Algorithmisieren im Grundschulalter

Sabrina Hoffmann[1], Katharina Wendlandt[2], Matthias Wendlandt[3]

Abstract: Informatik, insbesondere das Programmieren, wird zunehmend in Form von Projekten in Grundschulen in Deutschland thematisiert. Eine der wesentlichen Schlüsselkompetenzen zum erfolgreichen Programmieren ist das Algorithmisieren von Problemlösungen. In der vorliegenden Arbeit wird zunächst ein Stufenmodell des Algorithmisierens entworfen und darauf aufbauend ein Testverfahren zur Beurteilung der Fähigkeit zum Algorithmisieren gewonnen. Basierend auf diesem Testverfahren wurden 155 Grundschulkinder aus acht Grundschulklassen der Schulstufen 2-4 bezüglich ihrer Kompetenz zum Algorithmisieren untersucht. Die zugrundeliegende Fragestellung war, inwiefern sich die Fähigkeit zum Algorithmisieren in den Schulstufen 2-4 entwickelt. Mithilfe des Exakten Fisher-Tests konnten signifikante Unterschiede in vier verschiedenen Stufen der Fähigkeit zum Algorithmisieren zwischen den Klassenstufen nachgewiesen werden. Dabei fiel auf, dass der Unterschied zwischen der zweiten und dritten Klasse deutlicher ausfiel als zwischen der dritten und vierten Klasse. Die vorliegende Arbeit stellt den ersten Teil einer zweigeteilten Untersuchung dar. Der zweite Teil der Untersuchung wird sich mit der Trainierbarkeit des Algorithmisierens im Grundschulalter beschäftigen.

1 Einleitung

Informatikunterricht trägt in vielfältiger Weise zur kognitiven Entwicklung, der persönlichen Entfaltung und zu einem selbstbestimmten Handeln eines jeden Schülers/ einer jeden Schülerin bei. Informatische Kompetenzen wie das Algorithmisieren, das Modellieren und die Problemlösefähigkeit bilden heutzutage grundlegende Fähigkeiten, die in vielen Situationen auch unabhängig von Informatik verwendet werden können. Notwendiges Wissen über Datenschutz, Datensicherheit und Grenzen des Computers befähigt die SuS dazu, ein selbstbestimmtes Leben zu führen. Deshalb ist es eine Kernfrage, ob und inwieweit Informatik schon in der Grundschule angeboten werden kann. Diese Frage wurde bereits in einigen Projekten angegangen.

Mit der Zauberschule [BLS11] zeigen die Autoren einige interessante Ansätze zum Thematisieren von Informatik in der Grundschule. Die Themen unterteilten sich in Binärzahlen, Bilddarstellung und Fehlererkennung. Die Autoren berichten, dass es den Kindern viel Spaß gemacht habe. In [Sc01] konnte aus psychologischen Studien gefolgert werden, dass Grundschulkinder in der Lage seien, gewisse fundamentale Ideen der Informatik altersgerecht zu verstehen und anzuwenden. In [BD09] wird ein Projekt im Rahmen einer Arbeitsgemeinschaft mit Kindern der vierten Klasse beschrieben, welches das Thema Programmieren durch die drei Programmierumgebungen *Karol*, *Scratch* und *Lego Mindstoms* thematisiert. Die Reflexion des Projekts zeigt, dass *Karol* und *Lego Mindstoms* für einen

[1] Rotebergschule Dillenburg, Rühlstraße 3, 35683 Dillenburg, Deutschland
[2] Mathematisches Institut, Universität Giessen, Arndtstr. 2, 35392 Giessen, Deutschland
[3] Institut für Informatik, Universität Giessen, Arndtstr. 2, 35392 Giessen, Deutschland

solchen Kurs weniger geeignet sind, wohingegen *Scratch* eine gute Möglichkeit darstellt, Kinder dieser Altersstufe erste Programmiererfahrungen sammeln zu lassen. Ein weiteres Projekt, welches das Programmieren in Scratch [GW16] in der Grundschule thematisiert, wurde von der Hochschule Hannover in einer zweiten Grundschulklasse durchgeführt. Auch hierbei kommen die Autoren zu dem Schluß, dass „Grundschulkinder der zweiten Klasse in der Lage sind, Programmierkonzepte zu erlernen".

Die oben genannten Publikationen, bzw. Projekte, resultieren alle in dem Ergebnis, dass Grundschulkinder in der Lage sind zu programmieren. Diese Aussage erscheint gerade in der zweiten Klasse überraschend, da entwicklungspsychologische Aussagen dies nicht vermuten lassen würden und es stellt sich die Frage, *inwieweit* Grundschulkinder fähig sind zu programmieren. Nach Untersuchungen von Piaget [SL12] sind Kinder im präoperationalen Stadium (2-7 Jahre) nicht imstande, geistige Operationen vollständig durchzuführen. Diese Fähigkeit entwickelt sich erst im Verlauf des konkret-operationalen Stadiums (7-12 Jahre). Da das Grundschulalter die Zeit zwischen dem 6. und dem 10. Lebensjahr umfasst, sind grundlegende geistige Entwicklungsprozesse im Gange. Auch Untersuchungen mathematischer Kompetenzen im Grundschulalter bestätigen den Entwicklungsprozess. Erklärungen finden sich in der Fähigkeit Repräsentationsformen beziehungsweise Methoden zu kennen und nutzen zu können, beziehungsweise in den vorhandenen Fähigkeiten des Arbeitsgedächtnisses [RHP08, Gr06].

Die vorliegende Arbeit geht der Frage nach, auf welchem Komplexitätsniveau Grundschüler programmieren können und welche Entwicklungsphasen dabei sichtbar werden. Den Schwerpunkt der Untersuchung bildet dabei die Fähigkeit des Algorithmisierens.
Das Algorithmisieren ist eine der wesentlichen überfachlichen Kompetenzen, die im Informatikunterricht vermittelt werden. Ein Algorithmus ist eine präzise Folge von Anweisungen zum Lösen eines genau definierten Problems. Während die Kompetenz Algorithmen anzuwenden oft eine sehr spezielle und auf ein Fach oder eine Wissenschaft bezogene Tätigkeit ist, ist die Kompetenz des Algorithmisierens, also die Fähigkeit für ein gegebenes Problem eine präzise Folge von Lösungsschritten zu finden, eine Kompetenz, die in vielfältiger Weise, in verschiedenen Wissenschaften und vor allem im Alltag eine wichtige Rolle spielt. Sie ist die Fähigkeit, eine Problemlösung präzise und zielgerichtet zu konstruieren. Darüber hinaus liegt es im Charakter des Fachs Informatik, eine möglichst effiziente, also in möglichst wenig Zeit und mit möglichst wenig Platz, ablaufende Problemlösung zu beschreiben, die darüber hinaus noch automatisiert werden kann. Damit grenzt sich die Informatik deutlich von anderen Wissenschaften ab. Die Algorithmisierung findet sich deshalb auch in jeglicher Strukturierung der Informatik aus schulischer Sicht wieder (vgl. z.B. [Bi16], [Sc93]).

In Kapitel 2 wird zunächst ein Stufenmodell des Algorithmisierens entwickelt, welches die drei Dimensionen *Programmierelemente*, *Raumkomplexität* und *Problemkomplexität* umfasst. Die einzelnen Dimensionen werden jeweils in Stufen unterteilt. Aus diesem Stufenmodell werden in einem zweiten Schritt Testaufgaben entwickelt, um die Fähigkeiten zum Algorithmisieren von insgesamt 155 Grundschülern aus acht Klassen der Klassenstufen 2-4 an drei verschiedenen Schulen in Hessen bzw. Rheinland-Pfalz zu untersuchen. Die Aufgaben beziehen sich auf die Programmierung eines Hamsters [Bo13], der Aufga-

ben in einem Territorium bewätigen muss. Wie in den eingangs erwähnten Projekten wird auch hier mit konkret sichtbaren Problemstellungen gearbeitet, um die Aufgaben für die SuS leichter verständlich zu machen und die Motivation zu erhöhen. Die Ergebnisse befinden sich in Kapitel 4. Die dazugehörige Diskussion und ein Ausblick auf zukünftige Ideen befindet sich in Kapitel 5.

2 Das Stufenmodell

Zur Klassifizierung der Programmierfähigkeiten wurden Dimensionen des Programmierens [MHB15, BR12] untersucht. Auch die Taxonomie von Biggs [BC82] kann als Maß adaptiert werden. Jedoch sind beide Modelle zu grobkörnig für die Untersuchung der vorliegenden Problemstellung, da das Spektrum der Dimensionen für die Fähigkeiten von Grundschülern zu umfassend ist. Im Folgenden wird Algorithmisierung in drei verschiedene Dimensionen unterteilt. Die erste Dimension beschreibt die Möglichkeiten Algorithmen zu beschreiben. Hierbei werden drei verschiedene Stufen unterschieden:

1. Sequenzielle Befehle
2. Alternativen
3. Iteration

Als grundlegende Programmierelemente kommen normalerweise noch *Methoden* und in objektorientierten Programmiersprachen *Objekte* hinzu. Beide zählen wir nicht zur Algorithmisierung, sondern ähnlich wie auch in [Sc93] zur Modellierung, da sie die Funktionalität nicht erweitern, jedoch die Struktur verändern. *Rekursion* könnte als eigenständige Stufe betrachtet werden. Sie kann jedoch immer durch Iteration ersetzt werden und erweitert den Beschreibungsbereich somit nicht. Darüber hinaus ist die Rekursion ein fortgeschrittenes Konzept der Beschreibung von Lösungen, welches nur selten im Alltag intuitiv verwendet wird. Aus diesem Grund ist Rekursion für eine Einführung in der Grundschule sicherlich nicht geeignet. In höheren Klassenstufen kann sie jedoch als eigenständige Stufe mit aufgenommen werden.

Die zweite Dimension bezieht sich auf den Problemraum. Bei der gewählten Umgebung ist dies das Territorium des Hamsters (vgl. Bild 1). So kann ein Problem einen fest vorgegebenen oder einen beliebigen Raum betreffen. Der Algorithmus kann somit nur für genau einen Raum funktionieren oder für eine ganze Klasse von Territorien. Bei einem fest vorgegebenen Raum wird dahingehend unterschieden, ob der Weg fest vorgegeben ist oder nicht:

1. vorgegebener Raum, Weg vorgegeben
2. vorgegebener Raum
3. beliebiger Raum

Bei Problemen mit einem vorgegebenem Weg müssen die Kinder nicht selbst einen Weg finden, sondern dieser ist durch die Beschaffenheit des Raums vorgegeben (vgl. Bild 4 im Gegensatz zu Bild 5). Probleme in einem beliebigen Raum bedürfen der Fähigkeit, Lösungen nicht nur an konkreten Situationen festzumachen, sondern ein Muster oder etwas Vergleichbares zu erkennen. Ein Beispiel hierfür wäre „Beschreiben Sie einen Algorithmus, der Sie aus einem beliebigen Labyrinth führt". Nach Piaget [SL12] ist diese Fähigkeit, also die Fähigkeit, Verhalten unabhängig von einer konkreten Situation geistig zu verarbeiten, im konkret-operationalen Stadium (7-12 Jahre) noch nicht herausgebildet. Deshalb kann diese Stufe nur als Ausblick gewertet werden. Sie wird im Folgenden nicht untersucht.

Die dritte Dimension umfasst die Problemkomplexität. Hierbei werden ebenfalls drei verschiedene Stufen unterschieden:

1. 2-3 Änderungen
2. 4-8 Änderungen
3. beliebig viele Änderungen, Erkennung und Bearbeitung von Mustern

Untersuchungen zur Verarbeitungs- und Behaltensspanne [PSZ11] zeigen, dass 6-jährige Kinder dazu fähig sind, etwa 4 Wörter unmittelbar reproduzieren zu können. Die Verarbeitungsleistung ist noch geringer einzuschätzen, vgl. [Gr06]. Nach der Durchführung von Vortests mit Zweit- und Drittklässlern zeigte sich, dass Probleme mit vier oder mehr Änderungen, die sich die SuS merken mussten, deutlich schlechter gelöst wurden. Die letzte Stufe (beliebig viele Änderungen) kann nur in Kombination mit der dritten Stufe der Dimension *Problemraum*, also *beliebiger Raum*, auftreten. Deshalb wird sie in dieser Arbeit nicht untersucht.

Die unterschiedlichen Kombinationen von Komplexitätsstufen werden im Folgenden an einigen Stellen mit Zahlen abgekürzt. So beschreibt die Bezeichnung 3.2.1 ein Problem, dass unter Verwendung von Iteration, in einem vorgegebenen Raum mit maximal 2-3 Änderungen gelöst werden muss.

3 Methode

Insgesamt wurden von den SuS sieben Aufgaben bearbeitet. Alle Aufgaben wurden mit Zettel und Stift bearbeitet. Es wurde bewusst auf den Einsatz von Computern verzichtet, da sich bei der Verwendung von Computern die Möglichkeit bietet, eine richtige Lösung durch Austesten zu gewinnen. Dies misst jedoch nicht die Fähigkeit zum Algorithmisieren.

Bevor die konkreten Aufgaben beschrieben und analysiert werden, sei erwähnt, dass die Untersuchung nacheinander drei Befehlssätze verwendet hat. Befehlssatz 1 wurde in den Aufgaben 1-4 verwendet. Befehlssatz 2 in den Aufgaben 2-4. Für die Aufgaben 5-7 wurde ausschließlich Befehlssatz 3 verwendet.

Befehlssatz 1 wurde folgendermaßen definiert: **vor** - Er geht ein Kästchen vor, **links** - Er geht ein Kästchen nach links, **rechts** - Er geht ein Kästchen nach rechts, **zurück** - Er geht ein Kästchen zurück.

Befehlssatz 2 umfasst die Befehle: **vor bis zur Wand** - Er geht vor bis zur Wand, **links bis zur Wand** - Er geht nach links bis zur Wand, **rechts bis zur Wand** - Er geht nach rechts bis zur Wand, **zurück bis zur Wand** - Er geht zurück bis zur Wand.

Befehlssatz 3 umfasst die Befehle: **vor** - Er geht ein Kästchen vor, **linksum** - Er dreht sich nach links (90 Grad), **rechtsum** - Er dreht sich nach rechts (90 Grad), **Wand** - Er geht vor bis zur Wand.

Im Folgenden werden die Aufgaben erklärt. Zunächst die ersten beiden Aufgaben und Aufgabe 5:

1. Das ist unser Hamster Emil. Er hat einen neuen Käfig bekommen und kennt sich noch nicht so gut aus. Er wacht nachts auf und hat Hunger. Leider ist es sehr dunkel und er sieht nichts. Bitte hilf ihm, damit er das Futter findet (siehe Bild 1).
2. Emil will sein neues Spielzeug, einen Tunnel, ausprobieren. Er ist noch ein bisschen ängstlich. Du musst ihn führen. Du darfst kein Kommando zweimal hintereinander benutzen (vor vor ist nicht erlaubt!) (siehe Bild 2).
5. Emil hat Futter gefunden und möchte es in seinen Bau bringen. Bitte hilf ihm! Leite ihn zum Futter und dann zurück zum Ausgangspunkt.

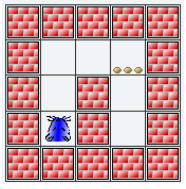

Bild 1. Aufgabe 1

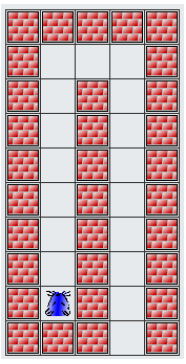

Bild 2. Aufgabe 2

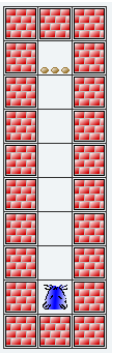

Bild 3. Aufgabe 5

Diese Aufgaben dienen zum einen als Einführung der Befehlssätze und „Warming up“, zum anderen sollen sie zeigen, dass die SuS die Befehle verstanden haben, wissen, wie sie ihre Lösungen aufschreiben sollen und prinzipiell keine links-rechts Schwächen haben. Sie zielen noch nicht darauf ab, eine der Stufen der Komplexität (vgl. Kapitel 2) zu messen.

Der erste Block von bewerteten Aufgaben (3+4) beschäftigt sich mit der Stufe der *Iterationen*. Hierbei wird die erste Stufe des Problemraums *(vorgegebener Raum, vorgegebener*

Weg) in Aufgabe 3 und die zweite Raumstufe in Aufgabe 4 *(vorgegebener Raum)* getestet. Die Problemkomplexität bleibt in der untersten Kategorie, da die Tests vor allem Aussagen über die Fähigkeit des Konzepts der Iteration geben sollen. Es werden also Stufe 3.1.1 und Stufe 3.2.1 getestet.

Die Aufgaben 3 und 4:

3. Puh! Der Tunnel ist ja noch gar nicht zu Ende und jetzt geht er auch noch im Zick-Zack weiter. Führe Emil durch den Tunnel.
   Emil, mache solange
   ...
   bis du aus dem Tunnel kommst!

4. Vom vielen Laufen hat Emil aber Hunger. Benutze möglichst wenig Anweisungen.
   Emil, mache solange
   ...
   bis du oben an der Wand bist. Gehe nun noch
   ...

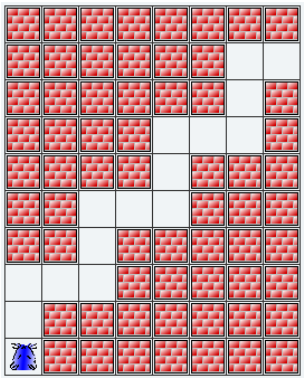

Bild 4. Aufgabe 3

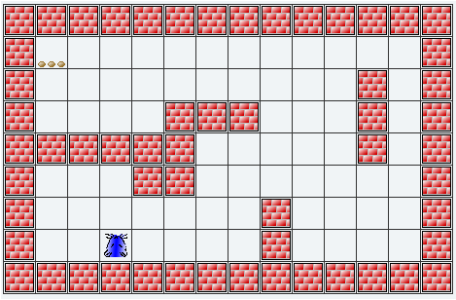

Bild 5. Aufgabe 4

In den Aufgaben 6 und 7 wurde die Problemkomplexität untersucht. So wurden bei beiden Aufgaben die ersten beiden Stufen auf unterstem Niveau betrachtet *(sequentielle Befehle, vorgegebener Raum, vorgegebener Weg)*. Aufgabe 6 misst die Problemkomplexität auf Stufe 1 *(2-3 Änderungen)* und Aufgabe 7 misst die Problemkomplexität auf Stufe 2 *(4-8 Änderungen)*. In Vortests zeigte sich, dass sich die Kinder Positionen im Raum durch Benutzung des Fingers auf dem Aufgabenzettel merkten. Deshalb wurde bei den Aufgaben 5-7 die Drehung aufgenommen und darauf geachtet, dass die SuS nur einen Stift zur Lösung benutzen konnten. Auf diese Weise konnte sichergestellt werden, dass die SuS sich gewisse Situationen merken mussten. Diese Fähigkeit wird beim Programmieren häufig benötigt, zum Beispiel bei Veränderungen von Variablen oder Arrays in einem Programm.

Die Aufgaben 6 und 7:

6 Emil ist entlaufen und ist in ein Rohr gekrabbelt. Er hat Angst und möchte deshalb nur noch vorwärts laufen, weil er dort alles gut sieht. Könntest du ihm helfen, wieder heraus zu kommen?

7 Emil findet wieder einmal das Futter nicht. Führe Emil zu den 6 Honigbällchen.

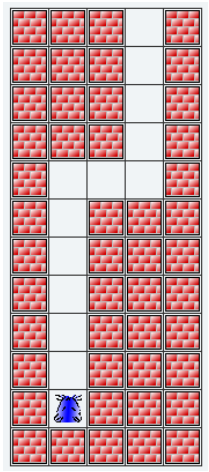

Bild 6. Aufgabe 6

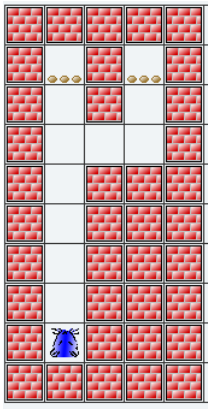

Bild 7. Aufgabe 7

Um die Objektivität der Untersuchung zu gewährleisten, wurde ein einheitlicher Ablaufplan für alle Untersuchungen aufgestellt. Sowohl die Erklärungen der Befehle, als auch die gezeigten Beispiele und der organisatorische Ablauf waren somit einheitlich und klar vorgegeben. Darüber hinaus wurden alle Untersuchungen unter Anleitung derselben Person durchgeführt.

4 Ergebnisse

Die bearbeiteten Aufgaben wurden bewertet und die Resultate in die diskrete Ordinalskala bestehend aus den drei Merkmalen „richtig gelöst“„teilweise gelöst“ und „nicht gelöst“ eingeordnet. Die Bewertung „teilweise gelöst“ wurde vergeben, wenn kleinere Fehler gemacht wurden, aber die Lösung des Problems erkennbar war. Syntaxfehler, bzw. Rechtschreibfehler, wurden nicht als Fehler gewertet. Die Ergebnisse der einzelnen Aufgaben sind in Kontingenztafeln dargestellt.

Die statistische Auswertung der Ergebnisse wurde mittels des *Exakten Fisher-Tests* durchgeführt. Die Berechnung erfolgte durch das Statistikprogramm *R*. Der Nachweis der Leistungsunterschiede zwischen den einzelnen Klassenstufen erfolgt zum Signifikanzniveau 1%. Die nachgewiesenen Unterschiede gelten als *hoch signifikant*.

Sind Unterschiede in den Leistungen zwischen der zweiten und dritten und zwischen der dritten und vierten Klasse nachweisbar, so kann hier aufgrund der Daten transitiv auf einen Leistungsunterschied zwischen den Klassen 2 und 4 geschlossen werden. Nur in Einzelfällen werden daher die Ergebnisse der zweiten und vierten Klasse direkt miteinander verglichen.

Aufgabe 3	Klasse 2	Klasse 3	Klasse 4	
richtig gelöst	5	31	39	75
teilweise gelöst	1	3	0	4
nicht gelöst	27	32	17	76
	33	66	56	155

Tabelle 1. Ergebnisse Aufgabe 3

Bei Aufgabe 3 ist zwischen der zweiten und dritten Klasse ein signifikanter Unterschied nachweisbar ($p = 0.002$). Auch wenn im Vergleich zu den SuS aus Klasse 3 ein höherer Prozentsatz der SuS aus Klasse 4 die Aufgabe richtig lösen konnte, ist dieser Unterschied nicht signifikant ($p = 0.014$). Ein deutlicher Leistungsunterschied besteht außerdem zwischen der zweiten und vierten Klasse ($p < 0.001$).

Aufgabe 4	Klasse 2	Klasse 3	Klasse 4	
richtig gelöst	1	16	29	46
teilweise gelöst	0	5	0	5
nicht gelöst	32	45	27	104
	33	66	56	155

Tabelle 2. Ergebnisse Aufgabe 4

Bei Aufgabe 4, der Aufgabe auf der höheren Problemstufe 3.2.1, lassen sich signifikante Unterschiede zwischen der zweiten und dritten Klasse ($p = 0.003$) und zwischen der dritten und vierten Klasse ($p = 0.001$) nachweisen. Es kann außerdem festgestellt werden, dass nur ein Kind aus der zweiten Klasse die Aufgabe lösen konnte.

Aufgabe 6	Klasse 2	Klasse 3	Klasse 4	
richtig gelöst	10	29	31	70
teilweise gelöst	7	17	14	38
nicht gelöst	16	20	11	47
	33	66	56	155

Tabelle 3. Ergebnisse Aufgabe 6

Bei Aufgabe 6 lässt sich kein signifikanter Unterschied zwischen den einzelnen Klassenstufen feststellen. Auch wenn angenommen werden muss, dass die SuS der zweiten und dritten Klasse die Aufgabe gleichermaßen lösen konnten ($p = 0.216$), genauso wie die SuS

der dritten und vierten Klasse ($p = 0.350$), so zeichnet sich doch ein Unterschied zwischen den SuS der zweiten und vierten Klasse ab ($p = 0.015$).

Aufgabe 7	Klasse 2	Klasse 3	Klasse 4	
richtig gelöst	6	34	38	78
teilweise gelöst	8	13	6	27
nicht gelöst	19	19	12	50
	33	66	56	155

Tabelle 4. Ergebnisse Aufgabe 7

Bei der höheren Problemkomplexität in Aufgabe 7 ist ein signifikanter Unterschied zwischen der zweiten und dritten Klasse nachweisbar ($p = 0.003$). Der Unterschied in den Ergebnissen der dritten und vierten Klasse ist nicht signifikant ($p = 0.184$). Es liegt ein deutlicher Unterschied zwischen den Leistungen der SuS der zweiten und vierten Klasse vor ($p < 0.001$).

5 Diskussion und Ausblick

Es zeigt sich ein deutlicher Leistungsunterschied zwischen der zweiten und der dritten Klasse. So konnten in den Aufgaben 3, 4 und 7 signifikante Unterschiede nachgewiesen werden. Auffällig ist auch, dass, bis auf eine einzige Ausnahme, kein Schüler und keine Schülerin der zweiten Klasse das Konzept der Iteration umsetzen konnte. Hingegen zeigt sich in der dritten Klasse, dass nahezu die Hälfte der Kinder unmittelbar fähig waren, dieses Konzept umzusetzen, sofern der Weg vorgegeben war. Bei nicht vorgegebenem Weg waren es nur noch ein Viertel der Drittklässler, die die Aufgabe lösen konnten. Der Unterschied zwischen zweiter und dritter Klasse wird auch in Aufgabe 7 sehr deutlich. So nimmt die Leistung bei einer höheren Anzahl von Veränderungen, die verarbeitet werden müssen, stark ab.

Der Unterschied zwischen den Klassen 3 und 4 zeigt sich vor allem bei dem Konzept der Iteration. So ist der Unterschied bei Aufgabe 3 mit vorgegebenem Weg zwar vorhanden, aber nicht signifikant. Bei Aufgabe 4 hingegen haben zirka ein Viertel der Drittklässler die Aufgabe korrekt gelöst, wohingegen mehr als die Hälfte der Viertklässler diese Aufgabe erfolgreich lösen konnte. Die sichtbaren Unterschiede in den Ergebnissen der Dritt- und Viertklässler in Aufgabe 7 müssen als zufällig angesehen werden.

Es zeigt sich, dass Aufgaben auf sehr niedrigem Niveau in allen Klassenstufen erfolgreich bearbeitet werden konnten. Sowohl das Konzept der Iteration als auch höhere Problemkomplexitätsstufen führten zu großen Problemen bei Schülerinnen und Schülern der zweiten Klasse. Das Konzept der Iteration kann unmittelbar erst ab der dritten Klasse verstanden und angewendet werden. Der sichere Umgang mit Iteration kann erst ab der vierten Klasse erwartet werden.

Insgesamt zeigt die Untersuchung klare Unterschiede im Übergang zwischen der zweiten und der dritten Klasse. Hinzu kommt, dass das Konzept der Iteration eine deutliche

Hürde in allen untersuchten Jahrgangsstufen ist. In einer Folgeuntersuchung muss überprüft werden, inwieweit diese Stufen in den betrachteten Altersklassen trainierbar sind. So sollte nach einem Eingangstest eine Unterrichtseinheit oder Ähnliches mit Grundschulkindern durchgeführt werden und infolgedessen ein zweiter Test durchgeführt werden. Hierbei muss vor allem die Verwendung des Programmierelements *Iteration* in allen Jahrgangsstufen untersucht werden. Desweiteren müssen die Bedingungen der zweiten Klasse genauer untersucht werden, da hier die prinzipielle Frage ist, ob Schülerinnen und Schüler der zweiten Klasse fähig sind, Lösungen für Probleme auf höheren Stufen zu erstellen. Aber auch für die Klassen 3 und 4 stellt sich die Frage, ob sich die Fähigkeiten für eine größere Anzahl von Schülerinnen und Schüler in einer Unterrichtseinheit erlernen, bzw. festigen lassen. Für die vierte Klasse stellt sich darüber hinaus die Frage, ob die Schülerinnen und Schüler schon in diesem Alter fähig sind, Probleme in einem beliebigen Raum zu lösen, da dies in klassischen Programmieraufgaben eine wichtige Rolle spielt.

Literatur

[BC82] John B. Biggs, Kevin F. Collis, *Evaluating the quality of learning: The SOLO taxonomy; structure of observed learning outcome*, Educ. Psychology Series, Academic Pr., 1982.

[BD09] Christian Borowski, Ira Diethelm, *Kinder auf dem Weg zur Informatik: Programmieren in der Grundschule*, In: Zukunft braucht Herkunft, INFOS 2009, 244–253, 2009.

[Bi16] Gerhard Röhner et al., *Bildungsstandards Informatik für die Sekundarstufe II*, Beilage zu LOG IN, Heft Nr. 183/184, 2016.

[BLS11] Nadine Bergner, Thiemo Leonhardt, Ulrik Schroeder *Zauberschule Informatik – Einblick in die Welt der Informatik für Kinder im Grundschulalter*, In: Informatik mit Kopf, Herz und Hand, INFOS 2011, 132–141, 2011.

[Bo13] Dietrich Boles, *Programmieren spielend gelernt mit dem Java-Hamster-Modell*, Springer-Vieweg-Verlag, 2013.

[BR12] Karen Brennan and Mitchel Resnick, *New frameworks for studying and assessing the development of computational thinking*, In: AERA, 2012.

[Gr06] Dietmar Grube, *Entwicklung des Rechnens im Grundschulalter*, Waxmann, 2006.

[GW16] Robert Garmann, Benjamin Wanous, *Code for competence - Programmieren für Zweitklässler mit ScratchJr*, In: Informatik für Kinder - 7. Münsteraner Workshop zur Schulinformatik, 71–80, 2016.

[MHB15] Andreas Mühling, Peter Hubwieser, Marc Benges, *Dimensions of programming knowledge.*, In: Situation, Evolution, and Perspectives, International Conference on Informatics in Schools, 32–44, Springer, 2015.

[PSZ11] Martin Pinquart, Gudrun Schwarzer, Peter Zimmermann, *Entwicklungspsychologie - Kindes- und Jugendalter*, 111–121, Hogrefe Verlag, 2011.

[RHP08] Kristina Reiss, Aiso Heinze, Reinhard Pekrun, *Mathematische Kompetenz und ihre Entwicklung in der Grundschule*, In: Kompetenzdiagnostik, 107–127, Verlag für Sozialwissenschaften, 2008.

[Sc01] Andreas Schwill, *Ab wann kann man mit Kindern Informatik machen?*, In: Informatikunterricht und Medienbildung, INFOS 2001, 13–30, 2001.

[Sc93] Andreas Schwill, *Fundamentale Ideen der Informatik*, Zentralblatt für Didaktik der Mathematik 1, 20–31, 1993.

[SL12] Wolfgang Schneider, Ulman Lindernberger *Entwicklungspsychologie*, 385–400, Beltz Verlag, 2012.