieren, TB 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 Sprachgebrauch und Sprache untersuchen und reflektieren, TB Sprachliche Verständigung untersuchen)

Heimat- und Sachunterricht

Die Gesellschaft für Didaktik des Sachunterrichts (2013, S. 63f.) 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 Zeit und Wandel, TB Dauer und Wandel)
- erklären anhand von Beispielen, welche Prinzipien bei einfachen technischen Erfindungen zu einer Arbeitserleichterung führen und inwiefern sie Kulturleistungen möglich machen.
(LB Technik und Kultur, TB 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.

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 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 Gestaltungselemente und Gestaltungsprinzipien)

Die Schülerinnen und Schüler

- handhaben Informatiksysteme zielorientiert, durchdringen ihre Funktionsweisen sowie grundlegenden Prinzipien und setzen sie zur Bewältigung neuer Herausforderungen ein.
  (Basiskompetenzen)
- strukturieren, modellieren und bereiten Daten und Informationen auf.
  (Suchen und Verarbeiten)
- verwenden analoge und digitale Werkzeuge zur effektiven Gestaltung kollaborativer und individueller Lernprozesse und teilen Resultate mit anderen.
  (Kommunizieren und Kooperieren)
- setzen Werkzeuge zur Realisierung verschiedener Medienprodukte ein.
  (Produzieren und Präsentieren)
- analysieren und bewerten Gestaltungsmittel, Strukturen und Wirkungsweisen von Informatiksystemen.
  (Analysieren und Reflektieren)

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:

5. Die Unterrichtsequenz aus Sicht der Lehrkräfte

Das folgende Kapitel beschreibt die einzelnen Elemente der Unterrichtssequenz, die von den Lehrkräfte im Rahmen des Projekts erprobt wurden. Anpassungen und Erweiterungen, die von den Lehrkräften am Konzept der TUM vorgenommen wurden, werden dabei mit einbezogen. Die Schilderungen und Beispiele verstehen sich als Anregungen, nicht als strikte Vorgaben.

Zur Strukturierung der Unterrichtssequenz werden die Kompetenzformulierungen für den Inhaltsbereich Algorithmen der Gesellschaft für Informatik verwendet, die für den Primarbereich formuliert werden (GI 2019, S. 13). Zu beachten ist, dass es sich dabei aus wissenschaftlicher Sicht nicht um empirisch fundierte Kompetenzstufen handelt, sondern vielmehr um Empfehlungen zur Strukturierung der Lehr-Lern-Prozesse (ebd, S. 8). Darüber hinaus kann man durch den beschränkten zeitlichen Rahmen der Unterrichtssequenz nicht von einem Kompetenzerwerb im eigentlichen Sinne ausgehen. Im Vordergrund steht vielmehr, das Interesse der Kinder für das Programmieren zu wecken sowie ihnen mögliche Talente und Interessen in diesem Bereich bewusst zu machen.

5.1 Was ist ein Algorithmus?

Die Schülerinnen und Schüler

- führen Algorithmen in ihrer Lebenswelt aus.
- beschreiben Algorithmen alltagssprachlich.

Da die meisten Grundschüler wenig Vorkenntnisse im Bereich der Programmierung mitbringen, besteht der erste Schritt darin, ihnen eine grundlegende Idee zu vermitteln, wie ein Algorithmus und ein Computerprogramm funktionieren. Sie arbeiten zunächst *unplugged*, das heißt ohne den Computer, und programmieren in Alltagssprache.

Die Schülerinnen und Schüler programmieren zunächst die Lehrkraft mittels mündlicher Anweisungen. Die Lehrkraft spielt einen Roboter, der kleine Aufgaben im Klassen-zimmer erfüllen soll, zum Beispiel das Fenster öffnen. Da dieser sich bei zu allgemeinen Anweisungen nicht rührt und nur präzise Befehle befolgt, wird schnell deutlich, dass jeder Schritt in einem Algorithmus verständlich, genau und eindeutig formuliert werden muss. Größere Vorgänge müssen dabei in Teilschritte zerlegt werden, die der Roboter nacheinander ausführt.

Praxisbeispiel:
Schriftliche Befehle für den Roboter

Die Anweisungen für den Roboter können nicht nur mündlich erfolgen, sondern auch in Kleingruppen aufgeschrieben werden (Abb. 6). Hier kann man den Schülerinnen und Schülern Programmieraufträge vorgeben (Abb. 5) oder sie ihre eigenen Ideen beschreiben lassen.

Die Anweisungen werden im Anschluss jeweils von einem Kind einer anderen Kleingruppe ausgeführt. Gemeinsam könenn die Kinder beurteilen, ob die Befehle präzise genug waren.

Abb. 5
Zur Vorgabe werden passende Anweisungen gegeben

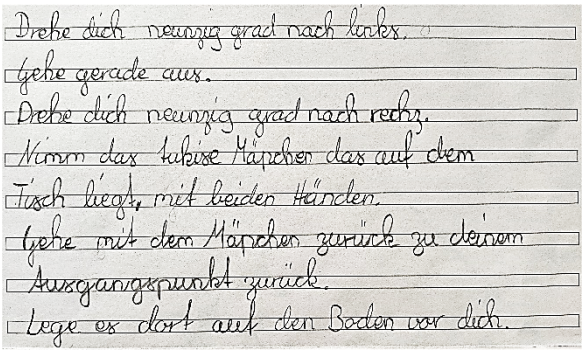


Abb. 6 Anweisungen zum Abholen einer Federtasche

Darüber hinaus wird herausgearbeitet, wo die Schülerinnen und Schüler Algorithmen in ihrem Alltag begegnen, zum Beispiel in Form von Bastelanleitungen oder Rezepten. In weiteren Aufgaben wird geübt, Abläufe in natürlicher Sprache zu beschreiben.

Praxisbeispiel: Vom Bild zur Sprache

Die Schülerinnen und Schüler verfassen zu einer bildlichen Anleitung (Abb. 7) sprachliche Anweisungen (Abb. 8). Dabei können sie die Anleitung auch durch zusätzlich Bilder ergänzen.

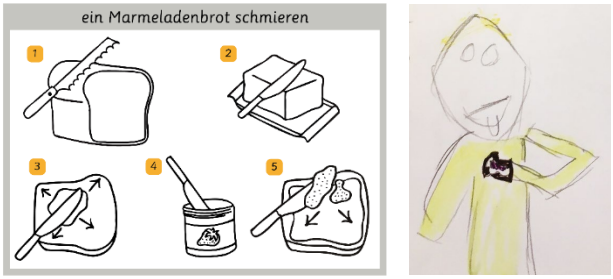


Abb. 7 Bildhafte Anleitung (links) mit Ergänzung (rechts)

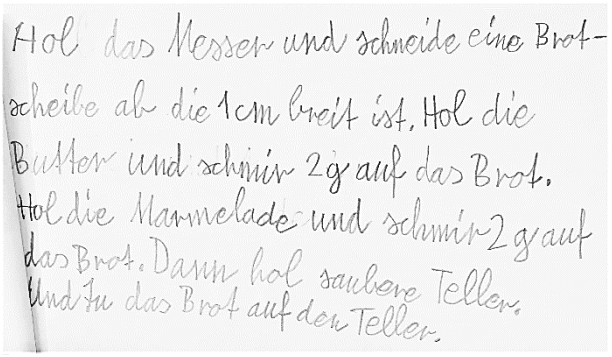


Abb. 8 Anweisungen zum Anrichten eines Marmeladenbrots

Praxisbeispiel: Stille Post einmal anders

Jedes Kind malt oben auf ein leeres Blatt Papier ein einfaches Bild (Abb. 9, links) und gibt es nach rechts an die Nachbarin bzw. den Nachbar weiter. Dieser beschreibt unter dem Bild, was er darauf sieht (Abb. 9, mitte) und knickt das gemalte Bild nach hinten um. Danach wird das Blatt erneut nach rechts weitergegeben, sodass nun jedes Kind die Beschreibung eines Bildes vor sich sieht. Zu dieser Beschreibung wird nun wieder ein Bild gemalt (Abb. 9, rechts).

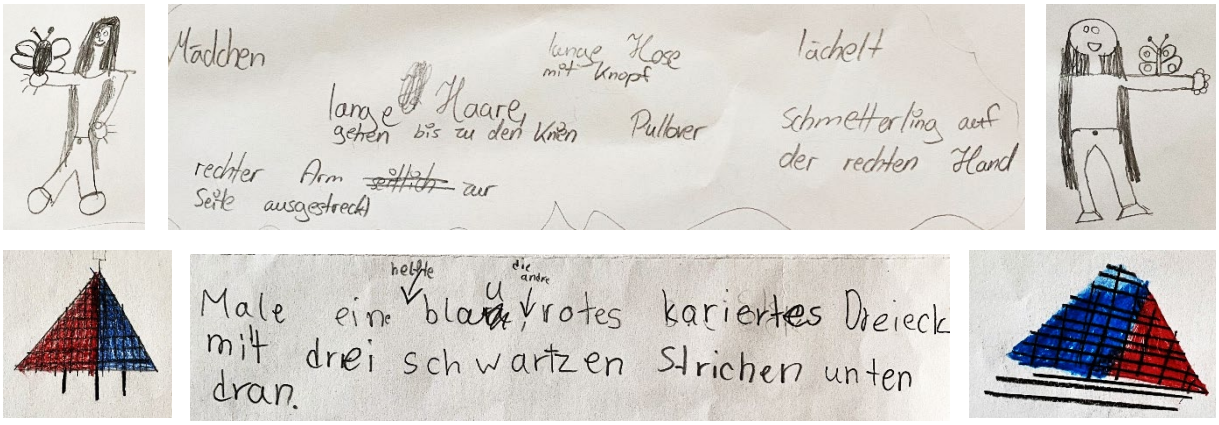


Abb. 9 Ergebnisse der stillen Post

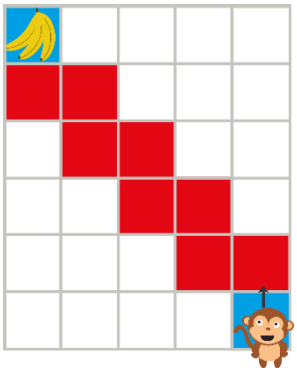
Im Anschluss falten die Schülerinnen und Schüler die Blätter auf und vergleichen, wie ähnlich sich die beiden Bilder sehen oder worin sie sich unterscheiden. Falls die Bilder sehr unterschiedlich aussehen, können sie untersuchen, an was es liegt – wurde nicht genau beschrieben oder nicht genau gelesen?

5.2 Programmieren *unplugged*

Die Schülerinnen und Schüler

- stellen Algorithmen in verschiedenen formalen Darstellungsformen dar.
- entwerfen, realisieren und testen Algorithmen mit den algorithmischen Grundbausteinen Anweisung, Sequenz, Wiederholung und Verzweigung.
- vergleichen Algorithmen unter Verwendung der Fachsprache.

Um das Darstellen von Algorithmen weiter zu vertiefen, programmieren die Schülerinnen und Schüler sich gegenseitig, um verschiedene Aufgaben in einem Parcours zu lösen, zum Beispiel den Weg eines Affen zu seinem Futter (Abb. 10, links). Sobald eine Aufgabe bearbeitet wurde, laufen sie die Lösung im Parcours ab. Falls nötig, verbessern sie die Programme gemeinsam. Im Anschluss können die Schülerinnen und Schüler sich eigene Aufgabenstellungen überlegen (Abb. 10, rechts).



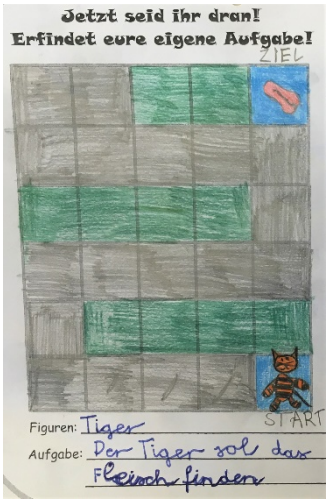


Abb. 10
Wege in einem Parcours beschreiben

Neben der Alltagssprache wird hier mit Symbolen und schließlich mit der blockbasierten Programmiersprache Scratch gearbeitet (Abb. 11). Um die kognitive Belastung für die Schülerinnen und Schüler möglichst gering zu halten, werden die Befehle der Programmiersprache zunächst *unplugged* verwendet: Für die Bearbeitung der Aufgaben wurden haptische Programmierblöcke angefertigt, die optisch den Programmierbefehlen in *Scratch* entsprechen (Abb. 11, rechts). Diese wurden mit Magneten und Klettverschlüssen ausgestattet, sodass die Schülerinnen und Schüler mit ihnen sowohl an der Tafel als auch auf Filzbahnen programmieren können.

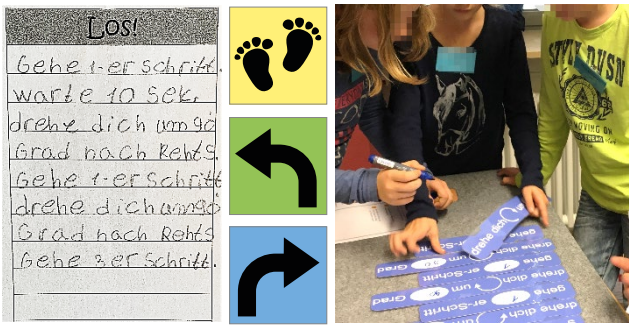


Abb. 11 Verschiedene Wege um einen Algorithmus darzustellen

Praxisbeispiel:
Testen von Lösungen

Um die Lösungen der Aufgaben zu testen, führen die Kinder die Anweisungen am Stück aus. Um die Orientierung nicht zu verlieren, können die Schülerinnen und Schüler die Wege dafür selbst ablaufen.

Den Parcours kann man z.B. mit Teppichfliesen legen, mit Malerkrepp abkleben (Abb. 12, oben) oder im Pausenhof mit Kreide aufmalen. Alternativ kann man die Lösungen auf dem Arbeitsblatt mit kleinen Figuren ablaufen, z.B. von Lego oder aus einem Überraschungsei (Abb. 12, unten).

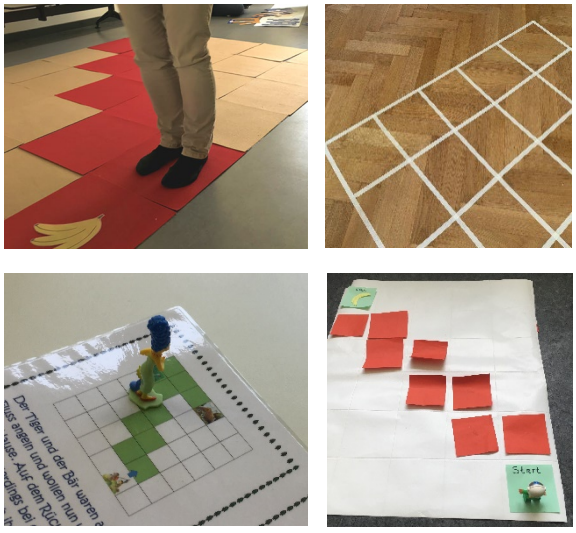


Abb. 12 Möglichkeiten zum Testen der Lösungen

Im Gegensatz zu den Symbolen können die Lernenden mit den Scratch-Befehlen nicht nur Sequenzen (mehrere Anweisungen hintereinander) programmieren, sondern auch Wiederholungen und bedingte Anweisungen.

Abb. 13 Mehrere Schülerlösungen der gleichen Aufgabe

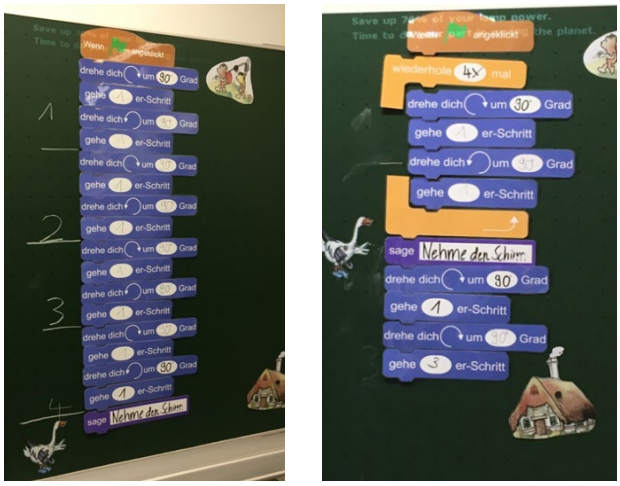


Abb. 14 Eine Sequenz (links) wird durch das Verwenden einer Wiederholung verkürzt (rechts)

Die Aufgaben wurden so gestaltet, dass es nicht nur einen richtigen Lösungsweg gibt bzw. man den gleichen Weg auf verschiedene Arten, programmieren kann, z.B. als Sequenz oder mit einer Wiederholung. Auf diesem Weg kann die Lehrkraft thematisieren, welche Vor- oder Nachteile die einzelnen Lösungsansätze mit sich bringen (Abb. 13) oder gezielt ein neues algorithmisches Strukturelement vorstellen (Abb. 14).

5.3 Programmieren am Computer

Die Schülerinnen und Schüler

- entwerfen, realisieren und testen Algorithmen mit den algorithmischen Grundbausteinen Anweisung, Sequenz, Wiederholung und Verzweigung.
- programmieren ein Informatiksystem.

Nach den *Unplugged*-Übungen programmieren die Kinder am Computer in *Scratch*. Man kann die kostenfreie Programmierumgebung entweder im Webbrowser verwenden oder herunterladen und lokal installieren. Letzteres bringt den Vorteil, dass zur Verwendung keine Internetverbindung notwendig ist. Scratch bietet eine Vielzahl von Figuren und Hintergründen, die programmiert werden können. Im Vergleich zum Programmieren *unplugged* stehen zudem weitaus mehr Programmierbefehle zur Verfügung.

Praxisbeispiel: 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 (Abb. 15), kann man das genaue Lesen und nachverfolgen von Code üben.

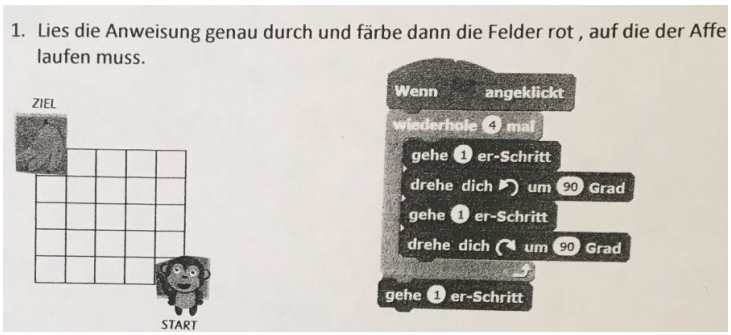


Abb. 15 Aufgabe für deren Lösung Code gelesen werden muss

Damit die Schülerinnen und Schüler sich ganz auf die Bedienung der Programmierumgebung konzentrieren können, bearbeiten sie darin zunächst einige Aufgaben, die bereits *unplugged* gelöst wurden (Abb. 16). In dieser Explorationsphase arbeiten die Schülerinnen und Schüler in Paaren.

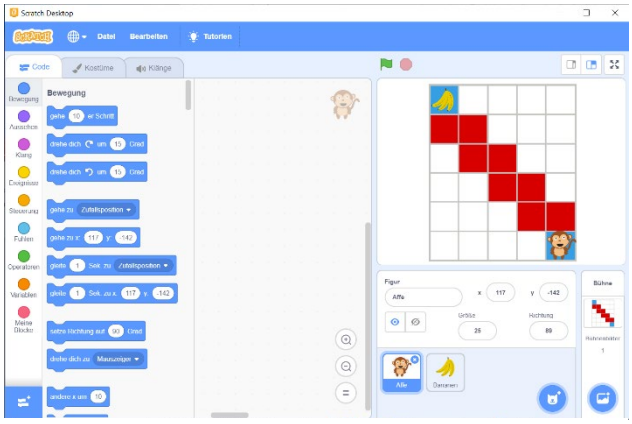


Abb. 16 Parcours-Aufgabe in *Scratch*

Im Anschluss bearbeiten die Schülerinnen und Schüler einen Lernzirkel mit zunehmend schwierigen Stationen, in dem die Grundfunktionen von *Scratch* nacheinander thematisiert werden. Ausgehend von Fragen der Bedienung führen die Zirkelkarten (Abb. 17) 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.

Alternativ bzw. ergänzend kann man den Schülerinnen und Schülern einzelne Programmierbefehle vorgeben, deren Funktion sie erkunden. Bewährt hat sich außerdem, in jeder Stunde kleine Programmieraufträge zu vergeben, z. B. dass sich eine Figur von Punkt A nach Punkt B bewegen soll. Eine Möglichkeit, das Vorgehen beim Lösen einer Programmieraufgabe gezielt zu demonstrieren, bietet der Ansatz des *Live Coding* (Waite und Grover 2020). Hier artikuliert die Lehrkraft die Überlegungen und Entscheidungen, die sie beim Bearbeiten der Aufgabe führt und trifft. Darüber hinaus können die Schülerinnen und Schüler auch in den Entwicklungsprozess einbezogen werden. Der Bildschirm der Lehrkraft sollte dafür von allen Lernenden einsehbar sein, z. B. über den Beamer. Das *Live Coding* ist keineswegs ungeplant – die Lehrkraft verfügt im Idealfall über eine genaue Vorstellung, welche Konzepte, Fehlerquellen, Fehlvorstellungen sie dabei thematisieren möchte.

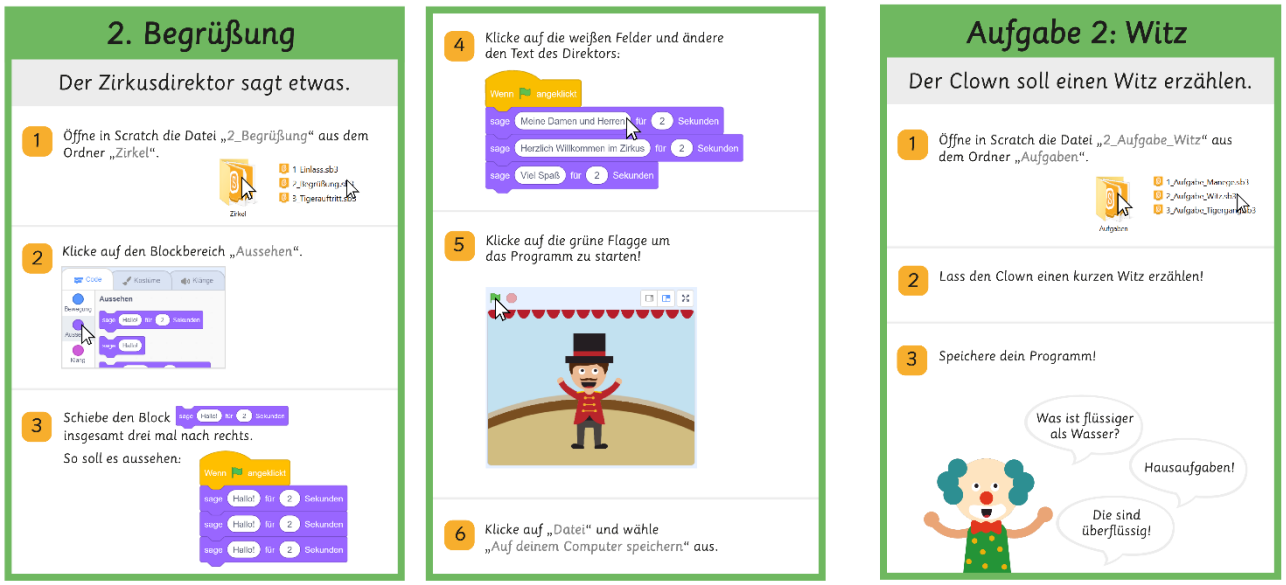


Abb. 17 Station des Scratch-Zirkels

Praxisbeispiel: 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 (Abb. 18). Zudem gibt die Lehrkraft Hilfestellungen bei konkreten Programmieraufgaben. Beispielsweise kann sie den Lösungsweg einer Aufgabe vor der Implementierung in *Scratch* „unplugged" veranschaulichen (Abb. 19, rechts).

Abb. 18 Blockkategorien in *Scratch*

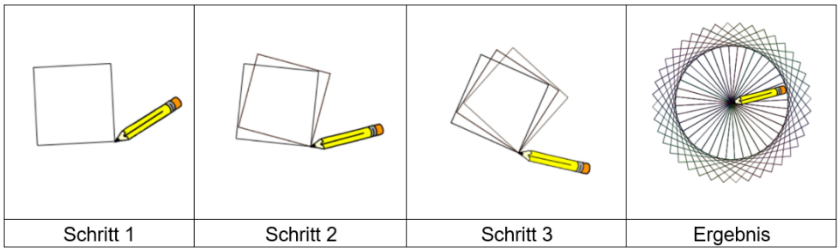


Abb. 19 Aus mehreren Quadraten formt sich ein Ornament (links) - der Prozess wird an einem Modell (rechts) verdeutlicht

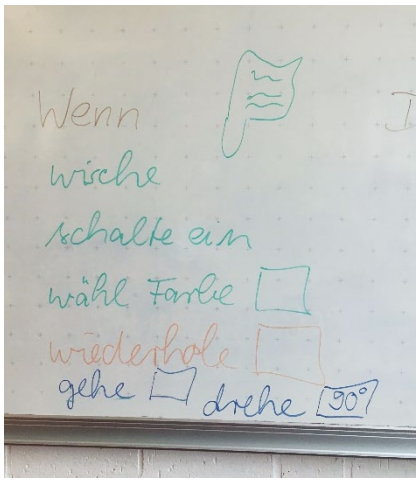


Abb. 20 Skizze einer Lösung an der Tafel

Eine andere Möglichkeit besteht darin, die Lösung einer Aufgabe im Plenum zu erarbeiten, bevor die Schülerinnen und Schüler sie in *Scratch* umsetzen, zum Beispiel durch das Skizzieren des Programms an der Tafel (Abb. 20). Darüber hinaus kann die Lehrkraft beim Auffinden von Fehlern im Programm helfen, in dem 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.

Besonders bei komplexeren Programmieraufgaben ist es hilfreich, den Problemlöseprozess der Schülerinnen und Schüler anzuleiten, z.B. durch Symbole die im Raum hängen und auf die sie jederzeit zurückgreifen können (Abb. 21).

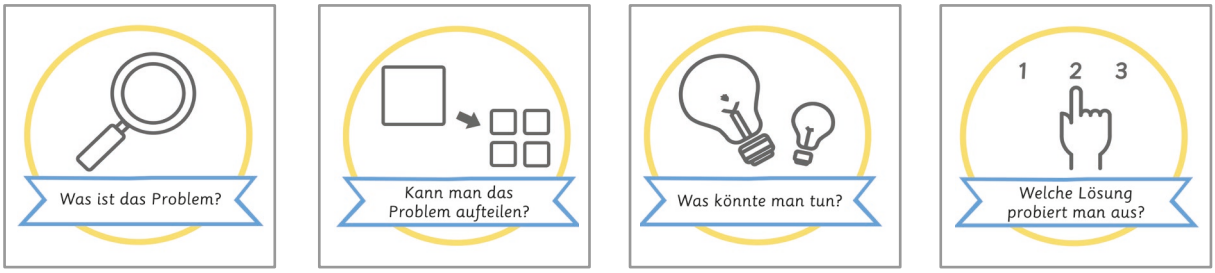


Abb. 21 Einzelne Schritte beim Lösen eines Problems

5.4 Programme planen

Die Schülerinnen und Schüler

- stellen Algorithmen in verschiedenen formalen Darstellungsformen dar.
- entwerfen, realisieren und testen Algorithmen mit den algorithmischen Grundbausteinen Anweisung, Sequenz, Wiederholung und Verzweigung.
- programmieren ein Informatiksystem.

Der letzte Schritt in der Unterrichtssequenz führt die Schülerinnen und Schüler hin zur Planung und Umsetzung eigener Programmideen. In Einzel- oder Partnerarbeit überlegt sich jedes Kind zum Beispiel eine kurze Geschichte, beschreibt diese in einem Drehbuch (Abb. 22, Abb. 23) und setzt sie schließlich in *Scratch* um.

Während man in den weiterführenden Schulen für die Planung von Algorithmen beispielsweise auf Ablaufdiagramme oder Struktogramme zurückgreift, eignen sich für die Grundschule andere Vorgehensweisen. 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 das Programmieren in *Scratch* empfiehlt es sich zudem, dass die Kinder im Vorfeld entscheiden, welche Akteure im Programm eine Rolle spielen sollen. Auf diesem Weg können sie später schneller mit dem Programmieren beginnen ohne zu viel Zeit auf die Auswahl der Figuren zu verwenden.

Für den Planungsprozess kann die Lehrkraft den Kindern eine Vorlage zur Verfügung stellen. Diese kann so gestaltet werden, dass sie auf sämtliche Projekte in *Scratch* angewandt werden kann (Abb. 22, Abb. 23), oder bereits einen Kontext für das Programm vorgibt (Abb. 24).

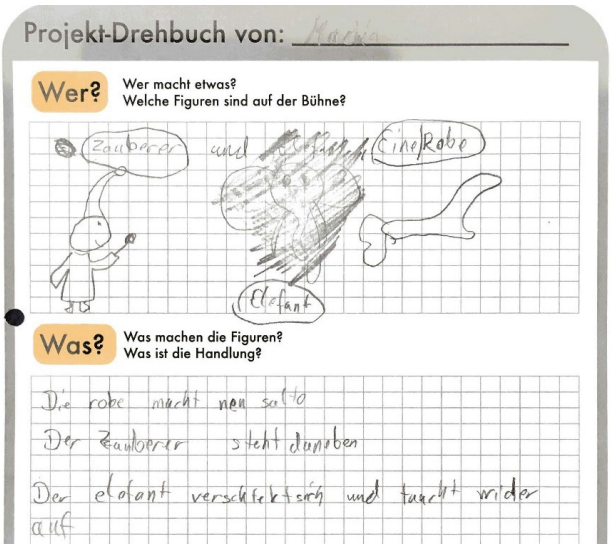


Abb. 22 Ausgefülltes Projekt-Drehbuch

Abb. 23 Vorlage für die Planung eines Programms

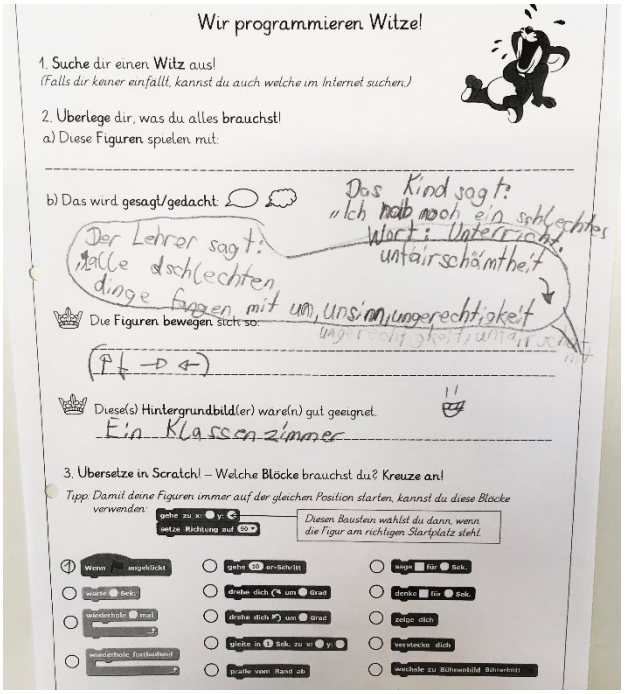


Abb. 24 Planungsvorlage für die Animation eines Witzes

Um die Schülerinnen und Schüler trotz einer offenen Aufgabenstellung nicht zu überfordern, kann man ihnen verschiedene Anforderungen vorgeben, die von den Programmen erfüllt werden sollen. Diese können sich einerseits auf den Inhalt der Programme beziehen – zum Beispiel kann ein Gespräch zwischen zwei Figuren, ein Witz oder eine Aufführung in einem Zirkus programmiert werden. Andererseits kann die Lehrkraft die zu verwendenden Programmstrukturen vorgeben, zum Beispiel eine Wiederholung oder verschiedene Startereignisse. Bei dieser Variante besteht allerdings die Gefahr, dass die Schülerinnen und Schüler die Strukturen nur einbauen, um die Vorgabe zu erfüllen, obwohl dies überhaupt nicht sinnvoll ist.

Am Ende der Unterrichtssequenz stellen die Schülerinnen und Schüler die eigenen Projekte vor und bekommen Gelegenheit, ihr Vorgehen sowie mögliche Schwierigkeiten zu kommentieren. Hier ist es optimal, wenn sie ihre Ergebnisse der Klasse über den Beamer zeigen können.

Praxisbeispiel: 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 (Martin 2017).

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 (Abb. 25).

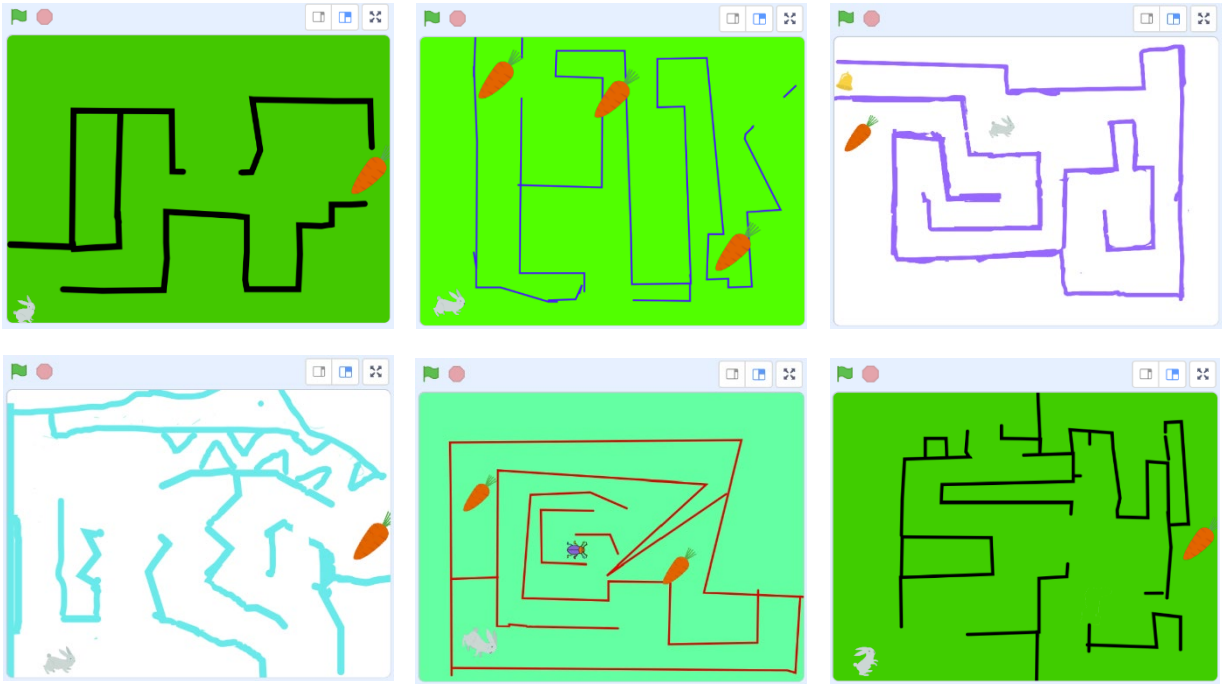


Abb. 25 Verschiedene Umsetzungen der gleichen Aufgabe

Praxisbeispiel: Arbeiten mit Computern

Unabhängig davon, ob mit den Schülerinnen und Schülern im Computerraum programmiert wird oder mobile Geräte, wie Tablets oder Laptops, zum Einsatz kommen, sollte man mit ihnen im Vorfeld bestimmte Regeln für den Umgang mit Informatiksystemen vereinbaren. Um Verbindlichkeit herzustellen und während des Unterrichts darauf zurückgreifen zu können, kann man diese für alle sichtbar im Raum aufhängen (Abb. 27).

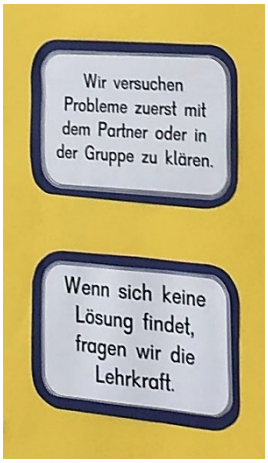


Abb. 26 Vorgehen bei Problemen

Diese Regeln legen einerseits fest, wie die Kinder mit den Geräten umgehen sollen, andererseits thematisieren sie das Verhalten bei möglichen Problemen (Abb. 26). Die Regeln können mit der Klasse weiter ausgestaltet werden, dass zum Beispiel alle Trinkflaschen bei Eintritt in den PC-Raum an einer bestimmten Stelle abgestellt werden und nur dort getrunken werden darf, oder alle Kinder die Hände von der Tastatur nehmen und mitklatschen, wenn die Lehrkraft einen Rhythmus klatscht. Letzteres hilft der Lehrkraft, die Aufmerksamkeit der Schülerinnen und Schüler auf sich zu ziehen, wenn sie z.B. etwas erklären möchte.

Abb. 27 Regeln im PC-Raum

Praxisbeispiel: Ein Projekt mit der ganzen Klasse

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 (Abb. 28).

Abb. 28 Ausschnitt aus dem Projekt der Klasse

5.5 Alternative Systeme

Die vorangegangenen Ausführungen in diesem Kapitel beschreiben die Unterrichtssequenz, die von der TUM im Vorfeld des Projekts entwickelt wurde, einschließlich verschiedener Modifikationen und Erweiterungen durch die Lehrkräfte der Projektschulen. Darüber hinaus ergänzten einzelne Projektschulen die Sequenz durch weitere Systeme, die im Rahmen der Fortbildungen vorgestellt wurden.

BeeBot und BlueBot

Der Roboter *BeeBot* bzw. *BlueBot*[31], der wie eine Biene gestaltet ist, kann mit verschiedene Pfeiltasten programmiert werden (Abb. 29). An den Projektschulen wurde er hauptsächlich als Ergänzung zu Beginn der Unterrichtssequenz eingesetzt oder in der ersten und zweiten Jahrgangsstufe. Die Schülerinnen und Schüler programmierten verschiedene Wege auf speziellen Matten (Abb. 30, links) oder lösten eigens gestaltete Aufgaben.

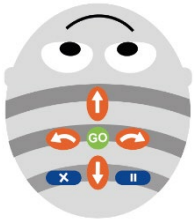


Abb. 29 BlueBot

Dazu gehörte sowohl das Lesen vorgegebener Programme (Abb. 30, Mitte) als auch das Aufstellen eigener Programme zur Lösung eines Problems (Abb. 30, rechts). Die Schülerlösungen wurden im Anschluss mit den Robotern ausgeführt und auf Korrektheit geprüft.

Um die Faszination, die von den Robotern ausgeht, als Motivation für das Lesen auszunutzen, wurde das Format der *Lese-Bot-Geschichte* entwickelt (Abb. 31). Hier müssen die Schülerinnen und Schüler Texte genau lesen, um den richtigen Weg in einem Feld zu finden.

ScratchJr

ScratchJr[32] ist wie *Scratch* eine visuelle Programmiersprache inklusive Programmierumgebung, mit der man die Handlungen verschiedener Figuren programmieren kann. Auch hier muss man den Code nicht eintippen, sondern aus Code-Bausteinen zusammensetzen. Die farbigen Blöcke enthalten jedoch nur Symbole und keinen Text (Abb. 32). *ScratchJr* enthält Blöcke mit denen Sequenzen und Wiederholungen programmiert werden können. Die Auswahl der Blöcke ist gegenüber *Scratch* überschaubar.

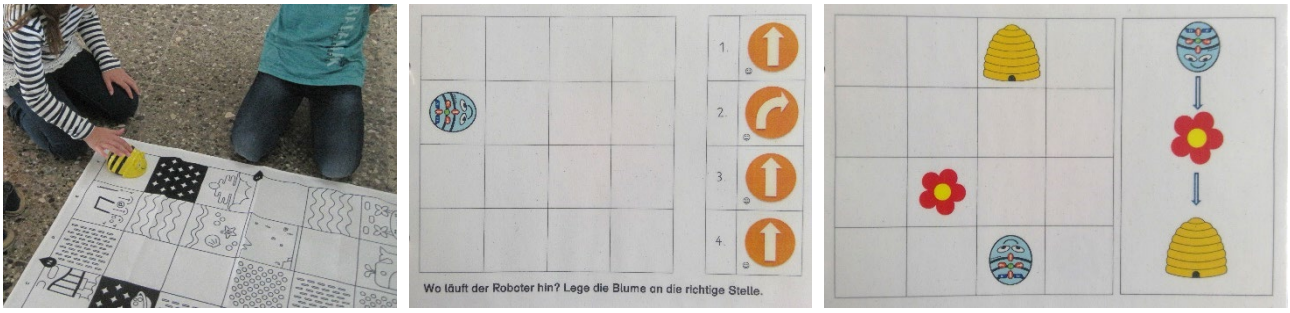


Abb. 30 Einsatzmöglichkeiten für die Bienenroboter

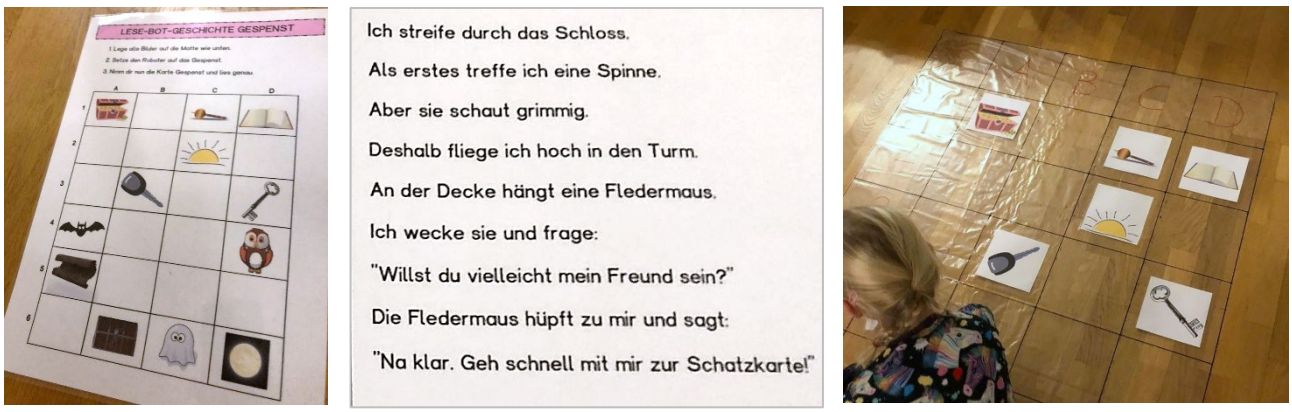


Abb. 31 Beispiel einer Lese-Bot-Geschichte

[31] https://www.b-bot.de/produkte/bee-bot-lernreihe/

[32] https://www.scratchjr.org/

Abb. 32 Die Oberfläche von ScratchJr

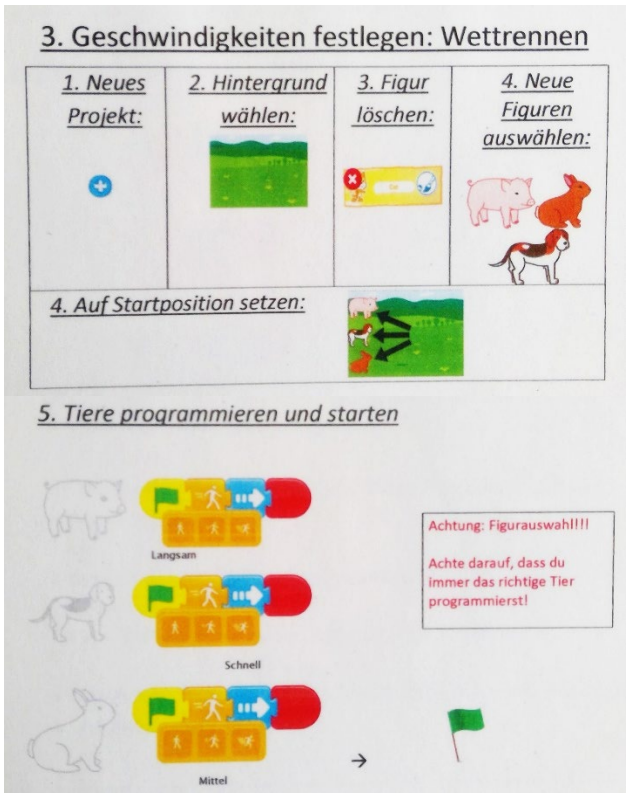


Abb. 33 Anleitung zum Programmieren mit *ScratchJr*

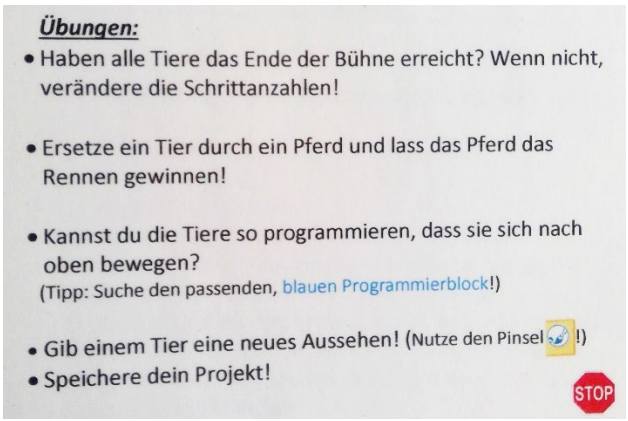


Abb. 34 Aufgaben, die in *ScratchJr* bearbeitet werden

[33] https://makeymakey.com/

Die App *ScratchJr* kann für die Betriebssysteme Android und iOS kostenfrei heruntergeladen werden. An einer Projektschule wurde *ScratchJr* in der Jahrgangsstufe 2 und 3 als Vorarbeit für das Programmieren in *Scratch* eingesetzt. Dafür wurden Materialien entwickelt, mit denen einzelne Programmstrukturen zunächst nachprogrammiert werden (Abb. 33). Wie im *Scratch*-Lernzirkel bearbeiten die Schülerinnen und Schüler dazu passende Aufgaben, in denen sie Programme verändern oder erweitern müssen (Abb. 34).

Makey Makey

Der *Makey Makey*[33] ist eine Platine, mit der man leitfähige Gegenstände in Computertasten oder die linke Maustaste umwandeln kann (Abb. 35). Die Platine wird mit einem USB-Kabel an den Computer angeschlossen. Im Anschluss können ein oder mehrere Objekte, zum Beispiel ein Stück Knetmasse oder eine Zitrone, mit den farbigen Krokodilklemmen an die Platine angeschlossen werden. Verbindet man zum Beispiel die Zitrone mit dem Anschluss für die Leertaste, wird bei jeder Berührung der Zitrone die Leertaste betätigt. Da man in *Scratch* die Computertasten als Startereignisse für Programme verwenden kann, lassen sich mit *Scratch* in Kombination mit dem *Makey Makey* auf Berührungen reagierende Programme erstellen.

In einer Schule wurde der *Makey Makey* als verbindendes Elemente zwischen der AG Programmieren und der AG Technik eingesetzt. In mehreren gemeinsamen Treffen wurde mit den Kindern die Leitfähigkeit von Objekten sowie das Programmieren in *Scratch* thematisiert.

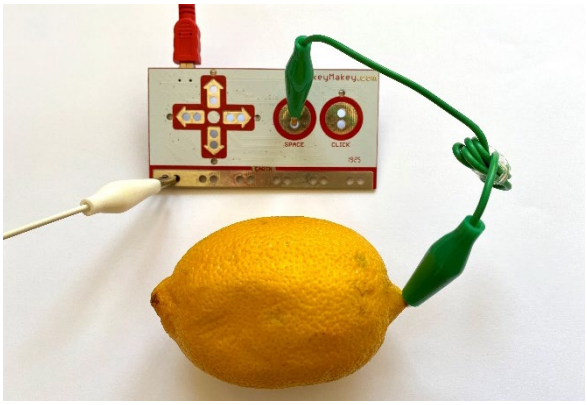


Abb. 35 Makey Makey-Set mit angeschlossener Zitrone

Calliope mini

Der *Calliope mini*[34] ist ein Einplatinencomputer (Abb. 36), der nach dem Vorbild des *BBC micro:bit*[35] entwickelt wurde. Er kann in verschiedenen blockbasierten Programmierumgebungen programmiert werden, z.B. *MakeCode*[36] oder *NEPO*[37]. Dafür schreiben die Schülerinnen und Schüler zunächst ein Programm, laden es herunter und übertragen es per USB-Kabel auf den *Calliope mini* (Abb. 37).

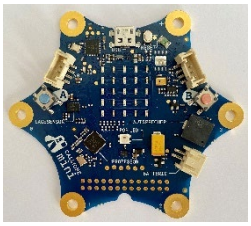


Abb. 36 Calliope mini

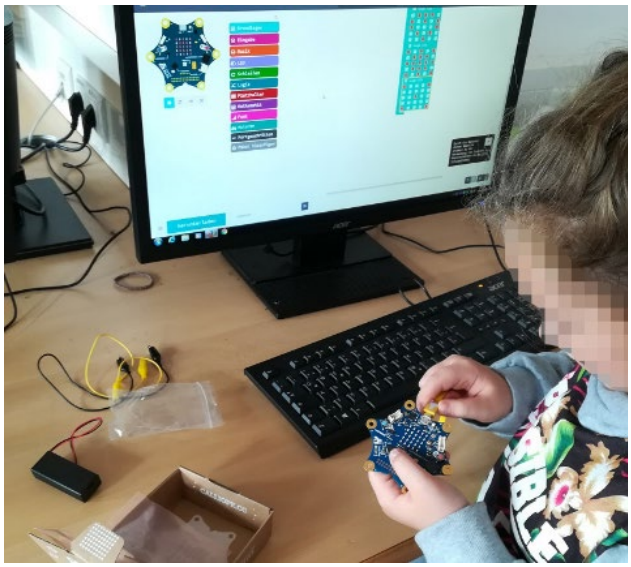


Abb. 37 Eine Schülerin schreibt ein Programm in *MakeCode* und überträgt es auf den *Calliope mini*

Der *Calliope mini* verfügt über einen Lage-, Bewegungs- und Helligkeitssensor sowie über Eingabe- und Ausgabeanschlüsse, zum Beispiel zwei Knöpfe (Eingabe) und ein Display aus 25 LEDs (Ausgabe). Er eignet sich daher gut zur Verdeutlichung des EVA-Prinzips.

Eine Erprobungsschule arbeitete bereits vor dem Projekt intensiv mit dem *Calliope mini* in der dritten und vierten Jahrgangsstufe und setzte ihn als Ergänzung zum Unterrichtskonzept der TU München ein. Im Vordergrund stand dabei, dass sich die Schülerinnen und Schüler ihre Umwelt mit selbst programmierten Messgeräten erschließen. Zum Beispiel wurde ein automatisches Licht für einen Tretroller programmiert, das sich einschaltet, sobald die Lichtstärke unter einen bestimmten Wert fällt (Abb. 38).

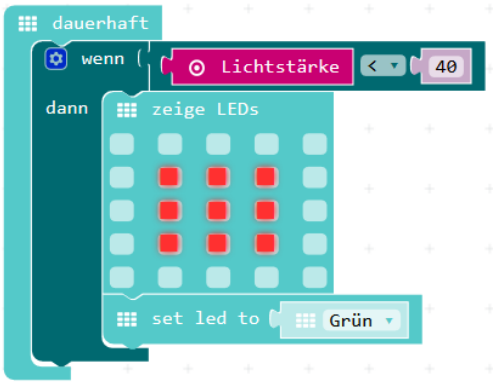


Abb. 38 Programm für den *Calliope mini* in *MakeCode*

[34] https://calliope.cc/
[35] https://microbit.org/
[36] https://makecode.calliope.cc/
[37] https://lab.open-roberta.org/

6. Entwurf der Lehrerfortbildung

Laut der Meta-Studie von Hattie (2010) üben Lehrkräfte den größten Einfluss auf die schulische Leistung der Lernenden aus. Sie schaffen im Unterricht nicht nur Anreize, in dem sie Lernformen, Materialien und Methoden auswählen und gemäß der jeweilige Klassensituation anpassen, sie begleiten und unterstützen die Lernenden in der aktiven geistigen Auseinandersetzung mit den Inhalten (StMBKWK 2014, S. 24).

Da die meisten Lehrkräfte der Grundschule keine oder nur wenige Vorkenntnisse in der Informatik mitbringen, sollten sie im Rahmen des Projekts *AlgoKids – Algorithmen für Kinder* die Chance erhalten, sich mit fachlichen und fachfachdidaktischen Inhalten auseinanderzusetzen und so ihr Professionswissen und Handeln weiterzuentwickeln. Dafür wurden drei mehrtägige Lehrerfortbildungen durchgeführt, die sich über die Projektlaufzeit verteilten (vgl. Abb. 3):

- **LFB 1:** 7.-9. Mai 2018
  6.-8. Juni 2018
- **LFB 2:** 12.-14. Dezember 2018
  17.-19. Dezember 2018
- **LFB 3:** 21.-22. November 2019

Der Schwerpunkt der ersten Fortbildung lag auf dem Kennenlernen der Unterrichtssequenz der TUMDDI sowie deren fachlicher Grundlagen. In der zweiten LFB bekamen die Lehrkräfte die Möglichkeit, sich über erste Erfahrungen in der Umsetzung der Unterrichtssequenz auszutauschen. Ferner setzten sie sich mit fachdidaktischen Inhalten auseinander. Diese wurden in der dritten Fortbildung weiter vertieft. Da die dritte LFB zum Projektende stattfand, wurde den Lehrkräften abschließend die Gelegenheit gegeben, Feedback und Anregungen zum Projekt zu äußern.

In diesem Kapitel wird zunächst ein Auszug der theoretischen Grundlagen für die Entwicklung der Fortbildungsveranstaltungen dargestellt. Danach wird der Ablauf sowie die Inhalte der einzelnen Lehrerfortbildungen beschrieben.

6.1 Design-Prinzipien

Wie bereits bei der Entwicklung der Unterrichtssequenz wurde für die Entwicklung des Fortbildungskonzepts der Ansatz der *Design-Based Research* gewählt (vgl. Abschnitt 4.2). Es folgt ein Auszug der theoretisch fundierten Design-Prinzipien, die für die Konzeption der Fortbildungen erarbeitet wurden.

Rahmenbedingungen

Um eine vertiefte Auseinandersetzung mit fachwissenschaftlichen und fachdidaktischen Inhalten zu ermöglichen, sollten sich Fortbildungsveranstaltungen über mehrere Tage erstrecken (Richter et al. 2020). Darüber hinaus sehen Garet et al. (2001) die Teilnahme mehrerer Lehrkräfte einer Schule als Merkmal einer wirksamen Lehrerfortbildung. Dies erhöhe einerseits die Wahrscheinlichkeit, dass die Fortbildungsinhalte nachhaltig im Unterricht verankert werden, und ermögliche andererseits einen gegenseitigen Austausch, der sich positiv auf das Verständnis der Lehrkräfte auswirken kann.

Conceptual Change nachvollziehen

In den Naturwissenschaften ist es üblich, dass Lehrkräfte die Möglichkeit bekommen, den *Conceptual Change* der Lernenden – die Veränderung der ursprünglichen Vorstellungen – selbst zu erleben. Auf diese Weise können sie dessen Bedeutung besser einschätzen und im Unterricht gezielt evozieren (Webb 1992, S. 63f.). Dazu kommt, dass Lehrerinnen und Lehrer Fortbildungen als zufriedenstellend erleben, wenn sie sich auf den konkreten Unterrichtsalltag beziehen (Lipowsky 2010).

Auswahl der Fachinhalte

Die Kenntnis einer Programmiersprache ist zwar eine notwendige, aber bei weitem nicht ausreichende Voraussetzung für das Programmierenlernen (Caspersen 2018). Vielmehr sollte der Fokus dabei auf den algorithmischen

Grundstrukturen liegen, die in fast jeder Programmiersprache zu finden sind sowie den Aspekten des Computational Thinking (Bell 2016; Bell und Duncan 2018).

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

Verschränkung von Fachwissenschaft und Fachdidaktik

Um Programmieren zu unterrichten brauchen Lehrkräfte nicht nur fundierte Fachkenntnisse, sondern auch Kenntnisse in der Didaktik der Informatik (Hill et al. 2008). Auch wenn noch nicht abschließend geklärt ist, was in der Informatik alles zu diesen fachdidaktischen Kompetenzen gehört und wie Lehrkräfte diese entwickeln, sollte man ihnen Kenntnisse über die Bedürfnisse der Lernenden, didaktische Ansätze und Methoden vermitteln (vgl. Hubwieser et al. 2013, Bender et al. 2015).

Eine Variante, Fachwissenschaft und Fachdidaktik in Fortbildungen zu verknüpfen, ist der Ansatz des situierten Lernens (Lave und Wenger 1991). Dabei setzen sich Lehrkräfte in authentischen Lernsituationen mit fachlichen Inhalten auseinander, bevor in einem nachgeschalteten Theorie-Input die didaktischen Hintergründe vorgestellt werden (Gebauer 2019).

Anregung von Reflexionsprozessen

Neben dem Aneignen von neuem Wissen, sollte für die Lehrkräfte in Fortbildungen die Möglichkeit bestehen, sich mit anderen Teilnehmenden über die Erprobung der neuen Inhalte auszutauschen und sich gegenseitig zu beraten (Lipowsky und Rzejak 2012). Dieser Austausch ermöglicht es ihnen einerseits, Einblicke in die Unterrichtspraxis anderer Lehrkräfte zu gewinnen und möglicherweise Anregungen für den eigenen Unterricht zu erhalten. Andererseits können sie Feedback zu ihren eigenen Unterrichtsideen und -praktiken erhalten und diese gegebenenfalls modifizieren (Aldorf 2016, S. 79).

Anknüpfen an Bekanntes

Insbesondere wenn Informatik nicht Teil des Lehrplans der Grundschule ist, sollte man den Lehrkräften 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 (Reding und Dorn 2017). 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).

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.

Fortlaufende Unterstützung

Die Forschung zeigt, dass Coaching-Angebote und andauernde Unterstützungsmaßnahmen das Implementieren neuer Lehrpläne, Werkzeuge oder Praktiken erheblich begünstigen können (Penuel et al. 2011; Roth et al. 2011; Sentance et al. 2013). Ein Coaching-Angebot erhöht darüber hinaus die Wahrscheinlichkeit, dass Lehrkräfte neue Inhalte oder Methoden angemessen implementieren (Devine et al. 2013). Im Idealfall sollte man innerhalb der Fortbildungen Zeit für das Besprechen möglicher Anliegen der Lehrkräfte einplanen. Voraussetzung dafür ist, dass sich die Fortbildungsmaßnahmen über einen längeren Zeitraum erstrecken.

6.2 Lehrerfortbildung 1

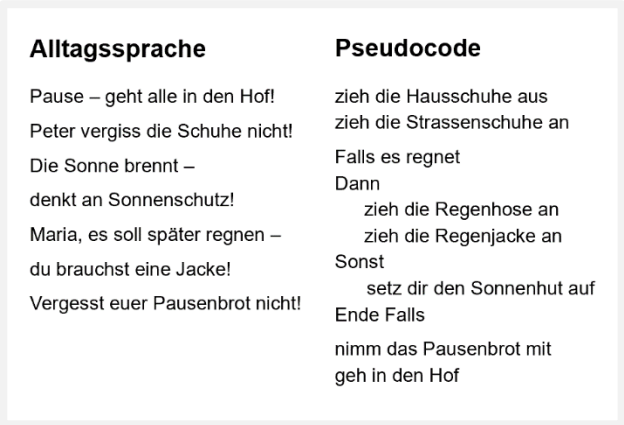
Die erste LFB wurde am Ende des Schuljahres 2017/18 an der ALP durchgeführt. Die Lehrkräfte hatten darin den ersten direkten Kontakt zur TUMDDI. Nach der Fortbildung starteten die ersten Unterrichtsversuche zum Programmieren an den Projektschulen.

Dem Erkenntnisgewinn der Schülerinnen und Schüler folgen

Zu Beginn der Fortbildung setzten sich die Lehrkräfte umfangreich mit der Unterrichtssequenz der TUM zum Programmieren in der Grundschule auseinander. Um ihnen den Perspektivwechsel von der Sichtweise des Lehrenden zu der des Lernenden zu ermöglichen, bearbeiteten sie ausgewählte Aufgaben aus der Unterrichtssequenz. Alle Übungen wurden gemeinsam besprochen. Die Reflexion der Lehrkräfte über den eigenen Lernprozess wurde zudem angereichert mit Schülerergebnissen und Videosequenzen, die bei der Erprobung der Unterrichtssequenz im Vorfeld des Projekts entstanden sind. Dies sollte den Lehrkräften helfen, die Lernprozesse auf Seiten der Schülerinnen und Schüler besser einzuschätzen und im eigenen Unterricht fördern zu können.

Sicherheit im Umgang mit den algorithmischen Grundstrukturen

Der zweite Fortbildungstag begann mit einem fachlichen Überblick zum Programmieren in der Schule. Den Lehrkräften wurde aufgezeigt, dass das Programmieren mehr ist als das bloße Erlernen einer Programmiersprache und der Fokus deshalb immer auf den der Programmierung zugrundeliegenden algorithmischen Strukturen liegen sollte: *Anweisung*, *Sequenz*, *Wiederholung* und *bedingte Anweisung*[38]. Nach einer kurzen Erläuterung des Begriffs Algorithmus wurden die einzelnen algorithmischen Grundstrukturen nacheinander behandelt. Jede Struktur wurde zunächst in Pseudocode[39] (Abb. 39) sowie Blöcken der Programmiersprache Scratch beschrieben (Tabelle 2) und danach in verschiedenen Programmieraufgaben vertieft. Zusätzlich zu den algorithmischen Grundstrukturen wurde das Variablenkonzept behandelt.



Alltagssprache	Pseudocode
Pause – geht alle in den Hof! Peter vergiss die Schuhe nicht! Die Sonne brennt – denkt an Sonnenschutz! Maria, es soll später regnen – du brauchst eine Jacke! Vergesst euer Pausenbrot nicht!	zieh die Hausschuhe aus zieh die Strassenschuhe an Falls es regnet Dann zieh die Regenhose an zieh die Regenjacke an Sonst setz dir den Sonnenhut auf Ende Falls nimm das Pausenbrot mit geh in den Hof

Abb. 39 Anweisungen in Alltagssprache und Pseudocode

Tabelle 2 Anweisung in Alltagssprache, Pseudocode und Scratch

Alltags-sprache	Pseudocode	Scratch
Zeichne ein Quadrat	Wiederhole 4 Mal Zeichne eine Linie Drehe Stift um 90° Ende Wiederhole	

Der theoretische Input wurde dabei möglichst kurzgehalten – der Schwerpunkt lag auf dem eigenständigen Arbeiten an den Aufgaben. Diese konnten die Lehrkräfte je nach Präferenz allein oder paarweise bearbeiten. Unterstützt wurden sie dabei durch Impulse und Hilfestellungen der Kursleiter. Nach jeder Übungsphase wurden die Probleme, die bei den Lehrkräften aufgetreten waren und die gemäß der Erfahrungen der Kursleiter häufig bei Schülerinnen und Schülern auftreten, im Plenum besprochen. Um Fehlvorstellungen zum Ablauf einzelner algorithmischer Strukturen zu vermeiden, wurden Codebeispiele in Scratch gemeinsam gelesen und nachvollzogen (Abb. 40).

[38] Es gibt weitere algorithmische Strukturen, die für die Programmierung relevant sind. Da sie während der Unterrichtssequenz der TUM nicht thematisiert werden, werden sie an dieser Stelle nicht erwähnt.

[39] Mit Pseudocode kann man Algorithmen darstellen. Er ist bereits ähnlich wie der Code eines Computerprogramms aufgebaut, benutzt jedoch nur natürliche Sprache.

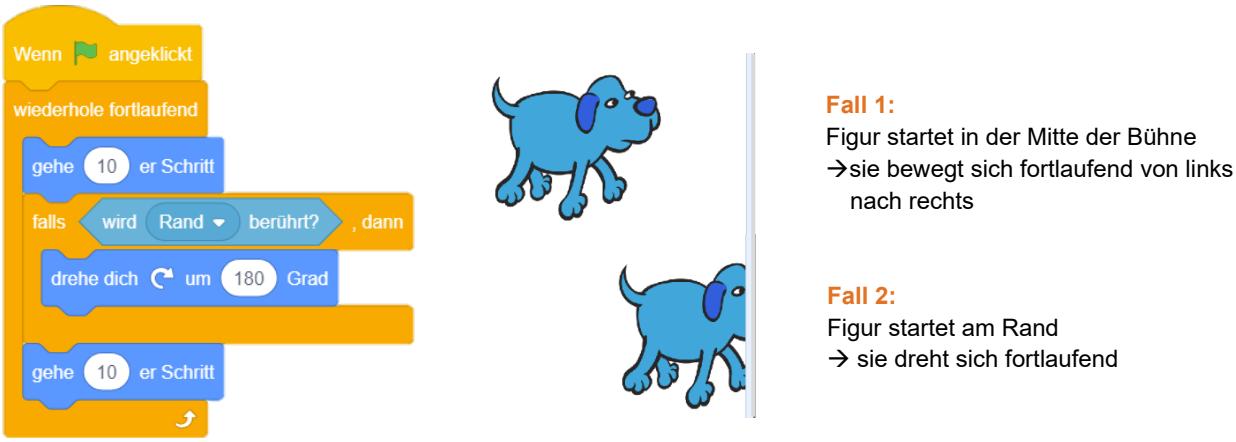


Abb. 40 Beispiel für eine Aufgabe in der Code interpretiert werden muss: „Warum dreht sich der Hund in Fall 2 permanent?“

Implementierung an den Projektschulen

Am Ende der ersten Fortbildungshalbwoche wurde thematisiert, wie sich die Lehrkräfte vorstellen können, die Themen Algorithmen und Programmieren in ihrem eigenen Unterricht umzusetzen.

Um den Austausch zwischen den Lehrkräften zu fördern, wurden Posterwände im Raum verteilt, auf denen Ideen, Inspirationen und Gedanken zu verschiedenen Themen gesammelt wurden, z.B. mögliche kurz-, mittel- und längerfristige Umsetzungen sowie Herausforderungen (Abb. 41). Die gesammelten Ergebnisse wurden im Anschluss im Plenum diskutiert. Weiterhin wurde der weitere Verlauf des Projekts besprochen.

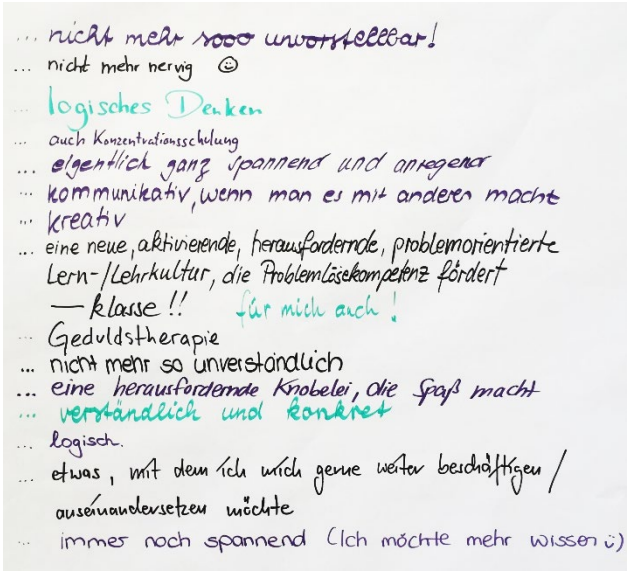


Abb. 41 Ausschnitt der Posterwand "Programmieren ist jetzt für mich..."

6.3 Lehrerfortbildung 2

Die zweite Fortbildung wurde ein halbes Jahr später im Dezember 2019 an der ALP durchgeführt. Im Vorfeld wurde mit den Lehrkräften vereinbart, dass sie bis dahin erste Unterrichtsversuche durchführen sollten.

Erfahrungsaustausch

Zu Beginn der Fortbildung wurde viel Zeit eingeplant, um den Lehrkräften eine umfangreiche Reflexion zu ermöglichen. Auf verschiedenen Plakatwänden zu den einzelnen Elementen der Unterrichtssequenz sammelten 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 (Abb. 42). Darüber hinaus wurden auf allen Plakaten die passenden Kompetenzformulierungen aus den Empfehlungen der Gesellschaft für Informatik und dem Lehrplan 21 der Schweiz aufgelistet. Mit Klebepunkten markierten die Lehrkräfte, welche Formulierungen ihnen am sinnvollsten erschienen. Auf einem weiteren Plakat notierten sie, welche Vorarbeiten sie als sinnvoll oder notwendig ansehen und was man weiterführend im Unterricht behandeln könnte. Die Ergebnisse sowie Best-Practice-Beispiele aus den Schulen wurden im Anschluss im Plenum besprochen.

Abb. 42 Schema einer Plakatwand zum Erfahrungsaustausch (oben) und Ergebnis (unten)

Aufgabenformate in Scratch

Um die Programmierkenntnisse der teilnehmenden Lehrkräfte weiter zu festigen, erhielten sie erneut die Gelegenheit, in Scratch zu programmieren und dabei gegebenenfalls auf die Unterstützung der Kursleiter zurückzugreifen. Um gleichzeitig ein fachdidaktisches Thema zu behandeln, standen bei den Aufgaben verschiedene Aufgabenformate in Scratch im Mittelpunkt (Abb. 44). Sämtliche Übungen wurden so entworfen, dass sie bestehende Aufgabenklassifikationen für Programmieraufgaben abdecken und sich darüber hinaus der Entwicklungsprozess der Lernenden beim Programmieren aufzeigen lässt (Geldreich et al. 2019c).

Zurechtfinden in Angebotsvielfalt

Neben Scratch gibt es noch viele andere Systeme, mit denen Kinder und Jugendliche das Programmieren lernen können. Die Lehrkräfte sollten sich einen Überblick über diese verschaffen und sie selbst ausprobieren, um sich eine eigene Meinung dazu bilden zu können.

An verschiedenen Stationen lernten sie weitere Möglichkeiten 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 erhielten die Lehrkräfte grundlegende Informationen zu dem jeweiligen Produkt und bearbeiteten eine einführende Aufgabe (Abb. 43). Anschließend wurden im Plenum die Vor- und Nachteile der einzelnen Produkte sowie mögliche Anknüpfungspunkte zur Unterrichtssequenz von *AlgoKids* gesammelt und diskutiert. Die Lehrkräfte, die bereits Erfahrung mit einem der Produkte gesammelt hatten, teilten diese.

Abb. 43 Arbeitsblatt zur Station *LEGO WeDo*

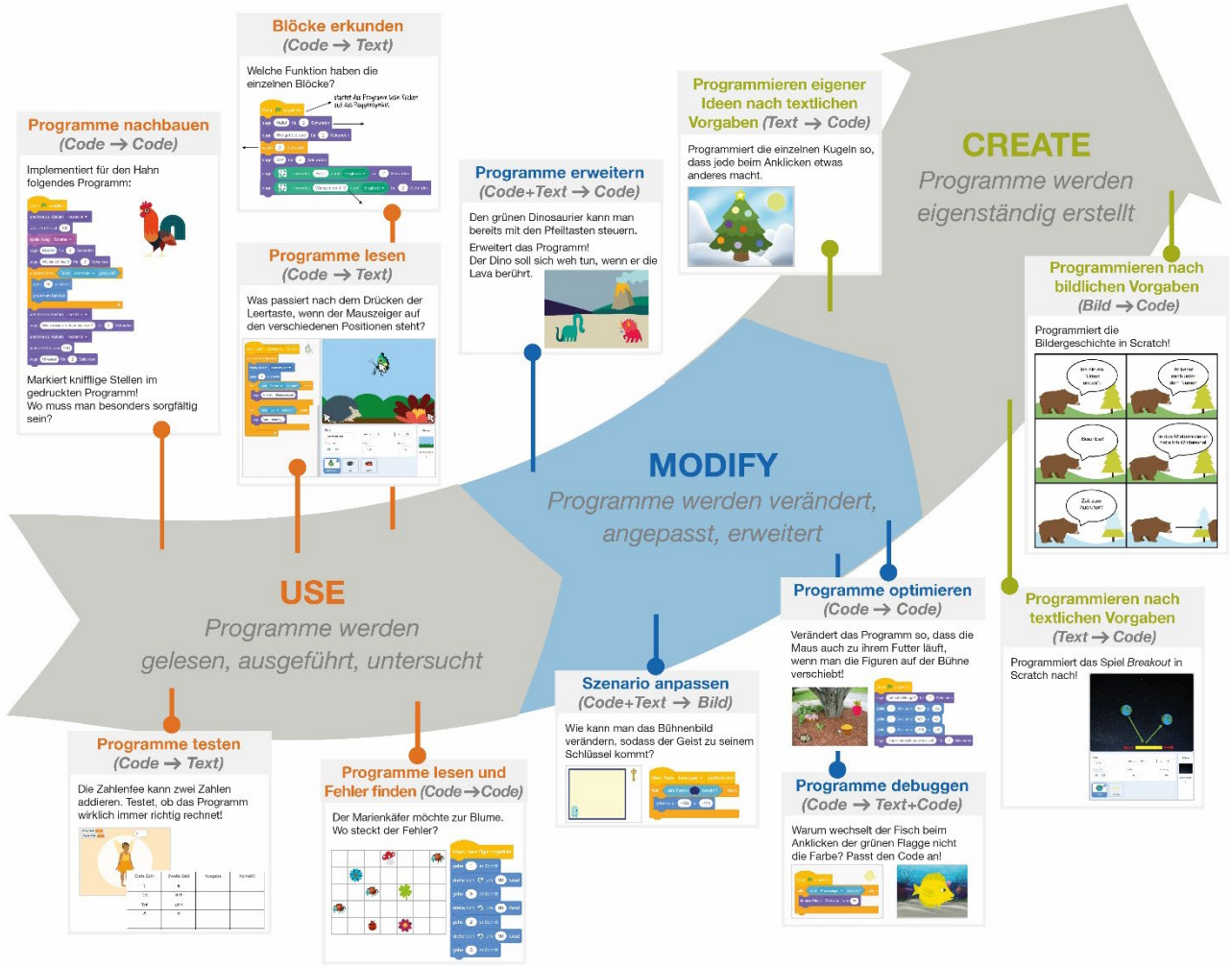


Abb. 44 Aufgabentypen in *Scratch*, die entlang des Use-Modify-Create-Ansatz gruppiert sind

Fachdidaktische Themen

Am letzten Fortbildungstag wurden verschiedene fachdidaktische Theorien vorgestellt: der Use-Modify-Create-Ansatz (Lee et al. 2011), PRIMM (Sentance und Waite 2017) sowie die Aufgabentaxonomie von Ruf et al. (2015). Um die Theorie mit der Praxis zu verknüpfen, ordneten die Lehrkräfte die Aufgaben, die sie am Vortag in *Scratch* bearbeitet hatten, in die Kategorien *use*, *modify* und *create* ein (Abb. 44). Hier arbeiteten sie zu zweit mit einem Satz Kärtchen, auf denen die Programmieraufgaben in Miniatur abgebildet waren und die sie entsprechend gruppieren konnten. Darüber hinaus wurde anhand verschiedener Lösungen der Aufgaben besprochen, bei welchen Programmierfehlern die Lehrkräfte bei ihren Schülerinnen und Schüler eingreifen müssen.

Lernstandserhebung

Zum Ende der Fortbildung erarbeiteten die Lehrkräfte Lernstandserhebungen auf der Grundlage der Empfehlungen der Gesellschaft für Informatik und dem Schweizer Lehrplan 21. Zu einzelnen Formulierungen wurden paarweise Aufgaben inklusive Erwartungshorizont für eine Lernstandserhebung entworfen. Im Anschluss wurden die Aufgaben untereinander ausgetauscht, bearbeitet und korrigiert.

6.4 Lehrerfortbildung 3

Die dritte Fortbildung fand im November 2019 in Räumlichkeiten der TU in München statt und ermöglichte allen Lehrkräften einen gemeinsamen Abschluss des Projektes. Am ersten Tag der Fortbildung arbeiteten die Lehrkräfte erneut an Programmieraufgaben in *Scratch*, auf deren Grundlage verschiedene fachdidaktische Themen im Plenum thematisiert werden konnten. Am zweiten Tag erhielten die Lehrkräfte die Möglichkeit, Feedback und Anregungen zum Projekt zu äußern sowie in der Gruppe zu diskutieren.

Programmierkontexte aus anderen Fächern

Da im LehrplanPLUS informatische Inhalte nicht explizit verankert sind, bearbeiteten die Lehrkräfte verschiedene Programmieraufgaben, die im Kontext anderer Fächer eingesetzt werden können. Zum Beispiel animierten sie ein kurzes Gedicht in *Scratch* (Abb. 45) oder programmierten geometrische Figuren (Abb. 46).

Abb. 45 Ein Gedicht (links) wird in Scratch animiert (rechts)

Abb. 46 Das Muster (links) wird durch ein Skript (rechts) erzeugt

Gute Programmierpraktiken

In Anknüpfung an die zweite Fortbildung, wurde in verschiedenen Aufgaben erneut thematisiert, wie korrekte Programme in *Scratch* aussehen. Beispielsweise wurden den Lehrkräften zwei Lösungen derselben Aufgabe zur Verfügung gestellt, bei denen sie Unterschiede in der Umsetzung finden und Kriterien für die Bewertung ableiten sollten. Anhand der Übungen wurden verschiedene positive Programmierpraktiken herausgearbeitet, wie z.B. sinnvolles Benennen von Variablen, Kommentieren von Code, Entfernen unnötiger Blöcke.

Fachdidaktische Vertiefung

In weiteren Übungen wurden verschiedene fachdidaktische Themen vertieft sowie weitere Methoden für den Unterricht vorgestellt. Beispielsweise konzentrierten sich Aufgaben auf das Verbessern von Programmen, auf den Ansatz des *Pair Programming* oder den Einsatz von *Code-Würfeln* (Abb. 47). Letztere können für den Einstieg in *Scratch* eingesetzt werden. Die Schülerinnen und Schüler bauen mit den Blöcken, die auf der Oberseite der Würfel abgebildet sind, ein Programm für eine beliebige Figur. Mit dieser Methode können die Programmierblöcke eingegrenzt sowie gezielt ausgewählte Blöcke im Unterricht behandelt werden. Gleichzeitig ermöglichen sie allen Schülerinnen und Schülern, individuelle Ergebnisse zu erzielen.

Abb. 47 Aus Papier gefaltete Würfel,
bei denen auf jeder Seitenfläche andere Blöcke abgedruckt sind

Computational Thinking ohne *Scratch*

Der letzte fachliche Block der Fortbildung setzte den Fokus auf einzelne Aspekte des *Computational Thinking* (siehe 4.1). In Kleingruppen bearbeiteten die Lehrkräfte fünf verschiedene Stationen und präsentierten diese anschließend im Plenum. Alle Stationen kamen

dabei ohne den Einsatz einer Programmiersprache aus. So stellten die Lehrkräfte z.B. einen Tanz-Algorithmus aus Symbolkarten mit unterschiedlichen Tanzposen auf (Abb. 48) oder erstellten mithilfe einer Digitalkamera einen Algorithmus für das Zusammenbauen einer Holzfigur (Abb. 49).

Abb. 48 Symbole für einzelne Tanzposen

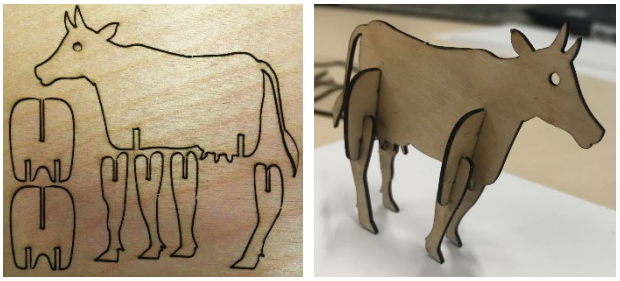


Abb. 49 Einzelteile und zusammengebaute Figur

Fazit zum Projekt

Zum Abschluss der Veranstaltung gaben die Lehrkräfte Feedback und Anregungen zu verschiedenen Themen. Zum Beispiel wurde diskutiert, inwieweit das Programmieren im Lehrplan der Grundschule verankert werden sollte, wie man Lehrkräfte in Zukunft weiterbilden könnte und welche Vernetzungsangebote sie als nützlich erachten. Als Grundlage für die Abschlussdiskussion sammelten die Lehrkräfte über den letzten Tag der Fortbildung auf Plakatwänden Stichpunkte zu den einzelnen Themen (Abb. 50, 51). Diese wurden im Plenum aufgegriffen und weiter besprochen.

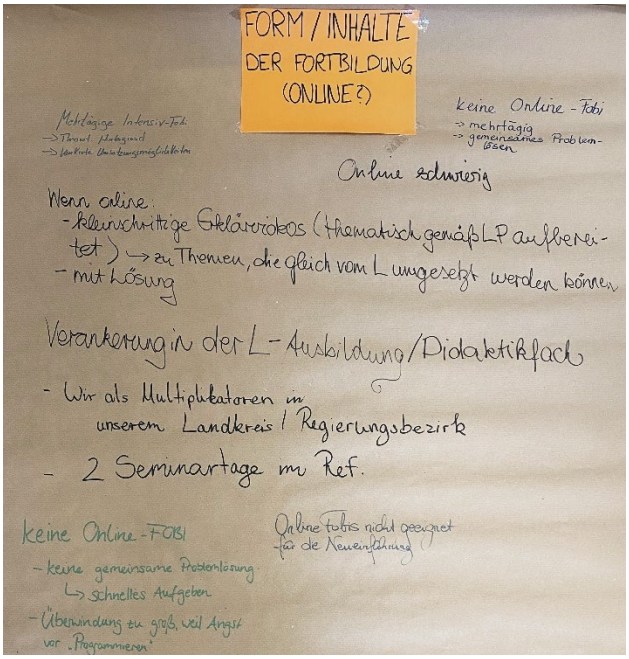


Abb. 50 Ausschnitt der Posterwand „Form/Inhalte der Fortbildung"

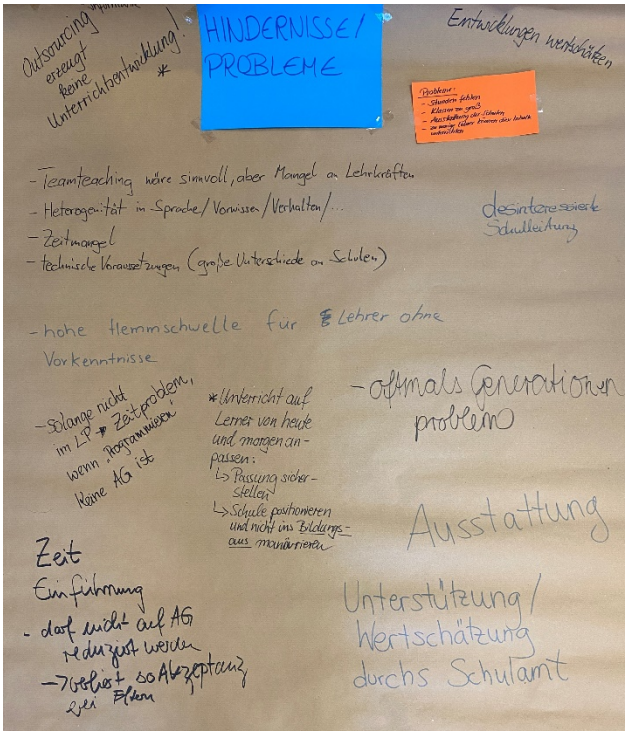


Abb. 51 Ausschnitt der Posterwand „Hindernisse/Probleme"

7. Evaluationskonzept

7.1 Ziele

Insgesamt sollten durch die Evaluation von AlgoKids fundierte Erkenntnisse in verschiedenen Domänen generiert werden, die sich an den Mehrebenen-Evaluationsmodellen von Kirkpatrick (1996) und Lipowsky (2010) orientieren. Die beiden Autoren identifizierten Kriterien unterschiedlicher Reichweite und Ausprägung, anhand derer sich die Wirksamkeit von Fortbildungen ableiten lässt. Lipowsky bezieht sein Evaluationsmodell dabei explizit auf die Wirkungen von Lehrerfortbildungen:

> *„Wirkungen von Fortbildungen lassen sich nach der Reichweite ihrer Wirkungen auf vier Ebenen verorten: Auf einer ersten Ebene lassen sie sich an den unmittelbaren Reaktionen und Einschätzungen der teilnehmenden Lehrkräfte, an ihrer Zufriedenheit und Akzeptanz festmachen, zum zweiten an kognitiven Veränderungen, also z.B. an der Erweiterung des Lehrerwissens, auf einer dritten Ebene an Veränderungen im unterrichtspraktischen Handeln und auf einer vierten Ebene an Veränderungen auf Seiten der Schüler/innen [..]“ (Lipowsky 2010, S. 52).*

Daraus wurden für AlgoKids die im folgenden beschriebenen Erkenntnisdomänen ED 1 bis ED 4 abgeleitet. Zusätzlich sollten Informationen über die Ausgangssituation der Lehrkräfte in der Erkenntnisdomäne E0 erfasst werden.

ED 0: Ausgangssituation der Lehrkräfte

Unmittelbar vor Projektbeginn wurden bei einer Reihe von Variablen die Ausgangswerte festgestellt. Um die Übertragbarkeit der Projektergebnisse zu gewährleisten, musste einerseits geklärt werden, ob bzw. welche der Lehrkräfte bereits über fachliche Vorerfahrungen verfügten. Andererseits war es notwendig, alle Variablen zu messen, für die im Laufe des Projekts Änderungen festgestellt werden sollten.

ED 1: Reaktionen und Einschätzungen der Lehrkräfte auf die Fortbildungsmaßnahmen

Ziel war es, herauszufinden, wie die Lehrkräfte subjektiv den Nutzen der Interventionen und Angebote des Projekts einschätzen und was sie zur Verbesserung für dessen Gestaltung und Ablauf vorschlagen.

ED 2: Kognitive Veränderungen bei den Lehrkräften

Dieser Bereich betrifft die kognitiven Veränderungen bei den Lehrkräften, die durch die Interventionen und Angeboten des Projekts bewirkt wurden. Den fachlichen Grundkenntnissen aus der Algorithmik kommen dabei ebenso eine Schlüsselfunktion für das Gelingen der Unterrichtsversuche zu wie den Überzeugungen und subjektiven Theorien der Lehrkräfte. Zum Beispiel sollte beleuchtet werden, wie sich das Selbstwirksamkeitsgefühl der Lehrkräfte in Bezug auf die Umsetzung der Projektinhalte über die Dauer des Projekts entwickelt.

ED 3: Veränderungen im unterrichtspraktischen Handeln der Lehrkräfte

Über die Projektlaufzeit sollte erfasst werden, in welchem unterrichtlichen Kontext, in welchem zeitlichen Umfang und in welcher Form die informatischen Inhalte in den Unterricht eingebaut werden. Darüber hinaus wurde erhoben, welchen Nutzen die Lehrkräfte darin für die Schule und den eigenen Unterricht sehen.

ED 4: Veränderungen auf Seiten der Schülerinnen und Schüler

Die Teilnahme von Lehrkräften an Fortbildungen kann sich bis auf deren Schülerinnen und Schüler auswirken (Kleickmann und Möller 2007). Im Rahmen des Projekts wurden keine Erhebungen mit Schülerinnen und Schülern durchgeführt. Die Lehrkräfte gaben jedoch Einschätzungen zum Lernfortschritt und Verhalten der Lernenden ab.

7.2 Erhebungen

Die Evaluation des Projekts erstreckte sich auf den Zeitraum zwischen den Lehrerfortbildungen zu Beginn und Ende des Projekts. Die

Evaluationsmaßnahmen schlossen alle teilnehmenden Lehrkräfte ein.

Für die schriftliche Befragung der Lehrkräfte wurden vier Fragebögen entwickelt. Diese Erhebungen richteten sich in Form von Paper-Pencil-Fragebögen jeweils gleichzeitig an alle teilnehmenden Lehrkräfte[40]. Sie verteilten sich auf insgesamt vier Erhebungszeitpunkte: zu Beginn der ersten Fortbildung, am Ende der ersten Fortbildung, zu Beginn der zweiten Fortbildung und am Ende des Projekts (Abb. 52). 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 erlaubte.

Die Ergebnisse der schriftlichen Befragungen wurden durch Rückmeldungen aus den Feedbackgesprächen ergänzt, die im Rahmen der Schulbesuche mit den Lehrkräften geführt wurden. Darüber hinaus gibt es mehrere Arbeitsergebnisse, die während der Fortbildungen erarbeitet wurden und ebenso berücksichtigt wurden.

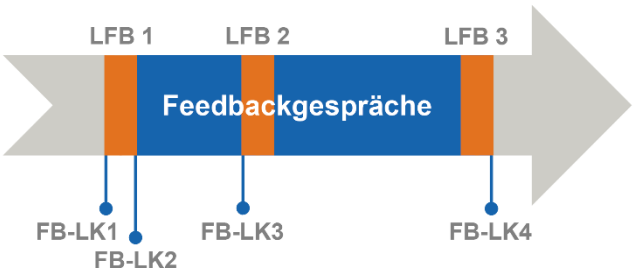


Abb. 52 Anlage der Evaluationsmaßnahmen über das Projekt

Fragebogen für Lehrkräfte FB-LK1

Die erste Befragung erfasste allgemeine Angaben zur Situation der Lehrkräfte, ihre Erwartungen an das Projekt AlgoKids, 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 et al. 2017, S. 57). Für die Operationalisierung der Sicherheit im Umgang mit Computern und Computeranwendungen wurde die entsprechende Skala von Richter et al. (2010) genutzt.

Fragebogen für Lehrkräfte FB-LK2

Am Ende der ersten Fortbildungshalbwoche wurden die Lehrkräfte gefragt, 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 wurde von Tsai et al. (2017) entwickelt und vorab ins Deutsche übersetzt. Die Items zur Selbstwirksamkeit in Bezug auf die Umsetzung der Fortbildungsinhalte im Unterricht wurden in Anlehnung an Jerusalem et al. (2009) konstruiert.

Fragebogen für Lehrkräfte FB-LK3

Zu Beginn der zweiten Fortbildung wurden die Lehrkräfte gefragt, ob sie die Fortbildungsinhalte bereits im Unterricht umsetzen konnten. Zudem wurden sie erneut befragt, inwiefern sie sich zutrauen, das Themengebiet *Algorithmen und Programmierung* in ihrem Unterricht umzusetzen und wie sie ihre eigene Programmierfähigkeit einschätzen. Zentral ist hierbei die Fragestellung, ob sich das Selbstwirksamkeitserleben der Lehrkräfte nach den ersten eigenen Unterrichtsversuchen veränderte.

Fragebogen für Lehrkräfte FB-LK4

Am Ende der dritten Fortbildung wurde ein letztes Mal erhoben, inwiefern sich die Lehrkräfte zutrauen, das Themengebiet *Algorithmen und Programmierung* im Unterricht umzusetzen

[40] 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 wie sie ihre eigene Programmierfähigkeit 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 Items wurde erneut ein Instrument von Vigerske (2017) verwendet.

Feedbackgespräche

Die Gespräche mit den Lehrkräften folgten keinem festgelegten Ablauf und thematisierten das Projekt im Allgemeinen sowie die Umsetzung an der jeweiligen Schule und ihre Erfahrungen. Dabei gingen die Lehrerinnen und Lehrer zum Beispiel auf das Umsetzungsformat, fachliche Anknüpfungspunkte an den LehrplanPLUS oder die Lernfortschritte ihrer Schülerinnen und Schüler ein.

Daten aus den Fortbildungen

Während der drei Fortbildungen wurden Plakatwände mit Stichworten und kurzen Sätzen zu verschiedenen Themen gefüllt (vgl. Kapitel 6). Am Ende der ersten Fortbildung wurde darauf notiert, wie die Themen im Unterricht umgesetzt werden könnten und welche Herausforderungen man dabei sieht. In der zweiten Fortbildung wurde Feedback zu den einzelnen Elemente der Unterrichtssequenz der TUM gesammelt. In der dritten Fortbildung zum Projektende dienten die Plakatwände als unterstützendes Element in der Abschlussdiskussion.

8. Ergebnisse der Evaluation

Durch die Evaluation von AlgoKids wurden Erkenntnisse in verschiedenen Domänen generiert (vgl. Abschnitt 7.1). Im Folgenden werden detaillierte Befunde zu den jeweiligen Themenbereichen beschrieben. Zur besseren Lesbarkeit wurden die Bezeichnungen der Domänen gekürzt.

8.1 Erkenntnisdomäne Ausgangssituation (ED 0)

Insgesamt nahmen 39 weibliche und zwei männliche Lehrkräfte am Projekt teil. Das Alter der Lehrkräfte bewegte sich zwischen *unter dreißig Jahren* und *über fünfzig Jahren* (siehe Kapitel 3). Die Berufserfahrung der Lehrkräfte zu Beginn des Projekts reichte von drei bis 39 Jahren, wobei fast die Hälfte der Teilnehmenden angab, über drei bis zehn Jahre Berufserfahrung zu verfügen (Abb. 53).

Vorerfahrungen in der Programmierung

Die Mehrheit der 41 teilnehmenden Lehrerinnen und Lehrer besaß keine Erfahrung im Programmieren. Fünf Lehrerinnen und Lehrer hatten bereits das Programmieren mit textuellen Programmiersprachen, Makros (z.B. in Word oder Excel) und realen Automatisierungssystemen (z.B. LEGO Mindstorms, LEGO WeDo, Fischertechnik) ausprobiert.

Insgesamt sieben Lehrkräfte hatten erste Erfahrungen im Programmieren mit blockbasierten Programmiersystemen gesammelt, elf hatten bereits ausprobiert, eine Website zu erstellen (z.B. in XML, HTML). Eine Lehrkraft gab an, selbst geschriebene Programme in text- und blockbasierten Programmiersprachen sowohl selbst produktiv verwendet als auch im Unterricht mit Schülerinnen und Schülern eingesetzt zu haben. Eine weitere Lehrkraft hatte bereits selbst Makros und Webseiten programmiert. Diese Ergebnisse spiegelten sich auch in der ersten Fortbildung wider. In der Vorstellungsrunde gaben mit einzelnen Ausnahmen alle Lehrkräfte an, dass Programmieren für sie ein großes Rätsel sei.

Technische Ausstattung in der Schule

Bezüglich der IT-Ausstattung der Schulen sowie deren Aktualität äußerten sich die Lehrkräfte zwar relativ ausgeglichen, es zeigte sich jedoch eine Tendenz zu einer stärker negativen Einschätzung (Abb. 54).

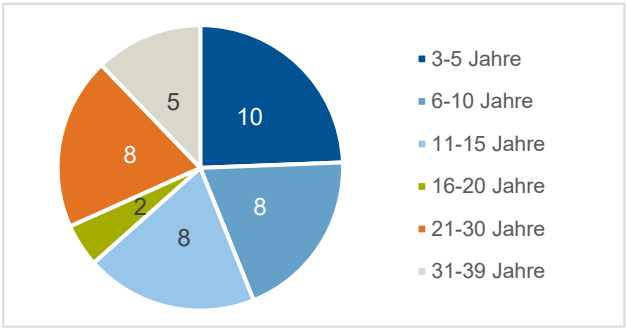


Abb. 53 Berufserfahrung der teilnehmenden Lehrkräfte (n=41)

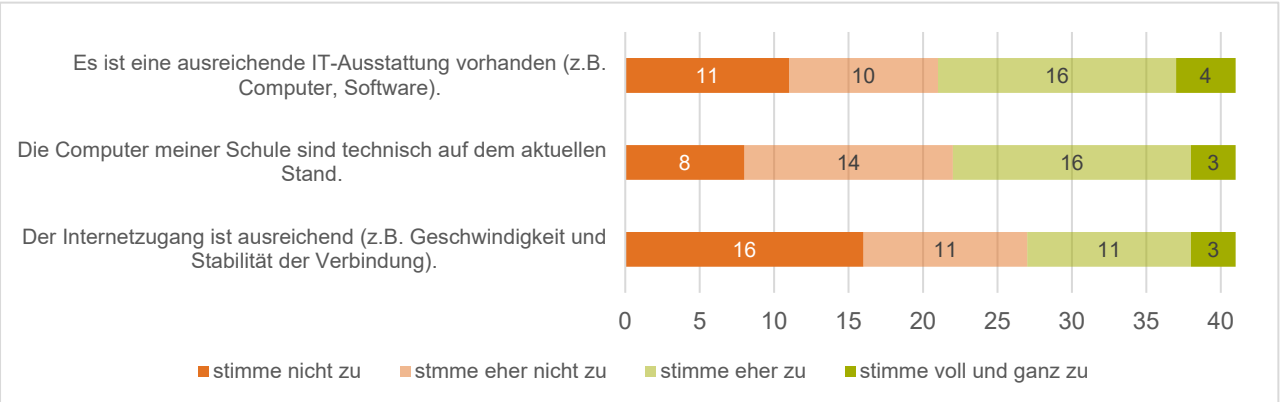


Abb. 54 Angaben zur technischen Ausstattung an den Projektschulen (n=41)

8.2 Erkenntnisdomäne Fortbildungen (ED 1)

Im Fragebogen FB-LK2 nahmen die Lehrkräfte konkret Bezug zur ersten Fortbildungshalbwoche (n=38). Alle weiteren Einschätzungen wurden im Fragebogen FB-LK4 erhoben und beziehen sich auf die gesamten Fortbildungen (n=33).

Fortbildung zu Projektbeginn

An der schriftlichen Befragung nach der ersten Fortbildungsveranstaltung nahmen 38 Lehrkräfte teil. Von diesen fühlten sich alle mindestens teilweise in der Lage, die Inhalte anzuwenden. Bis auf wenige Ausnahmen würden sie sich sogar zutrauen, die Inhalte an Kollegen weiterzugeben (Abb. 55). Die meisten Lehrkräfte hielten Dauer und Schwierigkeit der Fortbildung für angemessen. Fünf Lehrkräfte waren der Meinung, dass sie zu kurz war. Von den Lehrkräften gaben 33 an, mit der Fortbildung insgesamt *zufrieden* zu sein, fünf *eher zufrieden*. Die Erwartungen hinsichtlich des Nutzens für die Umsetzung an der Schule wurden *teilweise* (n=3), *weitgehend* (n=18) und *voll und ganz* (n=17) erfüllt.

Anwendung der Fortbildungsmaterialien

Am Ende des Projekts wurden die Lehrkräfte zu der Verwendung der Fortbildungsmaterialien befragt (Abb. 56). Es zeigte sich, dass alle befragten Lehrkräfte zumindest einen Teil des Materials im Unterricht verwendeten. Etwa dreiviertel gaben an, es inhaltlich oder methodisch weiterentwickelt zu haben.

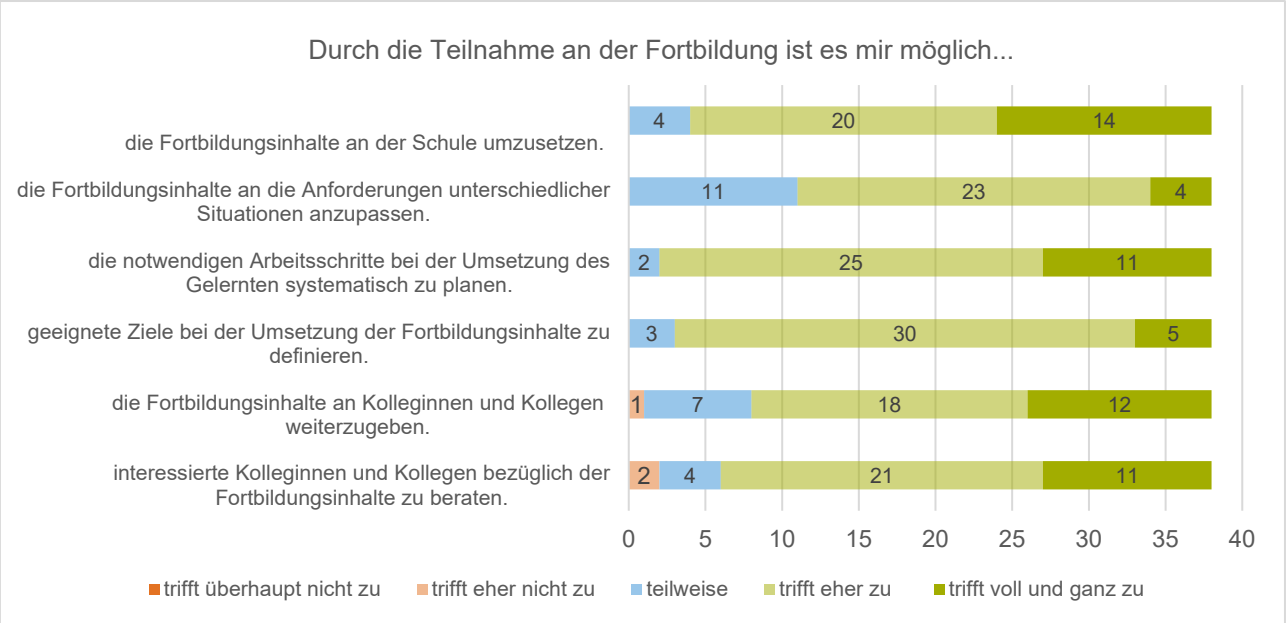


Abb. 55 Angaben zur Anwendung und Weitergabe der Fortbildungsinhalte (n=38)

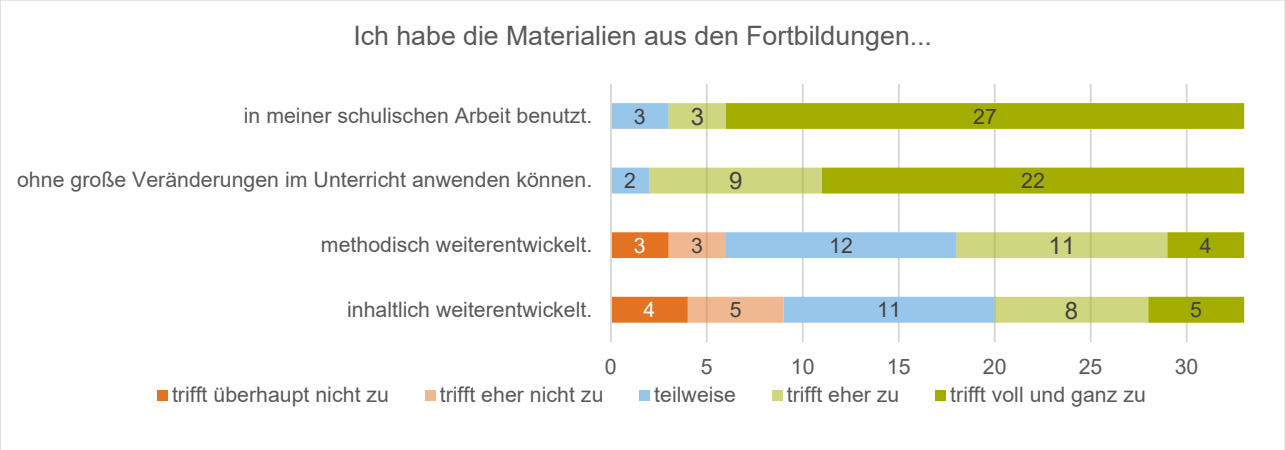


Abb. 56 Angaben zur Verwendung der Fortbildungsmaterialien (n=33)

In den Feedbackgesprächen äußerten sich die Lehrkräfte sehr positiv darüber, dass eine bestehende Unterrichtssequenz komplett zur Verfügung gestellt und in der Fortbildung selbst durchgearbeitet wurde:

Dadurch, dass wir das auch alles ausprobiert haben, habe ich mich echt sicher gefühlt.

Man konnte sofort loslegen ohne dass man sich noch tage- oder wochenlang hinsetzen musste, um noch irgendwas auszutüfteln oder vorzubereiten. Wir haben sofort anfangen können.

Das war der totale Aha-Effekt – was es da für Möglichkeiten gibt und wie kindgerecht das alles sein kann. Ohne dein Material hätte ich keine Ideen gehabt.

Programmierpraxis

Die Möglichkeit, sich intensiv mit dem Thema zu beschäftigen und selbst Programmierpraxis zu sammeln, wurde von allen Lehrkräften als unverzichtbar angesehen:

Zu wissen, ich mache jetzt diese Tage nichts anderes. Ich habe den Kopf frei und nicht zig andere Sachen zu tun. Das war toll.

Man kann sich vielleicht in Deutsch oder Mathe in Themen einlesen. Aber in Informatik – wenn man sich nicht damit auseinandersetzt, ist das nie greifbar. Erlesen funktioniert beim Programmieren nicht – man muss es gemacht haben.

Fortbildungsformat

Die Durchführung der Fortbildungen als Präsenzveranstaltungen wurde durchweg positiv bewertet. Besonders bezüglich der *Unplugged*-Aufgaben äußerten viele Zweifel darüber, ob sich ein Selbstlernkurs oder Online-Kurs umsetzen lassen würde:

Ich glaube die Begeisterung dafür entwickelt sich nur durch die persönliche Zusammenarbeit und Auseinandersetzung damit. Du hockst alleine vor dem Computer – das kann ich mir überhaupt nicht vorstellen. Und gerade am Anfang finde ich es auch wichtig, dass man Euch fragen konnte.

Wenn man es als Selbstlernkurs machen würde, da wären ja die ganzen unplugged-Übungen weg. Ich glaube, dass muss man gesehen haben. Wenn man das nur liest, kann man sich das nicht vorstellen.

Die Vertiefung von fachdidaktischen Themen in der zweiten Fortbildung sowie die Durchführung der Fortbildungen mit einigen Monaten Abstand, wurde von den Lehrkräften als sinnvoll erachtet:

Wenn etwas Zeit dazwischen liegt, kann man das alles ausprobieren. Man kann das alles testen und vielleicht hat etwas nicht funktioniert und man hat Fragen. Das ist einfach für den Austausch viel fruchtbarer, wenn Zeit dazwischen ist.

Diese Pause haben zum Ausprobieren ist gut. Weil es einfach etwas ganz anderes ist, wenn man alleine programmiert oder mit Schülern. Und man kommt dann mit ganz anderen Fragen zurück zur zweiten Fortbildung.

Bei der Abschlussdiskussion am Ende des Projekts kam zudem der Wunsch nach einer längerfristigen Vernetzung über die Projektlaufzeit hinaus auf. Hier fänden die Lehrerinnen und Lehrer regelmäßige Fortbildungen sowie die gemeinsame Entwicklung weiterer Unterrichtsszenarien sinnvoll.

8.3 Erkenntnisdomäne Lehrkräfte (ED 2)

Die Evaluation der kognitiven Veränderungen auf Ebene der Lehrkräfte bezieht sich auf deren persönliche Reflexion sowie Selbsteinschätzungen zu verschiedenen Themen. Eine Erhebung des fachlichen oder fachdidaktischen Wissens in Form eines Tests wurde nicht durchgeführt.

Nutzen für die eigene Person

In der Abschlussbefragung zum Projekt (n=33) gaben 28 Lehrkräfte an, dass sie durch die Umsetzung der Fortbildungsinhalte positive Auswirkungen auf ihre Professionalisierung feststellen konnten. 26 stellten eine positive Veränderung in ihrer Motivation und zwölf in ihrer Arbeitszufriedenheit fest. In Gesprächen merkten einige Lehrkräfte an, dass sie vor der Teilnahme am Projekt überhaupt nicht am Programmieren interessiert waren und es inzwischen als eine persönliche Bereicherung ansehen:

Ich bin sehr dankbar, dass ich die Chance hatte etwas zu machen, was ich von mir aus nie gemacht hätte. Es macht so viel Spaß und ich hab wirklich versteckte Talente entdeckt. Ich hatte tatsächlich die Einstellung als Frau, dass mich das Programmieren nicht interessiert. Und es interessiert mich doch! Ich möchte programmieren und ich möchte das verstehen.

Umsetzung der Fortbildungsinhalte

Die Lehrerinnen und Lehrer waren motiviert, die informatischen Inhalte im Unterricht umzusetzen und bewerteten die Umsetzung insgesamt als erfolgreich (Abb. 57). In den Feedbackgesprächen zeigten sich einige Lehrkräfte besorgt, dass sie nicht in der Lage sein würden, alle Fragen der Schülerinnen und Schüler beantworten zu können. Einige Lehrkräfte probierten ausgewählte Inhalte und Methoden zunächst mit einigen wenigen Schülern aus und wagten sich erst im Anschluss an eine größere Gruppe:

> *Ich habe zuerst nur mit sieben Kindern einige Sachen ausprobiert. So für acht Wochen sind wir jede Woche eine Stunde am Nachmittag in den Computerraum gegangen. Und dann fand ich es OK, eine AG alleine zu übernehmen. Weil ich auch wusste, dass ich mich erstmal an euer Konzept halten kann. Und daraus hat sich dann vieles entwickelt.*

Einige Lehrkräfte stellten fest, dass sie beim Programmieren mit ihren Schülerinnen und Schülern eine etwas andere Lehrerrolle einnahmen als gewöhnlich –fühlten sich dabei jedoch durchaus wohl. Trotz einiger anfänglicher Bedenken sahen viele sogar einen Vorteil darin, nicht immer alle Antworten zu kennen:

> *Die Lehrerrolle ist eigentlich so, wie sie im forschenden Unterricht sein sollte. Die Kinder programmieren am Computer und man kann ganz gezielt auf sie zugehen, auf sie eingehen, sie beraten. Sie entscheiden dann, was für sie am besten passt, denken selber weiter, sind aktiv und geben sich auch nicht mit vorgefertigten Lösungen zufrieden. Und sie können immer wieder ihre Ideen einbringen.*

> *Ich war ganz oft ratlos; bin bei einem Kind gestanden und musste sagen, dass ich keine Ahnung habe. Aber das war auch toll, weil die Kinder dann einfach gemerkt haben, dass der Lehrer auch nicht perfekt ist. Und man ist da auch ein Stück zusammengewachsen, wenn man gemeinsam überlegt, wie das gehen könnte. Manchmal ist dann der Schüler draufgekommen und manchmal man selber. Das war eine tolle Zusammenarbeit.*

Im Verlauf des Projekts wurde zu verschiedenen Zeitpunkten die Selbstwirksamkeitserwartung der Lehrkräfte hinsichtlich der Umsetzung informatischer Inhalte im Unterricht erhoben. Von 26 Lehrkräften lagen Daten von allen Messpunkten vor (Abb. 58). Hier zeigte sich, dass sie sich durchgängig zutrauten, im Unterricht *unplugged* zu programmieren und die Lernenden für die Thematik zu begeistern. Im Vergleich zum Messzeitpunkt nach der ersten Fortbildung (FB-LK2) zeigte sich, dass die Lehrkräfte sicherer wurden, auf die Fragen ihrer Schülerinnen und Schüler eingehen zu können und mit ihnen am Computer zu programmieren. Interessant ist, dass es einige Lehrkräfte nach ihren eigenen Erprobungen im Unterricht kritischer sehen, selbst den problematischsten Schülerinnen und Schülern die Inhalte vermitteln zu können.

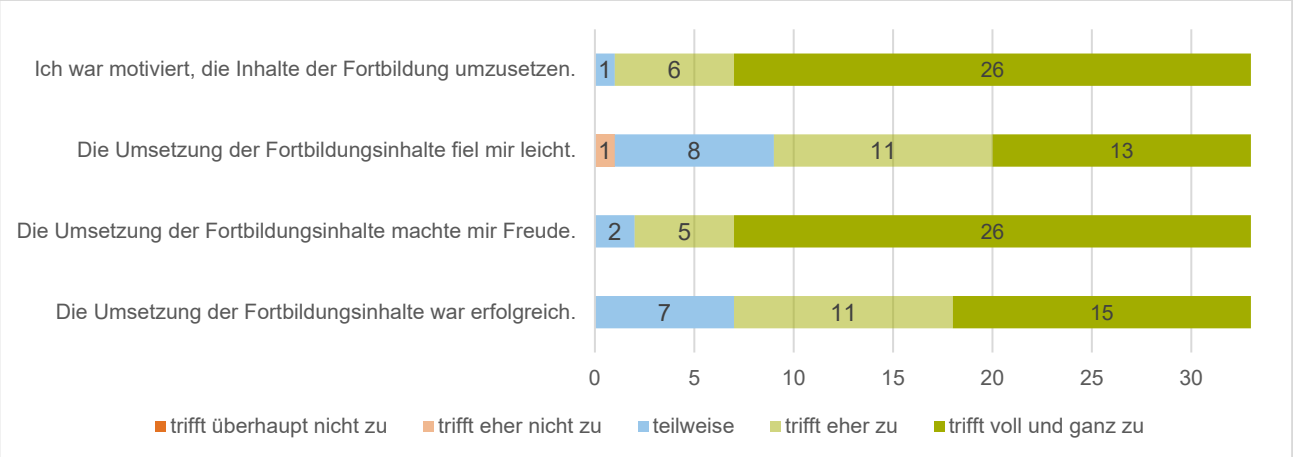


Abb. 57 Angaben zum persönlichen Empfinden der Umsetzung der Fortbildungsinhalte (n=33)

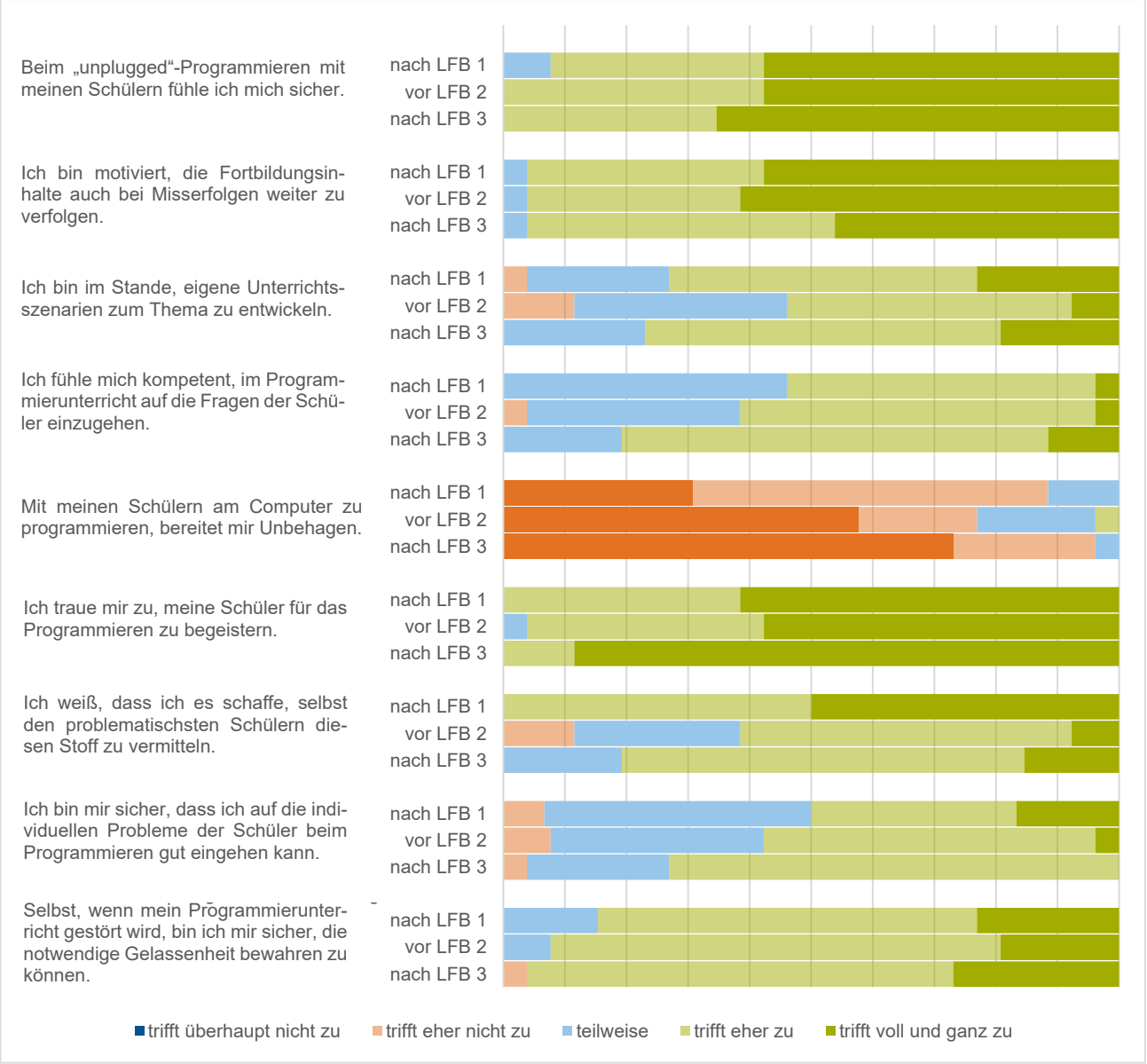


Abb. 58 Angaben zur Umsetzung der informatischen Inhalte im Unterricht (n=26)

Programmierfähigkeiten

An den gleichen drei Messpunkten beurteilten die Lehrkräfte ihre eigenen Fähigkeiten in Bezug auf einzelne Programmierkonstrukte, wie z. B. Wiederholungen oder bedingte Anweisungen, und Programmierpraktiken, wie z.B. das Verbessern von Programmen (Abb. 59). Hier zeigt sich, dass nach den ersten Unterrichtsversuchen im Bereich der Praktiken ein Abfall in der Selbstwirksamkeitserwartung zu verzeichnen war, der sich jedoch bis zum Ende des Projekts wieder aufhob.

Umgang mit Informatiksystemen

Zu Beginn und am Ende des Projekts machten die Lehrkräfte Angaben zu ihrem Umgang mit Computern und Computeranwendungen (Abb. 60). Aus den Datensätzen von 33 Lehrkräften lässt sich feststellen, dass sie sich nach dem Projekt sicherer im Umgang mit Computern fühlen und sich weniger leicht von auftretenden Problemen aus der Ruhe bringen lassen.

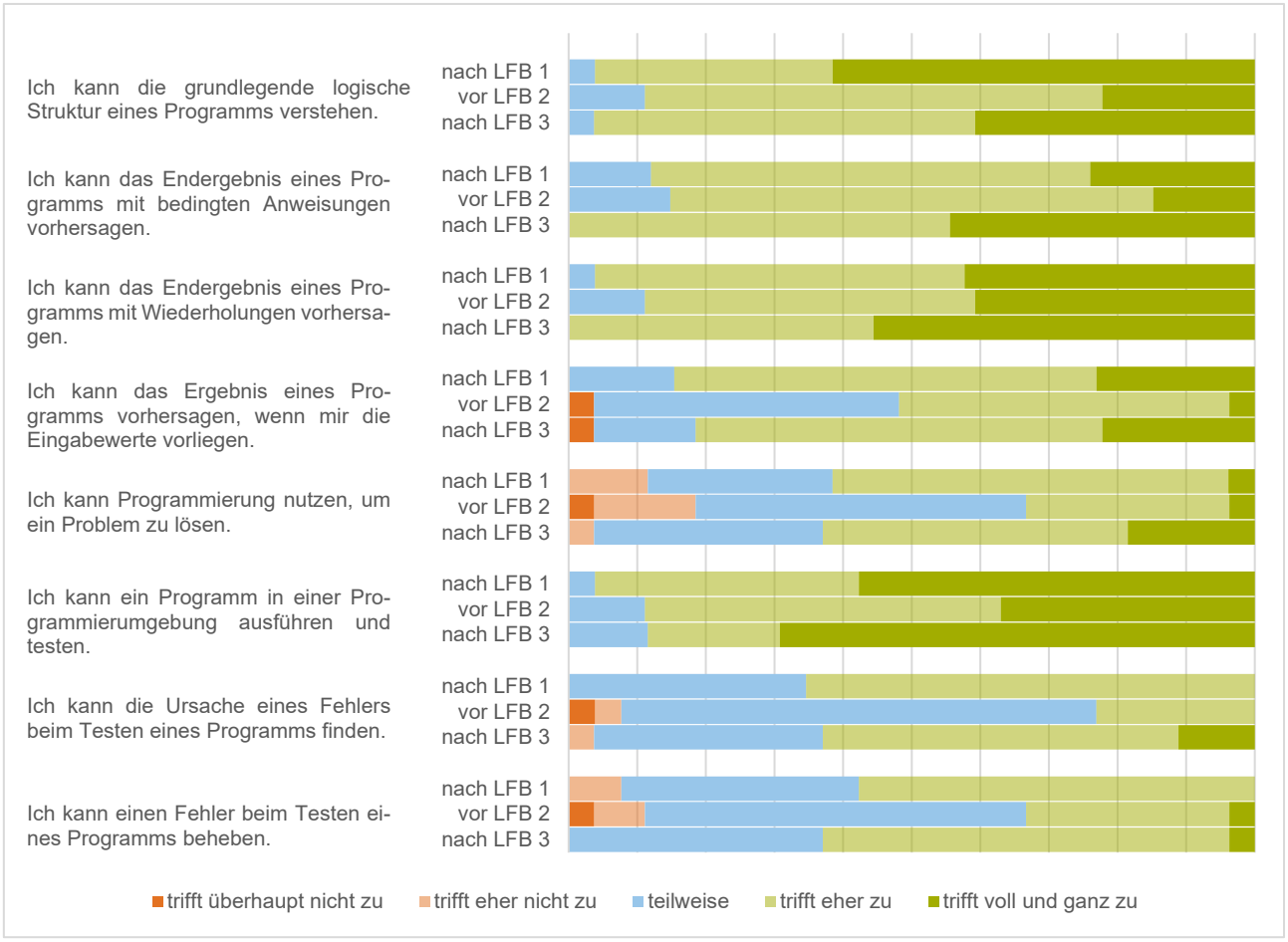


Abb. 59 Angaben zur eigenen Programmierfähigkeit (n=26)

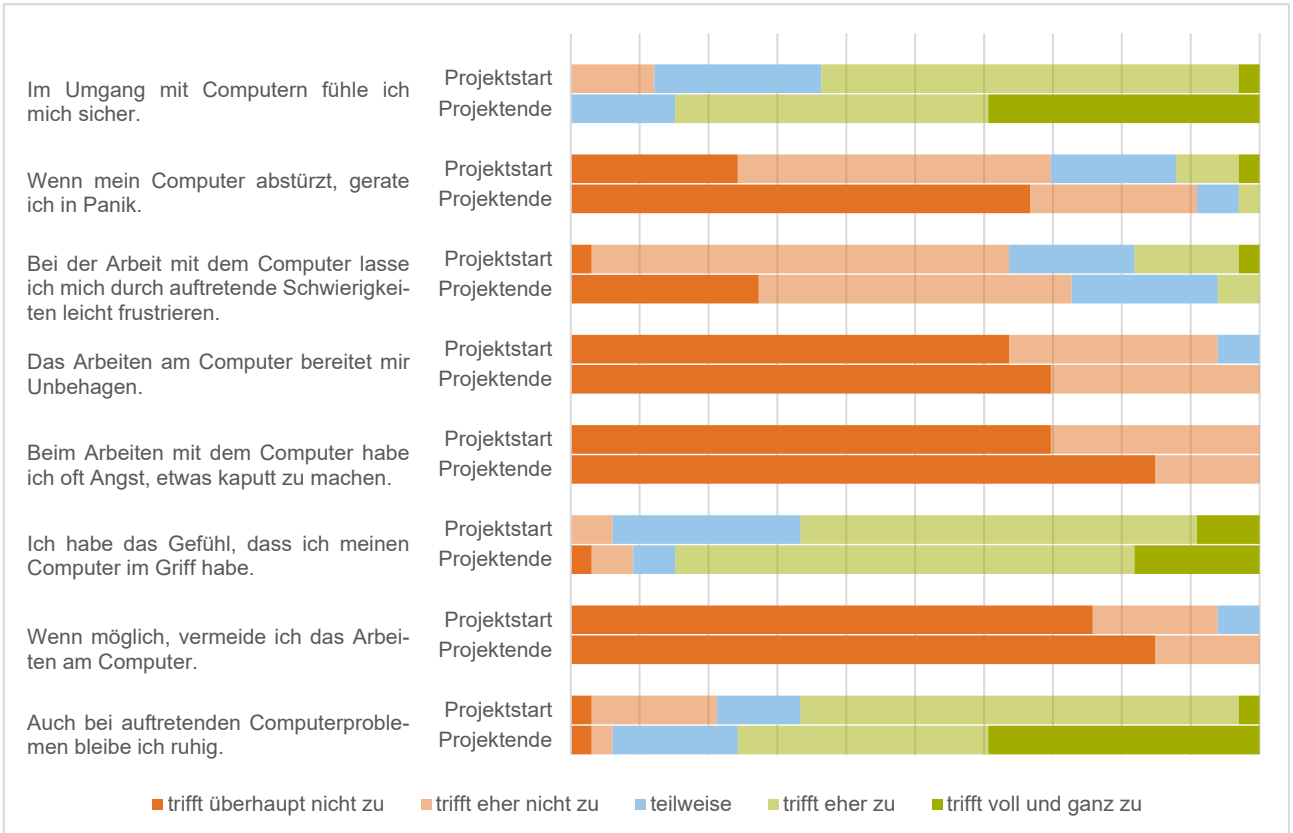


Abb. 60 Angaben zur Sicherheit im Umgang mit Computern und Computeranwendungen (n=33)

8.4 Erkenntnisdomäne Unterricht (ED 3)

Alle teilnehmenden Lehrkräfte erprobten das Programmieren im Unterricht. Im Fragebogen FB-LK4 wurden verschiedene Aspekte zur Umsetzung im Unterricht erhoben, die im Folgenden in Kombination mit weiteren Rückmeldungen der Lehrkräfte dargestellt werden.

Umsetzungsformat

Auch wenn sich alle Lehrkräfte bei ihrer unterrichtlichen Umsetzung sehr stark an der vorgestellten Sequenz orientierten, zeigten sich Unterschiede in den verschiedenen Implementierungen. Die Unterrichtssequenz wurde an den Projektschulen mit einer Ausnahme mit Schülerinnen und Schülern der Klassenstufe vier erprobt (n=19). Zwei Drittel der Schulen arbeiteten zusätzlich mit Kindern der Klassenstufe drei (n=13). Die Mehrheit der Projektschulen implementierte die Unterrichtssequenz im Klassenverband (*n=17*), ein Viertel der Schulen bot eine AG an (*n=5*), zu der zwischen sechs und sechzehn Kinder zugelassen wurden. An einer Schule wurden Elemente des Konzepts lediglich punktuell mit einzelnen Kindern ausprobiert. Drei Schulen, die im Klassenverbund programmierten, entschieden sich nach der zweiten Fortbildung dazu, *BlueBots* anzuschaffen, und diese in Zukunft zur Vorbereitung der Unterrichtssequenz einzusetzen. Eine Schule arbeitete bereits vor dem Projekt mit den Bienenrobotern. Zwei Schulen, die ausschließlich eine AG anboten, schafften im Laufe des Projekts einen programmierbaren Spielzeugroboter (*Cozmo*) und einen Satz von programmierbaren Bausätzen (*LEGO WeDo*) an. Darüber hinaus arbeitete eine Schule bereits vor Beginn des Projekts im Klassenverbund mit dem *Calliope*.

Umsetzung im Unterricht

Die Mehrheit der Lehrkräfte gab 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 Lehrerinnen und Lehrer gab an, die Inhalte angepasst, weiterentwickelt oder sogar eigene Ideen dazu entwickelt zu haben (Abb. 61).

Von 33 Lehrkräften, die an der Abschlussbefragung teilnahmen, gaben 30 an, dass sie die Fortbildungsinhalte mindestens teilweise gemeinsam mit Kolleginnen oder Kollegen umsetzten. 17 Lehrkräfte berichteten, dass sich ihre Arbeitsbelastung aufgrund der Umsetzung etwas erhöhte. Bei 16 Lehrkräften hatte die Umsetzung keine Auswirkungen auf die Arbeitsbelastung oder verringerte sie sogar.

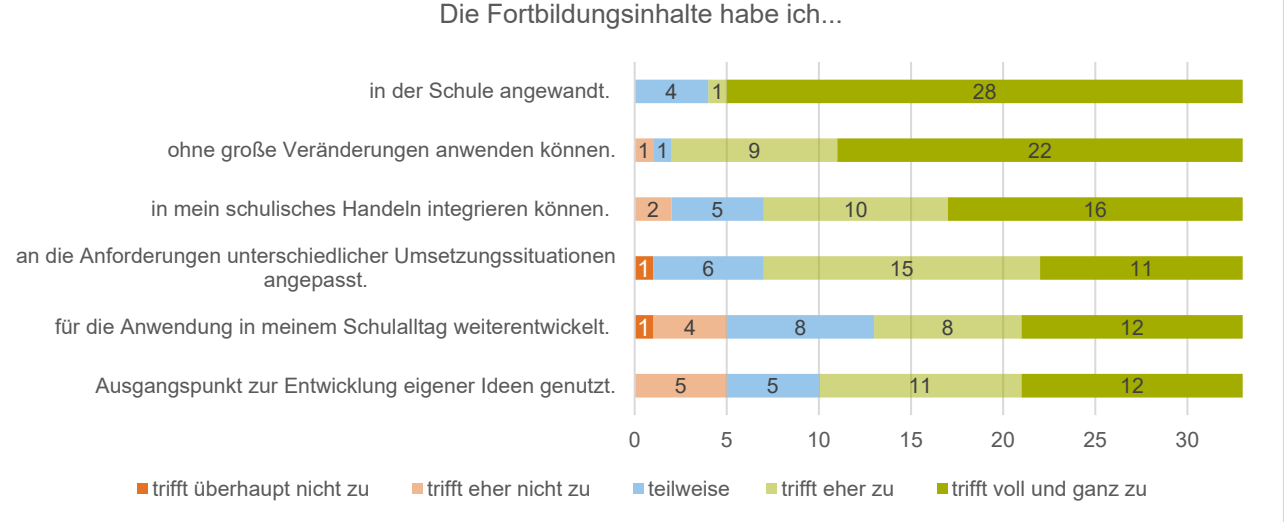


Abb. 61 Angaben zur Umsetzung der Fortbildungsinhalte (n=33)

Auswirkungen auf den Unterricht

Alle befragten Lehrkräfte gaben an, dass sie die Anwendung der Fortbildungsinhalte für sinnvoll halten. Die Hälfte von ihnen gab an, dass sie durch die Umsetzung der Fortbildungsinhalte positive Auswirkungen auf die Unterrichtsmethodik, Lehrer-Schüler-Interaktionen sowie die Schulentwicklung festgestellt haben. Alle bis auf fünf Lehrkräfte sahen positive Auswirkungen auf die Individualisierung der Lernprozesse (Tabelle 3).

Tabelle 3 Angaben zum Nutzen der Umsetzung (n=33)

Durch die Umsetzung der Fortbildungsinhalte konnte ich positive Auswirkungen feststellen in Bezug auf…	
Unterricht	
Methodik	17
Didaktik	9
(Fach)Inhalte	13
Lehrer-Schüler-Interaktion	17
individualisierte Lernprozesse	28
Schule	
Schulkonzept/Leitbild	10
Schulentwicklung	17
Profil der Schule	9
soziales Klima an der Schule	3

Hindernisse

Mehr als die Hälfte der Lehrer gab in der Schlussbefragung an, dass die Umsetzung der Fortbildungsinhalte durch fehlende Zeit oder Ausstattung erschwert wurde (Tabelle 4). Zwei Projektschulen mussten für die Umsetzung auf die Leihgeräte der TU München zurückgreifen.

Tabelle 4 Angaben zu Hindernissen in der Umsetzung (n=33)

Die Umsetzung der Fortbildungsinhalte wurde erschwert, da folgende Ressourcen nicht (ausreichend) zur Verfügung standen:	
Zeit	23
Ausstattung	21
Lehr-Lern-Materialien	6
Räumlichkeiten	8
finanzielle Mittel	17

Auch in den Feedbackgesprächen zeigte sich, dass fehlende Zeit und Probleme mit der technischen Ausstattung die Umsetzung an der Schule erschwerten:

> *Wir waren immer zu zweit in der Klasse. Da war das jetzt echt OK. Aber wenn ich mir denke, ich wäre jetzt alleine gewesen, da hätte ich es echt schwierig gefunden. Diese Orga, die Widrigkeiten mit dem Equipment!*

> *Es ist ein Zeitding! Wir haben an der Schule keinerlei Systembetreuung mit großer Stundenanrechnung oder sowas. Und so war ich jetzt einen ganzen Tag beschäftigt, bis alles installiert war. Und das ist der Punkt, an dem es scheitern könnte. Dass an Personal gespart wird und erwartet wird, dass man das freiwillig macht.*

Einige Lehrkräfte entschieden sich dafür, das Programmieren als AG anzubieten, weil sie während des regulären Unterrichts nicht genügend Zeit zur Verfügung stellen konnten:

> *Wir haben das Programmieren in eine AG ausgelagert. Weil wir es einfach schwer im Alltag hätten unterbringen können.*

> *Wenn man eine AG hat, dann hat man immer diese Stunde zur Verfügung. Und du hast Kinder drin, die fit sind und sich dafür interessieren.*

Obwohl einige Lehrkräfte von positiven Erfahrungen mit dem Programmieren in einer AG berichteten, äußerten sie aufgrund der hohen Schülerzahl Vorbehalte gegenüber dem Programmieren mit der gesamten Klasse:

> *Natürlich wünscht man sich in der AG manchmal, dass eine zweite Person im Computerraum dabei wäre die entweder hilft, wenn die Technik streikt oder den Kindern hilft, wenn man beschäftigt ist. Aber die Kinder sind da ganz entspannt und wissen, dass es manchmal einfach etwas länger dauert. Sie helfen sich dann auch teilweise gegenseitig oder probieren weiter das Problem zu lösen. Aber in der Klasse habe ich 29 Kinder - das wäre dann allein schwierig.*

Verankerung im Lehrplan

Für viele Lehrkräfte stellte es eine Schwierigkeit dar, dass informatische Inhalte nicht im Lehrplan verankert sind. Sie wünschten sich mehr Freiheit in der Stundentafel, um derartige Themen flexibler umsetzen zu können. Gleichzeitig waren viele aber auch der Meinung, dass es für viele Lehrkräfte problematisch sein könnte, wenn Informatik in zukünftigen Lehrplänen gefordert würde:

Es ist eigentlich schade, dass es nicht im Lehrplan ist, weil es wäre so viel Potenzial da in den Kindern. Ich denke, das könnte man ausschöpfen. Sie sind so motiviert und ohne Hemmungen und Ängste.

Irgendwie müsste man es im Lehrplan verankern, weil es sonst keiner macht. Aber es macht auch überhaupt kein Sinn, dass so zu institutionalisieren – Scratch bietet sich nicht dazu an, eine Probe zu schreiben!

Ich glaube, dass interessierte Lehrer sich auch damit beschäftigen, wenn es nicht im Lehrplan festgehalten ist. Und die, die keine Affinität haben, werden sich auch nicht damit beschäftigen, wenn es drinsteht.

Viele sind der Meinung, dass das Programmieren einfach eine neue Kulturtechnik wird in absehbarer Zeit und dass jeder zumindest einen Einblick bekommen muss. Aber ich glaube schon, dass der Platz dafür in der Grundschule erst noch geschaffen werden muss.

Die Mehrheit der Lehrkräfte befürwortete die Aufnahme des Programmierens in den Lehrplan der Grundschule. Uneinigkeit bestand jedoch über den Kontext, in dem dies geschehen soll bzw. geschehen könnte:

In HSU könnte man es am ehesten verorten. In Mathe kann man schon auch etwas machen mit genauen Anweisungen – weil die Mathematik ja eigentlich nicht anders funktioniert. Man muss eine gewisse Abfolge an Befehlen oder Regeln befolgen, wenn man etwas rechnet. Deutsch geht natürlich auch – dass sie ein Rezept als Anweisung schreiben müssen. Offline-Programmieren könnte man in Deutsch schon auch machen.

Ich würde es in Mathe verorten aufgrund des logischen Denkens, das einfach massiv dadurch gefördert wird.

Mathe, HSU, Deutsch – es ist halt alles ein bisschen!

Es ist absolut fächerübergreifend. Es ist etwas mathematisches, es ist ganz viel Deutsch. Der Bereich Medien ist ja im Lehrplan drin. Allerdings so vage – man kann sich auch gut drum drücken. Es müsste im Lehrplan deutlich gemacht werden, wie man es sinnvoll verknüpfen kann.

Es ist auch schön, wenn nur ein Experte von außen kommt und etwas ganz Besonderes mit den Kindern macht. Aber das ist halt dieser besondere Tag – dieser Projekttag. Und das sollte es ja eigentlich nicht sein. Man kann ganz viel so nebenher laufen lassen.

Unterrichtskonzept

Fast alle Lehrkräfte berichteten, dass das Leistungsniveau der Kinder im Umgang mit dem Computer sehr unterschiedlich war und sie deshalb eine Einheit zur Unterrichtssequenz hinzufügen bzw. vorschalten würden:

Ich würde schauen, dass die Kinder zuerst einen Grundstock an Fähigkeiten im Umgang mit dem Computer oder Tablet kriegen. Mausklick rechts, links, Doppelklick, Ordner öffnen und schließen.

Die Lehrkräfte äußerten sich positiv über die Heranführung an das Thema, in der die Schülerinnen und Schüler eine erste Idee über das Formulieren von Befehlen und beschreiben von Abläufen erhalten:

Den Lehrer zu programmieren, das ist richtig gut angekommen. Ganz viel Lachen und da war gleich schon eine positive Atmosphäre.

Das Programmieren des Lehrerroboters – dass sie einfach mal wahrnehmen, Programmiersprache muss eindeutig sein. Fand ich total klasse!

Neben dem Einstieg über das Programmieren des Lehrers, wurde auch das Beschreiben der bildlichen Anleitungen in natürlicher Sprache, z.B. das Schmieren eines Marmeladenbrots, als positiv bewertet. Einige Lehrkräfte zogen Hilfsmittel heran, mit denen die Schülerinnen und Schüler ihre Lösungen „live“ ausprobieren konnten:

Wir haben es dann nochmal ein bisschen anschaulicher gemacht. Wir haben wirklich Brot gekauft und haben sie sich selber programmieren lassen. Ich glaube, dadurch haben sie wirklich nochmal gemerkt „ich muss wirklich genau das sagen, was gemacht werden muss“.

Die Hinführung zur Programmierumgebung *Scratch* über die Aufgaben im Parcours wurden von allen Schulen umgesetzt und als essenziell für den Lernprozess der Schülerinnen und Schüler angesehen. Der Parcours wurde dabei auf unterschiedlichste Weise gelegt oder aufgezeichnet, zum Beispiel mit Teppichfliesen, Kreide oder mit Filzstift auf Leintüchern. Einzelne Aufgaben wurden teilweise als zu schwer eingeschätzt, sodass viele Lehrkräfte zusätzlich eigene, leichtere Aufgaben entwickelten. Vereinzelt entwickelten sie darüber hinaus Aufgabenformate, in denen Fehler in Programmen

gefunden werden mussten. Viele Lehrkräfte schilderten, dass die Schülerinnen und Schüler sich beim Programmieren ohne Computer (*unplugged)* sehr intensiv mit ihren Programmen auseinandergesetzten:

Wenn ein Fehler da ist, ist er nicht so einfach wegzuschieben wie am Computer. Ich glaube, man beschäftigt sich intensiver mit dem Programm. Sie denken nochmal darüber nach und reißen nicht gleich alles auseinander.

Also ohne die Unplugged-Übungen geht's für mich gar nicht. Sonst ist es für mich mehr Spielerei. Man hat diese Verinnerlichung, diesen Lernprozess, diese Einsicht.

Auch wenn die Lehrkräfte diesen Teil der Unterrichtssequenz als wichtig empfanden, waren sie in der Regel nicht der Meinung, dass in Gänze auf die Arbeit am Computer verzichtet werden sollte:

Es sind halt die Vorübungen fürs Programmieren am Computer. Sie haben dann vielleicht ein Verständnis von den Algorithmen und einzelnen Strukturen wie der Wiederholung oder einer Bedingung. Aber das muss dann doch konkret angewandt werden – sonst ist das ja wie Kaffee ohne Milch.

Sie haben am Computer einfach immer die konkrete Rückmeldung, ob das Programm funktioniert oder nicht. Macht die Figur das, was sie soll, oder nicht?

Die bereits erarbeiteten Anleitungen und Übungen im *Scratch*-Lernzirkel waren für die meisten Lehrkräfte eine Erleichterung. Während viele Lehrkräfte den Lernzirkel eins zu eins umsetzten, erweiterten ihn einige Lehrkräfte durch zusätzliche Aufgaben, in denen das bisher erarbeitete nochmals angewandt werden musste. Einige Lehrkräfte orientierten sich an den einzelnen Stationen, setzten jedoch andere Schwerpunkte oder wählten einen anderen methodischen Zugang:

Wir haben den Lernzirkel komplett durchgezogen – in einzelnen Stunden. Ich habe dann Schwerpunkte gelegt, zum Beispiel auf die Wiederholung. Ich habe die Kinder dann aber immer noch weitere Beispiele dazu programmieren lassen, dass sie das nochmal vertiefen konnten.

Ich habe mich zwar inhaltlich an die Stationen im Lernzirkel gehalten – habe die Kinder aber unabhängig von den Stationen programmieren lassen. Zum Beispiel haben wir erstmal mit Bewegungen angefangen – „Wir wollen, dass die Figur von da nach da geht. Wie bekommen wir das hin?“.

Die Lehrkräfte erachteten es als sehr wichtig, den Schülerinnen und Schülern konkrete Arbeitsaufträge in *Scratch* zu geben. Gleichzeitig beobachteten einige Lehrkräfte, dass die Kinder ihre Programme gerne personalisieren und den Wunsch äußern, freier zu arbeiten:

Vorgegebene Aufgaben finde ich ganz wichtig, bevor man frei etwas programmiert. Wenn sie ganz frei programmieren, programmieren sie einfach oft das, was sie können und was schon funktioniert. Wenn etwas nicht funktioniert, machen sie einfach was anderes und bleiben nicht dabei.

Am Lernzirkel hatten sie am Anfang sehr viel Freude – aber irgendwann war ihnen das dann zu blöd. Da kam der Wunsch, etwas Eigenes zu gestalten.

Auch wenn das Umsetzen eigener Programmideen von vielen Schülerinnen und Schülern gewünscht wurde, waren viele Lehrkräfte der Meinung, dass man ihnen zumindest einen inhaltlichen Rahmen vorgeben sollte:

Da war ich ein bisschen überrascht. Sie wollen selbst was machen, aber wissen eigentlich gar nicht, was sie wollen. Da dachte ich mir, es wäre gut ein Repertoire an Anregungen zu haben, die aber verschieden ausgestaltet werden können.

Also im Nachhinein würde ich sagen, das war fast etwas zu frei. Die wollen alles entdecken und im Endeffekt kommt nicht so viel dabei rum. Vor allem, wenn man nicht so viel Zeit hat, brauchen sie noch genauere Anleitungen.

Wir haben sie dann frei programmieren lassen, aber haben dann festgestellt, dass viele sich eher die ganzen Hintergründe und Figuren angeschaut haben als tatsächlich etwas zu programmieren.

Von den Lehrkräften, die über einen längeren Zeitraum mit ihren Schülerinnen und Schülern programmierten, wurde das Umsetzen eigener Programmideen dennoch sehr positiv bewertet:

Die eigenen Projekte, das sind wirklich so die Früchte von der Arbeit der Kinder. Die sind ihnen auch ganz wichtig.

Die Selbstreflexion kommt immer mehr. Wir können jetzt schon immer öfter sagen, lies nochmal dein Skript – was fehlt? Nach einem Schuljahr merkt man wirklich, dass sie an einem gewissen Grad der Selbstständigkeit angekommen sind, mit dem sie richtig viel anfangen können.

8.5 Erkenntnisdomäne Schülerinnen und Schüler (ED 4)

Viele der Lehrkräfte stellten durch die Umsetzung der Fortbildungsinhalte eine positive Auswirkung auf ihre Klasse fest:

Tabelle 5 Angaben zum Nutzen für die Klasse (n=33)

Durch die Umsetzung der Fortbildungsinhalte konnte ich positive Auswirkungen feststellen in Bezug auf…	
Kompetenzentwicklung der Schülerinnen und Schüler	31
Lernkultur der Klasse	13
Soziales Klima in der Klasse	14

In den Gesprächen zeigte sich ebenfalls, dass die Lehrkräfte das Programmieren für eine sehr nützliche Tätigkeit halten:

Zum einen ist es sehr motivierend, es ist modern, es ist einfach ein Medium, mit dem Kinder sinnvoll umgehen müssen. Und ich finde auch mit den ganzen Unplugged-Modulen davor – wir setzen sie ja nicht einfach vor den Laptop und die machen irgendwas. Das präzise Formulieren, die Bündelung einer Idee und die dann mit den Bausteinen als Programm darzustellen – das hat eine hohe Komplexität.

Ich glaube, es hilft den Schülern auch, ein bisschen strukturierter zu denken. Sie müssen sich wirklich einen Plan im Kopf machen – sie können schon auch Sachen ausprobieren, aber trotzdem muss man sich das genau überlegen. Das ist ja oft im Unterricht nicht so gegeben.

Einige Lehrkräfte äußerten jedoch auch Bedenken über die Ernsthaftigkeit des Programmierens im Unterricht:

Das war alles sehr spielerisch und sehr einfach! Also ich glaube, sie haben noch überhaupt nicht erkannt, dass das Informatik ist. Bei ihnen war das Programmieren mit einem riesen Spaßfaktor verbunden.

Auf der nächsten Ebene möchte ich, dass es ein bisschen an Ernsthaftigkeit zunimmt. Sie sollen sich wirklich überlegen, wie sie bestimmte Sachen Programmieren können. Aber man möchte sie ja auch nicht ausbremsen. Sie sind ja so voller Freude.

Es muss tatsächlich einen Mehrwert haben. Oft ist es gut für die Motivation – die ist auf jeden Fall da bei den Kindern. Es macht ihnen einfach Spaß. Ich finde, das hat schon eine große Berechtigung, weil wenn sie gerne in die Schule kommen, dann lernen sie vielleicht auch besser. Aber ob sie immer das lernen, was sie lernen sollen?

Motivation der Schülerinnen und Schüler

Ein weiteres wiederkehrendes Thema war die Motivation der Schülerinnen und Schüler sowie deren hohe Beteiligung:

Es ist so schön, wenn die Kinder nach der Stunde raus gehen und sagen „Wow, war das eine tolle Stunde heute“. Sie haben wirklich tolle Lernerfolge.

Die Schüler waren alle sehr interessiert und das schöne war, dass auch Kinder plötzlich total aktiv waren, die sich sonst eher sehr zurückhalten.

Die Kinder entwickeln eine Begeisterung dafür. Sie erzählen mir, dass sie sich Bücher zu Scratch gekauft haben, sich auf Scratch online angemeldet haben um ihre Projekte auch zu veröffentlichen. Oder sie erzählen mir von ihren älteren Geschwistern, die jetzt auch damit angefangen haben, weil sie ihnen davon erzählt haben.

Lehrkräfte, die das Programmieren in einer AG durchführten, merkten an, dass viel mehr Schülerinnen und Schüler die Erfahrung machen sollten, Programmieren zu lernen:

Die Motivation hält sich über das ganze Schuljahr. Wenn die AG mal ausfällt, sprechen mich die Kinder die ganze Woche auf dem Schulhof an: „Wieso ist kein Programmieren diese Woche?“. Es wäre schön, wenn mehr Kinder das Erlebnis hätten.

Genderaspekte

Viele Lehrkräfte zeigten sich sehr positiv überrascht darüber, dass sich auch Mädchen für das Programmieren interessierten:

Ich hatte schonmal eine Computer-AG angeboten. Da war das tatsächlich so, dass die Mädchen überhaupt nicht wollten und gesagt haben, sie können das nicht. Aber jetzt war das anders. Jetzt hatten sich viele Mädchen freiwillig für die AG angemeldet.

Positive Erfahrungen machen mit Informatik ist wichtig. Das habe ich jetzt gemerkt – viele Mädchen haben das Programmieren als versteckte Fähigkeit für sich entdeckt. Das macht ihnen Spaß, sie können das auch und haben Erfolgserlebnisse. Programmieren ist nicht nur für Jungs oder etwas, bei dem sie nicht durchsteigen.

Mehrere Lehrkräfte berichteten in den Feedbackgesprächen, dass die Jungen im Vergleich zu den Mädchen mehr Vorerfahrungen im

Umgang mit Computern aufwiesen und auch eher zuhause programmierten:

Ich würde sagen, die Jungs kennen sich besser mit dem Computer aus – was wahrscheinlich daran liegt, dass sie einfach zuhause mehr damit machen. Was nicht heißt, dass sie es besser können. Aber ich glaube, dass sie einfach mehr Erfahrung damit haben. Wobei die Mädchen oft mehr Geduld haben, wenn was nicht funktioniert.

Die Jungs sind tatsächlich die, die sich zuhause mehr damit beschäftigen. Sich Scratch runterladen, sich Bücher besorgen und zuhause programmieren. Oder mir erzählen, sie haben sich zuhause etwas überlegt und wollen es jetzt umsetzen. Von den Mädchen habe ich das so noch nicht gehört.

Heterogenität der Lerngruppen

Für viele Lehrerinnen und Lehrer war es eine Herausforderung, den unterschiedlichen Fähigkeiten und Wissensständen der Schülerinnen und Schüler gerecht zu werden:

Ein Schüler hat die AG nach einer Weile verlassen. Der war sehr fortgeschritten und hat schon in C programmiert – der Papa macht das beruflich. Die anderen hatten noch nie etwas vom Programmieren gehört.

Ich hatte Kinder, die kannten sich mit Laptops aus, ich hatte Kinder, die Scratch schon kannten und ich hatte Kinder, die hatten noch nie irgendein technisches Gerät in der Hand gehabt. Und das aufzufangen war am Anfang eine Herausforderung.

Die Bedienung vom Computer ist für viele ein Problem. Die haben mit dem Doppelklick zu kämpfen, damit wie man scrollt oder einen Ordner findet. Ich hatte manchmal das Gefühl, dass sie gar nicht auf das nächste Level beim Programmieren gekommen sind, weil das im Weg stand.

Ich hab etwas Bauchweh, dass wir irgendwann auf einem Niveau sind, wo auch ich nicht mehr helfen kann. Das passiert mir natürlich in keinem anderen Fach. Man stößt dann schon irgendwann an seine Grenzen. Ein paar Einheiten durchzumachen ist kein Problem – aber wenn das jetzt das ganze Schuljahr durchginge…

9. Zusammenfassung der Ergebnisse

Das Projekt *AlgoKids – Algorithmen für Kinder* ging von einer mehrperspektivischen Zielvorgabe aus: Das Unterrichtskonzept der TUM zum Programmieren in der dritten und vierten Jahrgangsstufe sollte an den Projektschulen erprobt und ggf. erweitert werden. Darüber hinaus sollte ein Fortbildungskonzept für Grundschullehrkräfte erarbeitet sowie umgesetzt werden. Während der 22-monatigen Projektlaufzeit wurde zudem untersucht, inwieweit sich die beteiligten Lehrkräfte das Unterrichten der informatischen Inhalte zutrauen und welchen Nutzen sie für ihre Schülerinnen und Schüler sehen. Insgesamt kann man feststellen, dass wir diese Ziele in zufriedenstellendem Maß erreicht haben. Im Folgenden fassen wir die wichtigsten Ergebnisse zu den einzelnen Fragestellungen zusammen.

9.1 Unterrichtssequenz

- Die Grundstruktur der ursprünglichen Unterrichtssequenz hat sich an den Projektschulen gut bewährt. Auf Basis der Rückmeldungen und Erweiterungen der Lehrkräfte wurde das Konzept weiter überarbeitet und durch Good-Practice-Beispiele ergänzt.
- Die Lehrerinnen und Lehrer schätzten die kindgerechte Aufbereitung der informatischen Inhalte und die Möglichkeiten für individualisiertes Lernen.
- Die *Unplugged*-Übungen wurden von allen Lehrkräften als notwendig erachtet – eine Umsetzung ausschließlich ohne Programmieren am Computer hielten sie jedoch nicht für sinnvoll.
- Die Lehrkräfte schilderten eine anhaltend hohe Motivation der Kinder beim Programmieren – sowohl der Jungen als auch der Mädchen.
- Das Programmieren in der Grundschule hielten die Lehrkräfte für sehr sinnvoll. Sie berichteten von einem Kompetenzzuwachs ihrer Schülerinnen und Schüler beim Programmieren und verzeichneten teilweise auch eine Veränderung in deren Arbeitsweise.
- Besonders erfreulich fanden die Lehrkräfte, wie gut die Schülerinnen und Schüler beim Programmieren zusammenarbeiteten und sich gegenseitig unterstützten.

9.2 Das Fortbildungskonzept

- Die Lehrkräfte fanden es gut, dass unser Fortbildungskonzept sowohl Erkenntnisse aus der Informatikdidaktik als auch aus der Erwachsenenbildung berücksichtigte. Dies bewährte sich in der Umsetzung insgesamt sehr gut.
- Die Lehrerinnen und Lehrer betonten die Notwendigkeit einer ausreichenden Fortbildung, um die erforderlichen Kenntnisse für die Umsetzung der Inhalte im Unterricht zu erwerben.
- Besonders positiv wurde angesehen, dass im Rahmen der Fortbildung eigene Programmierpraxis und zwischen den einzelnen Fortbildungen erste Erfahrungen in der Umsetzung an der Schule gesammelt werden konnten.
- Die teilnehmenden Lehrkräfte konnten sich nicht vorstellen, dass die Fortbildungen ausschließlich online durchgeführt werden könnten.
- Die Ergebnisse der Evaluation zeigen, dass die Lehrkräfte sich das Unterrichten der Inhalte durch entsprechende Fortbildung zutrauen.

9.3 Kritische Aspekte

- Als problematisch für die Umsetzung wurde die hohe Schülerzahl einiger Klassen geschildert.
- Die fehlende Verankerung der Inhalte und Kompetenzen im Lehrplan bereitete den Lehrerinnen und Lehrern Sorgen.
- Die technische Ausstattung der teilnehmenden Schulen war sehr unterschiedlich. In vielen Fällen waren die Geräte nur bedingt einsatztauglich.
- Oft fehlte eine ausreichende Systembetreuung.

10.Empfehlungen

Auf der Grundlage der Ergebnisse des Projekts können wir dem Staatsministerium guten Gewissens empfehlen, Programmieren in der Grundschule nach diesem Konzept an allen bayerischen Grundschulen einzuführen. Aus den Erfahrungen, die im Rahmen von AlgoKids gemacht wurden, lassen sich im Einzelnen folgende Empfehlungen ableiten.

10.1 Kurz- und mittelfristige Maßnahmen

Zum Programmieren für Kinder sind in den letzten Jahren zahlreiche Angebote kommerzieller und nichtkommerzieller Anbieter entstanden, die teilweise sehr intensiv beworben werden. Dadurch werden Eltern immer stärker unter Druck gesetzt, ihre Kinder in dieser Richtung zu fördern. Dies wird nahezu zwangsläufig die sozial bedingten Unterschiede in der individuellen Förderung weiter verstärken. Um diese Entwicklung aufzuhalten und zu verhindern, dass kommerzielle Anbieter aus den USA oder China unkontrollierten Einfluss auf unsere Kinder gewinnen, ist schnelles Handeln geboten. Nur so kann der Freistaat Bayern weiter die Oberhoheit über die Programmierausbildung seiner Grundschulkinder behalten.

Handreichung für Lehrerinnen und Lehrer

Wir schlagen daher vor, das Staatsinstitut für Schulqualität und Bildungsforschung (ISB) umgehend mit der Erstellung einer Handreichung zum Programmieren in der Grundschule zu beauftragen und dafür einen Arbeitskreis einzurichten, der sein Ergebnis spätestens zum Ende des Schuljahres 2021/22 abliefert. Die Handreichung sollte die Materialien und Erfahrungen aus dem Projekt AlgoKids berücksichtigen, insbesondere die folgenden Punkte.

Innerhalb des Projekts erwiesen sich verschiedene Inhalte und Methoden für das Programmieren in der Grundschule als erfolgreich (Tabelle 6). Diese bieten den Lehrkräften einen inhaltlichen Rahmen, gleichzeitig jedoch auch Raum zur weiteren Ausgestaltung im Sinne eines kompetenzorientierten Unterrichts. Der zeitliche Umfang und die Verteilung der einzelnen Einheiten kann verändert werden, es handelt sich bei den Angaben lediglich um Richtwerte.

Viele Kinder werden zum ersten Mal vor einem Computer sitzen, weil sie zu Hause nur Smartphones oder Tablets kennengelernt haben. Zu Beginn oder vor der Einheit zum Programmieren sollten daher die Grundlagen für das Arbeiten am Computer erarbeitet werden, z.B. das Öffnen eines Ordners oder die Bedienung der Maus.

Größere Klassen sollten für das Programmieren am Computer in zwei Gruppen aufgeteilt werden, die von je einer Lehrkraft betreut werden.

Lehrerfortbildung

Gleichzeitig mit der Erstellung der Handreichung sollte die ALP mit der Entwicklung und Umsetzung eines Fortbildungskonzepts zum Programmieren in der Grundschule beauftragt werden, weil ein adäquates Fortbildungsangebot den Erfolg der Maßnahme maßgeblich bestimmt. Die Entwicklung und Planung könnte im Juli 2021 abgeschlossen sein, so dass die ersten Fortbildungen im Herbst 2021 beginnen könnten.

Die Maßnahme sollte in Präsenz stattfinden und mindestens zwei Fortbildungshalbwochen je Lehrkraft umfassen.

Zwischen den beiden Halbwochen sollte ein ausreichender Zeitraum für eigene Unterrichtserfahrungen im Programmieren liegen, mindestens ein halbes Schuljahr.

Die Bedeutung der wichtigsten Fachkonzepte sollte dabei in Form eines Glossars kommuniziert werden, da die Lehrkräfte ansonsten keine Informatikausbildung aufweisen.

Für die beiden Veranstaltungen empfehlen wir die Themen und Methoden aus dem Fortbildungskonzept von AlgoKids (Tabelle 7).

Zur Vertiefung bestimmter Inhalte oder zu spezielleren Themen sollten zusätzlich kürzere Fortbildungen im Umfang eines Nachmittags angeboten werden.

An jeder Schule sollten mindestens zwei Lehrkräfte nach diesem Konzept fortgebildet werden, um sich einerseits gegenseitig zu unterstützen und andererseits Redundanz wegen möglicher Versetzungen oder Ausfälle abzusichern. Sie sind auch Ansprechpartner für andere Lehrkräfte an der jeweiligen Schule.

Aufgrund der erheblichen Anzahl von ca. 2400 Grundschulen in Bayern müssten demnach ca. 4800 Lehrkräfte fortgebildet werden. Bei einer Gruppenstärke von 30 Personen je Fortbildung müssten also 320 Halbwochen angeboten werden. Wenn man das Projekt über 5 Jahre verteilt, müsste die ALP je Kurshalbjahr 32 Kurse zum Programmieren in der Grundschule organisieren.

Dazu müsste vermutlich auf ein Multiplikatorensystem zurückgegriffen werden. Die Ausbildung der Multiplikatoren müsste allerdings im Vergleich zu den anderen Teilnehmern wesentlich umfangreicher und intensiver ausfallen. Wir schlagen vor, sich im Schuljahr 2021/22 zunächst auf die Ausbildung von 60 Multiplikatoren in je zwei vollen Kurswochen zu konzentrieren. Hierzu müsste die ALP also insgesamt 4 einwöchige Kurse anbieten, am besten am Anfang und am Ende des Schuljahres, damit dazwischen Unterrichtserfahrungen gesammelt werden können. Als Multiplikatoren würden sich natürlich vorrangig die Teilnehmer/innen des AlgoKids-Projekts anbieten.

TUMDDI würde die Multiplikatorenausbildung sehr gerne unterstützen, u.a. durch unseren MOOC zum objektorientierten Programmieren (*LOOP*[41]).

Zur überschulischen Vernetzung und zum Austausch von Material sollten regelmäßige, attraktive Angebote geschaffen werden. So sollten die Lehrkräfte dazu angeregt werden, ihre Ideen und Materialien über *mebis* zu teilen. Allerdings sollte dazu eine zentrale Qualitätskontrolle eingerichtet werden, um fachliche Korrektheit zu gewährleisten.

IT-Ausstattung und Systembetreuung

Damit die Einführung informatischer Inhalte in der Grundschule gelingen kann, sollten angemessene technische Rahmenbedingungen geschaffen werden. Nach unserer Erfahrung fehlen vielen Grundschulen geeignete Endgeräte entweder völlig oder in passender Zahl bzw. Ausstattung.

Zum Programmieren sollte jede Grundschule mindestens über einen Klassensatz (d.h. ca. 25) Laptops oder Tablets mit Internetzugang verfügen, auf dem die benötigten Programmierumgebungen lauffähig sind.

Viele Programmierumgebungen können über den Internet-Browser betrieben werden, so dass keine spezielle Software installiert werden muss. Allerdings müssen die Programme der Kinder gespeichert und ausgetauscht werden können, wozu entsprechende Dateisysteme auf einem lokalen Server oder einer Cloud benötigt werden.

Natürlich müssen die Geräte regelmäßig ausreichend gewartet werden, wozu Lehrkräfte grundsätzlich nicht herangezogen werden sollten, weil ihnen sowohl die nötige Zeit als auch die notwendige spezielle und aktuelle Ausbildung fehlen. Die jüngsten Pläne der Staatsregierung zur Systembetreuung an Schulen durch spezielle Firmen klingen dagegen sehr passend.

[41] https://www.edu.tum.de/ddi/lehre/moocs/loop/

Tabelle 6 Elemente des Unterrichtskonzeptes

Thema	Kurzbeschreibung	Umfang
Grundlagen von Algorithmen und Programmen	• Beschreiben von Algorithmen und ihren Abläufen in Alltagssprache • „Programmieren“ von natürlichen Personen (Lehrkraft, Schülerinnen und Schüler)	ca. 1 Doppelstunde
Programmieren *unplugged*	• Beschreiben von Abläufen mit Symbolen, z.B. von Wegen in einem Raster • Beschreiben von Abläufen mit haptischen Darstellungen von Programmieranweisungen • Anwenden und Zusammensetzen von algorithmischen Strukturelementen	ca. 1-2 Doppelstunden
Programmieren am Computer	• Programmieren eines „echten“ Informatiksystems mit einer blockbasierten Programmiersprache, z.B. *Scratch*, *Blockly* oder *Snap.* • Anwenden von Grundfunktionen des Systems und Implementierung von algorithmischen Strukturelementen	ca. 2-3 Doppelstunden
Aufgaben analysieren und Programme planen	• Analyse von schwierigeren Aufgabenstellungen • Umfangreichere Abläufe und Algorithmen entwerfen • Implementieren von umfangreicheren Algorithmen mit einer blockbasierten Programmiersprache	ca. 1-2 Doppelstunden

Tabelle 7 Struktur des Fortbildungskonzeptes

	Themenschwerpunkte
Fortbildungshalbwoche 1	• Ausprobieren einer Unterrichtssequenz in der „Schüler(innen)rolle“ • Diskussion der fachlichen Grundlagen, insbesondere der Fachkonzepte • Praktisches Programmieren: Anwendung der algorithmischen Strukturelemente • Kennenlernen verschiedener Aufgabentypen beim Programmieren (z.B. Programme analysieren, Programme ändern, Programme erstellen)
Fortbildungshalbwoche 2	• Erfahrungsaustausch über die eigenen Erfahrungen • Praktisches Programmieren: Unterstützung von Lernprozessen der Schülerinnen und Schüler, Finden und Verbessern von Fehlern • Einblick in weitere Möglichkeiten zum Programmieren in der Grundschule
zusätzliche Angebote	• Programmieren mit anderen Systemen (aktuell z.B. *Calliope mini*, *Makey Makey*, *Ozobot*[42], *Dash & Dot*[43])

[42] https://ozobot-deutschland.de/

[43] https://www.makewonder.de/dash/

10.2 Langfristige Maßnahmen

Neben diesen kurz- und mittelfristigen Maßnahmen sind zusätzliche langfristige Aktivitäten notwendig, um die Informatik an den Grundschulen nachhaltig abzusichern und auch auf neuere Entwicklungen fachlich und didaktisch kompetent reagieren zu können.

Lehrplanentwicklung

Informatik als zusätzliches Schulfach in der Grundschule einzuführen, ist in Anbetracht des Mangels an qualifizierten Lehrkräften wohl nur bedingt möglich und sinnvoll. Informatische Inhalte sollten dennoch bei der nächsten Revision des LehrplanPLUS explizit im Lehrplan verankert werden, beispielsweise in den Fächern Deutsch, Mathematik und dem Sachunterricht.

Unterrichtswerke

Sobald informatische Inhalte im Lehrplan verankert sind, sollten die Schulbuchverlage zur Erarbeitung von Unterrichtswerken zur Informatik in der Grundschule aufgefordert werden.

Lehrerausbildung

Um informatische Bildung in der Grundschule langfristig zu stabilisieren, muss Informatik als vollwertiges Fach in die erste Phase der Lehrerausbildung integriert werden. Zu diesem Zweck sollte zunächst ein Didaktikfach Informatik für das Lehramt Grundschule in der LPO I verankert werden.

In einem zweiten Schritt sollte auch ein Unterrichtsfach Informatik in die Liste der Fächerverbindungen aufgenommen werden, die gemäß § 35 (1) LPO I für Lehramtsstudierende zulässig sind. Zudem müsste man aktiv dafür werben, wenigstens einige Lehramtsstudierende für dieses Unterrichtsfach zu gewinnen. Diese Absolventen würden dringend zur fachgerechten und altersgemäßen Betreuung des Faches im Staatsministerium, Lehrplanentwicklung im ISB, Konzeption und Durchführung von Lehrerfortbildungen sowie zum Verfassen von Unterrichtswerken und Hilfsmaterial gebraucht. Andernfalls müssen diese Aufgaben entweder von fachfremden oder von Lehrkräften anderer Schularten übernommen werden.

Natürlich ist uns klar, dass diese Maßnahmen zum Teil sehr teuer und aufwändig sein werden. Wir sind jedoch überzeugt, dass sich der Aufwand langfristig lohnen wird, um Bayern, Deutschland und damit auch Europa dabei zu unterstützen, den aktuell gewaltigen Vorsprung der USA und Chinas im IT-Bereich aufzuholen.

Nur so wird Europa auf die Dauer in der Lage sein, seinen Wohlstand und seine Freiheit zu erhalten und eine prägende Rolle in der internationalen Staatengemeinschaft zu spielen, anstatt weiter vollständig von den IT-Großmächten abhängig zu bleiben.

11. Dokumentation des Projekts

11.1 Struktur des Projekts

Projektinitiative und -verantwortung:	Prof. Dr. Peter Hubwieser (TUM)
Projektleitung (Konzeption, Durchführung und Evaluierung):	Katharina Geldreich (TUM)
Ansprechpartnerin am StMUK:	Maria Wilhelm, MRin
Betreuung an der ALP:	Maria Stein, IRin
	Uwe Geuder, R (GS, MS)
Laufzeit:	März 2018 – Dezember 2019

Übersicht der Projektschulen:

11.2 Teilnehmende Schulen

Oberbayern

- Camerloher Grundschule München, Camerloherstraße 110, 80689 München
- Grundschule Burgheim, Schulgasse 4, 86666 Burgheim
- Grundschule Piding, Salzburger Str. 4, 83451 Piding
- Grundschule Wielenbach, Rosenstr. 9, 82407 Wielenbach
- Grund- und Mittelschule Vaterstetten, Gluckstr. 2, 85598 Baldham
- Mangfallschule Grundschule Kolbermoor, Rainerstraße 2, 83059 Kolbermoor

Niederbayern

- Bischof-Riccabona Grund- und Mittelschule Wallersdorf, Osenstr. 16, 94522 Wallersdorf
- Grundschule Stallwang, Kirchberg 32, 94375 Stallwang

Oberpfalz

- Barbara-Grundschule Amberg, Raiffeisenstraße 2, 92224 Amberg
- Grundschule Burgweinting, Obertraublinger Str. 22, 93055 Regensburg

Oberfranken

- Grundschule Burgebrach, Grasmannsdorfer Str. 3, 96138 Burgebrach
- Grundschule Marktredwitz, Bauerstraße 4 - 6, 95615 Marktredwitz

Mittelfranken

- Grundschule Gunzenhausen-Südstadt, Theodor-Heuss-Straße 1, 91719 Gunzenhausen
- Grundschule Wahlerschule , Holsteiner Straße 2a, 90427 Nürnberg
- Gustav-Weißkopf-Grundschule, Alter Postberg 7, 91578 Leutershausen

Unterfranken

- Astrid-Lindgren-Grundschule Hösbach, Jahnstraße 1-3, 63768 Hösbach
- Johann-Peter-Wagner-Mittelschule, Theres Alice-von-Swaine-Str. 12, 97531 Theres

Schwaben

- Erich-Kästner-Grundschule Neu-Ulm, Hasenweg 19, 89231 Neu-Ulm
- Grundschule Augsburg Bärenkeller, Bärenstr. 15, 86156 Augsburg
- Grundschule Griesbeckerzell-Obergriesbach, Schulweg 9, 86551 Aichach

11.3 Veranstaltungen

Fortbildungen

Datum	Ort	Thema/Referenten
07.05.2018- 09.05.2018	Dillingen	**Fortbildung 1** Unterrichtssequenz und fachliche Inhalte (Katharina Geldreich und Mike Talbot)
06.06.2018- 08.06.2018	Dillingen	**Fortbildung 1** Unterrichtssequenz und fachliche Inhalte (Katharina Geldreich und Mike Talbot)
12.12.2018- 14.12.2018	Dillingen	**Fortbildung 2** Erfahrungsaustausch und fachdidaktische Inhalte (Katharina Geldreich und Mike Talbot)
17.12.2018- 19.12.2018	Dillingen	**Fortbildung 2** Erfahrungsaustausch und fachdidaktische Inhalte (Katharina Geldreich und Mike Talbot)
21.11.2019- 22.11.2019	München	**Fortbildung 3** Fachdidaktische Inhalte und Fazit (Katharina Geldreich und Mike Talbot)

Schulbesuche

Insgesamt erfolgten 24 Schulbesuche im Zeitraum von Juli 2018 bis Juli 2019. Diese wurden allesamt von Katharina Geldreich, der Projektleiterin der TUM, koordiniert und durchgeführt.

Vorträge und Workshops über das Projekt

Das Projekt wurde von der Projektleitung auf verschiedenen Veranstaltungen in Vorträgen oder Workshops vorgestellt:

- Fachtag Digitale Bildung im Landkreis Mühldorf am Inn
- 7. Dillinger Schulleitertag – Leadership und Digitalisierung
- Jahrestagung des Oberbayerischen Seminars 2018
- Netzwerk-Konferenz CODE{affair} 2018
- 2. Symposium Informatisches Lernen zum Verstehen der digitalen Medienwelt

Darüber hinaus wurde das Projekt von den teilnehmenden Lehrkräften auf zahlreichen Schilfs, Workshops für Lehramtsanwärtern und Veranstaltungen zur digitalen Bildung im jeweiligen Regierungsbezirk vorgestellt.

Unser Dank gilt…

Allen, die uns bei der Entwicklung und Durchführung des Programmierzirkus und des Projekts „Algorithmen für Kinder“ unterstützt haben. Vor allem möchten wir an dieser Stelle (in chronologischer Reihenfolge) herzlich danken:

Unserem ehemaligen Team-Mitglied Alexandra Simon für die Mitarbeit am Programmierzirkus und den Publikationen dazu, Otto Lederer (damals MdL, jetzt Landrat in Rosenheim) für die wertvolle Unterstützung auf politischer Ebene, MdL Georg Eisenreich (damals Staatssekretär im Kultusministerium, jetzt Staatsminister der Justiz) für die Zusage der finanziellen Förderung durch das Ministerium, Mdgt. Walter Gremm und MR Matthias Stein für ihre engagierte Unterstützung des Projekts, MRin Maria Wilhelm für die hilfreiche Begleitung während der Umsetzung, dem ganzen Team der ALP, insbesondere dem Direktor Alfred Kotter, der Referentin Maria Stein und dem Referenten Uwe Geuder für die unbürokratische und unschätzbare Hilfe bei den Fortbildungen. Unserem Kollegen Mike Talbot sind wir sehr dankbar, dass er immer da war, wenn wir ihn gebraucht haben. Der ganz besondere Dank gilt natürlich allen Lehrerinnen und Lehrern, die am Projekt so engagiert und anhaltend mitgearbeitet, dafür auch viel Freizeit geopfert, wagemutige Unterrichtsexperimente begonnen und das Projekt schulintern bestens vertreten haben.

Abbildungen

Die meisten Abbildungen im vorliegenden Bericht sind im Rahmen des Projekts „Algorithmen für Kinder“ bzw. den Vorarbeiten der TUM entstanden. Die restlichen haben wir speziell für den Bericht angefertigt.

Die Fotografien zu Abb. 11, 14, 30, 31 und 37 wurden von Lehrkräften verschiedener Projektschulen zur Verfügung gestellt. Die restlichen Bilder haben die Autoren erstellt bzw. fotografiert.

Verschiedene Abbildungen enthalten Screenshots der Oberfläche von *Scratch* (CC-BY-SA-4.0 Creative Commons), eines Projekts der *Scratch Foundation* und der *Lifelong Kindergarten Group* am *MIT Media Lab*. *Scratch* steht kostenlos unter https://scratch.mit.edu zur Verfügung.

Abbildung 32 zeigt einen Screenshot der Oberfläche von *ScratchJr*, die im Rahmen einer Kooperation zwischen der *DevTech Research Group* der *Tufts University*, der *Lifelong Kindergarten Group* des *MIT Media Lab* und der *Playful Invention Company* entstanden ist. *ScratchJr* kann sowohl im *App Store* von *Apple* als auch im *Google Play Store* kostenlos heruntergeladen werden. Abbildung 38 enthält einen Screenshot der Oberfläche von *Microsoft MakeCode*, die kostenlos unter https://makecode.calliope.cc/ genutzt werden kann.

Die Verwendung einer dieser Abbildungen außerhalb des Abschlussberichts muss mit der TUMDDI abgesprochen werden.

Literatur

Aldorf, Anna-Maria (2016): Lehrerkooperation und die Effektivität von Lehrerfortbildung. Wiesbaden: Springer Fachmedien Wiesbaden.

Andre, Thomas; Whigham, Myrna; Hendrickson, Amy; Chambers, Sharon (1999): Competency beliefs, positive affect, and gender stereotypes of elementary students and their parents about science versus other school subjects. In: *J Res Sci Teach* 36 (6), S. 719–747. DOI: 10.1002/(SICI)1098-2736(199908)36:6<719::AID-TEA8>3.0.CO;2-R.

Bandura, Albert (1997): Self-efficacy: The exercise of control. New York: Worth Publishers.

Baum, Kevin; Kirsch, Nadine; Reese, Kerstin; Schmidt, Pascal, Wachter, Lukas; Wolf, Verena (2019): Informatikunterricht in der Grundschule? – Erprobung und Auswertung eines Unterrichtsmoduls mit Calliope mini. In: Arno Pasternak (Hg.): Informatik für alle. 18. GI-Fachtagung Informatik und Schule. INFOS. Gesellschaft für Informatik, S. 49–58.

Bell, Tim (2016): Demystifying Coding for Schools. What are we actually trying to teach? In: Jan Vahrenhold und Erik Barendsen (Hg.): Proceedings of the 11th Workshop in Primary and Secondary Computing Education. Münster, Germany, 13.-15.10.2016. New York, NY, USA: ACM.

Bell, Tim; Duncan, Caitlin (2018): Teaching Computing in Primary Schools. In: Sue Sentance, Erik Barendsen und Carsten Schulte (Hg.): Computer science education. Perspectives on teaching and learning in school. London, New York, Oxford, New Delhi, Sydney: Bloomsbury Academic.

Bell, Tim; Witten, Ian H.; Fellows, Mike (2015): CS Unplugged: An enrichment and extension programme for primary-aged students. 3. Aufl.

Bender, Elena; Hubwieser, Peter; Schaper, Niclas; Margaritis, Melanie; Berges, Marc; Ohrndorf, Laura et al. (2015): 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; Köster, Hilde; Magenheim, Johannes; Müller, Kathrin; Romeike, Ralf; Schroeder, Ulrik; Schulte, Carsten (2017): Zieldimensionen für frühe informatische Bildung im Kindergarten und in der Grundschule. In: Ira Diethelm (Hg.): Informatische Bildung zum Verstehen und Gestalten der digitalen Welt. INFOS. Bonn: Gesellschaft für Informatik, S. 15–24.

Berry, Miles (2015): QuickStart Primary Handbook. Swindon: BCS.

Berry, Miles (2018): "Computing" als neues Schulfach. In: *LOG IN* 189/190, S. 20–26.

Best, Alexander (2019): Bild der Informatik von Grundschullehrpersonen. Ergebnisse eines mehrjährigen Projekts zu informatikbezogenen Vorstellungen. In: Arno Pasternak (Hg.): Informatik für alle. 18. GI-Fachtagung Informatik und Schule. INFOS. Gesellschaft für Informatik, S. 59–68.

Beyer, Sylvia; Rynes, Kristina; Perrault, Julie; Hay, Kelly; Haller, Susan (2003): Gender Differences in Computer Science Students. In: Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education. Reno, NV, USA, 19.-23.02.2003. New York, NY, USA: ACM (SIGCSE '03), S. 49–53.

Bocconi, Stefania; Chioccariello, Augusto; Dettori, Giuliana; Ferrari, Anusca; Engelhardt, Katja (2016): Developing Computational Thinking in Compulsory Education. Implications for policy and practice. EUR: 28295 EN.

Brämer, Martin; Straube, Philipp; Köster, Hilde; Romeike, Ralf (2020): Eine digitale Perspektive für den Sachunterricht – ein Vorschlag zur Diskussion. In: *GDSU-Journal* (10).

Braught, Grant; Wahls, Tim; Eby, L. Marlin (2011): The Case for Pair Programming in the Computer Science Classroom. In: *ACM Trans. Comput. Educ.* 11 (1), S. 1–21. DOI: 10.1145/1921607.1921609.

Brennan, Karen (2013): Learning Computing through Creating and Connecting. In: *Computer* 46 (9), S. 52–59. DOI: 10.1109/MC.2013.229.

Brinda, Thorsten; Brügge, Niels; Diethelm, Ira; Knaus, Thomas; Kommer, Sven; Kopf, Christine et al. (2019): Frankfurt-Dreieck zur Bildung in der digital vernetzten Welt. Ein interdisziplinäres Modell. Online verfügbar unter https://www.keine-bildung-ohne-medien.de/frankfurter-dreieck/.

Brown, Neil C. C.; Sentance, Sue; Crick, Tom; Humphreys, Simon (2013): Restart: The Resurgence of Computer Science in UK Schools. In: *ACM Transactions on Computing Education* 1 (1).

Brown, Neil Christopher Charles; Kölling, Michael (2013): A tale of three sites. In: Beth Simon, Alison Clear und Quintin Cutts (Hg.): Proceedings of the ninth annual international ACM conference on International computing education research - ICER '13. the ninth annual international ACM conference. San Diego, San California, USA, 12.08.2013 - 14.08.2013. New York, New York, USA: ACM Press, S. 27.

Broy, Manfred (1998): Informatik Eine grundlegende Einführung. Berlin, Heidelberg: Springer Berlin Heidelberg.

Bundesministerium für Wirtschaft und Energie (2016): Digitale Bildung. Der Schlüssel zu einer Welt im Wandel. Berlin. Online verfügbar unter https://www.bmwi.de/Redaktion/DE/Publikationen/Digitale-Welt/digitale-bildung-der-schluessel-zu-einer-welt-im-wandel.html.

Calliope gGmbH (2020): Übersicht der Calliope-mini-Pilotschulen. Online verfügbar unter https://calliope.cc/schulen/pilotphase.

Caspersen, Michael E. (2018): Teaching Programming. In: Sue Sentance, Erik Barendsen und Carsten Schulte (Hg.): Computer science education. Perspectives on teaching and learning in school. London, New York, Oxford, New Delhi, Sydney: Bloomsbury Academic.

Chaudron, Stéphane (2015): Young Children (0-8) and digital technology. A qualitative exploratory study across seven countries. Luxembourg: Publications Office (EUR, Scientific and technical research series, 27052).

Condie, Rae; Munro, Bob (2006): The impact of ICT in school - a landscape review. Coventry: Becta.

Dagienė, Valentina; Jevsikova, Tatjana; Stupurienė, Gabrielė (2019): Introducing Informatics in Primary Education: Curriculum and Teachers' Perspectives. In: Sergei N. Pozdniakov und Valentina Dagienė (Hg.): Informatics in Schools. New Ideas in School Informatics. Cham: Springer International Publishing (11913), S. 83–94.

D-EDK - Deutschschweizer Erziehungsdirektoren-Konferenz: Lehrplan 21. Online verfügbar unter www.lehrplan.ch, zuletzt geprüft am 09.09.2019.

Devine, Mary; Houssemand, Claude; Meyers, Raymond (2013): Instructional Coaching for Teachers. A Strategy to Implement New Practices in the Classrooms. In: *Procedia - Social and Behavioral Sciences* 93, S. 1126–1130. DOI: 10.1016/j.sbspro.2013.10.001.

Diethelm, Ira; Schaumburg, Melanie (2016): IT2School – Development of Teaching Materials for CS Through Design Thinking. In: Andrej Brodnik und Françoise Tort (Hg.): Informatics in Schools: Improvement of Informatics Knowledge and Perception, Bd. 9973. Cham: Springer, S. 193–198.

Döbeli Honegger; Beat; Hielscher, Michael (2017): Vom Lehrplan zur LehrerInnenbildung - Erste Erfahrungen mit obligatorischer Informatikdidaktik für angehende Schweizer PrimarlehrerInnen. In: Ira Diethelm (Hg.): Informatische Bildung zum Verstehen und Gestalten der digitalen Welt. INFOS. Bonn: Gesellschaft für Informatik, S. 97–107.

Döbeli Honegger, Beat (2015): Digitale Kompetenzen von Lehrpersonen für den Lehrplan 21. Online verfügbar unter https://doebe.li/t17550.

Du Boulay, Benedict; O'Shea, T. I.M.; Monk, John (1999): The black box inside the glass box. Presenting computing concepts to novices. In: *International Journal of Human-Computer Studies* 51 (2), S. 265–277. DOI: 10.1006/ijhc.1981.0309.

Duncan, Caitlin; Bell, Tim; Atlas, James (2017): What do the Teachers Think? Introducing Computational Thinking in the Primary School Curriculum. In: Proceedings of the Nineteenth Australasian Computing Education Conference. Geelong, VIC, Australia, 31.01.-03.02.2017. New York, NY, USA: Academic Press, S. 65–74.

Duncan, Caitlin; Bell, Tim; Tanimoto, Steve (2014): Should your 8-year-old learn Coding? In: Proceedings of the 9th Workshop in Primary and Secondary Computing Education. Berlin, 05.-07.11.2014. New York, NY, USA: ACM, S. 60–69.

Euler, Manfred; Schüttler, Tobias; Hausamann, Dieter (2015): Schülerlabore: Lernen durch Forschen und Entwickeln. In: Ernst Kircher, Raimund Girwidz und Peter Häußler (Hg.): Physikdidaktik. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 759–782.

Falkner, Katrina; Vivian, Rebecca; Falkner, Nickolas (2014): The Australian digital technologies curriculum: challenge and opportunity. In: Jacqueline Whalley und Daryl D'Souza (Hg.): Proceedings of the Sixteenth Australasian Computing Education Conference. Auckland, New Zealand. New York: ACM.

Funke, Alexandra; Geldreich, Katharina (2017): Gender Differences in Scratch Programs of Primary School Children. In: Erik Barendsen (Hg.): Proceedings of the 12th Workshop on Primary and Secondary Computing Education. the 12th Workshop. Nijmegen, Netherlands, 08.11.2017 - 10.11.2017. New York, NY: ACM, S. 57–64.

Funke, Alexandra; Geldreich, Katharina; Hubwieser, Peter (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). Athens, Greece;, 25.-28.04.2017: IEEE, 1229-1236.

Garet, Michael S.; Porter, Andrew C.; Desimone, Laura; Birman, Beatrice F.; Yoon, Kwang Suk (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; Weitz, Katharina; Wolking, Maike; Schmid, Ute (2016): Computer science experimenter's kit for use in preschool and primary school. In: Jan Vahrenhold und Erik Barendsen (Hg.): Proceedings of the 11th Workshop in Primary and Secondary Computing Education. Münster, Germany, 13.-15.10.2016. New York, NY, USA: ACM, S. 66–71.

Gebauer, Susanne (2019): Praxisbezogene Beispiele vorschalten – den Theorie-Input nachschalten: Gestaltungsvarianten für Lehrer(innen)fortbildungen. In: Christian Donie, Frank Foerster, Marlene Obermayr, Anne Deckwerth, Gisela

Kammermeyer, Gerlinde Lenske et al. (Hg.): Grundschulpädagogik zwischen Wissenschaft und Transfer. Wiesbaden: Springer Fachmedien Wiesbaden, S. 162–168.

Geldreich, Katharina; Funke, Alexandra; Hubwieser, Peter (2016): A Programming Circus for Primary Schools. In: Andrej Brodnik und Françoise Tort (Hg.): Informatics in Schools: Improvement of Informatics Knowledge and Perception. Cham: Springer, S. 46–47.

Geldreich, Katharina; Funke, Alexandra; Hubwieser, Peter (2017): Willkommen im Programmierzirkus - Ein Programmierkurs für Grundschulen. In: Ira Diethelm (Hg.): Informatische Bildung zum Verstehen und Gestalten der digitalen Welt. 17. GI-Fachtagung Informatik und Schule ; 13.-15. September 2017 Oldenburg. Bonn: Gesellschaft für Informatik e.V. (GI) (GI-Edition - lecture notes in informatics (LNI) Proceedings, volume P-274), S. 327–334.

Geldreich, Katharina; Simon, Alexandra; Hubwieser, Peter (2019a): A Design-Based Research Approach for introducing Algorithmics and Programming to Bavarian Primary Schools. 53-75 Seiten / MediaEducation: Journal for Theory and Practice of Media Education, Heft 33: Medienpädagogik und Didaktik der Informatik. Eine Momentaufnahme disziplinärer Bezüge und schulpraktischer Entwicklungen. DOI: 10.21240/MPAED/33/2019.02.15.X.

Geldreich, Katharina; Simon, Alexandra; Starke, Elena (2019b): Which Perceptions Do Primary School Children Have about Programming? In: Unknown (Hg.): Proceedings of the 14th Workshop in Primary and Secondary Computing Education on - WiPSCE'19. the 14th Workshop in Primary and Secondary Computing Education. Glasgow, Scotland, Uk, 23.10.2019 - 25.10.2019. New York, New York, USA: ACM Press, S. 1–7.

Geldreich, Katharina; Talbot, Mike; Hubwieser, Peter (2019c): Aufgabe ist nicht gleich Aufgabe – Vielfältige Aufgabentypen bewusst in Scratch einsetzen. In: Arno Pasternak (Hg.): Informatik für alle. 18. GI-Fachtagung Informatik und Schule. INFOS. Gesellschaft für Informatik, S. 181–190.

Gesellschaft für Didaktik des Sachunterrichts (Hg.) (2013): Perspektivrahmen Sachunterricht. Vollständig überarbeitete und erweiterte Ausgabe. Bad Heilbrunn: Klinkhardt.

GI - Gesellschaft für Informatik e.V. (2016a): Dagstuhl-Erklärung. Bildung in der digitalen vernetzten Welt. Berlin.

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. (2016b): Bildungsstandards Informatik für die Sekundarstufe II. In: *LOG IN* 36 (183/184).

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).

Goecke, Lennart; Stiller, Jurik (2018): Informatische Phänomene und Sachunterricht. Beispiele für vielperspektivischen Umgang mit einem Einplatinencomputer. In: Marco Thomas und Michael Weigend (Hg.): Informatik und Medien: 8. Münsteraner Workshop zur Schulinformatik. Münster: Books on Demand.

Goode, Joanna; Margolis, Jane; Chapman, Gail (2014): Curriculum is not enough. In: J. D. Dougherty (Hg.): SIGCSE'14. Proceedings of the 45th ACM Technical Symposium on Computer Science Education ; March 5 - 8, 2014, Atlanta, Georgia, USA. the 45th ACM technical symposium. Atlanta, Georgia, USA, 05.03.2014 - 08.03.2014. Association for Computing Machinery; ACM Technical Symposium on Computer Science Education; SIGCSE. New York, NY: ACM, S. 493–498.

Google Inc. & Gallip 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. Online verfügbar unter https://goo.gl/iM5g3A.

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, UK, 18.07.2017-20.07.2017: IEEE.

Grover, Shuchi; Pea, Roy (2018): Computational Thinking: A Competency whose Time has come. In: Sue Sentance, Erik Barendsen und Carsten Schulte (Hg.): Computer science education. Perspectives on teaching and learning in school. London, New York, Oxford, New Delhi, Sydney: Bloomsbury Academic, S. 19–38.

Haselmeier, Kathrin (2019): Informatik in der Grundschule – Stellschraube Lehrerbildung. In: Arno Pasternak (Hg.): Informatik für alle. 18. GI-Fachtagung Informatik und Schule. INFOS. Gesellschaft für Informatik, S. 89–98.

Hassinen, Marko; Mäyrä, Hannu (2006): Learning Programming by Programming: a Case Study. In: Anders Berglund und Mattias Wigbberg (Hg.): Proceedings KolliCalling, S. 117–119.

Hattie, John (2010): Visible learning. A synthesis of over 800 meta-analyses relating to achievement. Reprinted. London: Routledge.

HdkF - Stiftung „Haus der kleinen Forscher“ (Hg.) (2018): Frühe informatische Bildung – Ziele und Gelingensbedingungen für den Elementar- und Primarbereich. Unter Mitarbeit von Nadine Bergner, Hilde Köster, Kathrin Müller, Johannes Magenheim, Ulrik Schröder, Ralf Romeike und Carsten Schulte. Berlin: Verlag Barbara Budrich.

Heintz, Fredrik; Mannila, Linda; Farnqvist, Tommy (2016): A review of models for introducing computational thinking, computer science and computing in K-12 education. In: 2016 IEEE Frontiers in Education Conference (FIE). 2016 IEEE Frontiers in Education Conference (FIE). Erie, PA, USA, 12.10.2016 - 15.10.2016: IEEE, S. 1–9.

Heintz, Fredrik; Mannila, Linda; Nordén, Lars-Ake; Parnes, Peter; Regnell, Björn (2017): Introducing Programming and Digital Competence in Swedish K-9 Education. In: Valentina Dagienė und Arto Hellas (Hg.): Informatics in Schools. Focus on Learning Programming. Cham: Springer International Publishing (10696), S. 117–128.

Hellmich, Frank (2011): Selbstkonzepte im Grundschulalter. Modelle, empirische Ergebnisse, pädagogische Konsequenzen. s.l.: W. Kohlhammer Verlag. Online verfügbar unter http://www.content-select.com/index.php?id=bib_view&ean=9783170228631.

Hermans, Felienne; Aivaloglou, Efthimia (2017): To Scratch or not to Scratch? In: Erik Barendsen und Peter Hubwieser (Hg.): Proceedings of the 12th Workshop on Primary and Secondary Computing Education. New York, NY, USA: ACM, S. 49–56.

Hill, Heather; Ball, Deborah; Schilling, Stephen (2008): Unpacking pedagogical content knowledge: Conceptualizing and measuring teachers' topic-specific knowledge of students. In: *Journal for Research in Mathematics Education* 39 (4), S. 372–400.

HITSA - Information Technology Foundation for Education (2015): ProgeTiger Programme 2015–2017.

HITSA - Information Technology Foundation for Education (2020): ProgeTiger Programme. Online verfügbar unter https://www.hitsa.ee/it-education/educational-programmes/progetiger.

Hubwieser, Peter (2007): Didaktik der Informatik. Berlin, Heidelberg: Springer Berlin Heidelberg.

Hubwieser, Peter; Magenheim, Johannes; Mühling, Andreas; Ruf, Alexander (2013): Towards a conceptualization of pedagogical content knowledge for computer science. In: Beth Simon, Alison Clear und Quintin Cutts (Hg.): ICER '13. Proceedings of the 2013 ACM Conference on International Computing Education Research. New York: ACM.

ISB - Staatsinstitut für Schulqualität und Bildungsforschung München (2017): Kompetenzrahmen zur Medienbildung an bayerischen Schulen.

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.

Jerusalem, Matthias; Drössler, Stephanie; Kleine, Dietmar; Klein-Heßling, Johannes; Mittag, Waldemar; Röder, Bettina (2009): Förderung von Selbstwirksamkeit und Selbstbestimmung im Unterricht. Skalen zur Erfassung von Lehrer- und Schülermerkmalen. Berlin: Humboldt-Universität zu Berlin.

K–12 Computer Science Framework (2016). Online verfügbar unter http://www.k12cs.org.

Kabátová, Martina; Kalas, Ivan; Tomcsányiová, Monika (2016): Programming in Slovak Primary Schools. In: *IOI* 10 (1), S. 125–159. DOI: 10.15388/ioi.2016.09.

Kellow, Jan-Marie (2018): Digital Technologies in the New Zealand Curriculum. In: *WJE* 23 (2). DOI: 10.15663/wje.v23i2.656.

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.

Kirkpatrick, Donald L. (1996): Evaluating training programs. The four levels. San Francisco: BK Berrett-Koehler.

Klafki, Wolfgang (2007): Neue Studien zur Bildungstheorie und Didaktik. Zeitgemäße Allgemeinbildung und kritisch-konstruktive Didaktik. s.l.: Beltz Verlagsgruppe. Online verfügbar unter http://www.content-select.com/index.php?id=bib_view&ean=9783407291493.

Kleickmann, Thilo; Möller, Kornelia (2007): Haben Lehrerfortbildungen einen Effekt auf Lernzuwächse bei Schülerinnen und Schülern? In: Dietmar Höttecke (Hg.): Naturwissenschaftlicher Unterricht im internationalen Vrrgleich. Münster: LIT Verlag, S. 506–508.

KMK (2016): Strategie der Kultusministerkonferenz zur Bildung in der digitalen Welt. Beschluss der Kultusministerkonferenz vom 08.12.2016. Hg. v. Sekretariat der Kultusministerkonferenz.

Kuhl, Maria (2008): Studienkultur Informatik neu denken. Geschlechterkonstruktionen im Informatikstudium an der Universität Dortmund und der Carnegie Mellon University. Zugl.: Dortmund, Univ., Diss, 2007. Aachen: Shaker (Soziologische Studien).

Kwon, Sei; Schroderus, Katri (2017): Coding in Schools: Comparing Integration of Programming into Basic Education Curricula of Finland and South Korea: Finnish Society on Media Education.

Lahtinen, Essi; Ala-Mutka, Kirsti; Järvinen, Hannu-Matti (2005): A study of the difficulties of novice programmers. In: José Cunha, William Fleischman, Viera K. Proulx und João Lourenço (Hg.): Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education. New York, New York, USA: ACM Press, S. 14.

Lamprou, Anna; Repenning, Alexander (2018): Teaching how to teach computational thinking. In: Irene Polycarpou und A. Special Interest Group on Computer ScienceC.M. Education (Hg.): Proceedings of the 23rd Annual ACM Conference

on Innovation and Technology in Computer Science Education. the 23rd Annual ACM Conference. Larnaca, Cyprus, 02.07.2018 - 04.07.2018. [S.l.]: ACM, S. 69–74.

Lave, Jean; Wenger, Etienne (1991): Situated Learning: Legitimate Peripheral Participation. Cambridge: Cambridge University Press.

Lee, Irene; Martin, Fred; Denner, Jill; Coulter, Bob; Allan, Walter; Erickson, Jeri et al. (2011): Computational thinking for youth in practice. In: *ACM Inroads* 2 (1), S. 32. DOI: 10.1145/1929887.1929902.

Leifheit, Luzia; Tsarava, Katerina; Ninaus, Manuel; Ostermann, Klaus; Golle, Jessika; Trautwein, Ulrich; Moeller, Korbinian (2020): SCAPA: Development of a Questionnaire Assessing Self-Concept and Attitudes Toward Programming. In: Michail Giannakos und Guttorme Sindre (Hg.): Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education. New York, NY, USA: ACM, 138-144.

Lipowsky, Frank (2010): Lernen im Beruf. Empirische Befunde zur Wirksamkeit von Lehrerfortbildung. In: F. H. Müller, A. Eichenberger, M. Lüders und J. Mayr (Hg.): Lehrerinnen und Lehrer lernen. Konzepte und Befunde der Lehrerfortbildung. Münster: Waxmann, S. 51–70.

Lipowsky, Frank; Rzejak, Daniela (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; Mooney, Aidan (2017): Computational Thinking in Education: Where does it fit? A systematic Literary Review. Maynooth, Ireland: Maynooth University.

Lorenz, Ramona; Bos, Wilfried; Endberg, Manuela; Eickelmann, Birgit; Grafe, Silke; Vahrenhold, Jan (Hg.) (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, Ne York: Waxmann.

Lowe, Tony A. (2018): Misconceptions and the Notional Machine in Very Young Programming Learners. In: *School of Engineering* (Paper 76).

Luxton-Reilly, Andrew; Sheard, Judy; Szabo, Claudia; Simon; Albluwi, Ibrahim; Becker, Brett A. et al. (2018): Introductory programming. A systematic literature review. In: Guido Rößling (Hg.): Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education. New York NY: ACM, S. 55–106.

Magenheim, Johannes; Schulte, Carsten; Schroeder, Ulrik; Humbert, Ludger; Müller, Kathrin; Bergner, Nadine; Fricke, Martin (2018): Das Projekt Informatik an Grundschulen. In: *LOG IN Informatische Bil-dung und Computer in der Schule* (189/190), S. 57–66.

Makris, Dimosthenis; Euaggelopoulos, Kleomenis; Chorianopoulos, Konstantinos; Giannakos, Michail (2013): Could you help me to change the variables? Comparing instruction to encouragement for teaching programming. In: Michael E. Caspersen, Maria Knobelsdorf und Ralf Romeike (Hg.): WiPSCE 2013. The 8th Workshop in Primary and Secondary Computing EducationNew York, New York, USA: ACM Press.

Margolis, Jane; Fisher, Allan (2002): Unlocking the Clubhouse: Women in Computing. Cambridge, Mass.: MIT Press.

Marsh, Herbert W.; Craven, Rhonda G. (2006): Reciprocal Effects of Self-Concept and Performance From a Multidimensional Perspective. Beyond Seductive Pleasure and Unidimensional Perspectives. In: *Perspect Psychol Sci* 1 (2), S. 133–163. DOI: 10.1111/j.1745-6916.2006.00010.x.

Marsh, Herbert W.; Martin, Andrew J. (2011): Academic self-concept and academic achievement. Relations and causal ordering. In: *The British journal of educational psychology* 81 (Pt 1), S. 59–77. DOI: 10.1348/000709910X503501.

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

Medienberatung NRW (2020): Medienkompetenzrahmen NRW. Münster/Düsseldorf. Online verfügbar unter https://medienkompetenzrahmen.nrw/fileadmin/pdf/LVR_ZMB_MKR_Broschuere.pdf.

MFS - Medienpädagogischer Forschungsverbund Südwest (2019): KIM-Studie 2018: Kindheit, Internet, Medien. Basisstudie zum Medienumgang 6- bis 13-Jähriger in Deutschland. Stuttgart.

Meerbaum-Salant, Orni; Armoni, Michal; Ben-Ari, Mordechai (2010): Learning Computer Science Concepts with Scratch. In: Michael E. Caspersen (Hg.): Proceedings of the Sixth international workshop on Computing education research. ACM Special Interest Group on Computer Science Education. New York, NY: ACM.

Miller, Susanne; Velten, Katrin (2015): Kinderstärkende Pädagogik in der Grundschule. 1. Aufl. Stuttgart: Kohlhammer Verlag (KinderStärken, Band 6). Online verfügbar unter http://gbv.eblib.com/patron/FullRecord.aspx?p=4340978.

Möller, Jens; Köller, Olaf (2004): Die Genese akademischer Selbstkonzepte. In: *Psychologische Rundschau* 55 (1), S. 19–27. DOI: 10.1026/0033-3042.55.1.19.

Moreno-Leon, Jesus; Roman-Gonzalez, Marcos; Robles, Gregorio (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.

Müller, Dorothee (2015): Informatikunterricht und Informatikselbstkonzept. Online verfügbar unter https://docplayer.org/10471394-Informatikunterricht-und-informatikselbstkonzept.html, zuletzt geprüft am 09.09.2019.

Müller, Kathrin; Schulte, Carsten; Magenheim, Johannes (2019): Zur Relevanz eines Prozessbereiches Interaktion und Exploration im Kontext informatischer Bildung im Primarbereich. In: Arno Pasternak (Hg.): Informatik für alle. 18. GI-Fachtagung Informatik und Schule. INFOS. Gesellschaft für Informatik, S. 139–148.

Murmann, Lydia; Schelhowe, Heidi; Bockermann, Iris; Engelbertz, Simon; Illginnis, Saskia; Moebus, Antje (2018): Calliope mini: Eine Explorationsstudie im pädagogisch-didaktischen Kontext - Abschlussbericht. Online verfügbar unter https://media.suub.uni-bremen.de/bitstream/elib/3457/1/00106848-1.pdf.

Nelson, Beryl (2014): The data on diversity. In: *Communications of the ACM* 57 (11), S. 86–95. DOI: 10.1145/2597886.

Pajares, Frank (2005): Gender Differences in Mathematics Self-Efficacy Beliefs. In: Ann M. Gallagher und James C. Kaufman (Hg.): Gender differences in mathematics. An integrative psychological approach. Cambridge, UK, New York: Cambridge University Press, S. 294–315.

Papert, Seymour; Harel, Idit (1991): Situating Constructionism. In: Seymour Papert und Idit Harel (Hg.): Constructionism.

Penuel, William R.; Gallagher, Lawrence P.; Moorthy, Savitha (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.

Peyton Jones, Simon; Muuß-Meerholz, Jöran (2014): Schulfach "Computing" ab Klasse 1. In: *c't* 14, S. 110–111.

Quigley, Claire (2017): Working towards a better gender balance in CoderDojo Scotland clubs.

Reding, Tracie Evans; Dorn, Brian (2017): Understanding the "Teacher Experience" in Primary and Secondary CS Professional Development. In: Josh Tenenberg und A. Special Interest Group on Computer ScienceC.M. Education (Hg.): Proceedings of the 2017 ACM Conference on International Computing Education Research. the 2017 ACM Conference. Tacoma, Washington, USA, 18.08.2017 - 20.08.2017. [S.l.]: ACM, S. 155–163.

Reese, Kerstin; Wolf, Verena (2018): Calliope mini in der Lehrerfortbildung. In: *LOG IN* (189/190), S. 108–111.

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.

Resnick, Mitchel; Rusk, Natalie; Cooke, Stina (1998): The Computer Clubhouse: Technological Fluency in the Inner City. In: Bishwapriya Sanyal, Donald A. Schön und William John Mitchell (Hg.): High technology and low-income communities. Prospects for the positive use of advanced information technology. Cambridge, Mass: MIT Press, S. 263–285.

Resnick, Mitchel; Silverman, Brian; Kafai, Yasmin; Maloney, John; Monroy-Hernández, Andrés; Rusk, Natalie et al. (2009): Scratch: Programming for All. In: *Commun. ACM* 52 (11), S. 60. DOI: 10.1145/1592761.1592779.

Rich, Kathryn M.; Strickland, Carla (2020): Testing and Debugging. In: Shuchi Grover (Hg.): Computer Science in K-12: An A-to-Z Handbook on Teaching Programming. Palo Alto, CA: Edfinity, S. 211–218.

Richter, Eric; Marx, Alexandra; Huang, Yizhen; Richter, Dirk (2020): Zeiten zum beruflichen Lernen. Eine empirische Untersuchung zum Zeitpunkt und der Dauer von Fortbildungsangeboten für Lehrkräfte. In: *Z Erziehungswiss* 23 (1), S. 145–173. DOI: 10.1007/s11618-019-00924-x.

Richter, Tobias; Naumann, Johannes; Horz, Holger (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.

Riefling, Markus; Fandrich, Anatolij; Diethelm, Ira (2020): Mit IT2School die digitale Welt analog und digital entdecken. In: Anabel Ternès von Hattburg und Matthias Schäfer (Hg.): Digitalpakt – was nun? Wiesbaden: Springer Fachmedien Wiesbaden, S. 293–300.

Romeike, Ralf; Reichert, Dominik (2011): PicoCrickets als Zugang zur Informatik in der Grundschule. In: Marco Thomas (Hg.): Informatik in Bildung und Beruf. 14. GI-Fachtagung "Informatik und Schule - INFOS 2011", 12. - 15. September 2011 an der Westfälischen Wilhelms-Universität Münster. Bonn: Ges. für Informatik (GI-edition Proceedings, 189), S. 177–186.

Roth, Kathleen J.; Garnier, Helen E.; Chen, Catherine; Lemmens, Meike; Schwille, Kathleen; Wickler, Nicole I.Z. (2011): Videobased lesson analysis. Effective science PD for teacher and student learning. In: *J. Res. Sci. Teach.* 48 (2), S. 117–148. DOI: 10.1002/tea.20408.

Ruf, A., Berges, M., Hubwieser, P. (2015). Classification of Programming Tasks According to Required Skills and Knowledge Representation. In Informatics in Schools. Curricula, Competences, and Competitions. 8th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives. Springer, Heidelberg, 57–68. DOI: doi.org/10.1007/978-3-319-25396-1_6

Sabitzer, Barbara; Antonitsch, Peter K.; Pasterk, Stefan (2014): Informatics concepts for primary education. In: Proceedings of the 9th Workshop in Primary and Secondary Computing Education. Berlin, 05.-07.11.2014. New York, NY, USA: ACM, S. 108–111.

Sands, Philip (2019): Addressing cognitive load in the computer science classroom. In: *ACM Inroads* 10 (1), S. 44–51. DOI: 10.1145/3210577.

Schmid, Ute; Gärtig-Daugs, Anja (2018): Notwendigkeit der Integration elementarinformatischer Lerneinheiten in den Vor- und Grundschulunterricht. In: *MedienPädagogik* 31, S. 78–106. DOI: 10.21240/mpaed/31/2018.03.29.X.

Schneekloth, Urlich; Pupeter, Monika (2010): Wohlbefinden, Wertschätzung, Selbstwirksamkeit: Was Kinder für ein gutes Leben brauchen. In: World Vision Deutschland e. V. (Hg.): Kinder in Deutschland 2010. 2. World Vision Kinderstudie. Orig.-Ausg. Frankfurt am Main: Fischer (Fischer, 18640), S. 187–221.

Schwarzer, Ralf; Jerusalem, Matthias (2002): Das Konzept der Selbstwirksamkeit. In: Matthias Jerusalem und Diether Hopf (Hg.): Selbstwirksamkeit und Motivationsprozesse in Bildungsinstitutionen. Weinheim: Beltz (Zeitschrift für Pädagogik Beiheft, 44), S. 28–53.

Schwill, Andreas (1993): Fundamentale Ideen der Informatik. In: *Zentralblatt für Didaktik der Mathematik* 25 (1), S. 20–31.

Sentance, Sue; Csizmadia, Andrew (2017): Computing in the curriculum. Challenges and strategies from a teacher's perspective. In: *Educ Inf Technol* 22 (2), S. 469–495. DOI: 10.1007/s10639-016-9482-0.

Sentance, Sue; Dorling, Mark; McNirol, Adam (2013): Computer Science in Secondary Schools in the UK: Ways to Empower Teachers. In: Ira Diethelm und Roland T. Mittermeir (Hg.): Informatics in Schools. Sustainable Informatics Education for Pupils of all Ages. 6th International Conference on Informatics in Schools: Situation, Evolution and Perspectives: Springer.

Sentance, Sue; Humphreys, Simon (2015): Online vs Face-To-Face Engagement of Computing Teachers for their Professional Development Needs. In: Andrej Brodnik und Jan Vahrenhold (Hg.): Informatics in Schools. Curricula, Competences, and Competitions. 8th International Conference on Informatics in Schools: Ljubljana, Slovenia, September 28 – October 1. Cham: Springer, S. 69–81.

Sentance, Sue; Waite, Jane (2017): PRIMM: Exploring pedagogical approaches for teaching text-based programming in school. In: Erik Barendsen und Peter Hubwieser (Hg.): Proceedings of the 12th Workshop on Primary and Secondary Computing Education. Nijmegen, Netherlands, 08.-10.11.2017. New York, NY, USA: ACM.

Shavelson, Richard J.; Hubner, Judith J.; Stanton, George C. (1976): Self-Concept. Validation of Construct Interpretations. In: *Review of Educational Research* 46 (3), S. 407–441. DOI: 10.3102/00346543046003407.

Simon, Alexandra; Geldreich, Katharina; Hubwieser, Peter (2019): How to Transform Programming Processes in Scratch to Graphical Visualizations. In: Unknown (Hg.): Proceedings of the 14th Workshop in Primary and Secondary Computing Education on - WiPSCE'19. the 14th Workshop in Primary and Secondary Computing Education. Glasgow, Scotland, Uk, 23.10.2019 - 25.10.2019. New York, New York, USA: ACM Press, S. 1–9.

Smith, Neil; Allsop, Yasemin; Caldwell, Helen; Hill, David; Dimitriadi, Yota; Csizmadia, Andrew Paul (2015): Master Teachers in Computing: What have we achieved? In: Proceedings of the Workshop in Primary and Secondary Computing Education. London, United Kingdom, 09.-11-11-2015. New York, NY, USA: ACM, S. 21–24.

Smith, Neil; Sutcliffe, Clare; Sandvik, Linda (2014): Code Club: Bringing Programming to UK Primary Schools through Scratch. In: J. D. Dougherty (Hg.): SIGCSE'14. Proceedings of the 45th ACM Technical Symposium on Computer Science Education ; New York, NY: ACM, S. 517–522.

SMK Sachen - Staatsministerium für Kultus Sachsen (2019): Lehrplan Grundschule Werken. Online verfügbar unter http://www.bildung.sachsen.de/apps/lehrplandb/.

Statter, David; Armoni, Michal (2020): Teaching Abstraction in Computer Science to 7 th Grade Students. In: *ACM Trans. Comput. Educ.* 20 (1), S. 1–37. DOI: 10.1145/3372143.

StMBKWK - Bayerisches Staatsministerium für Bildung und Kultus, Wissenschaft und Kunst (StMBKWK) (2014): LehrplanPLUS Grundschule. Lehrplan für die bayerische Grundschule. München: StMBKWK.

Storte, Davide; Webb, Mary; Bottino, Rosa Maria; Passey, Don; Kalas, Ivan; Bescherer, Christine et al. (2019): Coding, Programming and the Changing Curriculum for Computing in Schools. Report of UNESCO/IFIP TC3 Meeting at OCCE. Online verfügbar unter https://www.ifip-tc3.org/app/download/7193588451/UNESCO+meeting+at+OCCE+2018+report+final+edit+301019.pdf?t=1572465399.

Straube, Philipp; Brämer, Martin; Köster, Hilde; Romeike, Ralf (2018): Eine digitale Perspektive für den Sachunterricht? Fachdidaktische Überlegungen und Implikationen. In: *Widerstreit Sachunterricht* 24.

Sysło, Maciej M.; Kwiatkowska, Anna Beate (2015): Introducing a New Computer Science Curriculum for All School Levels in Poland. In: Andrej Brodnik und Jan Vahrenhold (Hg.): Informatics in Schools. Curricula, Competences, and Competitions. 8th International Conference on Informatics in Schools: Ljubljana, Slovenia, September 28 – October 1. Cham: Springer, S. 141–154.

Tausch, Reinhard; Tausch, Anne-Marie (1998): Erziehungspsychologie. Begegnung von Person zu Person. 11., korrigierte Aufl. Göttingen: Hogrefe Verl. für Psychologie. Online verfügbar unter http://www.sub.uni-hamburg.de/ebook/ebook.php?act=b&cid=527.

Tsai, Meng-Jung; Wang, Ching-Yeh; Hsu, Po-Fen (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; Moeller, Korbinian; Ninaus, Manuel (2018): Training Computational Thinking through board games. The case of Crabs & Turtles. In: *IJSG* 5 (2), S. 25–44. DOI: 10.17083/ijsg.v5i2.248.

van Blommenstein, Michiel (2016): Poland wants its kids to code - but time is against them. Hg. v. CBS Interactive. Online verfügbar unter https://www.zdnet.com/article/poland-wants-its-kids-to-code-but-time-is-against-them/.

Vigerske, Stefanie (2017): Transfer von Lehrerfortbildungsinhalten in die Praxis. Wiesbaden: Springer Fachmedien Wiesbaden.

Vivian, Rebecca; Falkner, Katrina; Falkner, Nickolas (2014a): Addressing the challenges of a new digital technologies curriculum. MOOCs as a scalable solution for teacher professional development. In: *Research in Learning Technology* 22. DOI: 10.3402/rlt.v22.24691.

Vivian, Rebecca; Falkner, Katrina; Szabo, Claudia (2014b): Can everybody learn to code? In: Päivi Kinnunen und Simon (Hg.): Proceedings of the 14th Koli Calling International Conference on Computing Education Research. New York, NY, USA: ACM, S. 41–50.

Waite, Jane; Grover, Shuchi (2020): Worked Examples & Other Scaffolding Strategies. In: Shuchi Grover (Hg.): Computer Science in K-12: An A-to-Z Handbook on Teaching Programming. Palo Alto, CA: Edfinity, S. 240–249.

Waite, Jane Lisa; Curzon, Paul; Marsh, William; Sentance, Sue; Hadwen-Bennett, Alex (2018): Abstraction in action. K-5 teachers' uses of levels of abstraction, particularly the design level, in teaching programming. In: *IJCSES* 2 (1), S. 14–40. DOI: 10.21585/ijcses.v2i1.23.

Webb, Mary; Davis, Niki; Bell, Tim; Katz, Yaacov J.; Reynolds, Nicholas; Chambers, Dianne P.; Sysło, Maciej M. (2017): Computer science in K-12 school curricula of the 2lst century. Why, what and when? In: *Educ Inf Technol* 22 (2), S. 445–468. DOI: 10.1007/s10639-016-9493-x.

Webb, Paul (1992): Primary science teachers' understandings of electric current. In: *International Journal of Science Education* 14 (4), S. 423–429.

Wenger, Etienne (1998): Communities of practice: Learning, meaning and identity. Cambridge: Cambridge University Press.

Wing, Jeannette M. (2006): Computational thinking. In: *Commun. ACM* 49 (3), S. 33. DOI: 10.1145/1118178.1118215.

Wolking, Maike; Schmid, Ute (2017): Mental Models, Career Aspirations, and the Acquirement of Basic Concepts of Computer Science in Elementary Education. In: Erik Barendsen und Peter Hubwieser (Hg.): Proceedings of the 12th Workshop on Primary and Secondary Computing Education. New York, NY, USA: ACM, S. 119–120.

Wong, Gary K.W.; Jiang, Shan (2018): Computational Thinking Education for Children: Algorithmic Thinking and Debugging. In: 2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE). IEEE, S. 328–334.

Worth, Karen (1999): The Power of Children's Thinking. In: National Research Council (Hg.): Inquiry: Thoughts, Views, and Strategies for the K-5 classroom: National Science Foundation, S. 25–31.

Xuequan, Mu (2018): Lithuania to introduce computer science lessons for primary schoolchildren. Hg. v. Xinhuanet. Online verfügbar unter http://www.xinhuanet.com/english/2018-08/28/c_137423534.htm, zuletzt geprüft am 16.06.2020.

Yadav, Aman; Mayfield, Chris; Zhou, Ninger; Hambrusch, Susanne; Korb, John T. (2014): Computational Thinking in Elementary and Secondary Teacher Education. In: *Trans. Comput. Educ.* 14 (1), S. 1–16. DOI: 10.1145/2576872.

Technische Universität München
TUM School of Education
Professur für Didaktik der Informatik
Prof. Dr. Peter Hubwieser, Katharina Geldreich

Arcisstraße 21
80333 München

www.edu.tum.de/ddi/