# Kompetenzorientierter Informatikunterricht in der Sekundarstufe I unter Verwendung der visuellen Programmiersprache Puck

Dissertation

zur Erlangung des akademischen Grades
doctor rerum naturalium (Dr. rer. nat.)

vorgelegt dem Rat der Fakultät für Mathematik und Informatik der
Friedrich-Schiller-Universität Jena

von Dipl.-Inf. Lutz Kohl
geboren am 08.11.1980 in Leinefelde

Gutachter
1. Prof. Dr. Michael Fothe, Jena
2. Prof. Dr. Bernd Zimmermann, Jena
3. Prof. Dr. Steffen Friedrich, Dresden

Tag der letzten Prüfung des Rigorosums: 27.03.2009

Tag der öffentlichen Verteidigung: 17.04.2009

# Danksagung

Diese Dissertation ist im Rahmen meiner wissenschaftlichen Tätigkeit an der Professur für Didaktik der Informatik/Mathematik (2003-2008 CASIO-Stiftungsprofessur) der Friedrich-Schiller-Universität Jena entstanden. Für das stete Fördern, Fordern und Motivieren, die Betreuung der Arbeit sowie die vielen gewinnbringenden Diskussionen bedanke ich mich herzlich bei Prof. Dr. Michael Fothe. Weiterhin gilt mein Dank den Gutachtern Prof. Dr. Steffen Friedrich und Prof. Dr. Bernd Zimmermann.. Ich danke Frau Dr. Carolin Ligges und Frau Kinga Szücz für anregende Diskussionen. Weiterhin sei den Lehrenden und Lernenden gedankt, die an den wissenschaftlichen Untersuchungen teilnahmen, sowie all jenen, die in sonstiger Weise zum Gelingen dieser Arbeit beigetragen haben. Meinen Eltern danke ich herzlich für vielfältige Unterstützungen und steten Ansporn.

Ganz besonderer Dank gilt meiner Frau Susann, die mich immer geduldig unterstützte und motivierte sowie unseren stets fröhlichen Kindern.

# Zusammenfassung

In dieser Arbeit wurde die aktuelle Diskussion um Bildungsstandards sowie die spezifische Situation in der Informatik dargestellt (vgl. Kapitel 2). Weiterhin wurde auf Kompetenzen, Kompetenzmodelle und Aufgaben eingegangen (vgl. Kapitel 3). Ausgehend von diesen Analysen wurde auf Grundlage der von der GI empfohlenen „Grundsätze und Standards für Informatik in der Schule“ ein Kompetenzmodell für den Inhaltsbereich „Algorithmen“ der 8. bis 10. Jahrgangsstufe entwickelt (vgl. Kapitel 5). Durch zusätzlich bereitgestellte Aufgaben können die in diesem Modell angegebenen Kompetenzen veranschaulicht, erworben und überprüft werden (vgl. Kapitel 6). Die bei der Entwicklung des Kompetenzmodells und der Aufgaben gewonnenen Erfahrungen wurden jeweils zusammengefasst.

Das Erwerben von Kompetenzen zum Inhaltsbereich „Algorithmen“ ist häufig mit dem Einsatz einer konkreten Programmiersprache und eines konkreten Programmiersystems verbunden. Die vom Autor erstellte visuelle Programmiersprache Puck wurde im Rahmen dieser Arbeit weiterentwickelt und von verschiedenen anderen visuellen Werkzeugen abgegrenzt. Weiterhin wurden Gründe dargestellt, die für den Einsatz solcher visueller Werkzeuge bei der Einführung in die Programmierung sprechen (vgl. Kapitel 4).

In einer Voruntersuchung an sechs Thüringer Schulen wurden erste Erfahrungen zu kompetenzorientiertem Informatikunterricht unter Verwendung der entwickelten Materialien und der visuellen Programmiersprache Puck zusammengetragen (vgl. Kapitel 7). Auf Grundlage dieser Ergebnisse wurden in der Hauptuntersuchung 84 Lehrerinnen und Lehrern im deutschsprachigen Raum das weiterentwickelte Kompetenzmodell, die zugehörigen Aufgaben und Puck zur Verfügung gestellt (vgl. Kapitel 8). Die Befragung von 40 Lehrpersonen, am Ende der Hauptuntersuchung ergab, dass es einem Großteil der Lehrpersonen möglich war, anhand der bereitgestellten Materialien kompetenzorientierten Informatikunterricht zu strukturieren, vorzubereiten, durchzuführen und auszuwerten. Es zeigte sich außerdem, dass die visuelle Programmiersprache Puck größtenteils als geeignet für eine Einführung in die Programmierung in den Jahrgangsstufen 8 bis 10 eingeschätzt wurde.

# Inhaltsverzeichnis

# 1 Einleitung

> *Doch Forschung strebt und ringt, ermüdend nie,*
> *Nach dem Gesetz, dem Grund, Warum und Wie.*
> (Johann Wolfgang von Goethe, Chinesisch-deutsche Jahres- und Tageszeiten, S. 14)

Angeregt durch die PISA-Ergebnisse ist im deutschen Schulsystem von einer traditionellen Steuerung des „Inputs“, d. h. von einer Festlegung der Strukturen, der inhaltlichen Vorgaben und der Ausstattung der Schulen, in den letzten Jahren eine Orientierung am „Output“ schulischer Bildung in den Vordergrund gerückt, d. h. eine Orientierung an den Ergebnissen der Lernenden (vgl. Ohrnberger u. a., 2007, S. 87). Nach Klieme u. a. (2003, S. 12) sorgt der Staat mit dieser „Outputorientierung“ nicht mehr durch detaillierte Richtlinien und Regelungen für Qualität, sondern durch Definition von Zielen, deren Einhaltung auch tatsächlich überprüft wird. Klieme u. a. (2003, S. 9 f.) empfehlen, diese Ziele als Bildungsstandards zu formulieren, die allgemeine Bildungsziele aufgreifen und festlegen, welche Kompetenzen bis zu einer bestimmten Jahrgangsstufe mindestens erworben werden sollten.[1] Sie nennen als Funktionen von Bildungsstandards, dass diese Schulen die Möglichkeit geben, sich an verbindlichen Zielen zu orientieren und dass auf ihrer Grundlage Lernergebnisse erfasst und bewertet werden. Als Aufgaben für die nähere Zukunft beschreiben Klieme u. a. (2003, S. 10) *die fachdidaktische Vertiefung in Kompetenzmodellen, [...] die Entwicklung von Aufgabenpools und Testverfahren sowie die Implementation an den Schulen.*[2] Die vorliegende Arbeit soll für das Fach Informatik einen Beitrag zu diesen Aufgabenfeldern liefern.

[1] Wie die Begriffe „Kompetenz“ und „Kompetenzmodell“ im Rahmen dieser Arbeit verwendet werden, wird im Kapitel 3 erläutert. Zum Begriff „Bildungsstandards“ sei auf die Ausführungen von Klieme u. a. (2003, S. 19) verwiesen.
Der Ansatz der Bildungsstandards wird teilweise mit dem lernzielorientierten Unterricht aus den 1960er und 1970er Jahren verglichen (vgl. z. B. Neuweg, 2004, S. 4). Auf eine ausführliche Darstellung des lernzielorientierten Unterrichts wird an dieser Stelle verzichtet; der interessierte Leser sei auf die Darstellungen von Jank und Meyer (1994, S. 299 ff.) sowie Möller (1995, S. 62 ff.) verwiesen.

[2] Alle in dieser Arbeit kursiv gesetzten Texte sind wörtliche Zitate.

### Entwicklung eines Kompetenzmodells mit zugehöriger Aufgabensammlung

Kompetenzmodelle als zentrale Bestandteile von Bildungsstandards sollen nach Klieme u. a. (2003, S. 22 und S. 119 ff.) zwischen abstrakten Bildungszielen und konkreten Aufgabensammlungen vermitteln, indem sie zwischen Teildimensionen innerhalb einer Domäne unterscheiden und verschiedene Niveaustufen beschreiben. Klieme u. a. (2003, S. 9) stellen dar, dass Kompetenzmodelle den Lehrkräften ein Referenzsystem für ihr professionelles Handeln bieten. Zum Erwerben und Überprüfen der Kompetenzen werden verschiedene Aufgaben benötigt (vgl. Blum, 2006, S. 18 und Herper, 2005, S. 74). Friedrich und Puhlmann (2007, S. 32) sehen diese Erarbeitung von Aufgaben als besonders wichtig an, *weil im Bereich der Informatik eine didaktisch-methodisch begründete Aufgabenkultur bisher wenig entwickelt ist und bedingt durch den Perspektivwechsel auch eine andere Sicht auf Aufgaben entsteht.* Es stellt sich die Frage:

**1. Forschungsfrage:** Wie kann ein Kompetenzmodell mit einer zugehörigen Aufgabensammlung für einen Bereich des Informatikunterrichts entwickelt werden?

Im Kapitel 2 wird das Konzept von Bildungsstandards vorgestellt, anhand von Argumenten eingeordnet und drei unterschiedliche Ansätze zur Festlegung der Ziele des Informatikunterrichts werden präsentiert. Auf diese grundlegenden Darstellungen wird im Verlauf der weiteren Arbeit Bezug genommen. Kompetenzmodelle und Aufgaben können mit unterschiedlichen Zielstellungen auf verschiedene Art und Weise entwickelt werden. Um eine Eingrenzung vorzunehmen, stellt Kapitel 3 mit Bezug auf Literatur und Beispiele dar, was unter Kompetenzen im Rahmen dieser Arbeit verstanden wird, welche Art von Kompetenzmodell wie entwickelt werden soll und welche Aufgabentypen unterschieden werden. Im Kapitel 5 wird ein Kompetenzmodell für einen Bereich des Informatikunterrichts entwickelt. In einer Reflexion werden anschließend Anforderungen für die Entwicklung eines solchen Kompetenzmodells genannt und erläutert. Nachfolgend werden im Kapitel 6 Aufgaben zum Veranschaulichen, Erwerben und Überprüfen der Kompetenzen des Modells vorgestellt. Auch hier werden in einem Reflexionsprozess Anforderungen zur Entwicklung solcher Aufgaben genannt und erläutert.

### Umsetzen von kompetenzorientiertem Informatikunterricht

Blum (2006, S. 18 f.) sieht es als Gefahr an, wenn erwartet wird, dass die Entwicklung von Bildungsstandards automatisch zu einer Verbesserung der Unterrichtsqualität führt. Für den langfristigen Kompetenzaufbau fordert er einen breit angelegten konsequent kompetenzorientierten Unterricht, bei dem sich *verwendete*

*Lernaufgaben an den angestrebten Kompetenzen orientieren, die sich wiederum in Testaufgaben konkretisieren können.* Es stellt sich die Frage:

**2. Forschungsfrage:** Inwieweit sind das in dieser Arbeit konstruierte Kompetenzmodell und die zugehörigen Aufgabensammlung als Grundlage für kompetenzorientierten Informatikunterricht in der Sekundarstufe I geeignet?

Ziel der Erprobung des entwickelten Kompetenzmodells und der zugehörigen Aufgabensammlung im Unterricht ist nicht Bildungsmonitoring oder Schulevaluation.[3] Vielmehr wird angestrebt, dass Lehrpersonen die entwickelten Materialien sinnvoll in einem kompetenzorientierten Unterricht einsetzen (vgl. Kapitel 3). In zwei Untersuchungen, die in anbetracht dieser Zielstellung größtenteils auf der Lehrerebene stattfinden, werden das Kompetenzmodell und die Aufgabensammlung erprobt (vgl. Kapitel 7 und 8). Die gewonnenen Ergebnisse zum Einsatz der Materialien können als eine Grundlage zur Weiterentwicklung derselben, zur gezielten Fortbildung von Lehrkräften und zur Entwicklung weiterer Kompetenzmodelle und Aufgaben für andere Bereiche des Informatikunterrichts genutzt werden (vgl. Kapitel 9).

### Einsatz von Puck zur Einführung in die Programmierung

Humbert (2005, S. 64) stellt fest, dass Informatiksysteme unverzichtbarer Bestandteil informatischer Lehr-/Lernprozesse sind. Die GI (2008, S. 8) legt in ihren Empfehlungen zu Grundsätzen und Standards für die Informatik in der Schule dar, dass Technik, als digitales Hilfsmittel im Informatikunterricht stets Medium, Werkzeug und Inhalt ist. Diese Verknüpfung des Informatikunterrichts mit zugehörigen (technischen) Werkzeugen kann in der Geschichte des Informatikunterrichts nachvollzogen werden (zur Geschichte des Informatikunterrichts vgl. Baumann, 1996, S. 101 ff.; Hartmann und Nievergelt, 2002; Hubwieser, 2004, S. 50 ff.; Humbert, 2005, S. 47 ff. sowie Koerber und Peters, 1998, S. 19 ff.). Auch beim im Rahmen der Untersuchungen dieser Arbeit durchzuführenden kompetenzorientierten Informatikunterricht soll ein (Software-) Werkzeug eingesetzt werden. Inwieweit das Programmieren Bestandteil eines Schulfachs Informatik sein sollte, wird in der Fachdidaktik Informatik kontrovers diskutiert (vgl. z. B. Gutknecht und Hromkovic, 2008). Es kann allerdings festgestellt werden, dass trotz aller Diskussionen in vielen Ländern das Arbeiten mit einer Programmiersprache Bestandteil des Informatikunterrichts ist (vgl. z. B. Thüringer Kultusministerium, 1999, S. 18). Eines der größten und frustrierendsten Probleme für Programmieranfänger war und ist das richtige Verwenden der Syntax einer Programmiersprache

[3] Was unter Bildungsmonitoring und Schulevaluation zu verstehen ist, wird im Abschnitt 2.1 erläutert.

(vgl. z. B. Kelleher und Pausch, 2005, S. 89 f.[4]). Die vom Autor im Rahmen von Studien- und Diplomarbeit entwickelte visuelle Programmiersprache Puck vermeidet Syntaxfehler (vgl. Kohl, 2004a und 2004b).[5] Erste Einsätze im Unterricht der Sekundarstufe II gaben Anlass zur Annahme, dass Puck dadurch geeignet ist, auch jüngeren Schülerinnen und Schülern eine Einführung in die Programmierung zu geben (vgl. Kohl, 2005). Es stellt sich die Frage:

**3. Forschungsfrage:** Eignet sich die visuelle Programmiersprache Puck, Schülerinnen und Schülern der Jahrgangsstufen 8 bis 10 eine Einführung in die Programmierung zu geben?

Im Kapitel 4 wird Puck vorgestellt, eingeordnet und von anderen Systemen abgegrenzt. Außerdem werden Problemfelder bei der Einführung in die Programmierung sowie Gründe, die für den Einsatz von visuellen Werkzeugen und speziell von Puck sprechen, dargestellt und es wird über den Entwicklungsprozess von Puck reflektiert. Bei den in den Kapiteln 7 und 8 dargestellten Untersuchungen zum kompetenzorientierten Informatikunterricht sollte Puck zur Einführung in die Programmierung in den Jahrgangsstufen 8 bis 10 eingesetzt und auf eine Eignung hierfür überprüft werden.

Im Kapitel 9 werden die Ergebnisse dieser Arbeit in Bezug auf die gestellten Forschungsfragen eingeordnet und ein Ausblick auf weitere mögliche Forschungsarbeiten wird gegeben.

[4] Originalzitat: *One of the largest and most frustrating challenges for novice programmers is syntax. Despite the attempts to make programming languages simpler and more understandable, many novices still struggle with syntax, for example, remembering the names of commands, the order of parameters, whether or not they are supposed to use parentheses or braces, and so on* (Kelleher und Pausch, 2005, S. 89 f.).

[5] Was in dieser Arbeit unter visuellen Programmiersprachen verstanden wird, wird im Abschnitt 4.2 erläutert.

# 2 Bildungsstandards und Schulinformatik

*So ein bißchen Bildung ziert den ganzen Menschen.*
(Heinrich Heine, Die Bäder von Lucca, S. 39)

In diesem Kapitel werden die allgemeine Diskussion um Bildungsstandards sowie verschiedene Ansätze, Ziele für die Schulinformatik festzulegen, vorgestellt. Auf die dadurch gelegten Grundlagen wird an verschiedenen Stellen in dieser Arbeit zurückgegriffen. Im Abschnitt 2.1 wird anhand der „Klieme-Expertise“ und der durch die KMK verabschiedeten Bildungsstandards auf die Entwicklungen in Deutschland eingegangen.[6] Anschließend wird im Abschnitt 2.2 eine Einordnung von Bildungsstandards anhand von Thesen und Argumenten vorgenommen, und es werden Schlussfolgerungen für die Informatik gezogen. Entsprechend der im Kapitel 1 dargestellten ersten Forschungsfrage soll in dieser Arbeit ein Kompetenzmodell mit zugehöriger Aufgabensammlung für einen Bereich des Informatikunterrichts entwickelt werden. Dafür erscheint es sinnvoll, sich an bestehenden Ansätzen, die Ziele des Informatikunterrichts festlegen, zu orientieren. Abschnitt 2.3 stellt das „Model Curriculum for K-12 Computer Science“ der ACM (2003) vor, ein in den USA entwickeltes und in der internationalen Diskussion vielbeachtetes Dokument, welches Ziele bzw. Inhalte für die gesamte Schulinformatik beschreibt.[7] Im Abschnitt 2.4 wird auf die „einheitlichen Prüfungsanforderungen Informatik“ eingegangen, die von der KMK (2004b) verabschiedet wurden und festlegen, welche Anforderungen ein Prüfling am Ende der Sekundarstufe II in Deutschland erfüllen muss. Im Abschnitt 2.5 werden die von der GI (2008) empfohlenen „Grundsätze und Standards für die Informatik in der Schule“ präsentiert, in denen Kompetenzen vorgeschlagen werden, die Schülerinnen und Schüler am Ende der 7. bzw. 10. Jahrgangsstufe in der Informatik an allen deutschen Schulen mindestens erworben haben sollen.[8]

[6] KMK steht für die ständige Konferenz der Kultusminister der Länder in der Bundesrepublik Deutschland

[7] ACM steht für Association for Computing Machinery. Die ACM ist eine internationale Fachgesellschaft für Informatik.

[8] GI steht für Gesellschaft für Informatik e. V..
Auf einen weiteren Vorschlag für Bildungsstandards im Informatikunterricht der Sekundarstufe I des Gymnasiums von Modrow (2005) soll aufgrund der umfangreicheren und aktuelleren GI-Empfehlungen zu Grundsätzen und Standards für Informatik in der Schule nicht weiter

## 2.1 Klieme-Expertise und KMK-Bildungsstandards

Die KMK beschloss im Jahr 2002, nationale Bildungsstandards in Kernfächern einzuführen. Das Bundesministerium für Bildung und Forschung (BMBF) gab ein Gutachten mit dem Ziel in Auftrag, das Konzept der Bildungsstandards fachlich zu klären und eine Rahmenkonzeption dafür vorzustellen, wie Bildungsstandards für das deutsche Schulsystem angelegt sein sollten und wie sie entwickelt und genutzt werden sollten (vgl. Klieme u. a., 2003, S. 14 f.). Das Gutachten wurde von führenden Experten aus Erziehungswissenschaft, empirischer Bildungsforschung, Lehr-Lern-Forschung, Bildungsrecht, pädagogisch-psychologischer Methodenlehre sowie den Fachdidaktiken der Mathematik und der Fremdsprachen unter der Koordination von Prof. Dr. E. Klieme erarbeitet und im Februar 2003 vorgestellt (vgl. Klieme u. a., 2003, S. 5 ff.).

Nach den Ergebnissen internationaler Vergleichsstudien wurde untersucht, was im Schulsystem von Ländern, die in diesen Studien besser abgeschnitten hatten, anders ist. Von Klieme u. a. (2003, S. 13) wurde unter anderem dargestellt, dass *die sehr guten Ergebnisse der skandinavischen Staaten und einiger anglo-amerikanischer Staaten, vermuten [lassen], dass Länder, die systematische Qualitätssicherung betreiben – sei es durch regelmäßige Schulleistungsstudien oder durch ein dichtes Netz von Schulevaluation – insgesamt höhere Leistungen erreichen.* Klieme u. a. (2003, S. 14) erläutern weiter, dass Bildungspolitiker als Antwort auf die Erkenntnisse vorschlugen, durch Bildungsstandards verbindlich festzuschreiben und regelmäßig zu überprüfen, welche Kompetenzen Schülerinnen und Schüler erreichen sollen. Um verschiedene Arten von Bildungsstandards zu unterscheiden, wurden von Klieme u. a. (2003, S. 32) anhand von drei Fragen unterschiedliche Varianten des Begriffs in der internationalen Diskussion erläutert, die nachfolgend dargestellt und an Beispielen des Informatikunterrichts veranschaulicht werden sollen:

1. Beziehen sich die Standards auf den Input oder auf den Output?

    - **opportunity-to-learn standards** beziehen sich auf den Input und die Prozesse des schulischen Lernens, wie z. B. Inhalt und Konzeption des Unterrichts, Ausstattung der Schulen, Qualifikation der Lehrkräfte. Beispiele: für jede Schülerin und jeden Schüler steht ein Computer zur Verfügung; Informatikunterricht wird von entsprechend ausgebildeten

eingegangen werden. Auch die im Rahmen des Projektes „Test your ICT-Knowledge“ entwickelten Standards für Medienbildung werden aufgrund der Ausrichtung dieser Arbeit auf den Informatikunterricht nicht weiter thematisiert (vgl. Keller und Moser, 2005 sowie Moser, 2006).

Lehrkräften erteilt; im Unterricht werden Anweisungen, Variablen, Verzweigungen und Schleifen behandelt.

- **outputorientierte Standards** beziehen sich hauptsächlich auf die Lernergebnisse der Schülerinnen und Schüler, dabei können anhand von zwei weiteren Fragen wieder verschiedene Typen unterschieden werden:

a) Auf welcher Abstraktionsebene werden die Standards formuliert?

   - Standards, die auf der Ebene **allgemeiner Bildungsziele und Wertvorstellungen** definiert sind.
     Beispiel: *Die Vision ist, dass informatisch gebildete Menschen alle informatischen Probleme, die ihnen in ihrem Leben begegnen werden, mit Selbstvertrauen anpacken und selbstständig allein oder im Team bewältigen können* (vgl. GI, 2008, S. 1).
   - Standards, die auf der Ebene mehr oder weniger **bereichsspezifischer Kompetenzen** formuliert sind.
     Beispiele: *Schülerinnen und Schüler kennen und verwenden arithmetische und logische Operationen. Schülerinnen und Schüler stellen Datentypen und Operationen formal dar und nutzen sie sachgerecht* (vgl. GI, 2008, S. 15).
   - **content standards** werden auf der Ebene von Lernzielen und -inhalten formuliert.
     Beispiel: *The Web page design component of this course should cover the following ideas: The use of hypertext links to load new pages or activate processes; Storing, compressing, encrypting, and retrieving image, video, and sound data; User interface design; Tools for expressing design (storyboard, site map)*(vgl. ACM, 2003, S. 15)
   - **performance standards** werden auf der Ebene von Testwerten formuliert (ein Grenzpunkt auf der Testwertskala markiert den Standard).
     Beispiel: Eine European Computer Driving Licence (ECDL) Advanced Prüfung gilt als bestanden, wenn mindestens 75 % der Fragen/Aufgaben korrekt gelöst wurden
     (vgl. ECDL Switzerland, 2008).

b) Welches Zielniveau wird spezifiziert?[9]

[9] Für Mindest-, Regel- und Maximalstandards werden keine Beispiele angegeben. Je nach Festlegung könnte ein Beispiel in einem Land Mindest-, in einem anderen Regel- und in wieder einem anderen Maximalstandards veranschaulichen.

- **Mindeststandards** beschreiben ein basales Niveau, das (fast) alle Schülerinnen und Schüler erreichen sollten.
- **Regelstandards** beschreiben ein durchschnittliches Erwartungsniveau.
- **Maximalstandards** beschreiben ein ideales Zielniveau.

Die Expertengruppe legte sich in dem Gutachten beim Begriff Bildungsstandards auf ergebnisbezogene (also outputorientierte) Bildungsstandards fest (vgl. Klieme u. a., 2003, S. 33). Diese sollten nach Klieme u. a. (2003, S. 33) auf bereichsspezifische Kompetenzen abzielen, sich an allgemeinen Bildungszielen orientieren und prinzipiell in Aufgaben und Testskalen umsetzbar sein.[10] Die Expertengruppe empfiehlt, *im Rahmen von Kompetenzmodellen verschiedene Stufungen von Kompetenzen zu unterscheiden, darunter aber eine bestimmte Kompetenzstufe als Mindeststandard auszuweisen* (vgl. Klieme u. a., 2003, S. 33). Kompetenzmodelle, die Inhalte auf Stufen der allgemeinen Bildung konkretisieren, werden von Klieme u. a. (2003, S. 62 ff.) auch als pragmatische Antwort auf Konstruktions- und Legitimationsprobleme von Bildungszielen angegeben. In dieser Arbeit wird ein solches Kompetenzmodell mit verschiedenen Stufen entwickelt.

Von Klieme u. a. (2003, S. 24 ff.) werden als Merkmale für gute Bildungsstandards Fachlichkeit, Fokussierung, Kumulativität, Verbindlichkeit für alle, Differenzierung, Verständlichkeit und Realisierbarkeit genannt. Diese werden als Kriterien zur Einschätzung von Bildungsstandards genutzt, z. B. von Fothe (2005, S. 48 f.) zur Einschätzung der EPA Informatik (vgl. Abschnitt 2.4). Im Abschnitt 2.5 werden die Merkmale guter Bildungsstandards ausführlicher erläutert und es wird überprüft, inwieweit die „Grundsätze und Standards für Informatik in der Schule“ der GI (2008) diese Merkmale aufweisen.

Klieme u. a. (2003, S. 49) raten nachdrücklich zu einer *deutlichen Trennung zwischen der Verwendung von standard-bezogenen Tests für Evaluation, Bildungsmonitoring und [...] als Entscheidungshilfe für individuelle Förderung einerseits, Noten und Abschlussprüfungen andererseits.* Auf negative Konsequenzen von „high stakes assessment“ in den USA wird hingewiesen (vgl. Klieme u. a., 2003, S. 48).[11]

[10] Nicht alle Kompetenzen, die sich an Bildungszielen orientieren, sind in Testskalen umsetzbar (vgl. hierzu Abschnitt 2.2).

[11] Als „high stakes assessment“ werden Tests bezeichnet, deren Ergebnisse direkte Auswirkungen auf den Getesteten haben, z. B. Zulassung zu einer Universität (vgl. Klieme u. a., 2003, S. 108).
Herzog (2006, S. 8) stellt zu diesem Thema folgendes dar: *Zu diesem 'High- Stakes Testing' gibt es inzwischen eine ansehnliche empirische Literatur, die etwas plakativ zusammengefasst zeigt, dass die Tests nicht messen, was sie zu messen vorgeben, dass schwache Schülerinnen*

In der Expertise werden vier Einsatzbereiche standardbezogener Tests unterschieden (vgl. Klieme u. a., 2003, S. 82 f.):

1. Beim **Überprüfen von Kompetenzmodellen** wird festgestellt, *ob diese Modelle tatsächlich die Aspekte der Kompetenzen von Lernenden, ihre Niveaustufung und ggf. ihre Entwicklung angemessen widerspiegeln.*
2. Beim **Systemmonitoring** *werden Tests verwendet, um Aussagen über das Kompetenzniveau von Schülerinnen und Schülern zu machen und Zusammenhänge mit schulischen wie außerschulischen Bedingungen aufzudecken.*
3. Bei **Schulevaluation** werden *Tests und andere Verfahren [...] eingesetzt, um zu prüfen, inwieweit eine Schule ihre pädagogischen Ziele erreicht.*
4. Bei der **Individualdiagnostik und der Förderung einzelner Schülerinnen und Schüler** werden spezifische Aussagen über die Stärken und Schwächen und damit den Förderbedarf einzelner Schülerinnen und Schüler gemacht. *Für diesen Zweck ist es in der Regel sinnvoller, einen kleineren Kompetenzbereich detaillierter zu erfassen.*

In dieser Arbeit soll unter anderem ein Kompetenztest entwickelt werden, den Lehrerinnen und Lehrer zur Individualdiagnostik und zur Beurteilung des durchgeführten Unterrichts einsetzen können. Dementsprechend wird der Kompetenztest – wie auch das gesamte Kompetenzmodell – einen kleinen Kompetenzbereich der Informatik thematisieren. Aussagen dazu, inwieweit das Modell und die in Stufen angegebenen Kompetenzen angemessen gewählt wurden, sollen größtenteils auf Grundlage der Einschätzungen der Lehrpersonen gemacht werden. Eine ausführliche empirische Überprüfung des Kompetenzmodells aufgrund der Lösungen zu Testaufgaben von Schülerinnen und Schülern erfolgt aufgrund der Ausrichtung der Untersuchungen auf das Lehrerhandeln nicht. Aussagen über Systemmonitoring und Schulevaluation werden nicht gemacht. Dementsprechend ist in dieser Arbeit hauptsächlich der 4. der genannten Einsatzbereiche standardbezogener Tests von Bedeutung.

Die KMK hat im Dezember 2003 Bildungsstandards für den mittleren Schulabschluss in den Fächern Deutsch, Mathematik und erste Fremdsprache (Englisch und Französisch) beschlossen (vgl. KMK 2003a; 2003b; 2003c). Diese sollen seit Sommer 2004 in allen 16 Ländern umgesetzt werden (vgl. KMK, 2004a, S. 18).[12] Wie von Klieme u. a. (2003, S. 33) gefordert, sind die Bildungsstandards der KMK

*und Schüler noch schwächer werden, dass sich der Unterricht an den Outputmessungen orientiert ('teaching to the test'), dass die Lehrerzentriertheit des Unterrichts (wieder) zunimmt und dass Lehrer wie Schüler demotiviert werden.*

[12] Im Oktober 2004 wurden von der KMK weitere Vereinbarungen über Bildungsstandards für den Primarbereich (Jahrgangsstufe 4) in Deutsch und Mathematik, für den Hauptschulabschluss

ergebnisbezogen formuliert, zielen auf bereichsspezifische Kompetenzen ab und sind prinzipiell in Aufgaben und Testskalen umsetzbar. Sie sind allerdings pragmatisch als Regelstandards festgelegt, weil nach Auffassung der KMK (2004a, S. 14) *notwendige Mindeststandards erst nach einem längeren Prozess der Erfahrung im Umgang mit Bildungsstandards formuliert werden können.* Auch wenn die KMK-Standards somit noch nicht alle Empfehlungen der Expertise von Klieme u. a. (2003) umsetzen, zeigen sie doch Veränderungen in Richtung einer Outputorientierung im Bildungswesen Deutschlands.[13] Die KMK hat weiterhin im Oktober 2007 festgelegt, die einheitlichen Prüfungsanforderungen für die Abiturprüfung (EPA) zu bundesweiten Bildungsstandards für die Abiturprüfung weiterzuentwickeln und zwar zunächst in den Fächern Deutsch, Mathematik und erste Fremdsprache (Englisch/Französisch) und ferner für die naturwissenschaftlichen Fächer (Biologie, Chemie, Physik).[14] Derzeit gibt es keinen Beschluss der KMK, Bildungsstandards für das Fach Informatik zu entwickeln. In den Abschnitten 2.3, 2.4 und 2.5 sollen deshalb verschiedene andere Ansätze zur Beschreibung der Ziele des Informatikunterrichts vorgestellt werden.

## 2.2 Diskussion zu Bildungsstandards

Um eine gewisse Einordnung des Konzeptes „Bildungsstandards“ vornehmen zu können, werden nachfolgend verschiedene Thesen und Argumente dargestellt. Dabei werden folgende drei Zielkomplexe von Bildungsstandards unterschieden:

- Strukturierung geforderter Kompetenzen schulischer Bildung

in den Fächern Deutsch, Mathematik und erste Fremdsprache (Englisch/Französisch) sowie für den mittleren Schulabschluss in den Fächern Biologie, Chemie und Physik beschlossen. Des Weiteren wurde 2006 das „Institut zur Qualitätsentwicklung im Bildungswesen“ (IQB) als wissenschaftliche Einrichtung der Länder an der Humboldt-Universität zu Berlin gegründet mit der Kernaufgabe, *die nationalen Bildungsstandards zu normieren, zu präzisieren, und weiterzuentwickeln. Weiterhin ist es ein Anliegen des IQB, die Schulen dabei zu unterstützen, die Bildungsstandards im Unterricht zu implementieren* (vgl. IQB, 2007).

[13] Reiss (2007, S. 20) bezeichnet die KMK-Bildungsstandards in Anbetracht der kurzen Entwicklungszeit als akzeptabel, da sie *beinhalten, was im jeweiligen Fach und auf der jeweiligen Klassenstufe zum grundlegenden Wissen gehört, und verdeutlichen, über welche Kompetenzen Schülerinnen und Schüler am Ende einer bestimmten Phase verfügen sollen.* Trotz verbleibender Kritikpunkte scheinen die KMK-Bildungsstandards auch für Vollmer (2005, S. 1) für die nächste Zukunft gesellschaftlich tragfähig zu sein.

[14] Dafür wurde das Institut zur Qualitätsentwicklung im Bildungswesen (IQB) in Zusammenarbeit mit der Amtschefskommission „Qualitätssicherung in Schulen“ beauftragt, die Erarbeitung der Bildungsstandards für die gymnasiale Oberstufe zu organisieren, ein Kompetenzmodell zu entwickeln und dabei die bisherigen Leistungen der EPA sowie andere bestehende Vereinbarungen zu berücksichtigen (vgl. KMK, 2007).

- Qualitätsentwicklung im Bildungssystem
- Gerechtigkeit im Bildungssystem

Am Ende eines jeden Zielkomplexes erfolgt jeweils eine Zusammenfassung mit Schlussfolgerungen für das Fach Informatik bzw. für diese Arbeit aus der Sicht des Autors.

### Strukturierung geforderter Kompetenzen schulischer Bildung

**Wenn Bildungsstandards festgelegt werden sollen, so muss zuvor über relevante Bildungsinhalte diskutiert werden.** Vollmer (2005, S. 1) würdigt die Diskussion, die durch die Entwicklung von Bildungsstandards und die parallele Veränderung im Schulsystem entstanden ist: *Die Tatsache, dass wir uns in vielen Fächern erstmals auf eine klare Benennung der Kernideen und damit der elementaren fachlichen Strukturen und ihrer Bildungspotenziale verständigt haben und konsequent von Modellen der fachlichen Kompetenz (statt von mehr oder minder „trägen" Wissensinhalten) ausgehen, kann gar nicht hoch genug gewürdigt werden. Die Bemühungen um eine kompetenzorientierte Ausrichtung des Lehrens und Lernens an unseren Schulen stellen sich als eine radikale Neuverortung dar und als wesentlicher Schritt in Richtung auf eine Neudefinition von Bildungsgehalt und von Bildungsperspektiven im Sinne des Aufbaus einer dynamischen, vernetzten Wissens- und Handlungsbasis für lebenslanges (Weiter-)Lernen.* Auch Neuweg (2004, S. 2 f.) begrüßt *die Chance einer neuen und breitflächigen Sensibilisierung für das Problem der Auswahl und Legitimation von Lehrzielen und -inhalten [sowie] die Aussicht auf eine deutlich reflektiertere Lösung dieses Problems.* Die KMK (2004a, S. 11) betont die Chance der Entwicklung einer anforderungsbezogenen Aufgabenkultur durch Bildungsstandards. Es stellt sich die Frage, wer festlegen soll, welche Kompetenzen in einem Fach von Schülerinnen und Schülern einer bestimmten Jahrgangsstufe erwartet werden können. Neuweg (2005, S. 3) behauptet, dass es *unter Fachwissenschaftler/inne/n und Fachdidaktiker/inne/n keine einheitliche Auffassung darüber gibt, was am Fach in welcher Weise relevant wäre. „Der" aktuelle Stand der wissenschaftlichen Fachdidaktik, der Grundlage für die den Standards unterliegenden Kompetenzmodelle sein soll, existiert so nicht.* Die Frage nach der Relevanz von geforderten Kompetenzen sollte nach Neuweg (2005, S. 3) in jedem Fall diskursiv, unter Einschluss von Stimmen der Öffentlichkeit, möglichst transparent diskutiert werden.

**Bildungsstandards machen Ziele und Anforderungen der Schule transparenter.** Durch Bildungsstandards werden nach Klieme u. a. (2003, S. 47 ff.) Ziele

und Anforderungen der Schule für alle Beteiligten transparenter. Sie können einen Bezugspunkt für Gespräche zwischen Lehrkräften, Lernenden und deren Eltern bieten (vgl. Klieme u. a., 2003, S. 48). Damit der Prozess des Lehrens und Lernens von allen Beteiligten verantwortungsvoll begleitet werden kann, ist es nach Reiss (2007, S. 27 f.) unverzichtbar, *dass sowohl Schülerinnen und Schülern als auch Eltern vermittelt wird, welche Kompetenzen im Verlauf der Schulzeit aufgebaut werden sollen. Alle müssen wissen, welche Ergebnisse man in einem konkreten Unterricht erwarten kann. [...] Will man die Verantwortung von Schülerinnen und Schülern sowie ihren Eltern stärken, dann wird es wichtig sein, in den Bildungsstandards die Ziele transparent, klar und einfach zu formulieren. Sie müssen so beschrieben werden, dass sie eindeutig mit Unterrichtsinhalten, Lernanforderungen und Aufgaben in Verbindung gebracht werden können.*

**Bildungsstandards können nicht alle Bildungsziele erfassen.** Herzog (2006, S. 10) beschreibt verschiedene Merkmale einer guten Schule, die nicht mit Bildungsstandards abbildbar sind. Er nennt unter anderem ein gutes Klima in der Lehrerschaft; Vorbereitung und Durchführung vieler Aktivitäten wie Feste, Ausflüge und Ausstellungen; eine unbürokratische Schulleitung sowie eine freundliche und lockere Atmosphäre. Klieme (2003, S. 16) räumt ein, dass es Bildungsziele im weiteren Sinne gibt, die nicht empirisch prüfbar sind, wie *z. B. die Idee von Mündigkeit, von Kreativität, von Entwicklung individueller Persönlichkeit.* Er stellt weiterhin dar, dass Bildungsstandards nicht alle Bildungsziele abdecken können, sondern sich lediglich auf einen zentralen Kern schulischer Bildung beziehen. Auch Bayrhuber (2007, S. 14) ist der Meinung, dass Kompetenzorientierung keinesfalls den gesamten Bildungsauftrag einer Schule umfasst und dass sich allgemeine Ziele wie die Entwicklung von Selbstbestimmung, sozialer Verantwortung, Leistungsbereitschaft und Motivation nicht in Leistungsstufen normieren lassen. In der Expertise von Klieme u. a. (2003, S. 99) wird dargelegt, dass – im Unterschied zu den inhaltlichen Vorgaben der Lehrpläne – Bildungsstandards mit ihren konkreten Kompetenzanforderungen einen Maßstab zur Bewertung von Lernergebnissen beinhalten. Die Konzentration auf das Messbare ist aber auch ein Kritikpunkt an den Bildungsstandards. Neuweg (2004, S. 5) stellt pointiert fest: *Die Einführung von Standards wird nicht einfach dazu führen, dass das gemessen wird, was wichtig ist, sondern dass vor allem das wichtig wird, was gemessen werden kann.* Das können nach seiner Auffassung aber im günstigsten Fall nur allgemeinverbindlich definierte, anspruchsvolle, konvergente kognitive Leistungen sein. Neuweg (2004, S. 5) legt weiter dar, dass *der „Rest“ [...] im Nicht-Gemessenen, Nicht-Honorierten und damit mit einiger Wahrscheinlichkeit im erst gar nicht ernsthaft Angestrebten*

*zu verschwinden [droht].*[15]

**Zusammenfassung und Schlussfolgerungen** Durch die Diskussion um Bildungsstandards werden auch die Ziele einer allgemeinen Schulbildung sowie die Lerninhalte verschiedener Fächer neu diskutiert. Es stellt sich allerdings die Frage, wer festlegen soll, welche Kompetenzen von Schülerinnen und Schülern zu fordern sind, bzw. was der Kern schulischer Bildung ist. Sicher ist, dass Bildungsstandards nicht alle Aspekte einer guten Schulbildung erfassen können. Dies gilt insbesondere, wenn das Erfüllen der Bildungsstandards überprüfbar sein soll. Fraglich ist, was mit den Zielen schulischer Bildung passiert, die nicht ohne Weiteres messbar sind. Trotzdem machen klar formulierte Bildungsstandards zumindest die messbaren Ziele und Anforderungen der Schule für Lehrkräfte, Eltern, Kinder und Abnehmer wie Betriebe und Universitäten transparenter.

Die neu entfachte Diskussion um die Ziele allgemeiner Schulbildung sowie die Lerninhalte einzelner Fächer stellt auch eine Chance für den Informatikunterricht dar. Gerade im vergleichsweise jungen Schulfach Informatik erscheint eine Strukturierung der geforderten Kompetenzen sinnvoll. Wenn informatische Kompetenzen, die sich an allgemeinen Bildungszielen orientieren, transparent benannt werden können, so kann sich dies positiv auf die Entwicklung des Faches auswirken und alle beteiligten Interessengruppen können sich an diesen Kompetenzen orientieren. Dabei muss der Versuchung widerstanden werden, den Informatikunterricht auf das Messbare zu reduzieren.

### Qualitätsentwicklung im Bildungssystem

**Bildungsstandards können zur Qualitätsentwicklung schulischer Bildung beitragen.** Nach Auffassung der KMK (2004a, S. 12) fördern Bildungsstandards *die Unterrichtsplanung im Hinblick auf definierte Leistungserwartungen, die diagnostische Kompetenz bei Lehrerinnen und Lehrern, den Umgang mit Heterogenität, die Evaluation von Unterricht durch interne und externe Verfahren und die Arbeit*

[15] Neuweg (2005, S. 4) stellt außerdem dar: *Zweifellos setzt eine Schule beispielsweise hohe Standards, wenn sie Neugierde weckt, Lebensfreude vermittelt oder zumindest versucht, sie nicht zu beschädigen, wenn sie intellektuelle Disziplin schult, Zivilcourage grundlegt oder Mut macht, sich 'letzten Fragen' zu stellen. Und gewiss setzt und erreicht ein Deutschlehrer hohe Standards, wenn einige seiner Schüler/innen sich durch die im Goethe-Faust aufgeworfene Sinnfrage ergriffen zeigen, ebenso wie dies für die Mathematiklehrerin gilt, die zumindest dann und wann sichtbar machen kann, dass und warum man die Mathematik als Formalwissenschaft in einem intellektuell-ästhetischen Sinne schätzen kann. Auf die jeweils zugehörigen 'Standard-Tests' und das entsprechende 'Standard-Monitoring' darf man freilich gespannt sein.*

*mit den Lehrplänen.* Paulsen und von Saldern (2005, S. 3) stellen fest, dass eine *eindeutige, hergeleitete Begründung für Bildungsstandards* fehlt. Insbesondere das gute Abschneiden skandinavischer Länder wird oft als Indiz für die Wirksamkeit von systematischer Rechenschaftslegung über die Ergebnisse schulischer Bildung angeführt (vgl. z. B. KMK, 2004a, S. 5 ff.). Es ist allerdings schwer zu beweisen, dass die Einführung von Bildungsstandards ein Schulsystem verbessert bzw. verschlechtert hätte, denn verschiedene Gesellschaften unterscheiden sich nicht nur in einem isolierbaren Bereich (vgl. Reiss, 2007, S. 21).

**Das Überprüfen des Erreichens von Bildungsstandards ist nur dann sinnvoll, wenn auch entsprechende Schlussfolgerungen gezogen und passende Maßnahmen ergriffen werden.** Nach Reiss (2007, S. 21 ff.) reicht die alleinige Einführung von Bildungsstandards nicht aus, um die Qualität schulischer Bildung zu verbessern, sondern es ist wichtig, begleitende Maßnahmen bereitzustellen. Neuweg (2005, S. 11) vergleicht den Übergang von einer Inputsteuerung zu einer Outputsteuerung mit dem Versuch, *ein Rennpferd dadurch schneller zu machen, dass man aufhört, es zu füttern, stattdessen aber laufend seine Rundenzeiten stoppt. Schulen erreichen Standards nicht durch ihre Messung, sondern durch die Zurverfügungstellung der den gesollten Output ermöglichenden Inputs. Es ist deshalb sehr zu hoffen, dass sich die Entscheidung für eine Planbewirtschaftung des Bildungswesens über Zielvorgaben, Messvorgänge und Ergebnisrückmeldungen mit einer Konzentration auch auf die Kontextbedingungen verbindet: auf die gezielte Aus- und Fortbildung des Personals, auf die Optimierung von Organisationsstrukturen und Rahmenbedingungen, auf das Setzen innovationsförderlicher Anreize und auf die gezielte Unterstützung von Entwicklungsprozessen.* Das Zitat von Neuweg zeigt, dass bei aller Kontrolle und Analyse des Outputs eines Bildungssystems natürlich entsprechende Maßnahmen – also der Input – nicht vergessen werden dürfen. [16]

Demmer (2003, S. 8) unterscheidet drei Rollen von Bildungsstandards in einem Gesamtkonzept:

- *[Nationale Bildungsstandards] können nach angelsächsischem Vorbild Element in einem Marktmodell sein, das auf Konkurrenz zwischen den Schulen und perspektivisch auf (Teil-) Privatisierung setzt. Grundgedanke ist, dass Wettbewerb per se die Qualitätsentwicklung positiv unterstützt; durch nationale Tests wird das Erreichen der Bildungsstandards jährlich überprüft. Die*

[16] Kurz (2006, S. 435) stellt Dimensionen eines umfassenden Qualititässicherungssystems vor, bei dem die Ergebnisse des gemessenen Outputs auf den Input und die Bildungsstandards wirken.

*Ergebnisse werden in Rankinglisten veröffentlicht.*

- *Nationale Bildungsstandards können – deutschsprachiger Tradition folgend – die frühe und permanente Selektion „vervollkommnen". In diesem Konzept werden für die „Gelenkstellen" des Schulsystems Bildungsstandards als Hürden konzipiert, die Schülerinnen und Schüler zu überwinden haben, um die Versetzung, den Übergang auf bestimmte Schulformen oder Schulabschlüsse zu erreichen.*

- *Nationale Bildungsstandards können aber auch nach skandinavischem Vorbild Element in einem demokratischen, auf Partizipation und Förderung ausgerichteten Qualitätskonzept sein. In diesem Fall sind Bildungsstandards Orientierungsmarken, für deren Erreichen die Gesellschaft bzw. die Schulen mit den jungen Menschen einen Lernkontrakt eingehen.*

Je nach der Rolle von Bildungsstandards im Gesamtkonzept sind unterschiedliche Konsequenzen denkbar. Auch der Zeitpunkt der Überprüfung sollte von den möglichen Reaktionen abhängen.[17] Um Bildungsstandards umzusetzen, ist es wichtig, dass diese in der Lehrerschaft Akzeptanz finden (vgl. Blum, 2006, S. 19 bzw. Reiss, 2007, S. 29). Lehrerinnen und Lehrer interessieren sich im Allgemeinen nicht für Probleme der nationalen bildungspolitischen Systemsteuerung, da sie täglich mit anderen Problemen konfrontiert sind (vgl. hierzu Herzog, 2006, S. 12). Reiss (2007, S. 28 f.) fordert, dass Lehrkräfte durch Fortbildungsmaßnahmen inhaltlich und methodisch auf den Wechsel von Lehrplänen zu Bildungsstandards vorbereitet werden müssen und dass Schulen mit schlechten Leistungen durch Förderprogramme unterstützt werden (vgl. hierzu auch Klieme u. a., 2003, S. 110 ff.).[18] Nach Paulsen und von Saldern (2005, S. 32) sollte sich die Rolle der Schülerinnen und Schüler mit der Einführung von Bildungsstandards idealer Weise von „Defizitträgern" zu „Förderzielen" wandeln.

[17] Klieme (2003, S. 18) hält es für vernünftig, standardbezogene Tests, die zur Evaluation genutzt werden, in einer Zwischenphase einzusetzen, da Schulen so von den Rückmeldungen vermutlich mehr profitieren können und sich dadurch auch nationale Prüfungen bzw. zentral gesteuerte Tests vermeiden lassen. Paulsen und von Saldern (2005, S. 25 ff.) weisen darauf hin, dass die Zahl der Tests unbedingt begrenzt werden sollte. Nach ihrer Vorstellung bedarf die konsequente Umsetzung von Bildungsstandards keiner parallelen Leistungserhebung. Sie stellen weiterhin dar, dass der Gefahr des „teaching to the test" mit guten Bildungsstandards, die dem Kriterium der Kumulativität entsprechen, entgegengewirkt und sogar ein „teaching to the objectives" ermöglicht werden kann, bei dem die Bildungsziele handlungsleitend sind (vgl. hierzu auch Blum, 2006, S. 18).

[18] Neuweg (2005, S. 11) fordert, dass Schulleitung und Schulaufsicht gegenüber Lehrerinnen und Lehrern bei mangelnden beruflichen Leistungen gegebenenfalls die Möglichkeiten des Beamtenrechts ausschöpfen sollten.

**Zusammenfassung und Schlussfolgerungen** Dass Bildungsstandards zur Qualitätsentwicklung schulischer Bildung beitragen können, wird meist nicht ausführlich begründet. Belege hierfür zu finden ist aufgrund der großen Anzahl an Faktoren, die die Qualität von Bildung z. B. in einem Land beeinflussen, allerdings auch schwer. Positive Effekte lassen sich überhaupt nur dann vermuten, wenn das Erreichen von Bildungsstandards nicht nur gemessen wird, sondern aufgrund von Testergebnissen gezielt Maßnahmen für Länder, Schulen, Lehrkräfte, Schülerinnen und Schüler ergriffen werden (können). Schulentwicklungsprogramme, Anreizsysteme für Schulen, Fortbildungen für Lehrkräfte und die individuelle Förderung von Schülerinnen und Schülern seien als Beispiele für mögliche Maßnahmen genannt. Der Zeitpunkt, zu dem das Erreichen der Bildungsstandards überprüft wird, sollte abhängig von den ergreifbaren Maßnahmen festgelegt werden.

Bevor Bildungsstandards zur Qualitätsentwicklung des Informatikunterrichts beitragen können, müssen Einsatzszenarien erprobt werden. Einen exemplarischen Beitrag hierzu sollen die in den Kapiteln 7 und 8 vorgestellten Untersuchungen liefern, in denen Lehrkräfte kompetenzorientierten Informatikunterricht auf Grundlage eines Kompetenzmodells unter Verwendung einer Aufgabensammlung und eines passenden Werkzeugs durchführen. Die beteiligten Lehrpersonen können in ihrem Unterricht Schülerinnen und Schüler mithilfe der bereitgestellten Materialien individuell fördern.

### Gerechtigkeit im Bildungssystem

**Einheitliche Bildungsstandards können einen Beitrag zu mehr Gerechtigkeit in einem Bildungssystem leisten.** Für Klieme (2003, S. 16 ff.) wird durch bildungsgang- und schulformübergreifende Mindestanforderungen zum Ausdruck gebracht, *dass sich unsere Gesellschaft dazu verpflichtet, jedem und jeder Jugendlichen bestimmte Kompetenzen mit auf den Weg zu geben.* Außerdem können seiner Meinung nach national einheitliche Bildungsstandards einen Beitrag dazu leisten, für mehr Gerechtigkeit im Bildungssystem zu sorgen, das heißt die Unterschiede zwischen Ländern in Bezug auf Benotungsmaßstäbe, soziale Disparitäten oder Unterschiede zwischen Zugewanderten und hier Geborenen abzubauen. Die KMK (2004a, S. 10) betont ebenfalls, dass durch die Entwicklung von Bildungsstandards und durch andere Maßnahmen versucht wird, die *Gleichwertigkeit der schulischen Ausbildung, die Vergleichbarkeit der Schulabschlüsse sowie die Durchlässigkeit des Bildungssystems sicher zu stellen* und somit der Forderung nach gleichen Bildungschancen für alle Schülerinnen und Schüler zu entsprechen. Neuweg (2004, S. 10) bemerkt kritisch: *Wer sich zu Mindeststandards* ***bekennt****, der wird besondere Anstrengungen unternehmen, um [...] SchülerInnen [aus unteren Sozialschichten*

*und Migrantenfamilien] in besonderer Weise zu unterstützen. Wer sie aber bloß* ***erfüllen*** *will, kann es sich leichter machen, indem er diese SchülerInnen entweder erst gar nicht aufnimmt oder aber sich zeitgerecht wieder von ihnen trennt – auch dann werden „die Daten stimmen".*

**Die Überprüfung von Bildungsstandards sollte keinen Einfluss auf Noten bzw. Übergangsempfehlungen haben.** Neuweg (2005, S. 5) stellt dar, dass die schulische Laufbahn von Schülerinnen und Schülern aufgrund von Entscheidungen festgelegt wird, die *elementaren Gerechtigkeitskriterien gegenwärtig nicht genügen.*[19] Gleiche Leistungen können in einem Land oder sogar in einer Schule zu unterschiedlichen Noten führen (vgl. z. B. Klieme, 2004a, S. 626). Es stellt sich die Frage, ob überprüfbare Bildungsstandards als weniger subjektives Kriterium hier Abhilfe schaffen können. In der Klieme-Expertise (2003, S. 10) wird ausdrücklich darauf hingewiesen, *dass Tests, die im Bildungsmonitoring und für die Schulevaluation eingesetzt werden, [...] Individualdiagnostik aus methodischen Gründen meist nicht erlauben. Von einer Verwendung der Standards bzw. standard-bezogener Tests für Notengebung und Zertifizierung wird abgeraten.*[20] Neuweg (2005, S. 7) stellt fest, dass standard-bezogene Testverfahren zwar objektiver sind, aber dass *das auf ausgedehnte Beobachtungen und vielfältige Leistungsmessungen zurückgehende Lehrer/innenurteil [...] relevanter, valider und reliabler sein kann.* Paulsen und von Saldern (2005, S. 32) legen dar, dass das Leistungsniveau der Lernenden auch von Faktoren bestimmt wird, auf die diese keinen Einfluss haben wie: *didaktische Qualität des Lehrkörpers, Ausstattung durch Land und Schulträger, Schülerzusammensetzung usw. Diesbezügliche Erkenntnisse dürfen nicht zulasten des Lernenden interpretiert werden, da diese am wenigsten Verursacher schlechter Ergebnisse sind.*

[19] Nach Paulsen und von Saldern (2005, S. 28) sind die Übergangsempfehlungen nach der Grundschule für ca. die Hälfte der Schülerinnen und Schüler falsch.

[20] Klieme u. a. (2003, S. 107 ff.) weisen ausdrücklich darauf hin, dass Individualdiagnostik nur bei entsprechendem Testdesign möglich ist und dass insbesondere die geringe individuelle Messgenauigkeit und das Bearbeiten von unterschiedlichen Testteilen durch verschiedene Schülerinnen und Schüler gegen die Weitergabe individueller Ergebnisse bei Monitoring- und Evaluationsstudien sprechen. Nach Reiss (2007, S. 29 f.) gibt es aufgrund der *Stichprobenauswahl [bei großen Schulleistungsstudien] [...] klare Grenzen, über welche Gruppen noch zuverlässige Aussagen gemacht werden können. Schwierig ist es, mit den aus diesen großen Studien gewonnenen Daten auch kleinere Gruppen zu beurteilen [...] Im Prinzip ausgeschlossen ist es, nur über solche Verfahren die Leistungen von Individuen zu bewerten.* Klieme u. a. (2003, S. 99 ff.) geben einen Überblick über verschiedene Fragestellungen und Ziele verschiedener Testverfahren bei der Überprüfung von Bildungsstandards.

**Zusammenfassung und Schlussfolgerungen** Wenn sich die schulische Bildung eines ganzen Landes an einheitlichen Standards orientiert, so kann das zu einer besseren Vergleichbarkeit von Abschlüssen führen. Das von Neuweg (2004, S. 10) überspitzt dargestellte „bloße Erfüllen“ der Standards muss dabei verhindert werden. Die Ergebnisse von Standardtests, die im Rahmen von Bildungsmonitoring bzw. Schulevaluation eingesetzt wurden, sind nicht zur Bewertung von Einzelleistungen, als Kriterien für Notengebung, Zertifizierung bzw. Übergangsempfehlungen geeignet.

Wenn der Informatikunterricht in verschiedenen Ländern und Schulen auf das Erfüllen derselben Bildungsstandards ausgerichtet ist, so kann damit ein erster Beitrag zur Gerechtigkeit im Bildungssystem geleistet werden. Ein Ziel dieser Entwicklung muss es sein, der digitalen Spaltung entgegenzuwirken, dass bedeutet*, dass niemand mehr ohne grundlegendes Verständnis moderner digitaler Hilfsmittel bleiben darf* (vgl. GI, 2004). Das in dieser Arbeit zu entwickelnde Kompetenzmodell und die zugehörigen Aufgaben sind primär für den Einsatz im Unterricht bestimmt und sollen Lehrerinnen und Lehrer bei der Individualdiagnostik unterstützen. Dementsprechend spricht nichts dagegen, es den Lehrkräften in den Untersuchungen freizustellen, den entwickelten Test im Unterricht auch zur Notengebung einzusetzen.

## 2.3 ACM Model Curriculum for K-12 Computer Science

Das „Model Curriculum for K-12 Computer Science“ wurde von der ACM (2003) für den Informatikunterricht in der Schule entwickelt.[21] Es orientiert sich sich an den folgenden Zielen (vgl. ACM, 2003, S. 6):

- Alle Schülerinnen und Schüler sollen von der Primarstufe an in grundlegende Konzepte der Informatik eingeführt werden.
- Informatik soll in den Sekundarstufen I und II für alle Schülerinnen und Schüler fachlich angemessen angeboten werden.
- Für interessierte Schülerinnen und Schüler sollen zusätzliche Kurse angeboten werden, durch die es möglich ist, sich intensiver mit der Informatik zu beschäftigen und sich auf Arbeitsleben oder Studium vorzubereiten.
- Alle Schülerinnen und Schüler, insbesondere Minderheiten, sollen ihr Wissen im Bereich der Informatik erweitern.

[21] K-12 ist die amerikanische Bezeichnung für den Unterricht vom Kindergarten über die Primarstufe bis zur Sekundarstufe, dabei steht die 12 für die Anzahl der Schuljahre.

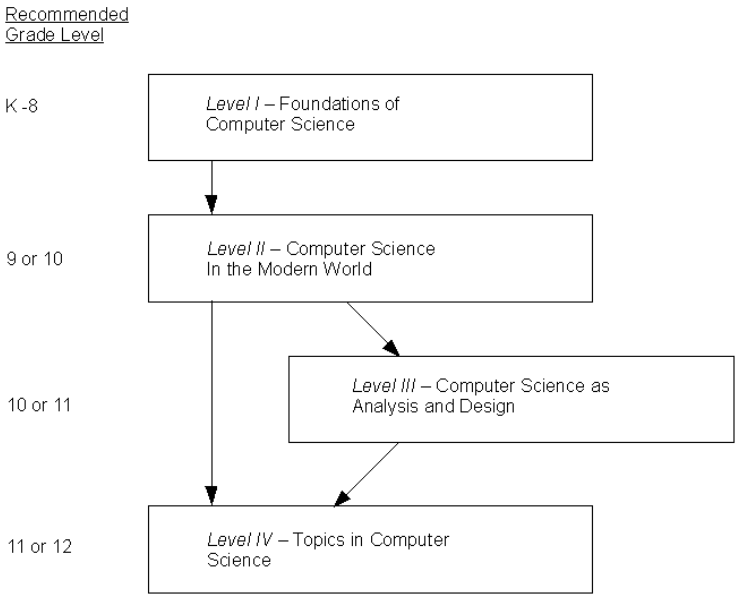



Abbildung 1: Struktur des Model Curriculum for K-12 Computer Science (vgl. ACM, 2003, S. 10)

Um ein eigenständiges Schulfach diesen Zielen entsprechend zu implementieren, sind nach Auffassung der ACM (2003, S. 20 ff.) verschiedene Herausforderungen zu bewältigen, wie z. B. die Ausbildung von Lehrkräften, das Entwickeln von inhaltlichen Bildungsstandards (bzw. Lehrplänen) und das Entwickeln von Unterrichtsmaterialien (sowohl Bücher als auch Material für die Arbeit mit dem Computer). Das Model Curriculum for K-12 Computer Science soll eine Basis für diese Entwicklungen bieten (vgl. ACM, 2003, S. 25). Es ist in vier Levels aufgeteilt, die im Folgenden kurz beschrieben werden (vgl. ACM, 2003, S. 10 ff.). Abbildung 1 zeigt die Struktur des Model Curriculum for K-12 Computer Science und verdeutlicht, dass nach dem Absolvieren von Level II direkt zu Level IV übergegangen werden kann.

**Level I – Foundations of Computer Science** In Level I werden die Kompetenzen angegeben, die Schülerinnen und Schüler am Ende der zweiten, fünften und achten Jahrgangsstufe erworben haben sollen (vgl. ACM, 2003, S. 11 ff.).:

- Am Ende der zweiten Jahrgangsstufe sollen die Schülerinnen und Schüler erste Erfahrungen mit Computern gemacht haben. Sie sollen Computer mit den typischen Ein- und Ausgabegeräten in verschiedenen Situationen einsetzen können, z. B. zur Unterstützung beim Lernen, als Kommunikationsmittel,

als Multimediasystem und als Nachschlagewerk. Dabei sollen auch erste typisch informatische Inhalte wie die Darstellung von Information mit Nullen und Einsen und einfache Sortieralgorithmen (ohne Computereinsatz) erlernt werden.

- Am Ende der fünften Jahrgangsstufe sollen alle Schülerinnen und Schüler in der Lage sein, verschiedene Aufgaben mit unterschiedlichen Werkzeugen am Computer selbstständig zu bewältigen. Außerdem sollen sie über den Einsatz und den verantwortlichen Umgang mit Technologie diskutieren können und ein einfaches Verständnis von Algorithmen wie Textkompression oder Suche (ohne Computereinsatz) entwickeln.
- Am Ende der achten Jahrgangsstufe werden von den Schülerinnen und Schülern verschiedene spezifische Kenntnisse in den Bereichen Werkzeuge, Software, Hardware und Simulation sowie deren praktischer Anwendung gefordert. Dabei sollen sie zusammenarbeiten und Produkte wie Internetseiten oder Videos entwickeln, veröffentlichen und präsentieren. Außerdem wird erwartet, dass sie einfache Probleme, die im täglichen Umgang mit Computern entstehen, lösen und aktuelle Veränderungen in der Informationstechnologie sowie deren Auswirkungen beschreiben können. Als typisch informatische Inhalte sollen die Schülerinnen und Schüler am Ende der achten Jahrgangsstufe Graphen als Werkzeug zur Veranschaulichung von Zuständen realer Systeme und die grundlegenden Konzepte der Logik und deren Nutzen verstehen.

Diese Kompetenzbeschreibungen des Level I orientieren sich an den „National Education Technology Standards for Students“ der ISTE (1998) und erweitern diese um Algorithmik und weitere typische informatische Inhalte (vgl. ACM, 2003, S. 12).[22] Um die geforderten Kompetenzen zu erreichen, wird von der ACM (2003, S. 10 f.) vorgeschlagen, kurze Module in vorhandenen Fächern wie Mathematik, den naturwissenschaftlichen Fächern und Sozialkunde (Social Studies) hinzuzufügen.

**Level II – Computer Science in the Modern World** Level II beschreibt einen einjährigen Kurs in der neunten oder zehnten Jahrgangsstufe, dessen Ziel es ist, allen Schülerinnen und Schülern eine Einführung in die Prinzipien der Informatik und ein Verständnis von Computerhardware, -software, -sprachen, -netzen und deren Auswirkungen in der modernen Welt zu vermitteln (vgl. ACM, 2003, S. 14 ff.). Um die von der ACM angegebenen inhaltlichen Ziele zu erreichen, wird vorgeschlagen, die Entwicklung von Algorithmen und Internetseiten im Unterricht zu behandeln (vgl. ACM, 2003, S. 14 f.).

[22] ISTE steht für International Society for Technology in Education.

Beim Thema „Algorithmen“ ist der Lehrkraft die Wahl der konkreten Programmiersprache und -umgebung freigestellt. Empfohlen wird die Vermittlung der folgenden Konzepte (vgl. ACM, 2003, S. 15):

- Variablen, Datentypen und die Darstellung von Daten im Computer
- die Reduzierung von Komplexität durch Top-Down-Vorgehen und objektorientiertes Design
- Prozeduren und Parameter
- Sequenzen, Verzweigungen, Schleifen
- Darstellungsmöglichkeiten wie Datenflussdiagramme, Pseudocode, UML, Struktogramme

Beim Thema „Erstellen von Internetseiten“ werden folgende Konzepte empfohlen (vgl. ACM, 2003, S. 15):

- das Nutzen von Links, um neue Seiten zu laden oder Prozesse zu aktivieren
- Speichern, Komprimieren, Entschlüsseln und Abrufen von Bildern, Videos und Sounds
- Design von Benutzungsschnittstellen
- Werkzeuge zum Darstellen von Designs (storyboard, site map)

**Level III – Computer Science as Analysis and Design** Level III beschreibt einen einjährigen Kurs in der zehnten oder elften Jahrgangsstufe, dessen Ziel es ist, die erworbenen Kenntnisse auf dem Gebiet der Informatik zu vertiefen (vgl. ACM, 2003, S. 16). Schülerinnen und Schülern soll es in diesem Kurs möglich sein, Informatik auch über die grundlegenden Konzepte hinaus kennenzulernen (vgl. ACM, 2003, S. 16). Um die von der ACM (2003, S. 16 f.) inhaltlich vorgeschlagenen Ziele zu erreichen sollen das Problemlösen mit Hilfe von Algorithmen und Programmierung unter Verwendung der Prinzipien der Softwareentwicklung vertieft werden. Dabei soll besonderer Wert auf die wissenschaftlichen und ingenieurmäßigen Merkmale der Informatik gelegt werden (vgl. ACM, 2003, S. 16). Die Wahl der konkreten Programmiersprache und -umgebung ist der Lehrkraft wieder freigestellt. Folgende Konzepte sollten vermittelt werden (vgl. ACM, 2003, S. 16):

- Methoden (Funktionen) und Parameter
- Rekursion
- Objekte und Klassen (Felder, Vektoren, Stapel, Schlangen und deren Einsatz)

- Programmieren von Grafiken
- Ereignisbehandlung und Interaktionen

Außerdem können verschiedene weitere Gebiete behandelt werden wie (vgl. ACM, 2003, S. 16 f.):

- Hardware und Systeme: logische Schaltungen, binäre Arithmetik, Maschinensprache, Betriebssysteme, Benutzungsschnittstellen, Compiler
- Ingenieurmäßige Softwareentwicklung: Anforderungen, Design, Teamarbeit, Testen und Warten, Dokumentieren, Software-Designwerkzeuge
- Soziale Aspekte der Softwareentwicklung: Grenzen des Einsatzes von Computern, Sprachhierarchien, Karrieremöglichkeiten im Bereich der Informatik

**Level IV – Topics in Computer Science** In diesem Kurs, der in der elften oder zwölften Jahrgangsstufe angesiedelt ist, sollen interessierte und qualifizierte Schülerinnen und Schüler aus verschiedenen Angeboten auswählen können, um ein tieferes Verständnis und erweiterte Fähigkeiten in einem spezifischen Bereich der Informatik zu erlangen (vgl. ACM, 2003, S. 17). Als Angebote sind genannt (vgl. ACM, 2003, S. 17 ff.):

- Eine Art Leistungskurs Informatik (Advanced Placement Course), dieser kann sich inhaltlich an den Ausführungen von The College Board (2007) anlehnen und richtet sich an Schülerinnen und Schüler, die mindestens Level I und II absolviert haben.
- Ein projektorientierter Kurs, bei dem Schülerinnen und Schüler, die mindestens Level I und II (bei einigen Themen auch Level III) absolviert haben, professionelle Fähigkeiten in einem bestimmten Bereich der IT-Anwendung erwerben. Als Beispiele werden genannt: Desktop Publishing, Präsentationen, Multimedia, Grafik, Design und Entwicklung von Internetseiten, Web Programmierung, Neue Technologien, Computer und Animation, Netztechnologien, Programmsimulation, objektorientiertes Design und Programmieren, effektiver Einsatz von Computeranwendungen.
- Ein Kurs, der zu einem Industrie-Zertifikat führt. Hierfür sollten die Schülerinnen und Schüler mindestens Level I und Level II absolviert haben. Bei produktspezifischen Angeboten sollte genau überprüft werden, ob generelle Konzepte oder produktbezogene Anwendungskenntnisse vermittelt werden. Als produktneutrale Zertifikate werden genannt: A+ Certified Technician, Certified Internet Webmaster (CIW) und i-Net+.

**Einordnung** Mit dem ACM Model Curriculum for K-12 Computer Science wurden Ziele des Informatikunterrichts angegeben und Möglichkeiten der inhaltlichen Ausgestaltung genannt. Dabei wurde versucht, an in den USA bereits existierende Strukturen, wie z. B. die National Education Technology Standards for Students, die Industriezertifikate oder die AP-Computer-Science-Kurse anzuknüpfen. Insgesamt soll ein eigenständiges Fach Informatik (Computer Science) etabliert werden, das über Benutzerfertigkeiten hinaus typisch informatische Inhalte vermittelt.
Das in dieser Arbeit zu entwickelnde Kompetenzmodell soll sich am Output schulischer Bildung orientieren. Die größtenteils inputorientierte inhaltliche Beschreibung des Unterrichts in den Levels II, III und IV sprechen gegen das Verwenden des ACM Model Curriculum for K-12 Computer Science als Grundlage für dieses Kompetenzmodell.

## 2.4 EPA Informatik

Um die Transparenz, Vergleichbarkeit und Einheitlichkeit der Prüfungsverfahren und -anforderungen in der Abiturprüfung zu sichern, wurden von der KMK (2002, S. 1 f.) einheitlichen Prüfungsanforderungen festgelegt, die konkrete Lern- und Prüfungsbereiche beschreiben und wichtige Hilfen zur Konstruktion von Prüfungsaufgaben und zur Bewertung von Prüfungsleistungen bereitstellen. In einem Beschluss der KMK (2004b) wurden die Länder gebeten, die aktualisierten EPA Informatik (und die EPA anderer Fächer) spätestens zur Abiturprüfung im Jahre 2007 umzusetzen. In den Prüfungsanforderungen werden nach einer Präambel fachliche Inhalte und Qualifikationen, Anforderungsbereiche sowie die schriftliche und die mündliche Prüfung beschrieben (vgl. KMK, 2004b). Außer diesen Festlegungen für die Gestaltung der Abiturprüfung sind die EPA Informatik mit einer Menge von konkreten Aufgabenbeispielen untersetzt (vgl. KMK, 2004b, S. 16 ff.). Fothe (2004, S. 18) erläutert, dass die EPA Informatik auf der Grundlage der derzeit geltenden Informatiklehrpläne der 16 Länder erarbeitet wurden und grundsätzlich nur Prüfungsanforderungen und keine Lehrpläne oder Unterrichtskonzepte enthalten. Er erwartet aber durch die inhaltliche Schwerpunktsetzung und Strukturierung der EPA eine Einflussnahme auf künftige Informatiklehrpläne.

Die EPA Informatik nennen die vier Kompetenzbereiche (vgl. KMK, 2004b, S. 4 f.):

- *Erwerb und Strukturierung informatischer Kenntnisse*
- *Kennen und Anwenden informatischer Methoden*
- *Kommunizieren und Kooperieren*

- *Anwenden informatischer Kenntnisse, Bewerten von Sachverhalten und Reflexion von Zusammenhängen*

Im Gegensatz zur Fassung von 1989 haben der Algorithmus und die technische Informatik in den EPA Informatik in der Fassung von 2004 einen geringeren Stellenwert; die Modellierung hat entsprechend der fachdidaktischen Erkenntnisse an Bedeutung gewonnen (vgl. Fothe, 2005, S. 48). So werden folgende sechs Modellierungstechniken genannt, von denen zwei im Grundkurs und drei im Leistungskurs beherrscht werden müssen (vgl. KMK, 2004b, S. 5 f.):

- *Objektorientierte Modellierung*
- *Datenmodellierung*
- *Zustandsorientierte Modellierung*
- *Modellierung von Abläufen mit Algorithmen*
- *Funktionale Modellierung*
- *Regelbasierte Modellierung*

Neben diesen grundlegenden Modellierungstechniken werden außerdem „Interaktion mit und von Informatiksystemen“ und „Möglichkeiten und Grenzen informatischer Verfahren“ als fachliche Inhalte beschrieben (vgl. KMK, 2004b, S. 6). Die Abituraufgaben einzelner Länder können bis zu einem Drittel landesspezifische in den EPA Informatik nicht genannte Themen beinhalten (vgl. KMK, 2004b, S. 12).

**Einordnung** Die EPA Informatik beschreiben Prüfungsanforderungen und beziehen sich somit auf den Output schulischen Lernens. Da die EPA dazu dienen sollen, Prüfungsaufgaben zu konstruieren bzw. auf ihre Eignung hin zu überprüfen, erscheint eine Einordnung der EPA Informatik als Mindest-, Regel- oder Maximalstandards nicht sinnvoll. Insbesondere die zahlreichen Aufgabenbeispiele der EPA Informatik veranschaulichen, welche Anforderungen an Prüflinge gestellt werden können.[23] Fothe (2005, S. 48 f.) charakterisiert die einheitlichen Prüfungsanforderungen für das Fach Informatik anhand der von Klieme u. a. (2003, S. 24 ff.) beschriebenen Merkmale guter Bildungsstandards und legt dar, dass die EPA Informatik bereits einige dieser Merkmale besitzen. Außerdem stellt er fest, dass die EPA Informatik so formuliert wurden, dass sie nicht alle drei Jahre überarbeitet werden müssen. Hierzu trägt insbesondere die oben genannte Regelung bei, dass

[23] Kritisch sieht Hartmann (2005) eine Beispielaufgabe zum abstrakten Datentyp Dictionary, da für die meisten Schülerinnen und Schüler ein Anwendungsbezug nur schwer herstellbar ist, der Bildungswert der Aufgabe für Kollegen anderer Fächer schleierhaft bleibt und eine Verknüpfung zu anderen Fächern unvorstellbar erscheint.

bis zu einem Drittel Aufgaben zu Themen gestellt werden können, die nicht in den EPA genannt sind. Fothe (2005, S. 49) stellt zur Diskussion, *ob künftige inhaltliche Regelungen für die Abiturprüfung als Regel- oder Mindeststandards formuliert werden sollten [...], ob es sinnvoll und machbar ist, die Anforderungsbereiche der EPA in Kompetenzstufen zu überführen und ob das Merkmal Kumulativität bewusst beim Erarbeiten von Prüfungsanforderungen beachtet werden sollte und wie dies ggf. geschehen kann.*[24] Mit der Ausrichtung auf die Abiturprüfung geben die EPA Ziele des Informatikunterrichts der Sekundarstufe II an. Dadurch erscheinen sie als Grundlage für das in dieser Arbeit zu entwickelnde Kompetenzmodell und die durchzuführende Untersuchung zum kompetenzorientierten Informatikunterricht in der Sekundarstufe I ungeeignet.

## 2.5 GI-Empfehlungen „Grundsätze und Standards für die Informatik in der Schule“

Die GI-Empfehlungen „Grundsätze und Standards für die Informatik in der Schule“ (nachfolgend kurz GI-Empfehlungen genannt) wurden vom Arbeitskreis „Bildungsstandards“ der GI entwickelt (vgl. GI, 2008). Die Arbeiten wurden von Lehrerinnen und Lehrern sowie Fachdidaktikerinnen und Fachdidaktikern unterstützt. Am Ende der Empfehlungen sind immerhin 80 Personen namentlich genannt, die besonders aktiv an der Entwicklung beteiligt waren. Die GI-Empfehlungen lehnen sich nicht nur im Titel, sondern auch strukturell an die „Principles and Standards for School Mathematics“ des NCTM (2000) an.[25] Beide enthalten eine Vision,

[24] Vorüberlegungen zur Entwicklung von Bildungsstandards für die Sekundarstufe II auf Grundlage der EPA Informatik präsentiert Fothe (2008).

[25] NCTM steht für National Council of Teachers of Mathematics.
Die vom NCTM im Jahr 2000 herausgegebenen „Principles and Standards for School Mathematics“ sind ein international bekanntes und einflussreiches Rahmenkonzept eines modernen Mathematikunterrichts, in dem versucht wird, mathematische Grundbildung über Schulstufen hinweg übergreifend zu beschreiben (vgl. Klieme u. a., 2003, S. 36. f.). Die Grundsätze und Standards des NCTM (2000, S. 2) beginnen mit einer Vision von „gutem“ Mathematikunterricht. Danach folgt ein Kapitel, in dem die Prinzipien vorgestellt werden, nach denen gehandelt werden sollte, um diese Vision zu erreichen (vgl. NCTM, 2000, S. 10 ff.). Im Anschluss an diese Prinzipien werden vom NCTM (2000, S. 29 ff.) Bildungsstandards, aufgeteilt in Inhalte und Prozesse, jahrgangsübergreifend vorgestellt. Darauf folgen detaillierte Erläuterungen zu den Bildungsstandards in den Altersstufen Kindergarten bis Jahrgangsstufe 2, Jahrgangsstufe 3 bis 5, Jahrgangsstufe 6 bis 8 und Jahrgangsstufe 9 bis 12 mit Beispielaufgaben (NCTM, 2000, S. 108 ff.). In einem letzten Kapitel mit dem Titel „Working Together to Achieve the Vision“ wird noch einmal auf die Vision von „perfektem“ Mathematikunterricht eingegangen (NCTM, 2000, S. 367 ff.).

Grundsätze die zum Erreichen der Vision nötig sind, und eine Aufteilung in Inhalts- und Prozessbereiche (vgl. GI, 2008 und NCTM, 2000).[26]

Die GI-Empfehlungen sind in fünf Kapitel unterteilt. Im Vorwort werden Entwicklung und Ziele erläutert, die Arbeiten werden kurz eingeordnet und ein Ausblick auf weitere Aufgaben wird gegeben (vgl. GI, 2008, S. V). Im ersten Kapitel werden von der GI (2008, S. 1 ff.) Grundsätze eines guten Informatikunterrichts beschrieben, dessen Ziel es ist, dass *informatisch gebildete Menschen alle informatischen Probleme, die ihnen in ihrem Leben begegnen werden, mit Selbstvertrauen anpacken und selbstständig allein oder im Team bewältigen können.* Die Grundsätze, die die Voraussetzungen eines solchen Unterrichts bilden, werden unter den folgenden Überschriften zusammengefasst (vgl. GI, 2008, S. 3 ff.):

- *Chancengleichheit*
- *Curriculum*
- *Lehren und Lernen*
- *Qualitätssicherung*
- *Technikeinsatz*
- *Interdisziplinarität*

Im zweiten Kapitel werden von der GI (2008, S. 11 ff.) je fünf Inhalts- und Prozessbereiche vorgestellt, in die die GI-Empfehlungen aufgeteilt sind: *Die Inhaltsbereiche charakterisieren mindestens zu erwerbende fachliche Kompetenzen. Die Prozessbereiche beschreiben, auf welche Art und Weise die Schülerinnen und Schüler mit den genannten Fachinhalten umgehen sollen* (vgl. GI, 2008, S. IV). Folgende Inhalts- und Prozessbereiche werden genannt GI (2008, S. 11):[27]

- *Inhaltsbereiche:*
  - *Information und Daten*
  - *Algorithmen*
  - *Sprachen und Automaten*

[26] Bei näherer Betrachtung unterscheiden sich die „Grundsätze und Standards“ von NCTM und GI aufgrund der Ausrichtung auf unterschiedliche Fächer erheblich.

[27] Warum gerade die genannten und keine anderen Inhalts- und Prozessbereiche gewählt wurden, wird nicht begründet und ist in Anbetracht der Menge an Experten, die an diesen Empfehlungen mitgearbeitet haben, und der Anzahl an Diskussionen wohl nur schwer nachzuvollziehen. Begründete Bildungsstandards, die sich, wie von Klieme u. a. (2003, S. 10, 21 ff. und 119 ff.) gefordert, auf erprobte Kompetenzmodelle stützen, sind zwar wünschenswert, aber in der derzeitigen Forschungslage der Fachdidaktik Informatik kaum realisierbar (vgl. hierzu auch Abschnitt 3.2).

  - *Informatiksysteme*
  - *Informatik, Mensch und Gesellschaft*
- *Prozessbereiche:*
  - *Modellieren und Implementieren*
  - *Begründen und Bewerten*
  - *Strukturieren und Vernetzen*
  - *Kommunizieren und Kooperieren*
  - *Darstellen und Interpretieren*

Nach einem allgemeinen Überblick über die Inhalts- und Prozessbereiche folgt eine tabellarische Aufschlüsselung der geforderten Kompetenzen in die Jahrgangsstufen 5 bis 7 und 8 bis 10 (vgl. GI, 2008, S. 12 ff.). Das dritte Kapitel der GI-Empfehlungen beschäftigt sich ausführlich mit den Inhaltsbereichen (vgl. GI, 2008, S. 23 ff.). Dabei wird Wert darauf gelegt, dass diese nicht zur Unterteilung des Unterrichts in Themengebiete (z. B. *drei Wochen Algorithmen gefolgt von zwei Wochen Informatiksysteme*) genutzt werden sollen, sondern dass die Kompetenzen in *beziehungsreichen Kontexten* erworben werden (vgl. GI, 2008, S. 23 ff.). Die Inhaltsbereiche sind mit einigen Beispielen angereichert (vgl. GI, 2008, S. 23 ff.). Im vierten Kapitel werden die Prozessbereiche detailliert beschrieben, wobei auch hier auf eine Vernetzung untereinander sowie mit den Inhaltsbereichen verwiesen wird (vgl. GI, 2008, S. 45 ff.). Zu jedem der oben genannten Inhalts- und Prozessbereiche gibt es eine Beschreibung der allgemein zu erwerbenden Kompetenzen sowie eine *Unterteilung dieser Aussagen in Kompetenzen für die Jahrgangsstufen 5 bis 7 und die Jahrgangsstufen 8 bis 10 [...]* (vgl. GI, 2008, S. 12 ff.).

**Einordnung** Nachfolgend sollen die GI-Empfehlungen bezüglich der unterschiedlichen von Klieme u. a. (2003, S. 24 ff.) vorgestellten Standardbegriffe eingeordnet werden (vgl. Abschnitt 2.1)[28]:

[28] In den GI-Empfehlungen wird auf unterschiedliche Standardbegriffe eingegangen und dargestellt, dass die als Mindeststandards konzipierten „Grundsätze und Standards für die Informatik in der Schule“ an Inhalten orientierte Handlungsfelder beschreiben und so zwischen einer Input- und einer Outputorientierung angelegt sind (vgl. GI, 2008, S. 3).

1. Beziehen sich die Standards auf den Input oder auf den Output?
   - In den GI-Empfehlungen wird von einem Schulfach Informatik ausgegangen, das in den Jahrgangsstufen 5 bis 10 durchschnittlich mit je einer Wochenstunde unterrichtet wird (vgl. GI, 2008, S. V). Außerdem werden weitere inputorientierte Anforderungen genannt wie z. B. Fachkompetenz der Lehrenden und eine durchschaubare informations- und kommunikationstechnische Infrastruktur einer Schule (vgl. GI, 2008, S. 7 ff.). Somit besitzen die GI-Empfehlungen Eigenschaften von **opportunity-to-learn standards** (vgl. auch GI, 2008, S. V).
   - Die GI-Empfehlungen orientieren sich allerdings hauptsächlich am Output schulischer Prozesse, denn es werden Kompetenzen genannt, die Schülerinnen und Schüler am Ende der Jahrgangsstufen 7 und 10 erworben haben sollen (vgl. GI, 2008, S. 14 ff.). Somit legen die GI-Empfehlungen Ziele des Informatikunterrichts fest und können als **outputorientierte Standards** bezeichnet werden (vgl. auch GI, 2008, S. V).

   a) Auf welcher Abstraktionsebene werden die Standards formuliert?
      - In den GI-Empfehlungen werden in der Vision und in den Grundsätzen **allgemeine Bildungsziele und Wertvorstellungen** genannt (vgl. GI, 2008, S. 1 ff.).
      - In den Inhalts- und Prozessbereichen der GI-Empfehlungen sind **bereichsspezifische Kompetenzen** angegeben (vgl. GI, 2008, S. 11 ff.).

   b) Welches Zielniveau wird spezifiziert?
      - Die GI-Empfehlungen beschreiben **Mindeststandards** *der informatischen Bildung in der Sekundarstufe I, d. h. ein Minimum an Kompetenzen, das jede Schülerin und jeder Schüler am Ende des 10. Jahrgangs, d. h. beim Mittleren Schulabschluss aufweisen sollte* (vgl. GI, 2008, S. 2).

Die GI-Empfehlungen orientieren sich am Output schulischer Bildung, indem sie die von den Lernenden zu erwerbende Kompetenzen angeben. Außerdem beziehen sie sich explizit auf die Sekundarstufe I. Insbesondere diese beiden Aspekte lassen die GI-Empfehlungen als geeignet erscheinen, um sie in dieser Arbeit als Grundlage für die Entwicklung eines Kompetenzmodells mit zugehöriger Aufgabensammlung für einen Bereich des Informatikunterrichts zu wählen. Um eine detailliertere Einschätzung vornehmen zu können, wird nachfolgend überprüft, inwieweit die

GI-Empfehlungen die von Klieme u. a. (2003, S. 24 ff.) beschriebenen Merkmale guter Bildungsstandards aufweisen (vgl. Abschnitt 2.1):[29]

**Fachlichkeit:** Klieme u. a. (2003, S. 24 ff.) geben als Merkmal guter Bildungsstandards an, dass diese sich jeweils auf einen bestimmten Lernbereich beziehen und Grundprinzipien einer Disziplin bzw. eines Unterrichtsfaches klar herausarbeiten.

Nach Auffassung des Autors sind die GI-Empfehlungen ausschließlich auf Informatik bezogen und arbeiten Grundprinzipien und Kernideen dieses Faches heraus. So finden sich z. B. die von Schubert und Schwill (2004, S. 89 ff.) genannten fundamentalen Ideen der Informatik „Algorithmisierung“, „Sprache“ und „strukturierte Zerlegung“ in den Kompetenzbereichen „Algorithmen“, „Sprachen und Automaten“ sowie „Strukturieren und Vernetzen“ wieder (vgl. GI, 2008, S. 30). Auch für die in der Informatik angewendeten Methoden sind jeweils Kompetenzen beschrieben. Als Beispiel seien Modellierung, Implementierung und der zielgerichtete Einsatz von Informatiksystemen genannt (vgl. z. B. GI, 2008, S. 37 ff.). Des Weiteren werden Kompetenzen zu informatiktypischen Arbeitsformen wie Kooperation und Projektarbeit angegeben (vgl. z. B. GI, 2008, S. 52 ff.). Damit wurde der Versuchung widerstanden, sich lediglich auf leicht messbare Kompetenzen zu beschränken (vgl. Abschnitt 2.2). Das Informatikstudium und die Fachwissenschaft selbst sind an deutschen Universitäten meist in die Bereiche praktische, technische und theoretische Informatik aufgeteilt.[30] Kompetenzen aus jedem dieser Bereiche finden sich in den Bildungsstandards wieder: Modellierung und Implementierung sind wichtige Bestandteile der praktischen Informatik (vgl. z. B. GI, 2008, S. 45 ff.). Kompetenzen zum Aufbau von Informatiksystemen werden meist in der technischen Informatik erworben (vgl. z. B. GI, 2008, S. 37 ff.). Sprachen und Automaten sind meist der theoretischen Informatik zugeordnet (vgl. z. B. GI, 2008, S. 34 ff.). Insgesamt wird nach Auffassung des Autors durch die GI-Empfehlungen zu Bildungsstandards Informatik ein Spektrum von Kompetenzen abgedeckt, das den Kern der wissenschaftlichen Disziplin Informatik altersgerecht aufbereitet und in dieser Tiefe nicht in anderen Fächern erworben werden kann.

[29] Eine erste Bewertung der GI-Empfehlungen von Friedrich und Puhlmann (2007, S. 32) kommt zu dem Schluss, dass diese den in Klieme u. a. (2003, S. 24 ff.) genannten Merkmalen guter Bildungsstandards genügen, wobei sich „Verständlichkeit“ und „Realisierbarkeit“ noch zeigen müssen.

[30] Im Jahr 2007 war das Informatikstudium an den Universitäten TU München, RWTH Aachen, FSU Jena in praktische, technische und theoretische Informatik aufgeteilt. Natürlich sind auch andere Aufteilungen möglich.

**Fokussierung:** Klieme u. a. (2003, S. 25 ff.) geben als Merkmal guter Bildungsstandards an, dass diese nicht die gesamte Breite des Lernbereiches bzw. Faches in allen Verästelungen abdecken, sondern sich auf einen Kernbereich konzentrieren.

In den GI-Empfehlungen gibt es keine expliziten Bezüge zu den in den sogenannten „Bindestrich-Informatiken“ (z. B. Bio-Informatik, Medieninformatik oder Wirtschaftsinformatik) benötigten Kompetenzen. Auch spezielle Themen der Informatik, die zwar in der Fachwissenschaft durchaus von Bedeutung sind, für die aber erweiterte mathematische Grundlagen benötigt werden (wie z. B. P =? NP und polynomielle Reduzierbarkeit), werden in den Bildungsstandards nicht genannt. Tiefer liegende Konzepte zum Beispiel der objektorientierten Programmierung werden nicht thematisiert. Nach Auffassung des Autors wurde der Gefahr, die Breite des Faches Informatik in all seinen Verzweigungen abzudecken, bei der Entwicklung der GI-Empfehlungen widerstanden und diese fokussieren somit bewusst auf Kernbereiche der Informatik.

**Kumulativität:** Klieme u. a. (2003, S. 25 ff.) geben als Merkmal guter Bildungsstandards an, dass diese sich auf die Kompetenzen beziehen, die bis zu einem bestimmten Zeitpunkt im Verlauf der Lerngeschichte aufgebaut worden sind und damit auf kumulatives, systematisch vernetztes Lernen abzielen.

Erste Kompetenzen in den Inhalts- und Prozessbereichen der GI-Empfehlungen sollen schon am Ende der 7. Jahrgangsstufe bei allen Schülerinnen und Schülern verfügbar sein, um diese auch in anderen Fächern nutzen zu können (vgl. GI, 2008, S. 14). Durch das Erwerben vertiefter Kompetenzen zu denselben Inhalts- und Prozessbereichen in den Jahrgangsstufen 8 bis 10 wird dem Spiralprinzip entsprochen.[31] In den GI-Empfehlungen (2008, S. 12 ff.) wird ausdrücklich darauf hingewiesen, dass die einzelnen Inhalts- und Prozessbereiche nicht getrennt voneinander zu behandeln sind, sondern dass stets Aufgaben bearbeitet werden sollen, bei denen verschiedene Inhalts- und Prozesskompetenzen gefordert sind und dass die einzelnen Kompetenzen im Unterricht in beziehungsreiche Kontexte zu stellen sind. Somit wird auf ein systematisches Vernetzen der erworbenen Kompetenzen abgezielt. Insgesamt ermöglichen die GI-Empfehlungen also kumulatives, systematisch vernetztes Lernen.

**Verbindlichkeit für alle:** Klieme u. a. (2003, S. 25 ff.) geben als Merkmal guter

[31] Das Spiralprinzip erläutert Humbert (2005, S. 34) mit Bezug auf Bruner (1974) wie folgt: *Im Laufe der [Schul-]Zeit sollte immer wieder auf die fundamentalen Ideen zurückgekommen werden – unter verschiedenen Gesichtspunkten und auf verschiedenen Niveaus.*

Bildungsstandards an, dass diese ein verbindliches Minimalniveau festschreiben, das von allen Lernenden erwartet wird und somit schulformübergreifend für alle Schülerinnen und Schüler gilt.

Die GI-Empfehlungen beschreiben Mindeststandards, *d. h. ein Minimum an Kompetenzen, das jede Schülerin und jeder Schüler am Ende des 10. Jahrgangs, d. h. beim Mittleren Schulabschluss aufweisen sollte* (vgl. GI, 2008, S. 2). Somit weisen die GI-Empfehlungen das Merkmal „Verbindlichkeit für alle“ auf.

**Differenzierung:** Klieme u. a. (2003, S. 25 ff.) geben als Merkmal guter Bildungsstandards an, dass diese nicht nur eine „Messlatte“ anlegen, sondern zwischen Kompetenzstufen differenzieren, die über und unter bzw. vor und nach dem Erreichen des Mindestniveaus liegen. So können Lernentwicklungen verständlich gemacht und weitere Abstufungen und Profilbildungen ermöglicht werden, die ergänzende Anforderungen in einem Land, einer Schulform oder einer Schule darstellen.

Im Vorwort der „Grundsätze und Standards für Informatik in der Schule“ wird ausdrücklich darauf hingewiesen, dass Mindeststandards entwickelt wurden und dass *in Schulen oder Schulzweigen, die ein entsprechendes Profil aufweisen, [...] eine Erweiterung hinsichtlich der Themenauswahl und des Anforderungsniveaus über diesen Anspruch hinaus sicher möglich und wünschenswert [ist]* (vgl. GI, 2008, S. V). Es wird nicht zwischen unterschiedlichen Kompetenzstufen differenziert, die vor oder nach dem Erreichen der Mindeststandards liegen. Die GI-Empfehlungen weisen somit nicht das Merkmal „Differenzierung“ auf.[32] Klieme u. a. (2003, S. 28 f.) halten es allerdings auch nicht für sinnvoll, auf nationaler Ebene Zusatzerwartungen, die über das Mindestkriterium hinaus gehen, zu formulieren. Die fehlende Differenzierung der GI-Empfehlungen kann Anreiz für wissenschaftliche Untersuchungen in der Fachdidaktik Informatik sein (vgl. hierzu z. B. Kohl und Fothe, 2007, S. 20).

**Verständlichkeit:** Klieme u. a. (2003, S. 25 ff.) geben als Merkmal guter Bildungsstandards an, dass diese klar, knapp und nachvollziehbar formuliert sind.

[32] Auch die Unterteilung in verschiedene Altersstufen kann nicht als Differenzierung in Kompetenzstufen interpretiert werden. Die einzige Möglichkeit einer Profilbildung für Länder, Schulen oder Schulformen wäre, wenn eine Schule festlegt: „In unserer Einrichtung erwerben alle Schülerinnen und Schüler bis zum Ende der 7. (bzw. 8. oder 9.) Jahrgangsstufe bereits die Kompetenzen der Mindeststandards, die am Ende der 10. Jahrgangsstufe erwartet werden.“ Diese Art von Profilbildung scheint aber insbesondere in Bezug auf die altersgerechte Auswahl der Kompetenzen nicht sinnvoll.

Die geforderten Kompetenzen der Inhalts- und Prozessbereiche sind in drei unterschiedlichen Abstraktionsstufen angegeben (vgl. GI, 2008, S. 12 ff.): Als Erstes werden die Kompetenzen über alle Jahrgangsstufen angegeben. Dann folgt eine tabellarische Übersicht, in der die Kompetenzen nach Jahrgangsstufen differenziert werden. Als Letztes ist noch eine ausführliche Darstellung aller Inhalts- und Prozessbereiche angegeben, in der auch auf spezifischere Fragestellungen und Beispiele eingegangen wird. Ob die Kompetenzbeschreibungen in den GI-Empfehlungen von Lehrerinnen und Lehrern als auch in der Öffentlichkeit nachvollzogen werden können, kann zum jetzigen Zeitpunkt nicht festgestellt werden.

**Realisierbarkeit:** Klieme u. a. (2003, S. 25 ff.) geben als Merkmal guter Bildungsstandards an, dass die Anforderungen eine Herausforderung für die Lernenden und die Lehrenden darstellen, aber trotzdem mit realistischem Aufwand erreichbar sind.

Zur Zeit der Erstellung der GI-Empfehlungen existierten für das Fach Informatik keine empirisch überprüften Kompetenzmodelle, die verschiedene Stufungen von Kompetenzen verdeutlichten (vgl. hierzu Abschnitt 3.5). Eine ähnliche Situation in anderen Fachdidaktiken führte dazu, dass die Bildungsstandards der KMK vorerst als Regelstandards entwickelt wurden (vgl. Abschnitt 2.1 bzw. KMK, 2004a, S. 14). Inwieweit die Anforderungen wirklich eine Herausforderung für die Lernenden und die Lehrenden darstellen und mit realistischem Aufwand erreichbar sind, kann nach Auffassung des Autors zum jetzigen Zeitpunkt nicht endgültig entschieden werden. Eine Überprüfung fällt vor allem deshalb schwer, weil die GI-Empfehlungen von einem Schulfach Informatik ausgehen, das durchschnittlich mit einer Stunde pro Woche in den Jahrgangsstufen 5 bis 10 unterrichtet wird (vgl. GI, 2008, S. V) Diese Bedingung ist derzeit in Deutschland nicht erfüllt (zur aktuellen Situation des Informatikunterrichts in den Ländern vgl. Weeger, 2007).

Entsprechend der obigen Einordnung weisen die GI-Empfehlungen für Bildungsstandards Informatik die Merkmale „Verbindlichkeit für alle“ und „Kumulativität“ auf. Sie differenzieren nicht zwischen verschiedenen Kompetenzstufen. Entsprechend der obigen Argumentation sind nach Auffassung des Autors auch die Merkmale „Fachlichkeit“ und „Fokussierung“ erfüllt. Inwieweit die GI-Empfehlungen die Merkmale „Verständlichkeit“ und „Realisierbarkeit“ ausweisen, kann zum jetzigen Zeitpunkt noch nicht entschieden werden. Insgesamt spricht die Analyse bezüglich der Merkmale guter Bildungsstandards für das Verwenden der GI-Empfehlungen als Grundlage des zu entwickelnden Kompetenzmodells, welches durch die Angabe von Kompetenzstufen in einem Bereich der informatischen Bildung zumindest einen exemplarischen Beitrag zur Differenzierung leisten kann.

## 2.6 Zusammenfassung

In diesem Kapitel wurden anhand der Klieme-Expertise und anhand der KMK-Standards aktuelle Entwicklungen zur Outputorientierung im Bildungswesen Deutschlands aufgezeigt. Thesen und Argumente zu Bildungsstandards wurden zusammengetragen. Konsequenzen für die Entwicklungen im Bereich der Schulinformatik wurden aufgezeigt. Es wurde festgestellt, dass die Diskussion um die Ziele allgemeiner Schulbildung und um die Lerninhalte einzelner Fächer auch eine Chance für den Informatikunterricht darstellt.
Drei Ansätze, mit denen versucht wird, Ziele des Informatikunterrichts festzulegen, wurden vorgestellt. Dabei wurde festgestellt, dass das ACM Model Curriculum for K-12 Computer Science aufgrund der größtenteils inputorientierten Angaben und die einheitlichen Prüfungsanforderungen Informatik aufgrund der Ausrichtung auf die Sekundarstufe II als Grundlage für die Entwicklung eines outputorientierten Kompetenzmodells für die Sekundarstufe I ungeeignet sind. Die GI-Empfehlungen „Grundsätze und Standards für die Informatik in der Schule“ geben auf verschiedenen Abstraktionsebenen Kompetenzen für die Jahrgangsstufen 5 bis 7 und 8 bis 10, aufgeteilt in Inhalts- und Prozessbereichen, an. Die Einordnung der GI-Empfehlungen zeigte, dass diese bereits einige der von Klieme u. a. (2003, S. 24 ff.) beschriebenen Merkmale guter Bildungsstandards aufweisen. Aufgrund der Ausrichtung auf die Sekundarstufe I und der outputorientierten Angabe von Kompetenzen zu verschiedenen Kompetenzbereichen, die im Informatikunterricht erworben werden sollen, erscheinen die GI-Empfehlungen als geeignet, um auf ihrer Grundlage ein Kompetenzmodell für einen Bereich des Informatikunterrichts mit zugehörigen Aufgaben zu entwickeln und im Unterricht zu erproben.

# 3 Kompetenzen, Kompetenzmodelle und Aufgaben

> *„standardorientiertes Unterrichten“: Jede einzelne Unterrichtsstunde und jede Unterrichtseinheit muss sich daran messen lassen, inwieweit sie zur Förderung und Weiterentwicklung inhaltsbezogener und allgemeiner Schüler-Kompetenz beiträgt, und der Unterricht über längere Zeiträume hinweg muss so konzipiert sein, dass der Aufbau von Kompetenzen im Zentrum steht.* (Blum, 2006, S. 17)

In der im Abschnitt 2.1 vorgestellten Klieme-Expertise nehmen Kompetenzmodelle eine zentrale Rolle ein. Sie vermitteln zwischen abstrakten Bildungszielen und konkreten Aufgabensammlungen (Klieme u. a., 2003, S. 71). Die Entwicklung von fachdidaktisch vertieften Kompetenzmodellen wird von verschiedenen Seiten gefordert (vgl. z. B. Bayrhuber, 2007, S. 12; Klieme u. a., 2003, S. 134 ff.; National Research Council, 2001, S. 2 ff. sowie Vollmer, 2005, S. 2). Entsprechend der im Kapitel 1 dargestellten Forschungsfragen soll in dieser Arbeit dieser Forderung entsprochen werden, indem ein Kompetenzmodell und eine zugehörige Sammlung von Aufgaben entwickelt und im Unterricht erprobt wird. In diesem Kapitel werden dafür erste Grundlagen gelegt. Im Abschnitt 3.1 wird geklärt, was unter Kompetenzen verstanden wird, bevor in den Abschnitten 3.2 und 3.3 auf verschiedene Varianten von Kompetenzmodellen und Aufgaben sowie Möglichkeiten der Entwicklung solcher Modelle bzw. Aufgaben eingegangen wird. Im Abschnitt 3.4 wird dargestellt, wie die im Rahmen dieser Untersuchung eingesetzten Materialien bei der Strukturierung, Vorbereitung, Durchführung und Auswertung von kompetenzorientiertem Informatikunterricht eingesetzt werden können. Schließlich werden im Abschnitt 3.5 Ansätze für Kompetenzmodelle in der Schulinformatik vorgestellt.

## 3.1 Kompetenz

Der Kompetenzbegriff wurde in den letzten Jahren auf unterschiedliche Art und Weise verwendet. In einer Definition von Weinert aus dem Jahr 1999 (zitiert nach Hartwig und Klieme, 2006, S. 128) werden Kompetenzen verstanden *als kontextspezifische kognitive Leistungsdispositionen, die sich funktional auf bestimmte*

*Klassen von Situationen und Anforderungen beziehen. Diese spezifischen Leistungsdispositionen lassen sich auch als Kenntnisse, Fertigkeiten oder Routinen charakterisieren.* Nach Hartwig und Klieme (2006, S. 129) stellt dieser von Weinert (1999) empfohlene Kompetenzbegriff, bei dem motivationale und affektive Faktoren ausgeschlossen werden, für die in der Bildungsforschung behandelten Fragestellungen eine gute Arbeitsgrundlage dar (vgl. auch Friedrich und Puhlmann, 2007, S. 24 und Klieme, 2004b, S. 10 f.). Klieme u. a. (2003, S. 21) stellen am Beispiel des Lernens einer Fremdsprache dar, dass kognitive Wissensinhalte mit Einstellungen, Werten und Motiven verknüpft sind. Dies soll an einem weiteren Beispiel aus der Informatik verdeutlicht werden: Wenn Schülerinnen und Schüler nicht motiviert sind, einen schriftlichen Entwurf für einen Algorithmus anzufertigen, dann wird sich das vermutlich auch beim Messen der Kompetenz, also in den Ergebnissen eines Kompetenztests, bemerkbar machen. Deswegen soll in dieser Arbeit die folgende, in vielen Bereichen der Bildungsforschung anerkannte, Definition nach Weinert (2002, S. 27 f.) verwendet werden, in der Kompetenzen verstanden werden als *die bei Individuen verfügbaren oder von ihnen erlernbaren kognitiven Fähigkeiten und Fertigkeiten, bestimmte Probleme zu lösen, sowie die damit verbundenen motivationalen, volitionalen und sozialen Bereitschaften und Fähigkeiten, die Problemlösungen in variablen Situationen erfolgreich und verantwortungsvoll nutzen zu können.*[33]

In dieser Arbeit werden aufgrund der geringen Erfahrungen mit motivationalen, volitionalen sowie sozialen Bereitschaften und Fähigkeiten in der Fachdidaktik Informatik diese nicht explizit formuliert bzw. überprüft. Wie das oben angegebene Beispiel zeigt, werden sie aber vermutlich implizit über die angegebenen Fähigkeiten und Fertigkeiten mit erfasst. Das Verwenden des von Weinert (2002, S. 27 f.) beschriebenen Kompetenzbegriffs ermöglicht es, dem Kompetenzmodell zu einem späteren Zeitpunkt solche Kompetenzen explizit hinzuzufügen.

## 3.2 Kompetenzmodelle

Nach Klieme u. a. (2003, S. 9) formulieren Kompetenzmodelle eine pragmatische Antwort auf die Konstruktions- und Legitimationsprobleme traditioneller Bildungs- und Lehrplandebatten und bieten Lehrerinnen und Lehrern ein Referenzsystem für ihr professionelles Handeln. Klieme u. a. (2003, S. 22) stellen dar, dass es das Ziel von Kompetenzmodellen ist, zwischen abstrakten Bildungszielen und konkreten Aufgabensammlungen zu vermitteln, indem sie zwischen Teildimensionen innerhalb einer Domäne unterscheiden und verschiedene Niveaustufen beschreiben. Nach

[33] volitional = willensmäßig

Klieme u. a. (2003, S. 74) erfüllen Kompetenzmodelle in Bezug auf Bildungsstandards also zwei Zwecke: *[Erstens] beschreiben sie das Gefüge der Anforderungen, deren Bewältigung von Schülerinnen und Schülern erwartet wird (Komponentenmodell); zweitens liefern sie wissenschaftlich begründete Vorstellungen darüber, welche Abstufungen eine Kompetenz annehmen kann bzw. welche Grade oder Niveaustufen sich bei den einzelnen Schülerinnen und Schülern feststellen lassen (Stufenmodell).* Klieme u. a. (2003, S. 22) fordern, dass jede Kompetenzstufe durch *kognitive Prozesse und Handlungen von bestimmter Qualität spezifiziert [ist], die Schüler auf dieser Stufe bewältigen können, nicht aber Schüler auf niedrigeren Stufen.* Des Weiteren können die Stufen auch als Entwicklungsschritte interpretiert werden, das heißt, dass die Lernenden beim Kompetenzerwerb die verschiedenen Stufen in einer gegebenen Reihenfolge durchlaufen (vgl. Klieme u. a., 2003, S. 77). Hammann (2004, S. 196 f.) beschreibt Kompetenzentwicklungsmodelle wie folgt: *Kompetenzentwicklungsmodelle geben in verschiedenen Stufen Fähigkeiten an, die Kompetenzverläufe illustrieren. Sie sind empirisch fundiert und beachten so beispielsweise entwicklungspsychologische Voraussetzungen, Lernschwierigkeiten und vorunterrichtliche Vorstellungen. In Kompetenzentwicklungsmodelle fließen die Ergebnisse empirischer Forschungen der Fachdidaktiken, der allgemeinen Pädagogik, der Entwicklungspsychologie sowie der pädagogischen Psychologie ein.*

Es kann festgestellt werden, dass im Bereich der Kompetenzmodelle eine gewisse Begriffsflut existiert. (Kompetenz-) Entwicklungs-, Komponenten-, Niveau-, Struktur-, Stufenmodelle seien hier als Beispiele genannt.[34] Zusammenfassend kann Folgendes festgehalten werden:

Kompetenzmodelle unterscheiden zwischen verschiedenen Komponenten (oder auch Dimensionen) eines Fachgebietes oder eines speziellen Kompetenzbereichs und geben für jede dieser Komponenten Stufen (oder auch Niveaus bzw. Niveaustufen) an, durch die Prozesse und Handlungen von bestimmter Qualität spezifiziert werden. Kann belegt werden, dass die Stufen eines Kompetenzmodells in einer bestimmten Reihenfolge durchlaufen werden, so wird von einem (Kompetenz-) Entwicklungsmodell gesprochen.

### Der Entwicklungsprozess von Kompetenzmodellen im Kontext

Helmke und Hosenfeld (2004, S. 61) stellen fest, dass es sich bei der Entwicklung von fachspezifischen Kompetenzmodellen um ein umfangreiches interdisziplinäres Forschungsprogramm handelt. Abbildung 2 nach Helmke und Hosenfeld (2004, S.

[34] Eine Unterscheidung zwischen Kompetenzstruktur- und Kompetenzniveaumodellen findet sich bei Hartwig und Klieme (2006).

62) zeigt, dass theoretisch begründete fachspezifische Kompetenzmodelle durch die wissenschaftlichen Felder der Fachdidaktik, Kognitionspsychologie, Entwicklungspsychologie, Lehr-Lernforschung und Methodologie mitgestaltet werden können, dass sie von Bildungsstandards und empirischen Überprüfungen beeinflusst werden, und dass sie Auswirkungen auf die Entwicklung von Bildungsstandards und Testaufgaben haben. Helmke und Hosenfeld (2004, S. 61) plädieren angesichts dringlicher und unabweisbarer Forderungen der Bildungspolitik für ein mehrgleisiges Verfahren, bei dem einerseits die Entwicklung von Kompetenzmodellen vorangetrieben wird, andererseits aber auch vorläufige Aussagen zu Kompetenzstufen gemacht werden, wenn noch keine umfassenden und bewährten Kompetenzmodelle entwickelt sind. Dementsprechend sollen in dieser Arbeit erste Aussagen zu einem Kompetenzmodell gemacht werden, auch wenn nicht der komplette in Abbildung 2 dargestellte Kreisprozess umgesetzt werden kann:

- In Anlehnung an die im Abschnitt 2.5 präsentierten GI-Empfehlungen wird ein Kompetenzmodell entwickelt und fachdidaktisch begründet (vgl. Kapitel 5).
- Zu diesem Modell werden Aufgaben bereitgestellt (vgl. Kapitel 6).
- Anschließend wird darüber berichtet, wie Kompetenzmodell und Aufgaben in zwei Szenarien von Lehrerinnen und Lehrern im Unterricht der Sekundarstufe I eingesetzt wurden und welche Rückschlüsse auf die Aufgaben und das Kompetenzmodell gezogen werden können (vgl. Kapitel 7 und 8).
- Trotz der Ausrichtung der Untersuchung auf das Lehrerhandeln sollen einige Aufgaben bzw. deren Lösungen durch Schülerinnen und Schüler exemplarisch untersucht werden. Eine vollständige empirische Überprüfung der Aufgaben kann im Rahmen dieser Arbeit nicht geleistet werden.

### Zum Festlegen von Kompetenzstufen

Es stellt sich die Frage, wie Kompetenzstufen festgelegt werden können. Hierzu werden nachfolgend zwei Möglichkeiten vorgestellt: Bei der im Jahr 2003 von der OECD durchgeführten Erhebung der mathematischen Kenntnisse und Fähigkeiten der Schülerinnen und Schüler in der PISA-Studie wurden 85 Aufgaben konstruiert, die verschiedene Inhalte, Prozesse und Situationen abdecken (vgl. OECD, 2004, S. 39 ff.).[35] Über eine viertel Million Schülerinnen und Schüler aus 41 Teilnehmerländern erhielten im Testszenario je ein Heft mit Aufgaben, die aus dem entwickelten

[35] OECD ist die Abkürzung für „Organisation for Economic Co-operation and Development“ also die Organisation für wirtschaftliche Zusammenarbeit und Entwicklung.
Mathematische Inhalte, Prozesse und Situationen wurden von der OECD (2004, S. 43) wie

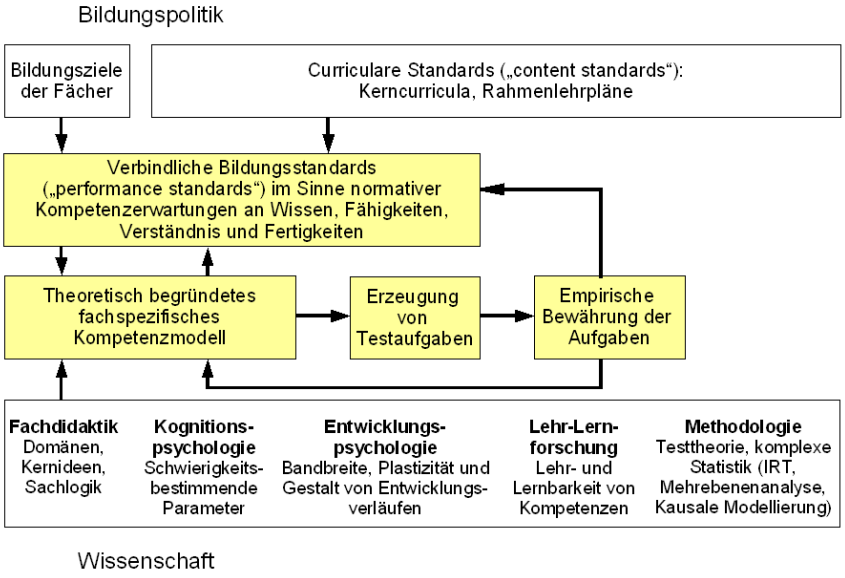



Abbildung 2: Ein Rahmenmodell zur Veranschaulichung des Kreisprozesses von Kompetenzmodellen und empirischen Tests nach Helmke und Hosenfeld (2004, S. 62)

Aufgabenpool ausgewählt wurden (vgl. OECD, 2004, S. 25 f.). Aufgrund der Auswertung der Lösungen konnte eine Skala der Mathematikleistungen entwickelt werden, um jede Testaufgabe mit einer Punktzahl entsprechend ihrem Schwierigkeitsgrad auf dieser Skala abzubilden und jeder Schülerin bzw. jedem Schüler einen Punktwert auf der Skala zuzuweisen, der ihren bzw. seinen geschätzten Fähigkeiten entspricht (vgl. OECD, 2004, S. 50).[36] Die Punktwerte wurden in

folgt beschrieben:
- *die mathematischen Inhalte, die bei verschiedenen Problem- und Fragestellungen gegeben sind*
- *die Prozesse, die aktiviert werden müssen, um die beobachteten Phänomene mit Mathematik in Verbindung zu bringen und dann die jeweiligen Probleme zu lösen*
- *die Situationen und Kontexte, die als Quellen für die Stimulusmaterialien herangezogen werden und in die die Probleme eingebettet sind*

[36] Im Bericht der OECD (2004, S. 50 f.) wird weiterhin ausführlicher dargestellt: *Die relative Kompetenz der Schüler, eine bestimmte Aufgabe zu lösen, lässt sich anhand des Anteils der von ihnen korrekt beantworteten Testaufgaben messen. Der relative Schwierigkeitsgrad der Testaufgaben kann auf der Basis des Anteils der getesteten Personen geschätzt werden, die alle Aufgaben korrekt lösen. Das für die Analyse der PISA-Daten verwendete mathematische Modell wurde durch iterative Verfahren umgesetzt, mit denen die Wahrscheinlichkeit geschätzt wird, dass eine bestimmte Person eine gegebene Testaufgabe korrekt beantwortet, und ebenso die Wahrscheinlichkeit, dass ein bestimmter Katalog von Testaufgaben von einem bestimmten*

sechs Kompetenzstufen eingeteilt, die Aufgabengruppen zunehmenden Schwierigkeitsgrades repräsentieren (vgl. OECD, 2004, S. 52). Die Kompetenzstufen wurden im Nachhinein anhand der oben beschriebenen inhaltlichen Dimensionen bezüglich der nötigen mathematischen Kompetenzen beschrieben (vgl. OECD, 2004, S. 53).[37]

Das in der PISA-Studie beschriebene Vorgehen erfordert eine empirische Untersuchung der Aufgabenlösungen einer größeren Schülerpopulation. Wie oben bereits beschrieben wurde, soll dies aufgrund der Ausrichtung dieser Arbeit auf das Lehrerhandeln hier nicht erfolgen. Reiss (2007, S. 25) hält es für wünschenswert, *Kompetenzstufenmodelle nicht nur durch Tests a posteriori festzulegen, sondern sie a priori zu definieren. Ein solches Vorgehen hat den Vorteil, dass die Verbindung zwischen curricularen Anforderungen, fachspezifischen Zusammenhängen und konkreten Aufgabensammlungen [...] kohärenter gestaltet werden kann.* Dementsprechend soll in dieser Arbeit ein Kompetenzmodell aus fachdidaktischer Sicht entwickelt werden.

Das Zusammentragen der Ergebnisse verschiedener empirischer Studien eines Fachgebietes ist eine weitere Möglichkeit, um Kompetenzstufen festzulegen. Ein Beispiel für eine derartige Entwicklung ist der gemeinsame europäische Referenzrahmen für Sprachen, der die Basis für die Entwicklung von zielsprachlichen Lehrplänen, curricularen Richtlinien, Prüfungen, Lehrwerken usw. in Europa ist (vgl. Trim u. a., 2001, S. 14). Er unterscheidet drei Referenzniveaus, die jeweils noch einmal in zwei Stufen aufgefächert sind (vgl. Trim u. a., 2001, S. 33 ff.):

- A Elementare Sprachverwendung
  - A1 Breakthrough
  - A2 Waystage
- B Selbstständige Sprachverwendung
  - B1 Threshold

*Schüler richtig beantwortet wird. Das Ergebnis dieser Verfahren ist ein aus Schätzwerten bestehender Datensatz, der es ermöglicht, eine kontinuierliche Skala zur Darstellung der mathematischen Grundbildung zu entwickeln. Entlang diesem Kontinuum kann die Position der einzelnen Schülerinnen und Schüler geschätzt und damit festgestellt werden, welches Niveau an mathematischer Grundbildung sie unter Beweis gestellt haben, und ebenso kann die Position einzelner Testaufgaben geschätzt und damit festgestellt werden, welches Niveau mathematischer Grundbildung jede Testaufgabe beinhaltet.*
Meyerhöfer (2004) zeigt, dass die Lösungshäufigkeit einer Aufgabe aufgrund der Vielfalt von Lösungszugängen inhaltlich nicht eindeutig interpretiert werden kann.

[37] Eine detaillierte Beschreibung der technischen Verfahren bei der PISA-Studie gibt die OECD (2005).

  - B2 Vantage
- C Kompetente Sprachverwendung
  - C1 Effective Operational Proficiency
  - C2 Mastery

Im Referenzrahmen sind für verschiedene Bereiche des Sprachenlernens verbale Beschreibungen der unterschiedlichen Kompetenzen auf Stufen angegeben, die in mehreren empirischen Studien ermittelt wurden (vgl. Trim u. a., 2001). So gibt es neben einer Globalskala z. B. ein Raster zur Selbstbeurteilung, in dem die geforderten Kompetenzen zum Hören, Lesen, zum Teilnehmen an Gesprächen, zum zusammenhängenden Sprechen und zum Schreiben auf den Stufen A1 bis C2 angegeben sind (vgl. Trim u. a., 2001, S. 35 ff.).

Der Ansatz, für einzelne Kompetenzbereiche eines Faches Untersuchungen vorzunehmen um anschließend aus den Ergebnissen dieser Untersuchungen ein komplexeres Kompetenzmodell zu konstruieren, erscheint sinnvoll und lohnenswert. Für die Informatik stellen die von Brinda und Schulte (2005) vorgestellten „Beiträge der Objektorientierung zu einem Kompetenzmodell des informatischen Modellierens“ einen ersten Schritt in dieser Richtung dar. Die Entwicklung, fachdidaktische Begründung und exemplarische Erprobung eines Kompetenzmodells zu einem Bereich der informatischen Bildung im Rahmen dieser Arbeit ist ein weiterer Schritt in dieser Richtung. Wenn weitere Modelle zu anderen Kompetenzbereichen entwickelt bzw. die im Abschnitt 3.5 vorgestellten Ansätze weiterentwickelt und erprobt werden, so können später auf Grundlage verschiedener Studien Kompetenzstufen einer gesamten informatischen Bildung festgelegt werden.

## 3.3 Aufgaben

Klieme u. a. (2003, S. 121 f.) empfehlen, parallel zu Kompetenzmodellen **Beispielaufgaben** bereitzustellen, anhand derer die Kompetenzen und Stufen illustriert werden können (vgl. hierzu auch Leuders, 2006, S. 81 ff.). Herper (2005, S. 74) stellt fest, dass zur „Überprüfung“ und zur „Herausbildung“ von Kompetenzen unterschiedliche Aufgaben benötigt werden (vgl. hierzu auch Leuders, 2006, S. 86 ff.). **Testaufgaben** müssen nach Klieme u. a. (2003, S. 124 f.) gezielt entlang von Kompetenzmodellen entworfen werden, damit eine auf die inhaltlichen und kognitiven Anforderungen der Kompetenzstufen ausgerichtete Interpretation der Ergebnisse erfolgen kann. Blum (2006, S. 18) nennt folgende Bedingungen, denen Testaufgaben im Rahmen von Bildungsstandards genügen müssen:

- *prinzipielle Verstehbarkeit ohne externe Unterstützung*
- *individuelle Bearbeitbarkeit in überschaubarer Zeit*
- *verlässliche Korrigierbarkeit*

Blum (2006, S. 18) stellt weiterhin dar, dass für **Lernaufgaben** die zur Kompetenzförderung **im Unterricht** eingesetzt werden, die oben genannten Beschränkungen wegfallen. Bei diesen Aufgaben für den Unterricht stehen seiner Auffassung nach Anregungsgehalt und Lernpotenzial im Vordergrund. Außerdem fordert Blum (2006, S. 18): *In jedem Fall muss klar sein, welches Kompetenzpotenzial in einer Aufgabe steckt, was von den Lernenden beim Bearbeiten erwartet wird und wie Bearbeitungen einzuordnen sind.*

Den oben genannten Aussagen entsprechend werden in dieser Arbeit drei Typen von Aufgaben unterschieden.

**Beispielaufgaben** verdeutlichen und konkretisieren die in einem Modell genannten Kompetenzen, sie dienen der Verständigung zwischen Lehrenden und Lernenden über Komponenten und Stufen eines Kompetenzmodells.

**Unterrichtsaufgaben** dienen dem Erwerb von Kompetenzen in verschiedenen Unterrichtssituationen.

**Testaufgaben** dienen der Überprüfung der von den Lernenden erreichten Kompetenzen.

Aufgaben aus der Informatikdidaktik die diesen drei Typen zugeordnet werden können wurden bereits präsentiert:[38]

- Beispielaufgaben aus den EPA Informatik wurden von Fothe (2005, S. 47 f.) erläutert.
- Unterrichtsaufgaben zum „Thema Sicherheit im Netz“ stellten Schubert u. a. (2005, S. 66 ff.) vor.
- Testaufgaben wurden von Koubek (2005, S. 61 ff.) (Thema „E-Mail“) und Durda (2007, S. 23 ff.) (Thema „Sprachen und Automaten“) präsentiert.

Herper (2005, S. 74) fordert, dass auch Aufgaben für Testinstrumente entwickelt werden sollten, *die außerhalb von üblichen Unterrichtssituationen in ähnlicher*

[38] Herper (2005, S. 75 f.) stellt Beispiele für Aufgaben zu „Algorithmen“ dar, die zum Erwerben und zum Überprüfen von Kompetenzen – also als Unterrichts- oder Testaufgaben – eingesetzt werden können. Weitere Aufgaben im Rahmen von Bildungsstandards zu verschiedenen Themen der Informatik nennen Humbert und Pasternak (2005, S. 25 ff.) sowie Puhlmann und Friedrich (2007).

*Weise wie bei PISA-Tests eingesetzt werden können.*[39] Solche Aufgaben, die für Bildungsmonitoring bzw. Schulevaluation genutzt werden können, sollen in dieser Arbeit nicht erstellt werden.

### Zur Entwicklung von Aufgaben

Klieme u. a. (2003, S. 125) fordern, dass Aufgabenentwürfe informell in einzelnen Klassen vorerprobt und dann einer systematischen Erprobungsstudie unterzogen werden um die Messqualität abzusichern. Als Ergebnis einer solchen Studie können nach Klieme u. a. (2003, S. 125) Rückschlüsse auf entwickelte Kompetenzmodelle gezogen werden, Kompetenzstufen an empirisch überprüften Beispielen illustriert oder Testinstrumente entwickelt werden. Eine empirische Überprüfung, bei der jede Aufgabe, wie von Klieme u. a. (2003, S. 124 f.) gefordert, *von mehreren Hundert Schülerinnen und Schülern bearbeitet* wird, wird aufgrund des explorativen Charakters der Arbeit und der Ausrichtung auf das Lehrerhandeln nicht geleistet (vgl. zu einer solchen Erprobung in der Informatik Durda, 2007). Trotzdem sollen soweit möglich aufgrund der Ergebnisse des Kompetenztests und der Einschätzung der Aufgaben durch die Lehrpersonen Rückschlüsse auf Kompetenzmodell und Aufgaben gezogen werden.

Klieme u. a. (2003, S. 84 ff.) stellen Prinzipien der Testentwicklung anhand von vier Fragen vor, die nachfolgend für die in dieser Arbeit verwendeten Testaufgaben beantwortet werden sollen:

- *Sollen die Testergebnisse im Vergleich zu den Testergebnissen anderer Schüler bzw. Schulen interpretiert werden (man spricht dann von „verteilungsorientierter“ – oder „normorientierter“ Testinterpretation) oder in Bezug auf ein inhaltliches Kriterium („kriteriumsorientiert“)?*

  Im Rahmen dieser Arbeit sollen die Testergebnisse in Bezug auf die in einem Kompetenzmodell festgelegten Kompetenzstufen interpretiert werden. Es soll nicht analysiert werden, wie einzelne Schülerinnen bzw. Schüler oder Schülergruppen relativ zu anderen Lernenden abschneiden.

- *Soll die Kompetenz innerhalb eines Lernbereichs/Faches auf einer einzigen Gesamtskala („eindimensional“) erfasst werden, oder sollen mehrere Teilkompetenzen unterschieden und getrennt erfasst werden („mehrdimensionale Skalierung“)?*

  Mit Hilfe des entwickelten Tests soll es möglich sein, verschiedene Kompetenzdimensionen zu erfassen. Dadurch ist es möglich, Rückschlüsse auf die

[39] Beispiele für Aufgaben, die im Bereich der Informatik als Testinstrumente eingesetzt werden können, geben Khalil (2006) und Puhlmann (2003; 2005).

Angemessenheit der Zieldimensionen zu ziehen und Defizitbereiche aufzudecken (vgl. Klieme u. a., 2003, S. 86).

- *Bearbeiten alle Schülerinnen und Schüler dieselben Testaufgaben, oder werden – durch sogenannte „Rotation“ von Testversionen innerhalb einer Klasse – den Schülerinnen und Schülern unterschiedliche Aufgaben vorgelegt, damit insgesamt mehr Aufgaben eingesetzt werden können?*

  Die Testaufgaben werden mit der Zielstellung der Individualdiagnostik entwickelt. Auf Grundlage der Testergebnisse soll es möglich sein, gezielte Fördermaßnahmen abzuleiten. Nach Klieme u. a. (2003, S. 86) ist es bei dieser Zielstellung selbstverständlich, *dass jede bzw. jeder einzelne der untersuchten Schüler die zur Beurteilung ihrer/seiner Stärken und Schwächen relevanten Testaufgaben vollständig bearbeitet.* Dementsprechend wird keine Rotation von Testaufgaben eingesetzt.

- *Soll ein und derselbe Test für alle Kompetenzniveaus und Schulformen gelten, oder verwendet man Aufgaben, die an das Leistungsvermögen der Gruppe bzw. des/der Einzelnen angepasst sind („verzweigtes“ bzw. „adaptives Testen“)?*

  Die entwickelten Testaufgaben sollen den unterschiedlichen Leistungsniveaus der Schülerinnen und Schüler angepasst werden. Nach Klieme u. a. (2003, S. 87) liegen die Vorteile solcher niveau-angepasster Tests *zum einen in einer höheren Messgenauigkeit der Testergebnisse, zum anderen aber auch in einer Aufrechterhaltung der Testmotivation, die bei einer Unter- oder Überforderung der Schüler durch zu leichte oder zu schwere Tests in der Regel nachlässt.*

## 3.4 Zur Umsetzung kompetenzorientierten Informatikunterrichts

Blums (2006, S. 17) Erläuterung zu „standardorientiertem Unterricht“ ist am Anfang dieses Kapitels wiedergegeben. In der Informatik gibt es derzeit keine Bildungsstandards, die auf einem abgestuften Kompetenzmodell beruhen. Im Rahmen der Untersuchungen dieser Arbeit soll aber über die in Mindeststandards festgelegten Kompetenzen hinaus ein weiterführender Kompetenzerwerb ermöglicht werden. Dehalb wird nachfolgend nicht von „standardorientiertem“, sondern von „kompetenzorientiertem“ Unterricht gesprochen.
Lankes u. a. (2006, S. 16) stellen dar, dass ein kompetenzorientierter Unterricht auf der einen Seite das Handwerkszeug sicher stellt, das Schülerinnen und Schüler in bestimmten Situationen zum Lösen von Problemen benötigen und auf der

anderen Seite viele Gelegenheiten bietet, dieses Handwerkszeug in immer wieder neuen, zunehmend komplexeren und offeneren Lernsituationen auszuprobieren. Die Reflexion der eigenen Lösungswege, die Diskussion über Alternativen, die Bewertung unter Berücksichtigung verschiedener Perspektiven sind nach Lankes u. a. (2006, S. 16) Kernprozesse eines an Kompetenzen ausgerichteten Unterrichts.

Um einen solchen Unterricht zu ermöglichen, in dessen Zentrum der Aufbau von Kompetenzen steht, werden bei den im Rahmen dieser Arbeit durchgeführten Untersuchungen verschiedene Mittel zur Verfügung gestellt, deren Zusammenwirken bei der Strukturierung, Vorbereitung, Durchführung und Auswertung von kompetenzorientiertem Informatikunterricht in Abbildung 3 dargestellt ist und im Folgenden erläutert wird:

1. Bildungsstandards, wie sie von Klieme u. a. (2003, S. 22) vorgeschlagen werden, stützen sich auf Kompetenzmodelle. Da in der Fachdidaktik Informatik (wie auch in anderen Fachdidaktiken) derzeit kaum erprobte Kompetenzmodelle vorliegen, wurden – ohne auf solche Modelle zurückzugreifen – von der GI (2008) direkt „Grundsätze und Standards für die Informatik in der Schule“ empfohlen. Durch diese GI-Empfehlungen wurden bereits Festlegungen, wie z. B. die Aufteilung der Kompetenzen in Inhalts- und Prozessbereiche getroffen, die nach Einschätzung des Autors für die Entwicklung des Informatikunterrichts in den einzelnen Ländern von Bedeutung sein werden (vgl. hierzu auch GI, 2008, S. VII). Wegen dieser zu erwartenden Auswirkungen erscheint es sinnvoll, zu analysieren, inwieweit bereits bestehende (Empfehlungen zu) Bildungsstandards als Grundlage bei der Erarbeitung von Kompetenzmodellen genutzt werden können.
2. Blum (2006, S. 18) legt dar, dass eine *direkte, quasi deduktive Ableitung von unterrichtlichem Handeln aus Bildungsstandards* nicht möglich ist. Klieme (2007) stellt mit Bezug auf die Expertise „Zur Entwicklung nationaler Bildungsstandards“ fest, dass in einem Unterricht, der auf einen systematischen Kompetenzaufbau abzielt, ein Kompetenzmodell von der Lehrkraft als *Basis für unterrichtliches Handeln genutzt wird.* In den Untersuchungen dieser Arbeit sollen Lehrkräfte ihren Unterricht an den in einem Modell geforderten Kompetenzen ausrichten. Das Kompetenzmodell kann als Hilfsmittel zur Strukturierung des Unterrichtsstoffes (Jahresplanung), zur Festlegung von Zielen bei der Vorbereitung einzelner Stunden und zur Analyse der erworbenen Kompetenzen bei der Auswertung des Unterrichts eingesetzt werden. Weiterhin ist es möglich, das Kompetenzmodell im Unterricht vorzustellen, damit Schülerinnen und Schüler ihre eigenen Leistungen beurteilen, Zielvorstellungen entwickeln und Konsequenzen für ihr Handeln ableiten können.

3. Nach Klieme (2007, S. 78) gilt die Entwicklung standardbezogener kompetenzorientierter Aufgaben heute als wichtigster Hebel *für eine anspruchsvolle, die Schülerinnen und Schüler kognitiv aktivierende, damit den kumulativen Aufbau von vernetztem Wissen fördernde Unterrichtsführung.* Blum (2006, S. 18) sieht es für kompetenzorientierten Unterricht als sinnvoll an, *dass sich die verwendeten Lernaufgaben an den angestrebten Kompetenzen orientieren, die sich wiederum in Testaufgaben konkretisieren können. Ein so verstandenes „Teaching to the Test" (der Unterricht muss dafür sorgen, dass die nachher erwarteten Kompetenzen auch erreicht werden können) ist höchst wünschenswert.* Lehrpersonen können die bereitgestellten Beispiel- und Unterrichtsaufgaben entsprechend den zu erwerbenden Kompetenzen bei der Planung ihres Unterrichts auswählen und bei dessen Durchführung einsetzen. Bei den Beispielaufgaben wurde versucht, die im Kompetenzmodell genannten Kompetenzen möglichst genau abzubilden, die Unterrichtsaufgaben wurden in die Komponenten und Stufen des Modells eingeordnet.

4. Nach Klieme u. a. (2003, S. 83) werden Tests im pädagogischen Alltag mit der Erwartung eingesetzt, *Aussagen über spezifische Stärken und Schwächen und damit den Förderbedarf einzelner Schülerinnen und Schüler zu machen.* Leuders (2006, S. 81) erläutert am Beispiel der Mathematik, dass Planungsentscheidungen leichter getroffen werden können, wenn bereits frühzeitig Aufgaben und Probleme festgelegt werden, um die Leistungen der Lernenden später zu überprüfen. Somit können auf ein Kompetenzmodell abgestimmte Testaufgaben also nicht nur zur Durchführung und Auswertung, sondern auch zur Strukturierung und Vorbereitung von Unterricht eingesetzt werden.

5. In den GI-Empfehlungen (2008, S. 8) wird dargelegt, dass Technik, als digitales Hilfsmittel im Informatikunterricht stets Medium, Werkzeug und Inhalt ist. Obwohl Informatikunterricht natürlich an verschiedenen Stellen auch ohne den Einsatz von Technik auskommt, so wird doch häufig ein (Software-) Werkzeug eingesetzt, welches auch Auswirkungen auf die Strukturierung, Planung und Durchführung des Unterrichts hat.[40]

Schott und Ghanbari (2008, S. 9) stellen dar, dass die Qualifikation, kompetenzorientiert unterrichten zu können voraussetzt, *dass die Lehrkräfte in der Lage sind, Schüler*

- *so zu unterrichten, dass diese die Chance haben, während des Unterrichts das zu lernen, was sie nach dem Unterricht können sollen und*

[40] Für Beispiele eines Informatikunterrichts ohne Computer sei auf die Ausführungen von Bell u. a. (2006) verwiesen.

- *kompetenzorientiert zu prüfen, das heißt eine Lernerfolgskontrolle durchzuführen, die feststellen kann, ob bzw. inwieweit die Schüler die im Unterricht angestrebten Kompetenzen erreicht haben.*

Durch die Bereitstellung der genannten Materialien soll ein Beitrag zum Erfüllen dieser Voraussetzungen kompetenzorientierten Unterrichtens geschaffen werden. Auf Grundlage der Erfahrungen in den Untersuchungen dieser Arbeit können zu einem späteren Zeitpunkt evtl. Unterrichtskonzepte zu kompetenzorientiertem Informatikunterricht entwickelt und Fortbildungen durchgeführt werden.

## 3.5 Kompetenzmodelle in der Schulinformatik

Auch in der Fachdidaktik Informatik werden Kompetenzmodelle mit zugehörigen Aufgabensammlungen gefordert (vgl. Herper, 2005, S. 74; Humbert, 2005, S. 65, Magenheim, 2005, S. 1 sowie Vollmost, 2005, S. 59 f.). Nachfolgend sollen Kompetenzmodelle für eine gesamte informatische Bildung sowie für Bereiche der informatischen Bildung vorgestellt werden, um aus diesen – soweit möglich – Schlussfolgerungen für die Entwicklung des Kompetenzmodells in der vorliegenden Arbeit zu ziehen.

### Kompetenzmodelle für eine gesamte informatische Bildung

Friedrich (2003, S. 134 ff.) stellte in Anlehnung an das Kompetenzmodell für Mathematik aus der PISA-Studie folgende fünf Stufen informatischer Kompetenz dar:

- *Stufe I: Bedienung von Informatikanwendungen*
- *Stufe II: Benutzung von Informatiksystemen*
- *Stufe III: Kenntnis fachsystematischer Grundlagen*
- *Stufe IV: Verständnis von Konzepten der Informatik*
- *Stufe V: Entwicklung und Bewertung von Informatiksystemen*

Diese Stufen wurden von ihm mit den Leitlinien informatischer Bildung „Interaktion mit Informatiksystemen“, „Wirkprinzipien von Informatiksystemen“, „Informatische Modellierung“ und „Wechselwirkungen zwischen Informatiksystemen, Individuum und Gesellschaft“ in einer Tabelle zu einem Kompetenzmodell verknüpft (zu den Leitlinien vgl. GI, 2000). Anschließend wurden von Friedrich (2003, S. 141 ff.) exemplarisch für eine Zelle dieser Tabelle inhaltliche Anforderungen und Arbeitsweisen sowie geeignete Werkzeuge und Anwendungen genannt.

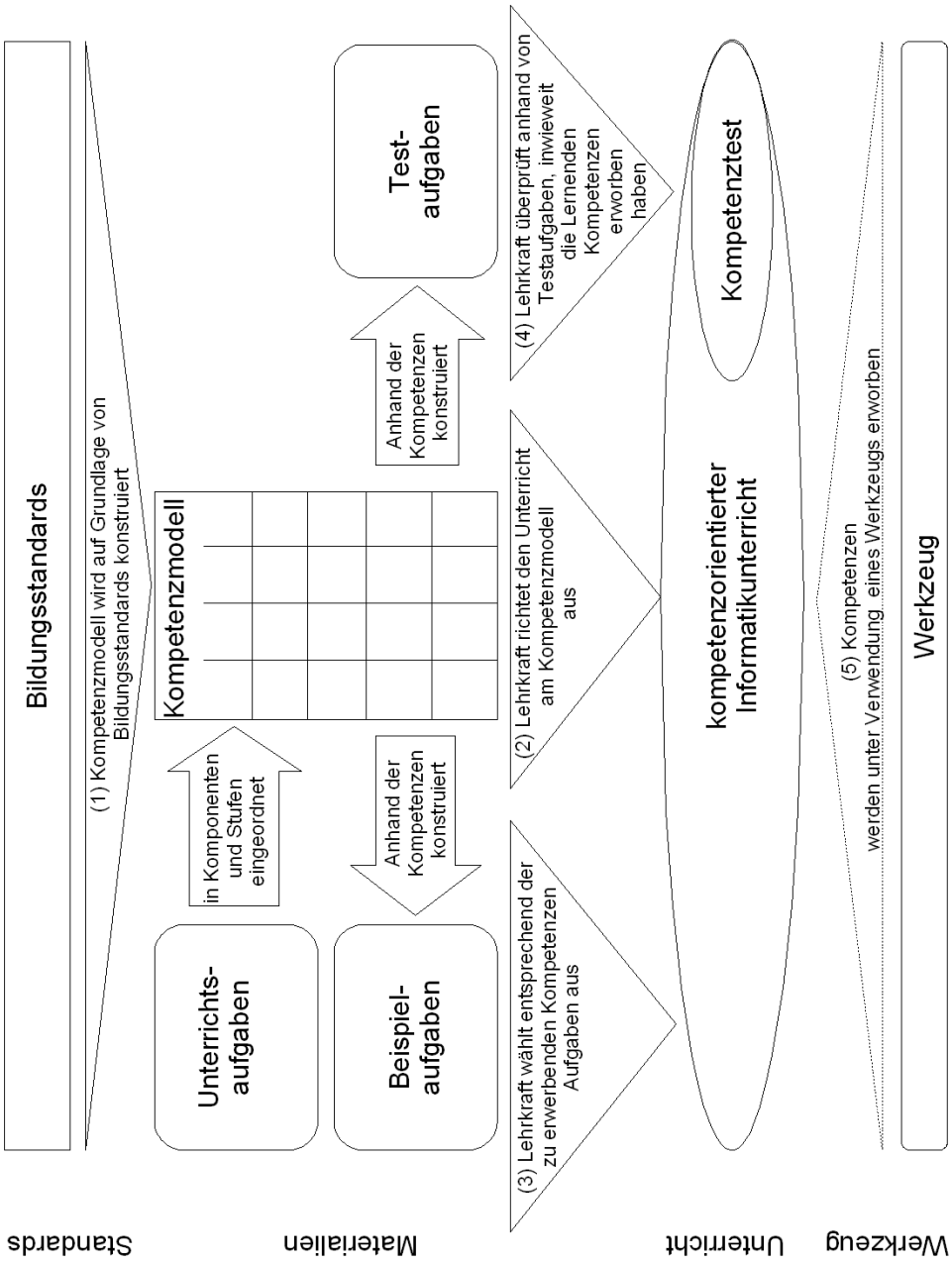



Abbildung 3: Zusammenwirken von Bildungsstandards, Kompetenzmodell, Aufgaben und Werkzeug bei der Strukturierung, Vorbereitung, Durchführung und Auswertung kompetenzorientierten Informatikunterrichts im Rahmen dieser Arbeit

Magenheim (2005) entwickelte ausgehend von den verschiedenen internationalen Kerncurricula ein „Model of ICT-Competence Classes“, in dem folgende drei Dimensionen angegeben sind:[41]

- *Usage of Media Functions of the Informatics System*

  *(Domain Specific Application, Cognitive Tools with Generic Functions, Communication / Co-operation Tools, Exploration and Evaluation Tools, Development Tools)*

- *Level of Application*

  *(Guided Use of Selected Basic Functions of the IS, Scenario Adequate Free Choice of Selected Basic Functions of the IS, Usage of Selected More Complex Functions of the IS, Competent Extensive Use of Systems Functions, Combined Co-ordinated Use of different IS)*

- *Level of System Comprehension*

  - *Knowledge*

    *(Understanding of Basic System Functions and Basic Concepts of Systems Hardware, Knowledge of Selected Views of the Informatics System (Algorithms, Source Code, GUI, Models, Protocols, Theory...), Comprehension of Fundamental Informatics Principles and Abstract Concepts (Meta-Models))*

  - *Transfer of Knowledge*

    *(Ability to Use ICT-Knowledge in Familiar Application Scenarios, Ability to Transfer ICT-Knowledge to a New Context)*

  - *Complexity of Construction*

    *(Comprehension of the Systems Coherence (Ability of Re-engineering), Using Information, Methods, Concepts and Theories in New Context to Build Systems (e.g. Design Pattern))*

  - *Assessment*

    *(Assessment of System Design its Functionality in Socio-Technical Context, Evaluation of IS Current Social Impact and IS in Historical Perspective)*

Magenheim (2005, S. 5) weist darauf hin, dass das vorgestellte Modell noch weiterentwickelt und auf Reliabilität und Validität sowie auf die Trennschärfe der Kategorien untersucht werden muss.

Puhlmann (2003, S. 148) entwickelte den Begriff der „informatischen Literalität“ und konkretisierte ihn in den folgenden drei Kompetenzklassen, die Arten der Beschäftigung mit den Informatik-Inhalten angeben:

- Klasse 1: Anwenden von Informatiksystemen

[41] ICT steht für Information and Communications Technology.
IS steht für Informatics System.

- Klasse 2: Gestalten von Informatiksystemen
- Klasse 3: verantwortungsvolles Entscheiden über den Einsatz und die Entwicklung von Informatiksystemen

Anschließend wurde von Puhlmann (2003, S. 149 ff.) je ein Beispiel zur Operationalisierung der Kompetenzklassen angegeben. Erste Aufgaben zum Thema Informationssicherheit im Internet wurden durch Khalil (2006) in dieses Kompetenzmodell eingeordnet.

Die drei vorgestellten Kompetenzmodelle zielen mit plausiblen Ansätzen auf die Strukturierung der gesamten informatischen Bildung ab. Inwieweit diese Kompetenzmodelle handhabbar und praktikabel sind, wird sich nach Auffassung des Autors allerdings erst zeigen, nachdem eine größere Menge von Aufgaben zu diesen Modellen entwickelt bzw. in diese Modelle eingeordnet worden ist. Der relativ hohe Abstraktionsgrad der genannten Kompetenzmodelle lässt sie eher zur Strukturierung der gesamten informatischen Bildung z. B. in einem Land oder an einer Schule bzw. zur Erfassung entsprechender Kompetenzen geeignet erscheinen, als zur Vorbereitung, Durchführung und Auswertung von konkretem kompetenzorientierten Unterricht. Wie in der ersten Forschungsfrage im Kapitel 1 beschrieben wurde, soll in dieser Arbeit ein Kompetenzmodell für einen Bereich des Informatikunterrichts entwickelt werden. Aufgrund dieser Zielstellung werden aus den vorgestellten Modellen keine Schlussfolgerungen für das weitere Vorgehen in dieser Arbeit gezogen.

#### Kompetenzmodelle für Bereiche der informatischen Bildung

Dorninger (2007, S. 8 ff.) beschreibt ein Kompetenzmodell für die Sekundarstufe II, welches die „Basic Skills“, also die grundlegenden Fähigkeiten, im Bereich der angewandten Informatik repräsentieren soll. Das zweidimensionale Kompetenzmodell wurde von der österreichischen Bildungsstandardgruppe „Angewandte Informatik an berufsbildenden höheren Schulen“ entwickelt (vgl. Dorninger, 2007, S. 8). Die erste Dimension beschreibt die Handlungskompetenzen, das sind fachlich orientierte Aktivitäten, die für die Bearbeitung und Nutzung der inhaltlichen Teilbereiche der angewandten Informatik erforderlich sind (vgl. Dorninger, 2007, S. 8). Als Handlungskompetenzen werden von Dorninger (2007, S. 8) „Verstehen“, „Anwenden“, „Analysieren“ und „Entwickeln“ angegeben. Des Weiteren werden inhaltsbezogene Kompetenzen genannt, die von Schülerinnen und Schülern bei der Auseinandersetzung mit fachlichen Inhalten erworben werden sollen und die beim Nutzen dieser Inhalte erforderlich sind (vgl. Dorninger, 2007, S. 8). Als inhaltliche Dimensionen werden von Dorninger (2007, S. 8) „Informatiksysteme“, „Publika-

tion und Kommunikation“, „Kalkulations- und Datenmodelle“, „Umfeldthemen“ sowie „Algorithmen und Datenstrukturen“ genannt. Die inhaltlichen Kompetenzen werden anschließend detaillierter als in anderen Kompetenzmodellen der Informatik beschrieben. Dies soll an einem Beispiel verdeutlicht werden: Zur Kompetenz *„Ich kann drucken“* aus dem Bereich Kalkulations- und Datenmodelle sind folgende Teilkompetenzen angegeben (vgl. Dorninger, 2007, S. 9): *ein und mehrere Arbeitsblätter drucken; Druckbereiche festlegen; Kopf- und Fußzeilen festlegen; Papierformate einstellen; Zeilen- und Spaltenwiederholungen festlegen; Seitenumbrüche festlegen; markierte Bereiche drucken; bestimmte Seiten drucken;*
Die konkreten Formulierungen ermöglichen es, direkt Aufgaben zu den in dem Modell geforderten Kompetenzen zu konstruieren. Die Werkzeugunabhängigkeit der Angaben ermöglicht eine Vielfalt der im Unterricht eingesetzten Systeme. Beide Aspekte sollen bei der Entwicklung des Kompetenzmodells in dieser Arbeit beachtet werden.

Schlüter und Brinda (2007, S. 283 ff.) stellen einen Ansatz zur Gewinnung von Kompetenzen und Kompetenzstufen als Bausteine eines Kompetenzmodells der theoretischen Informatik vor. Sie legen dar, wie die unterrichtsrelevanten Fachinhalte der theoretischen Informatik strukturiert werden sollen, nämlich inhaltlich nach den vier Teilgebieten „Automaten“, „Sprachen“, „Berechenbarkeit“, „Komplexität“ und tabellarisch dargestellt, so *dass aufeinander aufbauende Inhalte eines Kapitels von oben nach unten aufeinander folgen und Pfeile assoziative Sequenzen des Wissenserwerbs markieren* (vgl. Schlüter und Brinda, 2007, S. 288). Anschließend geben Schlüter und Brinda (2007, S. 289 f.), für das Beispiel reguläre Sprachen entsprechend der fünf Prozessbereiche der GI-Empfehlungen je fünf Kompetenzen für die Sekundarstufen I und II an (zu den GI-Empfehlungen vgl. Abschnitt 2.5). Nachfolgend stellen sie eine Beispielaufgabe vor, die mit verschiedenen Teilaufgaben Kompetenzen zu vier der fünf oben genannten Prozessbereiche enthält. Außerdem ordnen sie jeder Teilaufgabe eine Kategorie der Lernzieltaxonomie nach Bloom (1972) zu. Schlüter und Brinda (2007, S. 290 f.) schlagen vor, die Inanspruchnahme von Hilfestellungen zu jeder Teilaufgabe als Indikator für die Unterscheidung einzelner Kompetenzstufen zu interpretieren (vgl. Schlüter und Brinda, 2007, S. 291 f.). Sie ergänzen also die vorgestellte Aufgabe durch Hilfestellungen mit Lösungshinweisen, auf die jene Schülerinnen und Schüler zurückgreifen, welche die Aufgabe ohne Hilfe nicht lösen können. Je nach genutzter Hilfestellung entspricht die Lösung der Teilaufgabe dann auch unterschiedlichen Lernzielstufen, die als unterschiedliche Niveaustufen angenommen werden (vgl. Schlüter und Brinda, 2007, S. 291 f.). In einem Ausblick stellen Schlüter und Brinda (2007, S. 292 f.) dar, dass in weiteren Arbeiten ein erstes Kompetenzmodell anhand der am Beispiel erläuterten Vorgehensweise erstellt und anschließend in einer Feldstudie

empirisch überprüft werden soll.
Da zu dem von Schlüter und Brinda (2007, S. 283 ff.) vorgestellten Ansatz derzeit noch kein vollständiges Kompetenzmodell existiert und noch keine Erfahrungen mit dem Einsatz der Hilfestellungen im Informatikunterricht der Sekundarstufe I dokumentiert wurden, werden an dieser Stelle keine Rückschlüsse auf das weitere Vorgehen in dieser Arbeit gezogen.

Die relativ konkrete Beschreibung der Kompetenzen in den beiden letztgenannten Kompetenzmodellen lässt diese für die Strukturierung, Vorbereitung, Durchführung und Auswertung von kompetenzorientiertem Informatikunterricht geeigneter erscheinen, als die vorgestellten Kompetenzmodelle für eine gesamte informatische Bildung. Nach Auffassung des Autors wäre ein wichtiger nächster Arbeitsschritt auch hier die Bereitstellung bzw. Einordnung einer größeren Anzahl von Aufgaben in die beiden Kompetenzmodelle.

## 3.6 Zusammenfassung

In diesem Kapitel wurde dargestellt, was in dieser Arbeit unter Kompetenzen, Kompetenzmodellen, Beispiel- Unterrichts- und Testaufgaben verstanden wird. Auf die Entwicklung von Kompetenzmodellen und Aufgaben wurde eingegangen. Weiterhin wurde erläutert, welche Materialien im Rahmen dieser Arbeit entwickelt werden sollen, und wie diese bei der Umsetzung kompetenzorientierten Informatikunterrichts eingesetzt werden können. Existierende Kompetenzmodelle für die gesamte informatische Bildung und für einzelne Bereiche der informatischen Bildung wurden vorgestellt und eingeordnet.

# 4 Puck als visuelles Werkzeug im Informatikunterricht

> *Learning to program is not easy.* (du Bouley, 1989, S. 283)
>
> *Learning to program can be very difficult for beginners of all ages.* (Kelleher und Pausch, 2005, S. 83)
>
> *Learning to program is hard however.* (Robins u. a., 2003, S. 137)

Entsprechend der im Kapitel 1 vorgestellten dritten Forschungsfrage soll überprüft werden, inwieweit sich Puck eignet, um Schülerinnen und Schülern der Jahrgangsstufen 8 bis 10 eine Einführung in die Programmierung zu geben. Damit die Ergebnisse der in den Kapiteln 7 und 8 dargestellten Untersuchungen eingeordnet werden können, wird Puck nachfolgend vorgestellt. Abschnitt 4.1 gibt eine Einführung in Puck, in der auch die verwendeten Repräsentationsformen beschrieben werden. Um Puck von anderen Systemen abzugrenzen werden im Abschnitt 4.2 verschiedene visuelle Werkzeuge vorgestellt und Unterschiede zu Puck angegeben. Problemfelder bei der Einführung in die Programmierung sowie Gründe, die für den Einsatz visueller Werkzeuge und speziell für den Einsatz von Puck sprechen, werden im Abschnitt 4.3 näher betrachtet. Damit bei der Erstellung weiterer visueller Werkzeuge auf die bei der Entwicklung von Puck gewonnenen Erkenntnisse zurückgegriffen werden kann, wird im Abschnitt 4.4 reflektiert, inwieweit die von Arnold und Hartmann (2007a, S. 171 ff.) vorgestellten Empfehlungen zur Entwicklung von interaktiven Lernumgebungen bei Puck nachvollzogen werden können.

## 4.1 Eine kurze Einführung in Puck

Puck ist ein Programmiersystem zur Einführung in die imperative Programmierung, bei dem Programme aus Bausteinen zusammengesetzt werden. Das System ist so aufgebaut, dass keine Syntaxfehler möglich sind (vgl. Kohl, 2004a, S. 86 ff.). Der Umfang der Sprache ist den Anforderungen des Anfangsunterrichts entsprechend klein gehalten (vgl. Kohl, 2004a, S. 91). Abbildung 4 zeigt ein Beispiel eines Puck-Programms, das Geschenkvorschläge für Jungen und Mädchen unterschiedlichen

Alters macht. Die dort dargestellte Puck-Programmieroberfläche ist in drei Bereiche aufgeteilt:

- Auf der linken Seite befindet sich die Bausteinquelle. Von dort aus können die vorgegebenen Bausteine in den Arbeitsbereich gezogen werden.
- Im Arbeitsbereich, in der Mitte des Bildschirms, werden die Bausteine puzzleartig zu Programmen zusammengefügt. Prozeduren, Verzweigungen und Schleifen haben jeweils einen Rumpf mit erweiterbaren Anschlussstellen, an die Anweisungsbausteine angefügt werden können.
- Auf der rechten Seite der Puck-Programmieroberfläche befindet sich im unteren Bereich ein Textfeld für Kommentare, Notizen, Reflexionen und ähnliches. Darüber ist die Attributtabelle, in der spezifische Einstellungen an den Bausteinen vorgenommen werden können. So kann z. B. bei Variablen Name und Datentyp festgelegt werden. Veränderungen in der Attributtabelle werden meist direkt visuell dargestellt. So ändert sich z. B. die Farbe einer Variablen, wenn der Datentyp geändert wurde (Integer – blau, Boolean – grau).

Anhang A gibt eine Übersicht über die Bausteine, ihre Funktionalitäten, Einstellmöglichkeiten und Darstellungen. Das Ausgabefenster eines Puck-Programms ist in zwei Bereiche unterteilt. Im unteren Drittel erfolgt die Ausgabe von Texten. Im oberen Bereich werden Grafiken ausgegeben. Bei Grafikausgaben wird mit einem Koordinatensystem gearbeitet, bei dem sich der Nullpunkt in der linken unteren Ecke befindet. Diese Koordinatendarstellung ist Schülerinnen und Schülern aus dem Mathematikunterricht bekannt. Abbildung 5 zeigt die Ausgabe eines Programms, das das Zeichnen des „Hauses vom Nikolaus“ mit dem dazugehörigen Reim visualisiert.

Auf eine ausführlichere Beschreibung des Puck-Systems wird an dieser Stelle verzichtet. Der interessierte Leser sei auf die Internetseite `www.ipuck.de` verwiesen. Dort sind unter anderem Einführungsvideos, eine Online-Hilfe und Literaturverweise zusammengetragen (zu Literaturverweisen siehe auch Kohl, 2007a). Für eine detailliertere Darstellung der in Vorbereitung auf die Untersuchungen dieser Arbeit entwickelten Features sei auf die Artikel „Puck – ein Sommernachtstraum“ und „Puck – A Visual Programming System for Schools“ verwiesen (vgl. Kohl u. a., 2007 und Kohl, 2007b). Seit Beginn der Voruntersuchungen im Sommer 2006 wurde keine neue Puck-Version veröffentlicht (vgl. Kapitel 7). Die im Abschnitt 7.6 dargestellten Erfahrungen im Schuljahr 2006/2007 zeigten, dass Puck 2.4 gut im Unterricht eingesetzt werden kann.

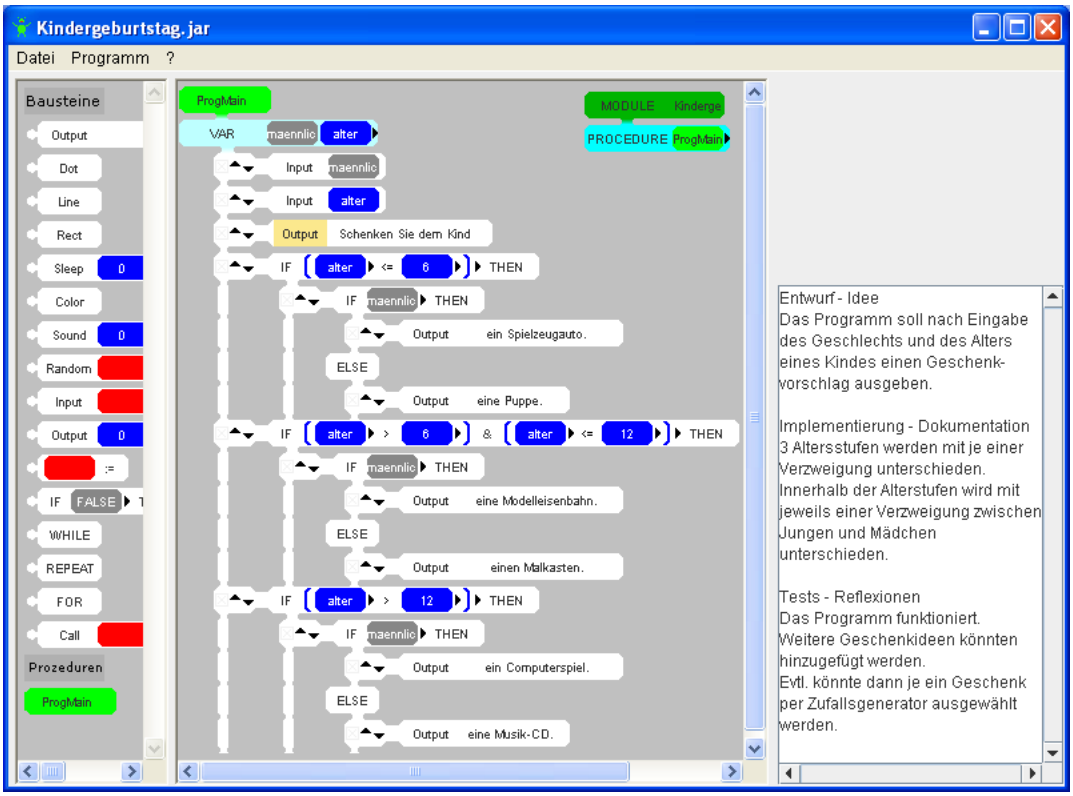



Abbildung 4: Ein Puck-Programm, das Geburtstagsgeschenke vorschlägt

### In Puck verwendete Repräsentationsformen

Nach Bruner (1988, S. 27 ff.) können die drei folgenden Repräsentationsformen unterschieden werden:[42]

enaktiv – Repräsentation durch Handlungen
ikonisch – Repräsentation durch Bilder
symbolisch – Repräsentation durch Symbole, wie z. B. Sprache oder Text

Außerdem unterscheiden Hartmann u. a. (2006, S. 116 f.) neben der echten enaktiven Repräsentation noch zwischen den folgenden beiden Repräsentationen:

*semi-enaktiv – Der Lehrer führt eine enaktive Demonstration durch, die Lernenden beobachten nur.*

[42] Bruner (1988, S. 27 ff.) stellt in Bezug auf das verwendete Medium einer Repräsentation dar, *daß man etwas auf drei verschiedene Weisen kennen kann: dadurch, daß man es tut, dadurch, daß man es sich bildlich vorstellt, und dadurch, daß man ein symbolisches Mittel wie z. B. die Sprache verwendet.*

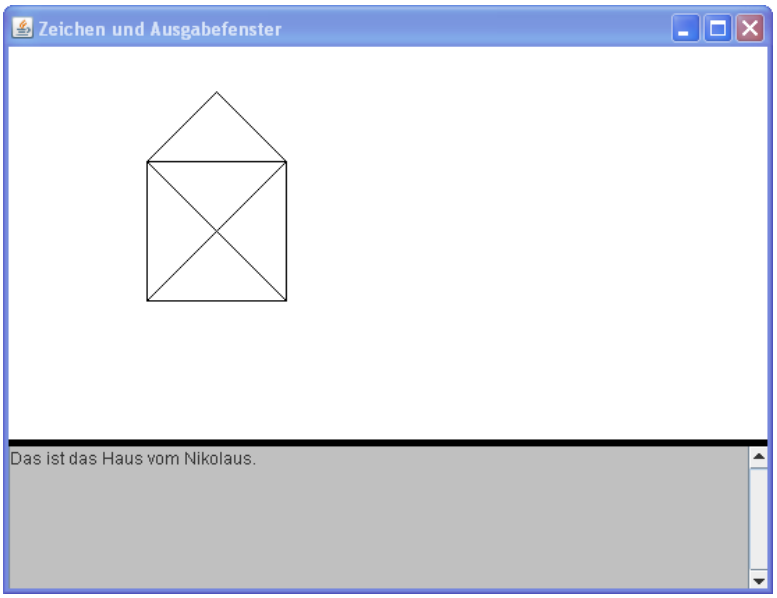



Abbildung 5: Die Ausgabe eines Puck-Programms

*virtuell-enaktiv – Die enaktiven Vorgänge werden durch Manipulationen von Objekten in einer computergestützten Umgebung simuliert.*

Die verschiedenen Repräsentationsformen werden in Abbildung 6 am Beispiel der Türme von Hanoi verdeutlicht werden, welches schon des öfteren als Anschauungsobjekt in diesem Kontext verwendet wurde (vgl. z. B. Aebli, 1994, S. 71 ff.):

- Eine textliche, also symbolische Darstellung des Problems geben Fothe und Kerner (1988, S. 18 ff.):

  *Als Modell eignen sich drei senkrechte Stangen, die auf einem Holzbrett befestigt sind. Auf einer der Stangen (Nr. 1) steckt am Anfang eine Anzahl unterschiedlich großer, in der Mitte durchbohrter Scheiben. Sie sind nach abnehmender Größe von unten nach oben übereinander gestapelt.*
  *Diese Scheiben werden nun nach folgenden Regeln versetzt:*
  *1. Stecke immer nur eine Scheibe von einer Stange auf eine andere um.*
  *2. Lege stets eine kleinere Scheibe auf eine größere.*
  *Alle Scheiben sind auf eine andere Stange (Nr. 2) zu versetzen.*

- Eine ikonische Darstellung in Form eines Bildes wie in Abbildung 6 hilft, das Geschriebene zu verstehen.

- Wenn das Umstapeln der Türme enaktiv, mit den Händen probiert werden

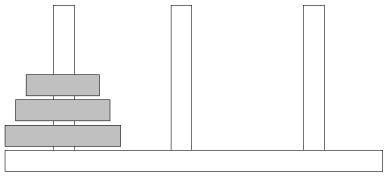


Abbildung 6: Ikonische Darstellung der Türme von Hanoi

soll, so eignet sich z. B. ein Holz-Spiel zum Anfassen.

- Da im Informatikunterricht meist nicht Holzmodelle in ausreichender Anzahl für alle Schülerinnen und Schüler zur Verfügung stehen, wird bei der Behandlung dieses Themas oft semi-enaktiv vorgegangen, d. h. die Lehrkraft, eine Schülerin oder ein Schüler führt das Spiel an einem Modell vor. Dabei kann auf Anleitungen und Hinweise der Schülerinnen und Schüler eingegangen werden.
- Eine weitere Variante ist, virtuell-enaktiv vorzugehen, also den Lernenden mit einem computergestützten Werkzeug, wie z. B. einem Applet, die Möglichkeit zu geben, die Scheiben am Computer zu bewegen.[43]

Wittmann (1976, S. 15) legt dar, dass die enaktive und die ikonische Form gewöhnlich leichter verständlich sind und deshalb die Verwendung dieser Darstellungsformen im Mathematikunterricht von allergrößter Bedeutung ist. Diese Argumentation kann nach Auffassung des Autors auch auf den Informatikunterricht übertragen werden. Hartmann u. a. (2006, S. 116) empfehlen mit Verweis auf Bruner (1988) im Unterricht Denkoperationen, wann immer möglich, auf mehreren Ebenen durchzuspielen.[44] Fothe (2007, S. 40 ff.) zeigt am Beispiel von Rollenspielen, dass gerade ein Wechsel zwischen den drei Darstellungsweisen interessante Möglichkeiten für die Unterrichtsgestaltung in der Informatik bietet. Humbert (2005, S. 34) empfiehlt zu berücksichtigen, dass die drei erstgenannten Repräsentationsformen aufeinander aufbauen und verweist im Blick auf die Reihenfolge auf das EIS-Prinzip (enaktiv-ikonisch-symbolisch).
Es erscheint ungünstig, das Unterrichtsthema Programmierung ausschließlich anhand der symbolischen – also der nach dem EIS-Prinzip letzten – Repräsentationsform einzuführen. Die Kombination verschiedener Repräsentationsformen

[43] Ein Applet zu den Türmen von Hanoi ist auf folgender Webseite zu finden: `http://www.mathematik.ch/spiele/hanoi_mit_grafik/` [Zugriffsdatum: 14.10.2008]

[44] Hartmann u. a. (2006, S. 122 f.) stellen dar, dass die enaktive Repräsentationsform sich besonders für den Einstieg in ein Thema eignet und geben verschiedene Unterrichtsbeispiele an, in denen enaktive und ikonische Repräsentationsformen genutzt werden.

ermöglicht das von Hartmann u. a. (2006, S. 116) empfohlene Durchspielen von Denkoperationen auf verschiedenen Ebenen. Außerdem wird dadurch der von Fothe (2007, S. 40 ff.) dargestellte Wechsel der Repräsentationsformen beim Thema „Programmierung“ im Unterricht ermöglicht.
Puck verbindet Elemente ikonischer, symbolischer und vituell-enaktiver Repräsentationsformen. Programme, Bausteine und deren Elemente sind ikonisch dargestellt. Die Bausteine von Anweisungen, Alternativen und Schleifen sind unterschiedlich aufgebaut und verschiedene Farben visualisieren z. B. die verwendeten Datentypen. Alle Bausteine enthalten aber auch ein aus der textuellen Programmierung bekanntes Schlüsselwort in symbolischer Darstellung. Programme werden aus Bausteinen wie ein Puzzle zusammengesetzt. Das Verbinden der Bausteine zu Programmen wird somit virtuell-enaktiv repräsentiert. Ein Übergang von Puck zu einer textuellen Programmiersprache kann außerdem durch das Generieren von Quelltext und Pseudocode vorbereitet werden (vgl. Kohl u. a., 2007; Kohl, 2007b).

## 4.2 Abgrenzung zu anderen visuellen Werkzeugen im Informatikunterricht

In den letzten Jahren sind verstärkt Werkzeuge für den Informatikunterricht entwickelt und im Informatikunterricht eingesetzt worden. Eine ausführliche Übersicht über textuelle und visuelle Programmierwerkzeuge, die die Hürden für Einsteiger verringern sollen, geben Kelleher und Pausch (2005; vgl. auch Kelleher, 2006, S. 221 ff.).
Im Folgenden wird dargelegt, was im Rahmen dieser Arbeit unter „visuellen Werkzeugen“ verstanden werden soll.

### Visuelle Werkzeuge

Softwaresysteme, die unter Verwendung visueller Elemente zur Vermittlung informatischer Inhalte im Unterricht eingesetzt werden, werden in dieser Arbeit als visuelle Werkzeuge bezeichnet.[45]
Im Folgenden werden zwei visuelle Werkzeuge vorgestellt. Diese und auch die nachfolgend dargestellten Beispiele werden jeweils kurz erläutert und an einem

[45] Schiffer (1998, S. 63) definiert den Begriff „visuell“ wie folgt:
*Visuell ist die Bezeichnung für jene Eigenschaft eines Objekts, durch die mindestens eine Information über das Objekt, die für das Erreichen eines Handlungsziels unverzichtbar ist, nur durch das visuelle Wahrnehmungssystem des Menschen gewonnen werden kann.*

Screenshot veranschaulicht. Literatur- und Internetverweise, die auch weiterführende Informationen enthalten, werden angegeben.

**LEO** ist eine Lernumgebung für objektorientiertes Modellieren im Informatikunterricht. Mit LEO können verschiedene vorgegebene objektorientiert modellierte Szenarien exploriert werden. In den Szenarien ist es möglich Objekte zu vorgegebenen Klassen zu erzeugen, Methoden aufzurufen und Attribute zu ändern. Leo unterstützt verschiedene Ansichten. So gibt es neben einer Klassen- und einer Objektsicht z. B. auch eine Realsicht, in der Objekte visuell dargestellt sind. Zum Explorationsmodul wurde auch ein Szenarieneditor entwickelt.
Der LEO-Screenshot in Abbildung 7 zeigt links die Modellierung verschiedener Klassen zum Thema Mobilkommunikation. Im mittleren Bereich – der Objektsicht – wurden zu den vorgegebenen Klassen dieses Szenarios verschiedene Objekte erzeugt, die im rechten Bereich – der Realsicht – visualisiert werden. In der Leiste am oberen Bildschirmrand können verschiedene Sichten ein- und ausgeblendet werden.
URL: `http://www.die.informatik.uni-siegen.de/pgleo/`
[Zugriffsdatum: 14.10.2008]
Literatur: Brinda, 2004

**LogicTraffic** ist eine interaktive Lernumgebung, bei der am Beispiel von Straßenkreuzungen in das Thema Aussagenlogik eingeführt wird. Durch die Steuerung von Ampeln mit Hilfe von Wahrheitstabellen und aussagenlogischen Formeln müssen Kollisionen an Straßenkreuzungen verhindert werden. Eine von Schülerinnen und Schülern entwickelte Konfiguration einer Ampelanlage kann mittels einer Simulation des Verkehrs getestet werden.
Im LogicTraffic-Screenshot in Abbildung 8 ist im linken Teil eine ampelgesteuerte Verkehrskreuzung dargestellt. Rechts davon wurden in der Wahrheitstabelle vom Benutzer die „sicheren“ Zustände mit einer „1“ markiert. Der Screenshot zeigt die Simulation der vorletzten Zeile der Wahrheitstabelle. Im unteren Bereich des Fensters können aussagenlogische Formeln auch direkt eingegeben werden.
URL: `http://www.swisseduc.ch/informatik/infotraffic`
[Zugriffsdatum: 14.10.2008]
Literatur: Arnold, 2007

Nachfolgend werden verschiedene Kategorien visueller Werkzeuge zum Thema „Algorithmen“ unterschieden und an Beispielen veranschaulicht. Die meisten Beispiele können mehreren der nachfolgend vorgestellten Kategorien zugeordnet werden. Für die Darstellungen wurde jeweils die nach Auffassung des Autors typischste Eigenschaft eines Werkzeugs zur Einteilung in eine Gruppe gewählt. Es wird

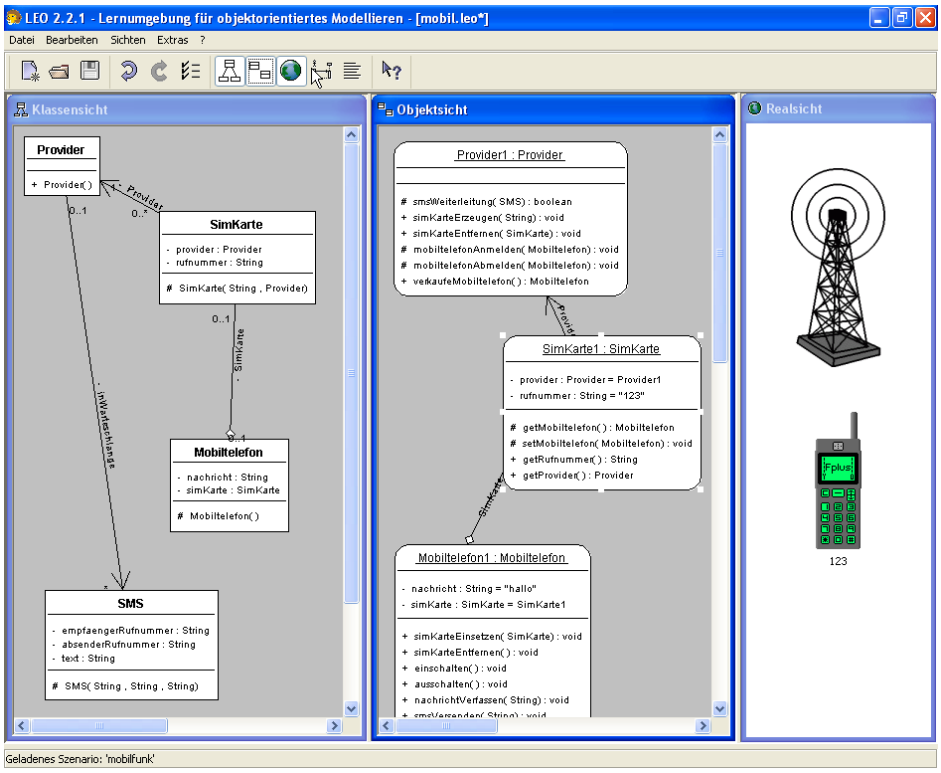



Abbildung 7: Screenshot LEO

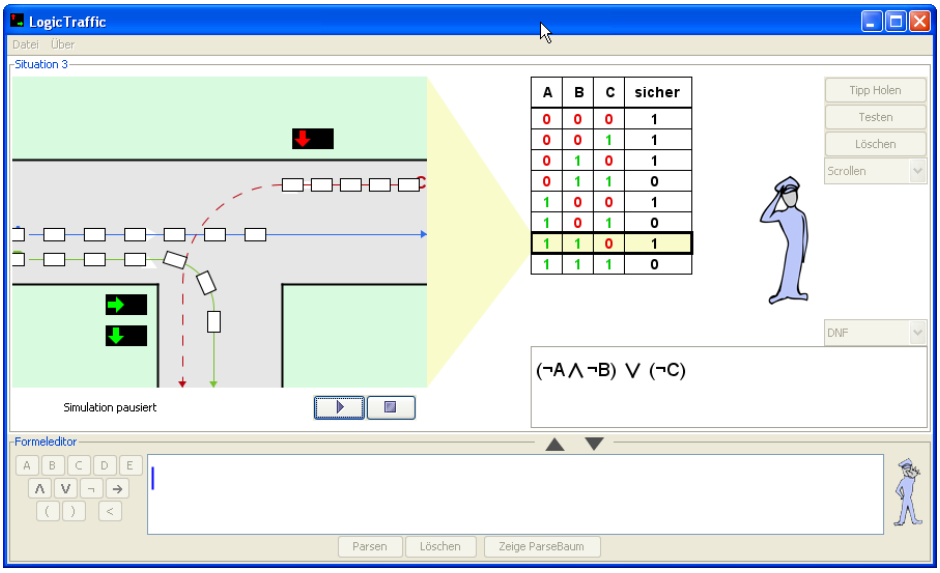



Abbildung 8: Screenshot LogicTraffic

jeweils analysiert, inwieweit Puck in die entsprechende Kategorie einzuordnen ist und was Puck von den vorgestellten Beispielen unterscheidet.

### Werkzeuge zur Softwarevisualisierung

Softwarevisualisierung hat das Ziel, softwaretechnische Sachverhalte zu illustrieren, das heißt zu zeigen, *wie Software strukturiert ist, was sie macht, wie sie funktioniert, warum sie funktioniert oder warum nicht* (vgl. Schiffer, 1998, S. 67). Schiffer (1998, S. 63 ff.) definiert Softwarevisualisierung als *werkzeugunterstützte, visuelle Darstellung von Software für Analysezwecke, d.h. für Untersuchungen der statischen und dynamischen Eigenschaften von Software.* Dabei können Eigenschaften visueller und textueller Programme visualisiert werden (vgl. Schiffer, 1998, S. 67). Nachfolgend werden zwei Beispiele für solche Werkzeuge vorgestellt.

**BlueJ** ist ein Programmiersystem für die Sprache Java, das speziell für den Einstieg in die objektorientierte Programmierung entwickelt wurde. Zu Java-Klassen können in BlueJ direkt per Mausklick Objekte interaktiv erstellt werden. Diese werden durch abgerundete Rechtecke visualisiert und können durch Methodenaufrufe manipuliert werden. Außerdem ist es jederzeit möglich, die Attributwerte der Objekte zu inspizieren. Die Benutzungsschnittstelle und der Debugger der BlueJ-Umgebung sind auf die Bedürfnisse von Programmierneulingen zugeschnitten. Klassendiagramme werden in BlueJ visuell dargestellt. Das Hinzufügen bzw. das Entfernen eines Vererbungspfeils, wirkt sich direkt auf den Quelltext der entsprechenden Dateien aus. Ansonsten erfolgt die Implementierung der Klassen in BlueJ in einem textuellen Editor. Im BlueJ-Screenshot in Abbildung 9 ist rechts oben ein Klassendiagramm dargestellt. Buttons, die das Hinzufügen von Klassen bzw. von Beziehungen zwischen diesen ermöglichen, befinden sich oben links. Im unteren Bereich sind drei Objekte als abgerundete rote Rechtecke visualisiert.
URL: `http://www.bluej.org` [Zugriffsdatum: 14.10.2008]
Literatur: Barnes und Kölling, 2000 sowie Kölling u. a., 2003

**Jeliot 3** ist ein Programm, das darstellt, wie Java-Programme ausgeführt werden. Dazu generiert Jeliot 3 automatisch eine Animation, in der Methodenaufrufe, Variablenbelegungen, Kontrollstrukturen und Operationen während der Ausführung eines Java-Programms Schritt für Schritt veranschaulicht werden. Der Screenshot zu Jeliot 3 in Abbildung 10 zeigt auf der linken Seite den Quelltext eines Java Programms und auf der rechten Seite die dazu generierte Animation. Im unteren Teil des Fensters befinden sich Steuerelemente für die Animation sowie eine Ausgabekonsole.
URL: `http://cs.joensuu.fi/~jeliot/index.php`

[Zugriffsdatum: 14.10.2008]
Literatur: Ben-Ari u. a., 2002

In der aktuellen Version von Puck gibt es im Gegensatz zu den beiden vorgestellten Werkzeugen keine Möglichkeit der Softwarevisualisierung, da dieses Feature in keiner Phase der Entwicklung von Lehrkräften gefordert wurde.

### Programmierwerkzeuge mit visueller Ausgabe

In verschiedenen Programmierwerkzeugen für Einsteiger wird versucht, Mittel zur Verfügung zu stellen, um mit möglichst wenig Aufwand Programme entwickeln zu können, die visuelle Elemente enthalten. Einige Werkzeuge stellen dafür eine Miniwelt zur Verfügung, in der ein programmierbarer Aktor Aufgaben erfüllen muss (vgl. hierzu Brusilovsky u. a., 1997, S. 65 ff. bzw. Hartmann u. a., 2004, S. 10 f.) Andere Werkzeuge ermöglichen Anfängern, mit einfachen Befehlen grafische Ausgaben zu erstellen. Programmierwerkzeuge mit visueller Ausgabe nutzen eine textuelle oder visuelle Programmiersprache, um ein Informatiksystem mit visuellen Elementen zu erstellen oder zu modifizieren. Erste Entwicklungen in dieser Richtung wurde von Papert (1985) mit der Programmierung von Turtle-Grafiken vorgestellt. Eine Übersicht über Minilanguages mit geringem Sprachumfang geben Brusilovsky u. a. (1997). Im Folgenden werden zwei Programmierwerkzeuge vorgestellt, bei denen es relativ einfach ist, visuelle Ausgaben zu erzeugen.

**Robot Karol** ist ein System zur Einführung in die Programmierung, bei dem ein Roboter mit einfachen Befehlen programmiert wird. Die entwickelten Programme steuern den visuell dargestellten Roboter, wodurch ein direktes Überprüfen der Ergebnisse möglich wird. In der verwendeten pseudocodeähnlichen Programmiersprache mit deutschen Schlüsselwörtern, werden keine Variablen verwendet.[46]

Im Screenshot in Abbildung 11 befindet sich im linken oberen Bereich ein textuelles Programm, das den Roboter in der rechten oberen Hälfte des Screenshots steuert. In der Mitte des Bildschirms sind Steuerelemente für die Programmausführung, die Modifikation der Miniwelt und für die manuelle Steuerung des Roboters zu sehen. Im unteren Bereich des Fensters ist auf der linken Seite das Programm als Baum dargestellt. Rechts daneben werden in einer Konsole bei der Programmausführung auftretende Fehlermeldungen angezeigt.

URL: `http://www.schule.bayern.de/karol/` [Zugriffsdatum: 14.10.2008]
Literatur: Freiberger, 2002; Freiberger und Krsko, 2003; Pattis, 1995

[46] Die Idee vom Roboter Karol geht auf Pattis (1995) zurück. Es gibt verschiedene Umsetzungen dieser Idee (vgl. Brusilovsky u. a., 1997).

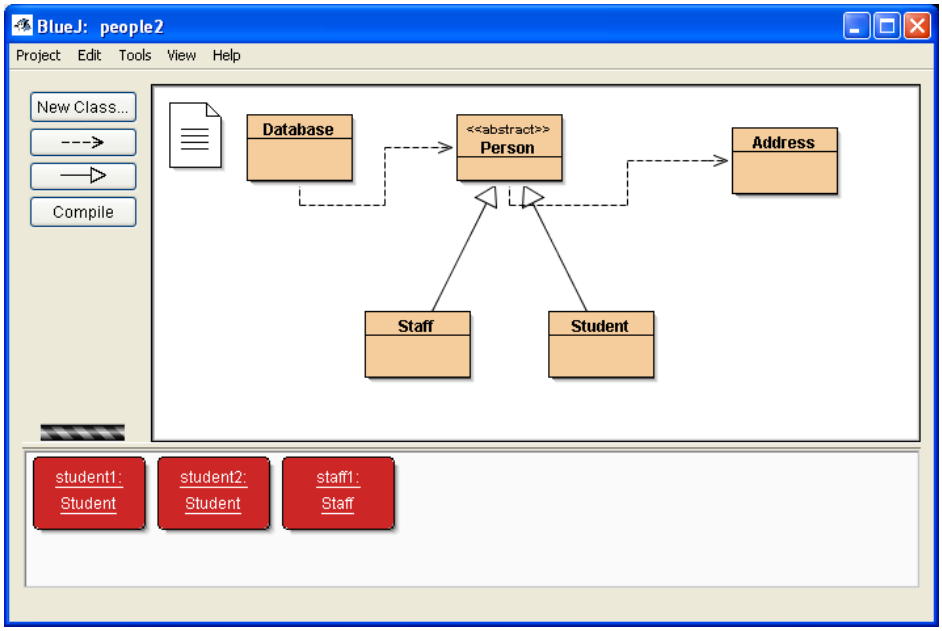



Abbildung 9: Screenshot BlueJ

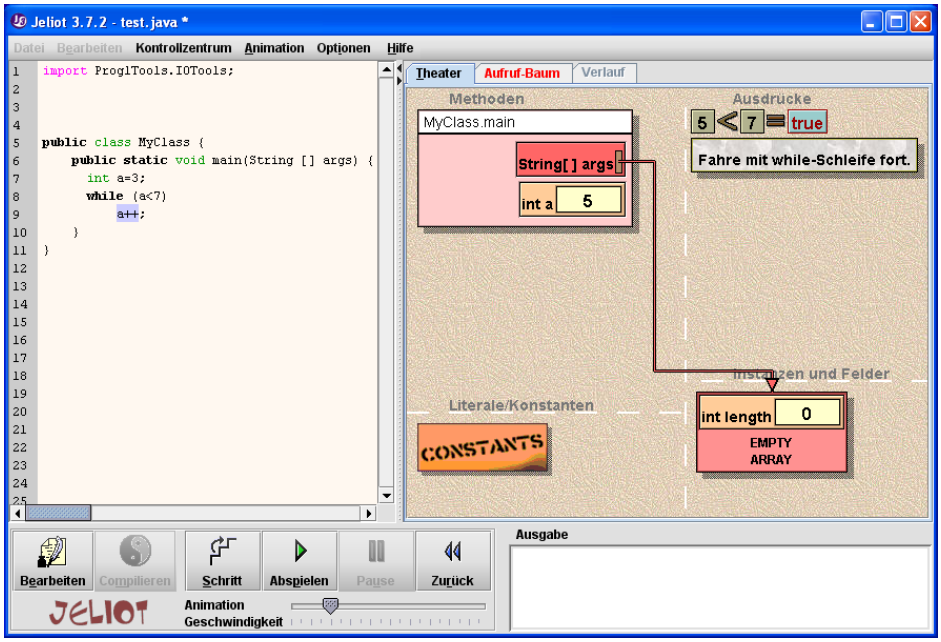



Abbildung 10: Screenshot Jeliot 3

**Greenfoot** ist ein Programmiersystem, bei dem unter Verwendung einer speziell für Einsteiger entwickelten Java-Bibliothek Programme erstellt werden können. Die Programme nutzen Aktoren, die sich in einer zweidimensionalen Welt bewegen können. Greenfoot stellt eine überschaubare Menge einfacher Methoden bereit, mit der sich eine Vielzahl von Aufgaben mit Hilfe von objektorientierter Programmierung lösen lässt. Das System ist insbesondere für die Entwicklung von kleineren Spielen geeignet, wodurch die Motivation der Schülerinnen und Schüler geweckt werden kann. In Greenfoot erfolgt die Programmierung in einem textuellen Editor.
Im Greenfoot-Screenshot in Abbildung 12 ist eine Miniwelt dargestellt, in der eine vom Benutzer zu steuernde Schlange kleine Frösche einsammeln muss. Die Klassenstruktur des Spiels ist auf der rechten Seite des Bildschirms dargestellt. Im unteren Bildschirmbereich befinden sich Buttons und ein Schieberegler, mit denen die Programmausführung gesteuert werden kann.
URL: `http://www.greenfoot.org/` [Zugriffsdatum: 14.10.2008]
Literatur: Henriksen und Kölling, 2004

In Puck wurde keine Miniwelt umgesetzt, wie das bei Robot Karol der Fall ist. Puck ermöglicht Anfängern mit den Bausteinen Dot, Line, Rect/Ellipse und Color auf einfache Weise visuelle Ausgaben zu erstellen und kann somit auch als ein Programmierwerkzeug mit visueller Ausgabe charakterisiert werden. Im Vergleich zu Greenfoot ist die Vielfalt der mit Puck umsetzbaren Programme durch die Anzahl der Bausteine eingeschränkt. Diese Einschränkung ermöglicht aber die spezielle Form der visuellen Programmierung, durch die Einstiegshürden reduziert werden können (vgl. Kapitel 7 und 8).

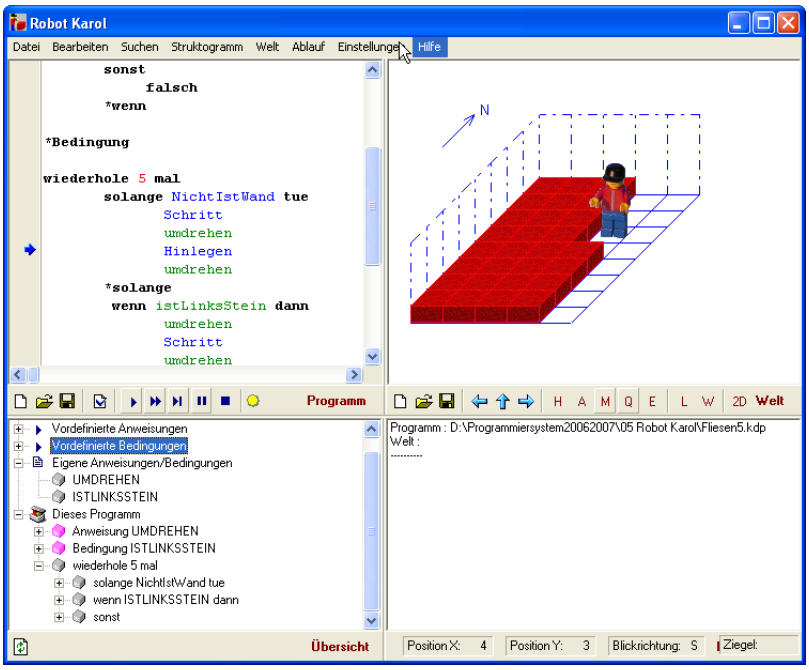



Abbildung 11: Screenshot Robot Karol

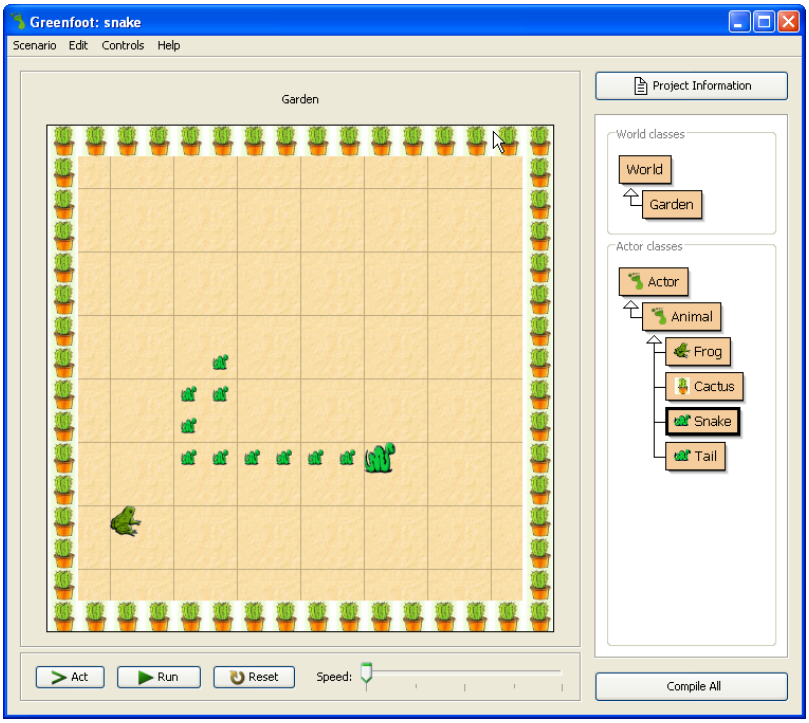



Abbildung 12: Screenshot Greenfoot

### Visuelle Softwarebeschreibungssprachen

Eine visuelle Softwarebeschreibungssprache ist nach Schiffer (1998, S. 63 ff.) *eine visuelle Sprache zur Beschreibung bestimmter Aspekte von Software. Aus den damit erstellten Beschreibungen ist Programmcode maschinell generierbar.*[47] Diese Aspekte können z. B. die Erstellung einer Benutzungsschnittstelle durch einen visuellen (GUI-)Editor oder die Generierung einer Grundstruktur von Klassen aus (UML-)Klassendiagrammen sein. Andere Aspekte der zu entwickelnden Software, wie z. B. die Implementierung der Methoden einer Klasse, werden mit textuellen Programmiersprachen beschrieben. Zwei Beispiele für Entwicklungsumgebungen, die visuelle Softwarebeschreibungssprachen nutzen, werden im Folgenden vorgestellt:

**Java-Editor** ist eine Entwicklungsumgebung für die Sprache Java, für den Einsatz in Schulen. Das System enthält Softwarebeschreibungssprachen für UML-Klassendiagramme und Benutzungsschnittstellen. Die Programmierung von Methoden erfolgt im Java-Editor textuell.
Der Screenshot des Java-Editors in Abbildung 13 zeigt auf der linken Seite ein Fenster, in der Labels, Textfelder und ein Button zu einer Benutzungsschnittstelle zusammengefügt wurden. Im rechten Bereich des Screenshots ist eine Javaklasse mit Attributen und Methoden visuell dargestellt. Veränderungen an dieser visuellen Repräsentation der Klasse wirken sich direkt auf den Quelltext derselben aus. In der Mitte des Fensters befindet sich ein Editor, in dem zumindest die Methoden der Java-Klassen textuell implementiert werden müssen.
URL:
`http://lernen.bildung.hessen.de/informatik/javaeditor/index.htm`
[Zugriffsdatum: 14.10.2008]
Literatur: Röhner, 2005

**Delphi** ist gleichzeitig der Name einer Entwicklungsumgebung und einer Programmiersprache. Für das Erstellen von Benutzungsschnittstellen gibt es in Delphi einen visuellen Editor, in dem Buttons, Textfelder, Panels, Labels und andere visuelle Elemente platziert werden können. Zu der so visuell entwickelten Benutzungsschnittstelle wird dann eine Datei generiert, in der die Anordnung der Komponenten textuell beschrieben ist. Auch in Delphi erfolgt die Implementierung von Methoden textuell.
Im Delphi-Screenshot in Abbildung 14 sind oben Textfelder, Buttons und

[47] Schiffer (1998, S. 63) definiert den Begriff „visuelle Sprache“ wie folgt:
*Eine visuelle Sprache ist eine formale Sprache mit visueller Syntax oder visueller Semantik und dynamischer oder statischer Zeichengebung.*

andere Elemente angeordnet. Aus diesen wurde in der Mitte des Screenshots eine Benutzungsschnittstelle zusammengefügt. Links unten ist ein textueller Editor dargestellt, in dem z. B. die beim Klick auf die Buttons auszuführenden Methoden implementiert wurden. Rechts unten befindet sich der Objektinspektor, in dem verschiedene Attribute zu den auf der Oberfläche platzierten Elementen verändert werden können.
URL: `http://www.borland.com/de/products/delphi/index.html`
[Zugriffsdatum: 14.10.2008]
Literatur: Matthäus, 2006; Orlamünder u. a., 2002

Im Gegensatz zu den beiden vorgestellten visuellen Softwarebeschreibungssprachen werden in Puck nicht nur bestimmte Aspekte der Programmierung visuell beschrieben, sondern der gesamte Prozess der Programmentwicklung erfolgt mit Hilfe einer visuellen Sprache. Die beiden vorgestellten Werkzeuge sind aufgrund der textuellen Darstellung von Algorithmen universell einsetzbar und Programme beliebiger Größe können entwickelt werden. Die Probleme, die mit Puck bearbeitet werden können, sind durch die durchgängig visuelle Programmierung in Funktionalität und Größe beschränkt. Ob dies problematisch für eine Einführung in die Programmierung ist, soll im Rahmen dieser Arbeit untersucht werden.

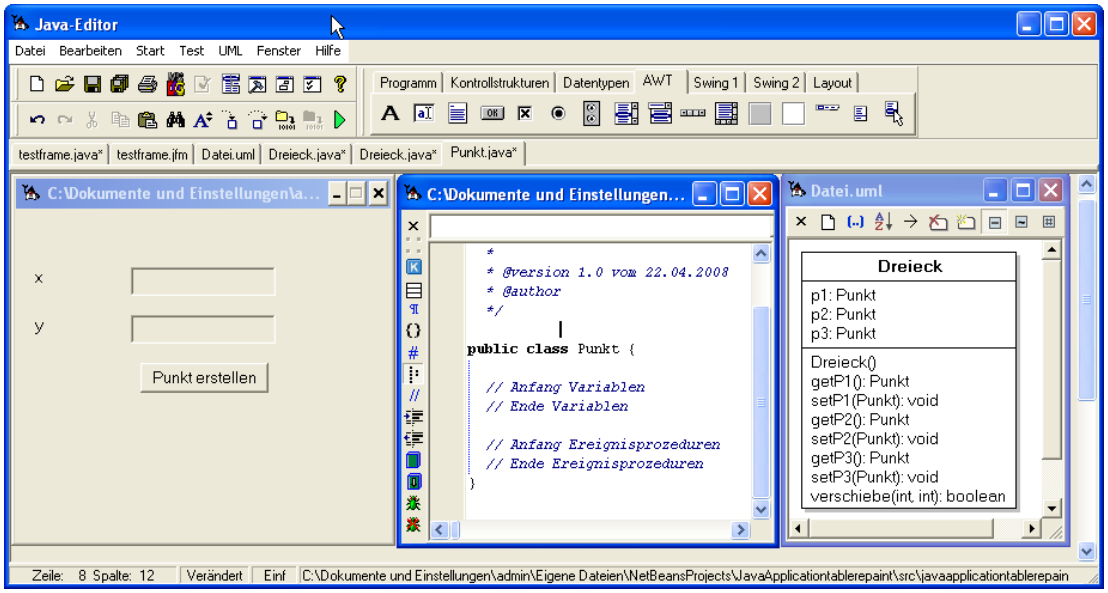



Abbildung 13: Screenshot Java Editor

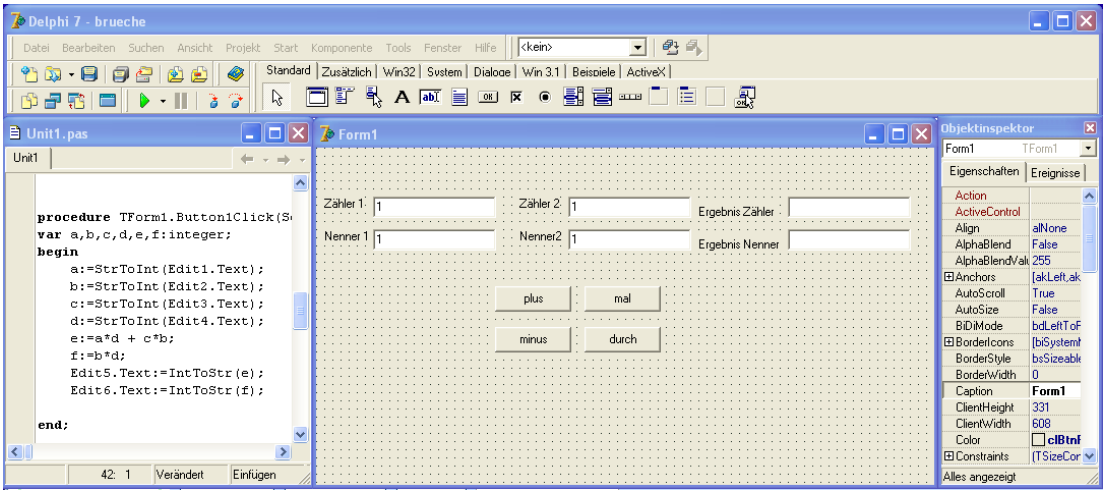



Abbildung 14: Screenshot Delphi

### Visuelle Programmiersprachen

Nach Schiffer (1998, S. 64) ist eine visuelle Programmiersprache *eine visuelle Sprache zur vollständigen Beschreibung der Eigenschaften von Software.* Eine Übersicht über Vor- und Nachteile visueller Programmiersprachen findet sich bei Schiffer (1998, S. 69 ff.). Im Folgenden werden zwei Beispiele für visuelle Programmiersprachen, die in Schulen eingesetzt werden, vorgestellt.

**Kara** ist die Bezeichnung für ein Programmiersystem und einen gleichnamigen Marienkäfer, der zusammen mit Kleeblättern, Pilzen und Baumstümpfen in einer einfachen, grafisch dargestellten Welt „lebt". Kara kann mit Hilfe von endlichen Automaten visuell programmiert werden. Bei der Programmausführung wird visualisiert, in welchem Zustand sich der Marienkäfer befindet und welche Aktionen er gerade ausführt. Mit dem Kara-Programmiersystem können auch die Themen Parallelität (Multi-Kara), Turing-Maschinen (Turing-Kara) und Roboter (Lego-Kara) bearbeitet werden. Um den Übergang von Kara zu textuellen Sprachen zu erleichtern, ist es auch möglich, Kara mit Java, JavaScript, Python oder Ruby zu programmieren.
In der linken Hälfte von Abbildung 15 ist Karas Welt dargestellt, die mithilfe der Symbole am rechten Bildschirmrand verändert werden kann. Marienkäfer, Baumstümpfe, Pilze und Kleeblätter können per „Drag and Drop" bewegt, hinzugefügt oder entfernt werden. Am linken Bildschirmrand befinden sich Buttons, mit deren Hilfe Kara manuell gesteuert bzw. zum Aufnehmen oder Ablegen eines Kleeblattes aufgefordert werden kann. Im oberen Teil des in der rechten Hälfte von Abbildung 15 dargestellten Programmierfensters ist ein Zustandsraum dargestellt, der per „Drag and Drop" manipuliert werden kann. Darunter können in Abhängigkeit von Karas Sensoren entsprechende Aktionen und Zustandsübergänge visuell programmiert werden.
URL: `http://www.swisseduc.ch/informatik/karatojava/kara/`
[Zugriffsdatum: 14.10.2008]
Literatur: Hartmann u. a., 2004

**Scratch** ist eine visuelle Programmiersprache, bei der Programme aus Bausteinen puzzleartig zusammengefügt werden. Mit Scratch-Programmen können durch grafisch dargestellte Aktoren, die sich in einer Ebene bewegen, leicht visuelle Ausgaben erzeugt werden. Beim Testen von Programmen wird visuell veranschaulicht, welche Teile der entwickelten Software gerade abgearbeitet werden. Entwickelt wurde Scratch hauptsächlich in den USA für den Einsatz in Computer Club Houses, in denen Kinder und Jugendliche mit Scratch Animationen, kleine Spiele und Programme entwickeln können.[48]

[48] Scratch wird auch im Unterricht eingesetzt (vgl. hierzu z. B. Romeike, 2007a).

Ein motivierendes Feature von Scratch ist es, dass Projekte per Knopfdruck in einer Galerie auf der Scratch-Homepage zur Verfügung gestellt werden können. Das Ziel der Entwickler vom Massachusetts Institute of Technology, Jugendliche für das Thema Programmierung zu begeistern, wurde angesichts von mehr als 100.000 veröffentlichten Projekten auf der Scratch-Homepage erfüllt.
Auf der linken Seite des Scratch-Screenshots in Abbildung 16 befinden sich vordefinierte visuelle Bausteine, die im mittleren Bereich des Scratch-Fensters zu einem Programm zusammengesetzt werden können. Im rechten Teil der Scratch-Programmierumgebung sind unten die verschiedenen im Programm eingesetzten Aktoren aufgelistet. Darüber wird die Ausgabe visualisiert, eine virtuelle Welt, in der sich die Aktoren bewegen und verschiedene Aktionen ausführen können.
URL: `http://scratch.mit.edu/` [Zugriffsdatum: 14.10.2008]
Literatur: Maloney u. a., 2004; Romeike, 2007a

Kara, Puck und Scratch kombinieren jeweils verschiedene Repräsentationsformen. Alle drei Systeme können aber als visuelle Programmiersprache angesehen werden, denn mit den verwendeten visuellen Sprachen kann eine Software bzw. ein einfaches Programm vollständig beschrieben werden. In Kara beschränken sich die lösbaren Aufgaben auf die Welt des Marienkäfers und die visuelle Programmierung erfolgt mit Automaten. Im Gegensatz dazu wird in Puck imperativ programmiert und es können Aufgaben gelöst werden, die typischerweise im Anfangsunterricht textueller Programmiersprache zum Einsatz kommen wie z. B. die im ersten Teil des Heftes „Problemlösen mit Python“ von Fothe (2002) vorgestellten Aufgaben. Scratch wurde zeitlich parallel mit Puck entwickelt, die Konzepte der visuellen Programmerstellung ähneln sich. Einige Gemeinsamkeiten und Unterschiede der beiden Systeme werden in Tabelle 1 analysiert. Trotz des ähnlichen Ansatzes können bei genauerer Betrachtung Unterschiede in Zielgruppe, Zielstellung, Programmierparadigma, den umsetzbaren Konzepten und den möglichen Ergebnissen der Programmierung festgestellt werden.

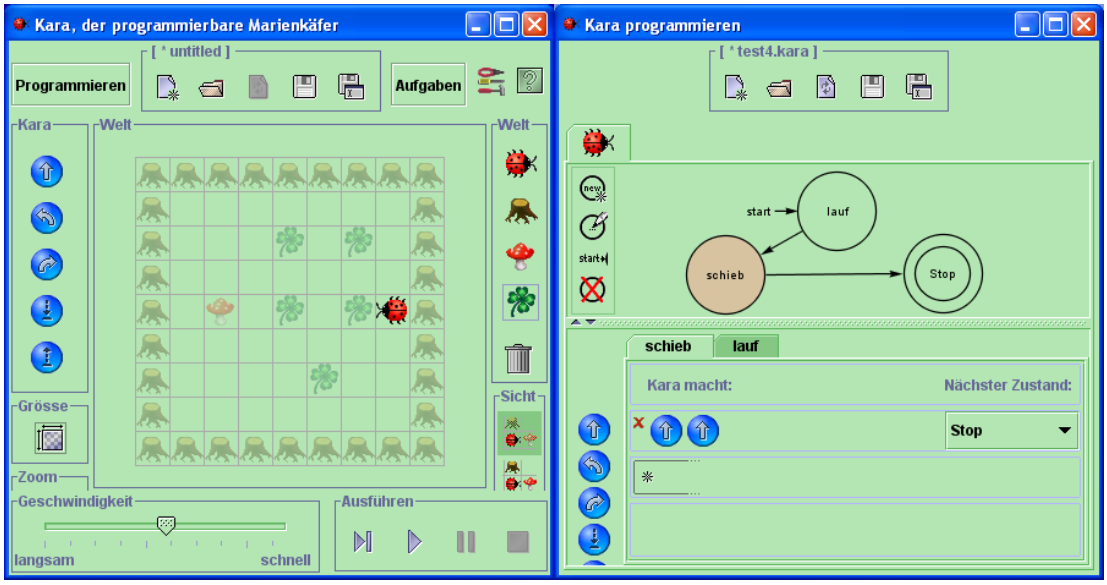



Abbildung 15: Screenshot Kara (links: Karas Welt; rechts: Kara programmieren)


Abbildung 16: Screenshot Scratch

| | Scratch | Puck |
|---|---|---|
| Programmierung | Programme werden aus Bausteinen zusammengesetzt | |
| Zielgruppe | Kinder in Computer-Clubs in den USA | Schülerinnen und Schüler an allgemeinbildenden Schulen |
| Zielstellung | Kinder für das Thema Programmierung begeistern | Ein Werkzeug nach den Anforderungen von Lehrenden zu entwickeln, mit dem im Unterricht ein erster Eindruck von imperativer Programmierung vermittelt werden kann |
| Programmier-paradigma | größten Teils imperative Programmierung, vereinzelt objektorientierte Konzepte | imperative Programmierung |
| Umsetzbare Konzepte | Kontrollstrukturen, Ereignissteuerung, Nachrichtenaustausch | Kontrollstrukturen, Datentypen, Prozeduren, Parameter |
| Mögliche Ergebnisse der Programmierung | interaktive Geschichten, Animationen, Spiele, Musikstücke | typische imperative Programme des Anfangsunterrichts inkl. einfacher grafischer Ausgaben und Sounds |
| Besonderheiten | Programme können auf einer Internetplattform präsentiert werden | Für den Umstieg auf eine textuelle Programmiersprache kann Quelltext generiert werden |

Tabelle 1: Gemeinsamkeiten und Unterschiede zwischen den visuellen Programmiersprachen Puck und Scratch

## 4.3 Gründe für den Einsatz visueller Werkzeuge bei Programmieranfängern

Wie die Zitate am Anfang dieses Kapitels zeigen, ist die „Einführung in die Programmierung“ ein Thema, das nicht leicht zu lehren bzw. zu lernen ist (vgl. z. B. du Bouley, 1989, S. 283; Bergin und Reilly, 2005, S. 293; Börstler, 2007, S. 9; Feldgen und Clua, 2004, S. S1H-11; Kelleher und Pausch, 2005, S. 83 oder Robins u. a., 2003, S. 137). Einen Überblick über Literatur und Forschungslage zur Einführung in die Programmierung geben Ala-Mutka (2003) sowie Robins u. a. (2003).[49] Im Folgenden sollen Problemfelder bei der Einführung in die Programmierung sowie Gründe, die für den Einsatz visueller Werkzeuge sprechen, dargestellt werden. Eine Klassifikation von Nachteilen, die im Zusammenhang mit Visualisierung von Informationen auftreten, geben Bresciani und Eppler (2008).

### Problembereiche bei der Einführung in die Programmierung

**Verschiedene Fertigkeiten müssen gleichzeitig erlernt werden** Du Bouley (1989, S. 283 f.) beschreibt die folgenden fünf sich überschneidenden Bereiche, die bei einer Einführung in die Programmierung erlernt werden müssen:

**orientation:** eine generelle Orientierung, wofür Programmierung gut ist, welche Probleme durch Programmierung gelöst werden können und welche Vorteile sich aus dem Erlernen dieser Fähigkeit ergeben

**notional machine:** eine Modellvorstellung vom Computer und von der Ausführung von Programmen

**notation:** die Syntax und die Semantik von Programmiersprachen

**structures:** Schemas und Pläne, die es ermöglichen, informatische Probleme zu lösen

**pragmatics:** Fähigkeiten, die bei der Planung, bei der Entwicklung, beim Testen und bei der Fehlersuche bzw. -korrektur, auch im Zusammenhang mit Werkzeugen, benötigt werden

Diese fünf Bereiche stellen für du Bouley (1989, S. 284) potentielle Quellen für Schwierigkeiten bei der Einführung in die Programmierung dar, die insbesondere durch ihre Kombination einen „Schock“ für Anfänger bedeuten können.[50]

[49] Einen Überblick zur Einführung in die objektorientierte Programmierung gibt Börstler (2007).

[50] Das Originalzitat von du Bouley (1989, S. 284) lautet: *None of these issues are entirely separable from the others, and much of the 'shock' [...] of the first few encounters between the*

**Probleme mit der Syntax einer textuellen Programmiersprache** Kelleher und Pausch (2005, S. 89f.) bezeichnen Syntax als eine der größten und frustrierendsten Herausforderungen für Anfänger. Die nachfolgend dargestellten Untersuchungen belegen dies: Garner u. a. (2005) sowie Robins u. a. (2006) gehen davon aus, dass effektivere Lernumgebungen kreiert werden können, wenn das Erlernen der ersten Programmiersprache besser verstanden wird, und untersuchten deshalb die Verteilung von Problemen in Programmierkursen für Anfänger an einer Universität. In den Jahren 2003 und 2004 wurden dafür insgesamt 19008 bei Studierenden auftretende Probleme in zwei Programmierkursen bei ca. 470 Studentinnen und Studenten erfasst und in 27 (2003) bzw. 36 (2004) Kategorien eingeordnet (vgl. Garner u. a., 2005 und Robins u. a., 2006). Die am häufigsten auftretenden Fehler (3070) waren syntaktischer Art.[51]

**Unmotivierende Probleme/Aufgaben** Bei klassischen Einführungen in die Programmierung wurden häufig mathematische Probleme, wie z. B. die Berechnung der durchschnittlichen Tagestemperatur, die Berechnung von Fibonacci-Zahlen oder die Summe von Zahlen bearbeitet (vgl. z. B. Rust, 1985, S. 367 oder Feldgen und Clua, 2003, S. T3C-T24). Heute kennen Schülerinnen und Schüler grafische Oberflächen, Animationen, Spiele, Web-Applikationen usw. bereits durch ihren täglichen Umgang mit dem Computer (vgl. Landerer, 2006, S. 100 ff.). Wenn in einem Einführungskurs über einen längeren Zeitraum Programme entwickelt werden, die über mathematische Berechnungen und Manipulationen von Zeichenketten nicht hinausgehen, so wirkt dies nicht motivierend (vgl. Brusilovsky u. a., 1997, S. 65 ff. oder Feldgen und Clua, 2004, S. SH1-11). Auch sind mathematische Probleme wie die oben genannten für Schülerinnen und Schüler der Sekundarstufe I aufgrund fehlender Vorkenntnisse nicht immer geeignet. In einer empirischen Studie von Bergin und Reilly (2005) wurde festgestellt, dass intrinsische Motivation und Ergebnis in einem Einführungskurs zum Programmieren stark korrelierten.[52]

*learner and the system are compound by the students' attempt to deal with all these different kinds of difficulty at once.*

[51] Die Kategorie, in die Syntaxfehler eingeordnet wurden, hieß „Basic mechanics". Robins u. a. (2006, S. 172) geben dafür folgende Definition an:
*Trivial problems with little mechanical details (where these are not better described by some other problem). Braces, brackets, semi–colons. Typos and spelling. Java and file naming conventions. Import statements (when forgotten, when misunderstood see Problem S12). Formatting output. Tidiness, indenting, comments.*
*This category only covers trivial problems (e.g. accidentally mismatched {}). When the underlying issue is actually a conceptual one (e.g. they don't understand the structure that the {} describe) use the best matching specific problem.*

[52] Nach Hubwieser (2004, S. 15) versteht man unter Motivation *einen kurz andauernden Zustand des Angetriebenseins.* Intrinsische Motivation *zielt auf die Beschäftigung mit der Sache selbst*,

### Gründe für den Einsatz visueller Werkzeuge zur Einführung in die Programmierung

Um den oben beschriebenen anfänglichen „Schock“ bei einem Einstieg in die Programmierung zu vermeiden, werden bei Programmieranfängern häufig didaktisch reduzierte visuelle Werkzeuge eingesetzt. So wird oft ein vereinfachtes Modell vom Computer und der Programmausführung (notional machine) zu Grunde gelegt: Bei einigen Werkzeugen mit visueller Ausgabe wie Robot Karol, Kara oder dem Java-Hamster-Modell gibt es einfache Befehle, die einen imaginären Roboter, Marienkäfer bzw. Hamster in einer Welt steuern (vgl. Abschnitt 4.2 bzw. Brusilovsky u. a., 1997 sowie Boles, 2002 und 2004 zum Java-Hamster-Modell). Die wenigen Grundbefehle sind für Programmierneulinge leicht verständlich und *aufgrund des visuellen Programmablaufs sehen die Schüler sofort, was sie programmiert haben und ob ihr Programm funktioniert* (vgl. Hartmann u. a., 2004, S. 11).
Auch die Syntax und Semantik (notion) wird für den Einstieg in die Programmierung oft didaktisch reduziert: Die oben beschriebenen Studien von Garner u. a. (2005) sowie Robins u. a. (2006) zeigen, dass ein großer Teil der bei Anfängern auftretenden Probleme beim Erlernen einer Programmiersprache syntaktischer Art sind. In den vielen aktuellen visuellen Programmiersprachen, wie z. B. Kara und Scratch sind Syntaxfehler aufgrund der Konstruktion des Systems nicht möglich (vgl. Abschnitt 4.2). Dass dies vorteilhaft für Programmierneulinge sein kann, wurde bereits in empirischen Studien belegt (vgl. Cunniff u. a., 1986; Calloni u. a., 1997 sowie Cilliers u. a., 2005).[53] Auch in visuellen Softwarebeschreibungssprachen können spezifische Aspekte von Software wie z. B. Benutzungsschnittstellen oder Klassendiagramme erstellt werden, ohne dass der Benutzer die Syntax der zugehörigen Anweisungen kennen muss (vgl. Abschnitt 4.2). Einzelne Werkzeuge vereinfachen aber auch die textuelle Syntax, die zur Erstellung von Programmen

extrinisiche Motivation *zielt auf äußere Begleitumstände wie Lob, gute Noten* (vgl. Hubwieser, 2004, S. 16).

[53] In der Studie von Cunniff u. a. (1986) wurde die textuelle Programmiersprache Pascal mit der visuellen Sprache FPL im Rahmen eines Programmierkurses an einer Universität verglichen (zu FPL vgl. Taylor u. a., 1986). Cunniff u. a. (S. 181) stellten bereits 1986 fest: *The absence of any syntax-related bugs in the FPL programs is noteable.*
Calloni u. a. (1997) beschreiben zwei Untersuchungen während Einführungskursen in die imperative und die objektorientierte Programmierung, bei denen die Studentinnen und Studenten in zwei Gruppen aufgeteilt wurden. Calloni u. a. (1997) stellen dar, dass die Gruppe, die parallel die visuelle Programmiersprache BACCII++ und die textuelle Sprache C++ nutzte, in den Abschlussprüfungen signifikant besser abschnitt als die Gruppe, die lediglich C++ nutzte.
Auch eine Studie von Cilliers u. a. (2005) ergab, dass das parallele Verwenden der Programmierumgebung Delphi mit der visuellen Programmiersprache B# vorteilhaft gegenüber dem Einsatz von Delphi als alleiniges Werkzeug war (zu B# vgl. Greyling und Brown, 2002).

mit visueller Ausgabe nötig ist: So wird beispielsweise beim Java-Hamster-Modell der Beginn der Hauptprozedur nicht – wie in Java üblich – durch *public static void main (String[] args)* sondern lediglich durch *void main()* vereinbart (vgl. Boles, 2002, S. 84). Semantische Vereinfachungen sind seltener, aber auch möglich. So gibt es z. B. in Robot Karol Schleifen, bei denen durch die Angabe einer Zahl die Anzahl der Wiederholungen eingestellt werden kann. Dadurch wird es möglich, Schleifen auch ohne Kenntnis des Variablenkonzepts zu benutzen. Das ist in vielen industriell eingesetzten Programmiersprachen nicht möglich.
Ein weiterer Bereich, bei dem didaktisch reduziert werden kann, sind die Entwicklungsumgebungen, mit denen Implementierung, Test und Fehlersuche bzw. -korrektur (pragmatics) durchgeführt werden. Nach der Auffassung von Börstler (2007, S. 15) *kommen häufig Umgebungen zum Einsatz, die für AnfängerInnen viel zu komplex und anspruchsvoll sind, wie z. B. Eclipse. Dies führt unweigerlich zu einer höheren kognitiven Belastung.* Viele der im Abschnitt 4.2 vorgestellten visuellen Werkzeuge, die speziell für die Einführung in die Programmierung konzipiert sind, versuchen mit möglichst wenigen Buttons/Elementen auszukommen.
Es kann festgehalten werden, dass auf verschiedene Weisen versucht wird, die Einstiegshürden für Programmierneulinge zu vermindern. Visualisierungen sind bei der didaktischen Reduzierung ein wichtiges Hilfsmittel.

Landerer (2006, S. 118) stellt dar, dass Schülerinnen und Schüler Programme entwickeln wollen, *die sie selbst ansprechen* und zwar so, *wie es ihrer gewohnten Arbeitsweise nahe kommt.* Das heißt, sie wollen in visuell aufbereiteten, interaktiven Umgebungen arbeiten und auch Programme erstellen, die denen, die sie im täglichen Leben nutzen, ähnlich sind und nicht Text in einen Editor eingeben, aus dem nach mehreren Übersetzungsversuchen und Fehlermeldungen ein textuelles Programm entsteht, das Zahlen und Zeichenketten manipuliert (vgl. hierzu Landerer, 2006, S. 116 ff.).
Einige Entwickler von visuellen Werkzeugen und zugehörigen Aufgaben versuchen diesen Ansprüchen der Schülerinnen und Schüler zu entsprechen und somit eine Motivation für das Erlernen der Programmierung zu erreichen indem sie Werkzeuge bereitstellen, die es Anfängern ermöglichen, relativ schnell und einfach visuelle Ausgaben zu erzeugen (z. B. Turtle-Grafiken). Andere Werkzeuge stellen einfache, abgeschlossene, visuell dargestellte Welten zur Verfügung, in denen Probleme gelöst werden sollen (z. B. Java-Hamster-Modell, Kara, Robot Karel), wieder andere ermöglichen es auch Programmierneulingen, beim Einstieg in die Programmierung bereits Animationen und Spiele zu erstellen (z. B. Greenfoot, Scratch).

Es wurde dargestellt, dass mit visuellen Werkzeugen versucht wird, die Einführung in die Programmierung zu erleichtern, und zwar durch didaktisch reduzierte Entwicklungsumgebungen, durch das Vermeiden von Syntaxfehlern und durch das

Bereitstellen motivierender Aufgaben. Diese drei Ansätze wurden auch in Puck aufgegriffen. Durch die Einschränkung auf 16 Bausteine und zwei Datentypen und die einfach gehaltene Entwicklungsumgebung soll einer Überforderung des Benutzers entgegengewirkt werden. Durch die Programmierung mit Bausteinen, die nur in syntaktisch korrekter Weise kombiniert werden können, werden Syntaxfehler verhindert.[54] Durch die Möglichkeit, neben textuellen auch visuelle und akustische Ausgaben zu erstellen, können neben mathematischen auch viele andere motivierende Probleme und Aufgaben bearbeitet werden (für entsprechende Aufgaben siehe Kohl, 2008a).

## 4.4 Einordnung der Entwicklungsarbeiten an Puck

Nachfolgend wird dargestellt inwieweit Empfehlungen zur Entwicklung interaktiver Lernumgebungen von Arnold und Hartmann (2007b) bei der Entwicklung von Puck nachvollzogen werden können.[55] Die oben genannten Empfehlungen wurden am Beispiel des Systems InfoTraffic veranschaulicht; eine empirische Erprobung dieses Systems erfolgte nicht (vgl. Arnold, 2007, S. 74 ff.). Somit ist derzeit auch keine Aussage darüber möglich, ob das Umsetzen dieser Empfehlungen zu einer im Unterricht erfolgreich eingesetzten interaktiven Lernumgebung führt. Im Folgenden soll für jede der zehn Empfehlungen reflektiert werden, inwieweit diese auch bei der Entwicklung des Puck-Systems von Bedeutung waren. In den Kapiteln 7 und 8 wird dann dargestellt, inwieweit sich Puck im Unterricht bewährt hat.

**1. Braucht es dazu überhaupt den Computer?** Für Arnold und Hartmann (2007b, S. 178) macht es keinen Sinn, *Lernumgebungen zu entwickeln für Themen, die man genau so gut oder besser ohne Computer vermitteln kann. Speziell interessant sind deshalb gerade abstrakte und schwierige Themen, bei denen im Unterricht durch Individualisierung den unterschiedlichen Kenntnissen und Lerntempi Rechnung getragen werden kann.*

Programme, als Ergebnisse der Programmierung werden von Computern ausgeführt. Deshalb ist eine Einführung in die Programmierung mit Computern durchaus sinnvoll. Programmieren kann als abstraktes und schwieriges Thema bezeichnet werden (vgl. Abschnitt 4.3). Auch unterschiedliche Vorkenntnisse und

[54] Für eine Klassifizierung von Fehlern, die beim Erstellen von Programmen auftreten sowie eine Analyse, welche dieser Fehler mit Puck verhindert werden können sei auf die Diplomarbeit des Autors verwiesen (vgl. Kohl, 2004a, S. 81 ff.)

[55] Puck ist eine Lernumgebung zum Thema Programmierung, die Interaktivität ermöglicht und somit auch als interaktive Lernumgebung charakterisiert werden kann.

Lerntempi sind im Programmierunterricht nicht ungewöhnlich (vgl. z. B. Baldwin und Kujis, 2004, S. 285 sowie Perkins u. a. 1989, S. 261). Deshalb macht es Sinn, für eine Einführung in die Programmierung den Computer und speziell eine visuelle Programmiersprache wie Puck einzusetzen.

**2. Ist das Unterrichtsthema auch in 10 Jahren noch relevant?** Arnold und Hartmann (2007b, S. 178) stellen dar, dass der Aufwand für das Entwickeln interaktiver Lernsoftware im Allgemeinen so groß ist, dass er sich nur für längerfristig relevante Themen rechtfertigt.

Die Einführung in die „Algorithmik“ wird bereits seit Mitte der 1970er Jahre im Informatikunterricht vermittelt und ist auch heute noch aktuell (vgl. z. B. Baumann, 1996, S. 113, Hubwieser, 2004, S. 51 und Thüringer Kultusministerium, 1999). Somit ist dieser Bereich in dem sich ständig wandelnden Fach Informatik von dauerhafter Bedeutung. Aufgrund der Einordnung von „Algorithmisierung“ als fundamentale Idee der Informatik kann davon ausgegangen werden, dass dieser Inhalt auch in 10 Jahren noch relevant sein wird (vgl. hierzu auch Abschnitt 5.1 bzw. Schubert und Schwill, 2004, S. 89 f.).

**3. Use-Cases: Ist Interaktivität möglich?** Da sich nicht jedes Thema für eine computergestützte Mensch-Maschinen-Interaktion eignet, schlagen Arnold und Hartmann (2007b, S. 178 f.) vor, *bereits in der Spezifikationsphase des Projektes Musteraufgaben zusammenzutragen, welche mithilfe der geplanten Lernumgebung bearbeitet werden sollen. [...] Aufgrund der Musteraufgaben kann abgeschätzt werden, ob die Lernumgebung zu einer wirklichen Interaktion mit den Lernenden führt und ob verschiedene kognitive Stufen angesprochen werden.*

Da Algorithmen zum größten Teil bereits computergestützt unterrichtet wurden, stellte sich beim Entwickeln des Puck-Systems die Frage nach der Eignung dieses Themas für die Mensch-Maschinen-Interaktion nicht. Beispiele für Aufgaben, die zur Einführung in die Programmierung genutzt werden können, standen schon vor der Entwicklung von Puck in großer Anzahl zur Verfügung (vgl. z. B. Fothe, 1989). Insbesondere bei der Überlegung, die Datentypen auf Integer und Boolean zu beschränken, wurde geprüft, ob dies die Anzahl der zu bearbeitenden Aufgaben nicht zu sehr einschränken würde. Die bereits erprobten Materialien zum Programmieren mit Python von Fothe (2002, Kapitel Python I) sowie selbstständig zusammengetragene Aufgaben führten zur genannten Beschränkung der Datentypen, die zur Einfachheit des Systems beiträgt. Durch die im Rahmen dieser Arbeit entwickelten Bausteine Color, Rectangle/Ellipse, Sleep, Sound und Random wurde es ermöglicht, eine Vielzahl von weiteren Aufgaben umzusetzen

(für Aufgabenbeispiele vgl. Kohl, 2008a). Die Abbildungen 17 und 18 zeigen, dass bereits während der Entwicklung von Puck mit konkreten Aufgaben gearbeitet wurde.

**4. Paper Based Prototyping** Arnold und Hartmann (2007b, S. 179) argumentieren, dass es sich bei Lernumgebungen, bei denen der Benutzeroberfläche eine besonders hohe Bedeutung zukommt, lohnt, so lange wie möglich nur auf Papier zu arbeiten und Klebeband, Post-Its, Schere sowie Flipcharts zu nutzen. Für das schnelle Erstellen und Variieren von Programmoberflächen empfehlen sie Präsentationswerkzeuge.

Auch bei der Entwicklung des Puck-Systems wurden viele Entwürfe – insbesondere für die Darstellung der Bausteine und deren Anordnung im Arbeitsbereich – auf dem Papier entwickelt. Auch wenn einzelne Design-Entscheidungen nach der Entwicklung des ersten Prototyps verbessert wurden, so kann doch festgestellt werden, dass der Einsatz von Paper-Based-Prototyping zu einer erheblichen Arbeitserleichterung führte. Als einfaches Werkzeug wurden neben den oben genannten auch Folienausdrucke zur Präsentation und Diskussion in einem Seminar von Lehramtsstudenten eingesetzt. Durch das Verwenden von Folienausdrucken, Schere und Projektor konnte frühzeitig die Drag and Drop-Funktionalität diskutiert werden. Abbildung 17 zeigt Entwürfe für erste Puck-Bausteine, die mit einem Zeichenprogramm erstellt, auf Folie ausgedruckt, ausgeschnitten und zu einem Programm, das den größten gemeinsamen Teiler berechnet, zusammengefügt wurden.

**5. Rapid Prototyping** Arnold und Hartmann (2007b, S. 180) empfehlen, einen ersten Prototyp direkt nach dem Festlegen der Benutzungsschnittstelle und der Use-Cases zu implementieren, diesen möglichst schnell einzelnen ausgewählten Testpersonen zu zeigen sowie ihn anschließend experimentell zu erproben.

Der in Abbildung 18 dargestellte erste Prototyp des Puck-Systems wurde einer Arbeitsgruppe von Thüringer Informatiklehrern bereits im Jahr 2004, also während der Entwicklung, präsentiert. Nach dem Fertigstellen der ersten Version von Puck erfolgten Tests an zwei Jenaer Schulen im Frühjahr 2005. Dabei konnten viele Anregungen für Verbesserungen und Weiterentwicklungen gesammelt werden. Ein Beispiel sind die Bausteine Color und Rect/Ellipse, die aufgrund der Anforderungen der Lehrpersonen entwickelt wurden.

**6. Technische Anforderungen: so einfach wie nur möglich** Arnold und Hartmann (2007b, S. 180) fordern Einfachheit und den Verzicht auf unnötige Funktio-

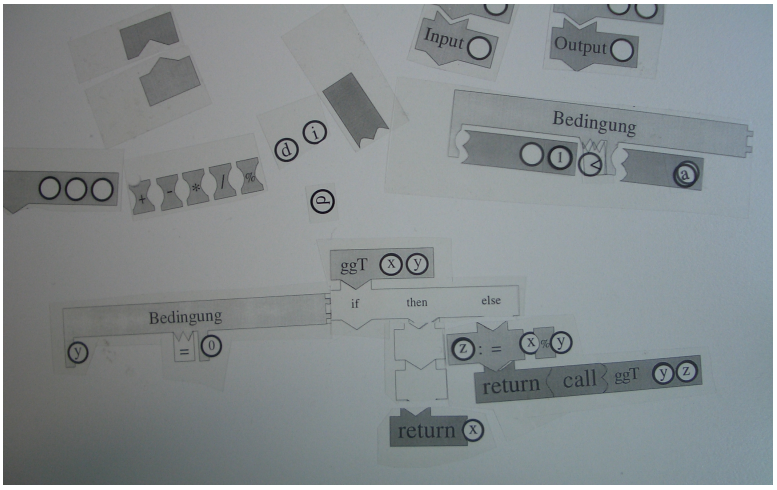



Abbildung 17: Entwürfe für erste Puck-Bausteine auf Folie gedruckt, ausgeschnitten und zu einem Programm zusammengefügt, das den größten gemeinsamen Teiler berechnet

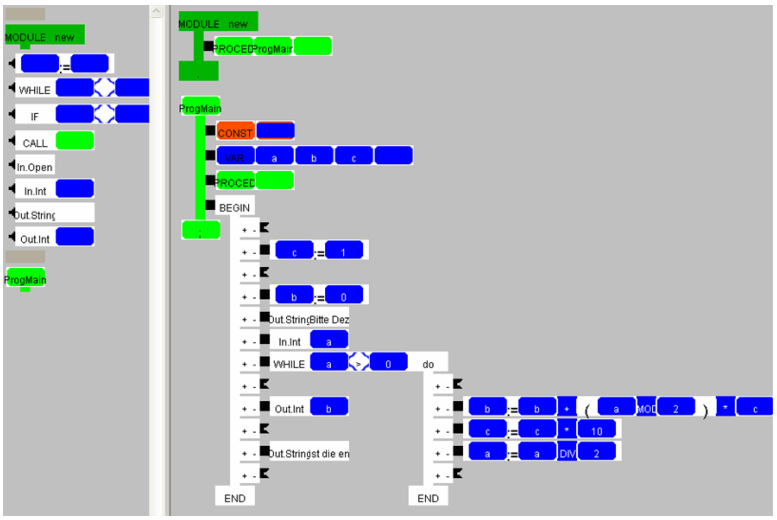



Abbildung 18: Musteraufgabe für ein Programm zum Umrechnen von Dezimal- in Dualzahlen bei einem frühen Prototypen des Puck-Systems

nen, außerdem sollten Lernumgebungen auf verschiedenen Plattformen einsetzbar sein, um den großen Entwicklungsaufwand zu rechtfertigen.

Bei Weiterentwicklungen im Rahmen dieser Arbeit wurde stets versucht, das System so einfach wie möglich zu lassen oder es sogar noch zu vereinfachen. Experimente mit den Datentypen Double und Character wurden aufgrund der Erhöhung der Komplexität der visuellen Darstellung abgebrochen. Das Puck-System wurde in Java implementiert und benötigt lediglich den Java Development Kit (JDK), der für verschiedene Plattformen zur Verfügung steht.[56] Auf spezielle Effekte, die moderne Grafikkarten oder schnelle Prozessoren erfordern, wurde bei der Entwicklung von Puck verzichtet.

**7. Frühzeitige Erprobung** Arnold und Hartmann (2007b, S. 180) empfehlen, erste Tests mit ganzen Schulklassen so früh wie möglich durchzuführen, denn *diese Tests zwingen dazu, konkrete Aufgabenstellungen und begleitendes Unterrichtsmaterial auszuarbeiten, und decken in der Regel eine ganze Reihe von Schwachstellen auf. Schüler denken und handeln oft anders, als Lehrpersonen sich das vorstellen!*

Wie oben bereits dargestellt, wurden schon wenige Monate nach der Entwicklung der ersten Puck-Version erste Tests mit konkreten Aufgabenstellungen in zwei Schulklassen durchgeführt. Die Erfahrungen dieser und verschiedener weiterer Tests zeigten, dass der Empfehlung von Arnold und Hartmann (2007b, S. 180) vollkommen zugestimmt werden kann: Durch den Einsatz im Unterricht konnten Schwachstellen aufgedeckt und beseitigt werden. Bestehende Aufgabenstellungen zur Einführung in die Programmierung konnten bei der Arbeit mit Puck teilweise übernommen werden. Weiterhin wurden – auch in Vorbereitung auf die Untersuchungen im Rahmen dieser Arbeit – frühzeitig Aufgabenstellungen und begleitendes Unterrichtsmaterial entwickelt (vgl. Kohl, 2006a; Kohl u. a., 2007; Kohl, 2008a).

**8. Sparsamkeit bei der Benutzungsschnittstelle** *Die Benutzeroberfläche einer Lernumgebung soll im Idealfall selbsterklärend sein*, deshalb fordern Arnold und Hartmann (2007b, S. 181) diese nach den ersten Tests einem Reviewprozess zu unterwerfen, bei dem für jeden Button, jede Beschriftung usw. kritisch hinterfragt wird, *ob dieses Element wirklich benötigt wird und ob es an der richtigen Stelle platziert ist.*

[56] Aufgrund der großen Verbreitung des Betriebssystems Windows an deutschsprachigen Schulen wurde Puck bisher lediglich auf verschiedenen Windows-Versionen eingesetzt. Aufgrund der Angaben der Lehrpersonen in der im Kapitel 8 dargestellten Hauptuntersuchung liegt es nahe, Puck auch für Linux-Systeme nutzbar zu machen.

Nach den ersten Tests des Puck-Systems fand der von Arnold und Hartmann (2007b, S. 181) geforderte Review- und Verbesserungsprozess statt. Dabei wurden z. B. das Vereinbaren von globalen Variablen sowie die Möglichkeit, innerhalb einer Prozedur Unterprozeduren zu deklarieren, entfernt, denn davon wurde im Unterricht kein Gebrauch gemacht. In der ersten Puck-Version war das Ausführen der Programme noch recht umständlich. Zu einem visuell erstellten Programm musste das zugehörige textuelle Programm gespeichert, mit einem externen Editor geöffnet und dort kompiliert sowie ausgeführt werden. Mit der Zielstellung, im Einführungsunterricht ausschließlich Puck zu verwenden, wurde das System so weiterentwickelt, dass es nun möglich ist, die visuell entwickelten Programme direkt auszuführen. Durch diese Auflösung der Abhängigkeit von einer textuellen Programmiersprache wurden die Benutzung und die Installation des Systems vereinfacht. Weiterhin konnte der Menüeintrag für das Speichern des textuellen Programms somit aus dem Standardsystem entfernt werden.

**9. Verbreitung der Lernumgebung** Arnold und Hartmann (2007b, S. 181) fordern, um überhaupt eine Wirkung der Lernumgebung im Unterricht zu erreichen, diese Schülerinnen und Schülern, Lehrkräften und anderen Benutzern zur Verfügung zu stellen. Dabei sollten nach Meinung von Arnold und Hartmann verschiedene Verteilerkanäle bedient und didaktische Begleitmaterialien bereitgestellt werden.

Zur Verbreitung des Puck-Systems wurden folgende Verteilerkanäle genutzt:

- Puck wurde Lehrkräften in regionalen Fortbildungen präsentiert.
- Puck wurde in Fachzeitschriften der Didaktik der Informatik vorgestellt (vgl. Kohl, 2006b; Kohl u. a., 2007).
- Puck wurde auf Fachtagungen der Didaktik der Informatik und auf Messen präsentiert (vgl. Kohl, 2005, 2007a).
- Für das Puck-System wurde eine umfangreiche Webseite entwickelt: `www.ipuck.de` [Zugriffsdatum: 14.10.2008] .
- Über die Hauptuntersuchung dieser Arbeit wurden mehr als 500 Schulen in Deutschland, Österreich und der Schweiz per E-Mail informiert. In dieser E-Mail wurde auch auf das Puck-System hingewiesen (vgl. Abschnitt 8.1).

Um die Verbreitung des Puck-Systems zu ermöglichen, wurden folgende didaktische Begleitmaterialien entwickelt und im Internet sowie teilweise in gedruckter Form zur Verfügung gestellt:

- eine Dokumentation bzw. Hilfe-Seiten zum Puck-System

- eine Einführung in Puck mit fünf Beispielen (vgl. Kohl, 2006a)
- eine Unterrichtseinheit zur Einführung in die Programmierung unter Verwendung von Puck (vgl. Kohl u. a., 2007)
- Arbeitsmaterial zur Untersuchung „Kompetenzorientierter Informatikunterricht mit der visuellen Programmiersprache Puck“ inkl. einer umfangreichen Aufgabensammlung (vgl. Kohl, 2008a)

**10. Sicherstellung von Unterhalt und Kontinuität** Um den Unterhalt einer Lernumgebung zu garantieren, muss nach Arnold und Hartmann (2007b, S. 181) sichergestellt werden, dass nach der Veröffentlichung etwaige Fehler behoben, evtl. Erweiterungen entwickelt und Anpassungen an neue Software-Versionen bereitgestellt werden. Arnold und Hartmann (2007b, S. 181) argumentieren, dass Lehrkräfte für die Einarbeitung in eine neue Lernumgebung sowie für die Aufbereitung des Unterrichtsstoffes Zeit nur dann investieren, wenn abzusehen ist, dass das eingesetzte System über längere Zeit nutzbar bleibt, was insbesondere bei Systemen, die im Hochschulumfeld entstehen, nicht immer sichergestellt ist. Bei der Umsetzung der an die Entwickler herangetragenen Wünsche nach Erweiterungen einer Software empfehlen Arnold und Hartmann (2007b, S. 181) vorsichtig zu sein, denn *häufige Überarbeitungen und Erweiterungen verunsichern die Benutzer und führen dazu, dass bereitgestellte Unterrichtsmaterialien aktualisiert werden müssen.* Außerdem misst sich die Qualität von Lernumgebungen nicht an der Menge von Features, und die Erweiterung des Programmcodes erschwert die Wartung (vgl. Arnold und Hartmann, 2007b, S. 181). Deshalb empfehlen Arnold und Hartmann (2007b, S. 181) das Motto „Weniger ist mehr“.

Das Puck-System wurde nach der ersten Veröffentlichung im Jahr 2004 bis zur Fertigstellung dieser Arbeit im Jahr 2008 an der Friedrich-Schiller-Universität Jena betreut. Fehler wurden behoben, Erweiterungen entwickelt und Anpassungen an neue Softwareversionen, wie der Übergang von Java 1.4 auf Java 5.0, wurden vorgenommen. Seit der Präsentation von Version 2.4 im Sommer 2006 wurden – trotz intensiven Einsatzes im Unterricht – keine größeren Probleme bei der Arbeit mit dem Puck-System festgestellt. Bei der Entwicklung wurde stets nach dem oben genannten Motto „Weniger ist mehr“ gehandelt und so wurde Puck nicht voreilig um weitere Datentypen oder Bausteine erweitert. Inwieweit zusätzlich entwickelte Features von Lehrkräften genutzt wurden, wird im Kapitel 7 dargestellt. Puck wurde von Anfang an unter der GNU General Public License der Free Software Foundation (2007) entwickelt. Damit wurden die Voraussetzungen dafür geschaffen, dass das System in der Zukunft durch eine Gruppe von interessierten Nutzern weiter betreut wird.

**Ergebnis der Analyse** Es konnte festgestellt werden, dass alle zehn von Arnold und Hartmann (2007b, S. 171 ff.) vorgestellten Empfehlungen zur Entwicklung von interaktiven Lernumgebungen auch bei der Entwicklung der visuellen Programmiersprache Puck nachvollzogen werden konnten. Somit konnte die Nützlichkeit der Entwicklungsempfehlungen neben dem von Arnold (2007) vorgestellten System Info-Traffic an einem weiteren Beispiel demonstriert werden. In den Kapiteln 7 und 8 wird dargestellt, wie überprüft wurde, inwieweit sich Puck für den Einsatz im Unterricht eignet. Kann gezeigt werden, dass das Umsetzen der Empfehlungen von Arnold und Hartmann (2007b) beim Entwickeln des Puck-Systems zu einer im Unterricht erfolgreich eingesetzten interaktiven Lernumgebung (bzw. visuellen Programmiersprache) führte, so ist dies ein Beleg für die Sinnhaftigkeit dieser Empfehlungen, der dazu führen kann, dass neue interaktive Lernumgebungen zielgerichteter entwickelt werden können.

## 4.5 Zusammenfassung

In diesem Kapitel wurde die visuelle Programmiersprache Puck vorgestellt. Es wurde gezeigt, dass in Puck Aspekte der virtuell-enaktiven, der ikonischen und der symbolische Repräsentationsform kombiniert werden. Anschließend wurden verschiedene visuelle Werkzeuge vorgestellt, die im Informatikunterricht eingesetzt werden. Dabei wurden vier Varianten von visuellen Werkzeugen zum Thema „Algorithmen“ unterschieden, nämlich Werkzeuge zur Softwarevisualisierung, Programmierwerkzeuge mit visueller Ausgabe, visuelle Softwarebeschreibungssprachen und visuelle Programmiersprachen. Beispiele für die verschiedenen Varianten visueller Werkzeuge wurden angegeben und von Puck abgegrenzt.
Probleme, die bei der Einführung in die Programmierung entstehen, wurden analysiert. Es wurde festgestellt, dass gerade die Kombination verschiedener von Programmieranfängern zu erlernender Fertigkeiten problematisch ist. Dabei gestaltet sich unter anderem das richtige Verwenden der Syntax von textuellen Programmiersprachen als schwierig. Des Weiteren trug in der Vergangenheit das Bearbeiten mathematischer Probleme nicht immer zur Motivation der Schülerinnen und Schüler bei. Es wurde gezeigt, dass sich einige dieser bei der Einführung in die Programmierung auftretende Probleme durch visuelle Werkzeuge und speziell durch Puck vermindern lassen.
Eine Reflexion über die Entwicklungsarbeiten ergab, dass im Entstehungsprozess von Puck die von Arnold und Hartmann (2007b, S. 171 ff.) vorgestellten Empfehlungen zur Entwicklung interaktiver Lernumgebungen umgesetzt wurden.

# 5 Die Entwicklung des Kompetenzmodells „Algorithmen“

> *Kompetenzmodelle und -erwartungen sind unverzichtbar, um zwischen relativ abstrakten, verallgemeinerten Bildungszielen einerseits, konkreten Aufgabensammlungen andererseits zu vermitteln. Vor allem die Kompetenzstufen sind ein zentrales Hilfsmittel für die Konstruktion von Aufgaben, sei es zu Zwecken der Lernplanung oder für die Testentwicklung.* (Klieme u. a., 2003, S. 24)

In diesem Kapitel wird dargestellt, wie das Kompetenzmodell „Algorithmen“ für die Jahrgangsstufe 8 bis 10 entwickelt wurde. Damit wird ein Beitrag zur Beantwortung der ersten Forschungsfrage geleistet (vgl. Kapitel 1). Im Abschnitt 5.1 wird die Wahl des Themas begründet und ein Einsatzszenario für das zu erstellende Kompetenzmodell wird vorgestellt. Im Abschnitt 5.2 werden auf Grundlage der GI-Empfehlungen „Grundsätze und Standards für Informatik in der Schule“ konkrete Kompetenzen zu vier Komponenten genannt. Im Abschnitt 5.3 werden drei Kompetenzstufen festgelegt um so eine differenzierte Leistungsbeurteilung sowie die individuelle Förderung von Schülerinnen und Schülern besser zu ermöglichen. Die Komplexität der Stufen des Kompetenzmodells wird im Abschnitt 5.4 mit Hilfe der SOLO-Taxonomie analysiert. Die Beziehungen des Kompetenzmodells zu den Inhalts- und Prozessbereichen der GI-Empfehlungen werden im Abschnitt 5.5 aufgezeigt. Im Abschnitt 5.6 werden in einem Reflexionsprozess Anforderungen für die Entwicklung eines Kompetenzmodells für einen Bereich der Informatik zusammengefasst.

## 5.1 Wahl des Themas und Einsatzszenario

Der ersten Forschungsfrage entsprechend soll sich bei der Entwicklung des Kompetenzmodells in dieser Arbeit auf einen Bereich des Informatikunterrichts konzentriert werden. Warum gerade der Inhaltsbereich „Algorithmen“ gewählt wurde, soll nachfolgend anhand von verschiedenen Argumenten verdeutlicht werden:

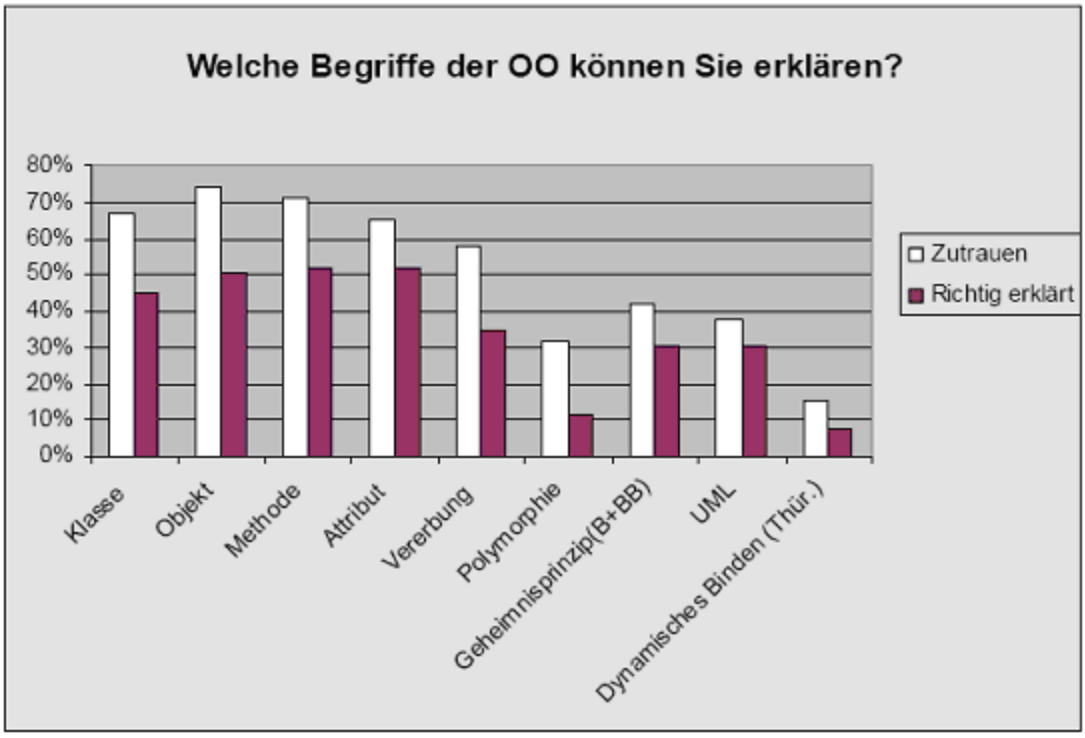
- Mit fundamentalen Ideen wird versucht, langlebige Konzepte eines Fachs zu identifizieren (zu fundamentalen Ideen der Informatik vgl. Schwill, 1993).

Schubert und Schwill (2004, S. 86 ff.) untersuchten den Softwareentwicklungsprozess auf fundamentale Ideen mit der Begründung, dass die Entwicklung von Software der Schlüssel zur besseren Nutzbarmachung des Computers sei. Sie identifizieren „Algorithmisierung“, „strukturierte Zerlegung“ und „Sprache“ als Masterideen, die in allen Stadien der Softwareentwicklung eine herausragende Rolle spielen. Diese Identifizierung von Algorithmisierung als Masteridee soll ausreichen, um zu begründen, dass das Thema „Algorithmen“ wichtig für die Informatik im Allgemeinen und für den Informatikunterricht im Besonderen ist.

- Schubert und Schwill (2004, S. 29) stellen dar, dass sich die meisten Informatiklehrkräfte in unterschiedlichen Maßnahmen fort- und weitergebildet haben und dass es kaum ausgebildete Informatiklehrerinnen und -lehrer gibt. Bei einer Untersuchung der Kompetenzen von Lehrpersonen zum Thema Objektorientierung in Berlin, Brandenburg und Thüringen wurde festgestellt, dass nur ein geringer Anteil der im Rahmen von Fortbildungen befragten Lehrpersonen tiefer liegende Konzepte der objektorientierten Programmierung erklären konnten (vgl. Kohl und Romeike, 2006). In Abbildung 19 ist für verschiedene Konzepte der Objektorientierung angegeben, wie viel Prozent der Untersuchungsteilnehmer sich zutrauten, das jeweilige Konzept zu erklären und wie viel Prozent tatsächlich eine richtige Erklärung abgaben (vgl. Kohl und Romeike, 2006).[57]
  In den GI-Empfehlungen wird dargelegt, dass *Fachkompetenz des Lehrenden [...] eine notwendige [...] Bedingung für erfolgreiches Lehren und Lernen im Informatikunterricht [ist].* Humbert (2005, S. 47) stellt dar, dass *die der Algorithmenorientierung zuzurechnenden Konzeptionen in der gymnasialen*

[57] Fast die Hälfte der Lehrerinnen und Lehrer hatte das Thema schon unterrichtet. Ca. 50 % erklärten die Begriffe Klasse, Objekt, Methode und Attribut richtig. Bei den tiefer liegenden objektorientierten Konzepten wie Vererbung, Polymorphie und Geheimnisprinzip konnten nur von wenigen Befragten Kompetenzen nachgewiesen werden.
Ähnliche Erfahrungen mit den Kompetenzen der Lehrerinnen und Lehrer zum Thema Objektorientierung in anderen Ländern finden sich bei Gieseke (2007, S. 34) und Röhner (2007, S. 37):
In einem Artikel von Gieseke (2007, S. 34) heißt es: *Aufgrund der Situation der Informatik in Niedersachsen sollte behutsam an aktuelle informatische Inhalte – wie zum Beispiel objektorientierte Modellierung – herangeführt werden.*
Röhner (2007, S. 37) stellt in einem Artikel über das Zentralabitur in Hessen fest: *Der Fortbildungsbedarf rührt insbesondere daher, dass der im Jahr 2003 eingeführte Lehrplan als neues Halbjahresthema Objektorientierte Modellierung ausweist. Mit den objektorientierten Konzepten und Inhalten müssen sich die Lehrkräfte aber erst vertraut machen, bevor sie diese erfolgreich vermitteln können.*
Diese Aussagen legen die Vermutung nahe, dass sich die Situation in den beiden Ländern ähnlich verhält wie in den Ländern der dargestellten Untersuchung.

*Oberstufe verbreitet aufgenommen wurden und Bestand haben.* Die meisten Informatiklehrerinnen und -lehrer konnten deshalb vielfältige Erfahrungen zu diesem Thema sammeln und sind nach Auffassung des Autors infolgedessen fachlich sicher. Dadurch erscheint dieses Thema für die Einführung von Kompetenzmodellen im Informatikunterricht geeigneter als z. B. das Thema Objektorientierung.




B+BB = nur in Berlin und Brandenburg erhoben
Thür. = nur in Thüringen erhoben

Abbildung 19: Ergebnisse einer Befragung von 69 Informatiklehrerinnen und -lehrern zum Thema Objektorientierung (vgl. Kohl und Romeike, 2006, S. 72)

- Alle drei im Kapitel 2 vorgestellten Ansätze zur Festlegung von Zielen des Informatikunterrichts enthalten das Thema „Algorithmen“ in irgendeiner Weise. In den GI-Empfehlungen (2008) wurden sogar Mindeststandards für den Inhaltsbereich „Algorithmen“ vorgestellt (vgl. Abschnitt 2.5). Diese Mindeststandards können bei der Entwicklung eines Kompetenzmodells als Orientierungsgrundlage genutzt werden.

- Mit der visuellen Programmiersprache Puck ist es möglich, in das Thema „Algorithmen“ einzuführen, ohne Schülerinnen und Schüler bei der Programmierung mit Syntaxfehlern zu konfrontieren (vgl. Kapitel 4). Gespräche mit Lehrkräften zeigten, dass diese Interesse an einer solchen visuellen Programmiersprache haben. Dieses Interesse kann für eine Untersuchung von kompetenzorientiertem Informatikunterricht zum Thema „Algorithmen“ genutzt werden.

Aufgrund der genannten Argumente wurde das Thema „Algorithmen“ für die Entwicklung des Kompetenzmodells gewählt. Mit Bezug auf die Darstellungen im Abschnitt 3.4 beschreibt das nachfolgende Szenario aus der Sicht einer Lehrkraft, mit welchem Ziel das Kompetenzmodell „Algorithmen“ und zugehörige Aufgaben in dieser Arbeit entwickelt wurden.

**Szenario:** Eine Lehrkraft möchte in der 8., 9. oder 10. Jahrgangsstufe ein Halbjahr lang das Thema „Algorithmen“ unterrichten. Dabei möchte sie erreichen, dass alle Schülerinnen und Schüler zumindest die in den GI-Empfehlungen geforderten Mindeststandards erlangen. Einige Schülerinnen und Schüler sollen aber auch darüber hinaus weitergehende Kompetenzen erwerben. Die Lehrkraft will den Unterricht an einem Kompetenzmodell ausrichten und eine in dieses Modell eingeordnete Sammlung von Aufgaben zur Unterrichtsvorbereitung und -durchführung nutzen. Am Ende des Halbjahres möchte die Lehrperson mit einem Test überprüfen, inwieweit die Schülerinnen und Schüler die im Modell angegebenen Kompetenzen erworben haben. Auf Grundlage der Ergebnisse des Tests will sie anschließend den Unterricht auswerten.

Diesem Szenario entsprechend sollen nachfolgend das Kompetenzmodell und zugehörige Aufgaben entwickelt werden.

## 5.2 Beschreibung von Kompetenzen zu vier Komponenten

Im Abschnitt 3.4 wurde dargelegt, dass es bei der Entwicklung eines Kompetenzmodells sinnvoll ist, sich an existierenden (Empfehlungen zu) Bildungsstandards zu orientieren. Die Analyse der drei im Kapitel 2 vorgestellten Ansätze zur Festlegung der Ziele informatischer Bildung ergab, dass sich im Rahmen dieser Arbeit die im Abschnitt 2.5 vorgestellten GI-Empfehlungen aufgrund der Ausrichtung auf die Sekundarstufe I und der outputorientierten Angabe von Kompetenzen am besten als Grundlage für die Entwicklung eines Kompetenzmodells eignen. In den GI-Empfehlungen (2008, S. 13) werden im Inhaltsbereich Algorithmen folgende

Kompetenzen gefordert:[58]

*Schülerinnen und Schüler aller Jahrgangsstufen*

*kennen Algorithmen zum Lösen von Aufgaben und Problemen aus verschiedenen Anwendungsgebieten und lesen und interpretieren gegebene Algorithmen,*

*entwerfen und realisieren Algorithmen mit den algorithmischen Grundbausteinen und stellen diese geeignet dar.*

Aus diesen Anforderungen für alle Schülerinnen und Schüler wurden für die Jahrgangsstufen 8 bis 10 die in der linken Spalte von Tabelle 2 genannten Kompetenzen abgeleitet (vgl. GI, 2008, S. 15 f.). Diese wurden von der GI (2008, S. 30 ff.) anschließend noch konkretisiert und an einzelnen Beispielen erläutert.

Im Folgenden sollen die in den GI-Empfehlungen geforderten Kompetenzen vier Komponenten zugeordnet werden. Damit wird dem im Abschnitt 3.2 beschriebenen ersten Zweck von Kompetenzmodellen entsprochen, nämlich das Beschreiben des Gefüges von Anforderungen, deren Bewältigung von Schülerinnen und Schülern erwartet wird (vgl. Klieme u. a., 2003, S. 74). Durch die Zuordnung der Kompetenzen zu vier Komponenten wird das Kompetenzmodell systematisiert, die Diskussion über dieses Modell in der Wissenschaft, in einem Lehrerkollegium oder auch im Unterricht wird vereinfacht und die systematische Entwicklung von Aufgaben, insbesondere von Kompetenztests wird vorbereitet (vgl. Kohl und Fothe, 2007, S. 20). Die Komponenten stellen dabei keine Reihenfolge oder Gewichtung dar. Tabelle 2 veranschaulicht, wie die sechs in den GI-Empfehlungen geforderten Kompetenzen in die vier Komponenten aufgeteilt wurden und zeigt jeweils konkrete Kompetenzen, die nachfolgend ausführlicher beschrieben werden.

**Komponente A – Eigenschaften von Algorithmen** In der Komponente A wird gefordert, dass Schülerinnen und Schüler die Eigenschaften von Algorithmen sowie den sich daraus ergebenden Algorithmusbegriff erklären können. Dies ist eine Voraussetzung für das Überprüfen der wesentlichen Eigenschaften von Algorithmen, das aus den Empfehlungen der GI (2008, S. 15) direkt in das entwickelte Kompetenzmodell übernommen wird. Des Weiteren sollen die Schülerinnen und Schüler noch Beispiele für Probleme nennen, die algorithmisch lösbar bzw. nicht algorithmisch lösbar sind. Durch diese Beispiele wird der Algorithmusbegriff für Schülerinnen und Schüler praktisch greifbar.

[58] Dem Autor der Arbeit lagen bereits vor der Veröffentlichung der GI-Empfehlungen Entwurfsfassungen vor.

| Kompetenzen der GI-Empfehlungen der Jahrgangsstufen 8 bis 10<br><br>*Schülerinnen und Schüler* | Komponente des Kompetenzmodells „Algorithmen“ | Kompetenzen des Kompetenzmodells „Algorithmen“ der Jahrgangsstufen 8 bis 10<br><br>Die Schülerinnen und Schüler |
|---|---|---|
| *- überprüfen die wesentlichen Eigenschaften von Algorithmen* | A<br>Eigenschaften von Algorithmen | - erklären den Algorithmusbegriff und die wesentlichen Eigenschaften von Algorithmen<br>- überprüfen die wesentlichen Eigenschaften von Algorithmen<br>- nennen Probleme, die mithilfe von Algorithmen lösbar bzw. nicht lösbar sind |
| *- stellen die algorithmischen Grundbausteine formal dar*<br>*- verwenden Variablen und Wertzuweisungen* | B<br>algorithmische Grund-bausteine und Datentypen | - erklären die algorithmischen Grundbausteine wie Variablen, Wertzuweisungen, Verzweigungen, Wiederholungen und wenden diese Erklärungen an<br>- stellen die algorithmischen Grundbausteine dar<br>- verwenden Datentypen |
| *- lesen formale Darstellungen von Algorithmen und setzen sie in Programme um*<br>*- modifizieren und ergänzen Quelltext eines Programms nach Vorgaben* | C<br>Arbeit mit Algorithmen | - analysieren gegebene Algorithmen<br>- prüfen Algorithmen<br>- setzen gegebene Algorithmen in Programme um<br>- modifizieren und ergänzen Algorithmen bzw. Programme |
| *- entwerfen, implementieren und beurteilen Algorithmen* | D<br>Programm-entwicklung | - entwerfen Programme<br>- implementieren Programme mit einem Programmiersystem<br>- testen Programme auf ihre Funktionalität |

Tabelle 2: Aufteilung der Kompetenzen aus den GI-Empfehlungen in vier Komponenten und erste Verfeinerung in einzelne Kompetenzen

**Komponente B – algorithmische Grundbausteine und Datentypen** In den GI-Empfehlungen wird der abstrakte Begriff der algorithmischen Grundbausteine nicht explizit definiert. Als Beispiele werden nur Folge, Verzweigung und Wiederholung genannt (vgl. GI, 2008, S. 30). Im Rahmen des hier vorgestellten Kompetenzmodells werden auch Variablen und Wertzuweisungen als algorithmische Grundbausteine verstanden und dementsprechend werden die zugehörigen Kompetenzen mit in der Komponente B aufgeführt.
Neben dem in den GI-Empfehlungen genannten Darstellen algorithmischer Grundbausteine werden im Kompetenzmodell „Algorithmen“ auch das Erklären und Anwenden derselben gefordert. Außerdem impliziert die in den GI-Empfehlungen genannte Forderung, Variablen und Wertzuweisungen zu verwenden, dass zumindest ein Datentyp bekannt ist. Dies wird in einer eigenen Kompetenz formuliert. Die beschriebenen Kompetenzen werden in den Komponenten C und D benötigt, wenn zum Beispiel bei der Modifikation eines Algorithmus algorithmische Grundbausteine angewendet oder beim Entwurf eines Programms dargestellt werden müssen. Trotzdem sollen die Kompetenzen der Komponente B in eigenen Aufgaben überprüft werden. Nur so kann festgestellt werden, ob grundlegende Konzepte verstanden wurden, auch wenn Anwendungsaufgaben der Komponenten C und D nicht gelöst werden konnten.

**Komponente C – Arbeit mit Algorithmen** Bei der Konstruktion von Aufgaben zeigte sich, dass ein Algorithmus gelesen bzw. analysiert werden muss, bevor er modifiziert werden kann. Es ist also sinnvoll, diese beiden Kompetenzen in der Komponente „Arbeit mit Algorithmen“ zu vereinen, da ansonsten in Testsituationen je ein Algorithmus zum Analysieren und ein weiterer zum Modifizieren vorgegeben werden müsste. Dies würde die Überprüfung der Kompetenzen durch die Verlängerung des Aufgabentextes und der Testdauer unnötig erschweren.[59] In den GI-Empfehlungen wird das Lesen und Umsetzen von Algorithmen gefordert. Dies wird im Kompetenzmodell jeweils in einer eigenen Kompetenz zum Ausdruck gebracht. Das weiterhin geforderte Modifizieren und Ergänzen des Quelltextes von Programmen nach Vorgaben, wird mit dem Modifizieren und Ergänzen von Algorithmen zusammengefasst. In welcher Form die Programme bzw. Algorithmen dabei gegeben sein sollen, wird nicht festgelegt. Somit ist die Kompetenz weiter gefasst und kann z. B. auch das Modifizieren eines als Struktogramm gegebenen Algorithmus beinhalten. Außerdem wird in den Erläuterungen der GI (2008, S. 33) gefordert, *dass Schülerinnen und Schüler Werkzeuge (beispielsweise Durchlauftabellen) zur schrittweisen Prüfung von Algorithmen sicher anwenden können.*

[59] In der Voruntersuchung wurde noch zwischen den Komponenten Analysieren und Modifizieren von Algorithmen unterschieden (vgl. Abschnitt 7.1).

Auch das Prüfen von Algorithmen wird als eigene Kompetenz in das Modell aufgenommen.

**Komponente D – Programmentwicklung** In den GI-Empfehlungen werden der Entwurf, das Implementieren und das Beurteilen von Algorithmen gefordert. Weitergehend wird von der GI (2008, S. 33) erläutert: *Im Kern sollte erreicht werden, dass Schülerinnen und Schüler sowohl diesen Gesamtprozess des Problemlösens von der Problemstellung über ein Modell, einen Algorithmus, das Programm bis zur Interpretation der Ergebnisse an Beispielen kennen lernen als auch in der Lage sind, einzelne Teilschritte auch einmal losgelöst von einander zu bearbeiten.* Hieraus werden für das Kompetenzmodell die Kompetenzen Entwerfen, Implementieren und Testen von Programmen abgeleitet. Der Begriff Programm verdeutlicht, dass die Schülerinnen und Schüler an dieser Stelle praktisch am Computer mit Algorithmen arbeiten. Nach welchen Kriterien Algorithmen bzw. Programme beurteilt werden sollen, wird in den GI-Empfehlungen nicht festgelegt. Es wird lediglich gefordert, dass *auch Fähigkeiten zur Interpretation der Ergebnisse schrittweise entwickelt werden* (vgl. GI, 2008, S. 33). Für dieses Kompetenzmodell wird das Testen eines Programms auf Funktionalität als ein erstes Kriterium der Beurteilung genannt.

Diese grundlegende Beschreibung der Komponenten wird nachfolgend durch die Angabe von nach Stufen differenzierten Kompetenzen konkretisiert.

## 5.3 Festlegen von Kompetenzstufen

Mit dem Festlegen von Kompetenzstufen wird dem im Abschnitt 3.2 beschriebenen zweiten Zweck von Kompetenzmodellen entsprochen: Das Liefern einer wissenschaftlich begründeten Vorstellung darüber, welche Abstufungen eine Kompetenz annehmen kann bzw. welche Grade oder Niveaustufen sich bei den einzelnen Schülerinnen und Schülern feststellen lassen (vgl. Klieme u. a., 2003, S. 74). In der Fachdidaktik Informatik kann nicht auf umfassende Ergebnisse empirischer Studien bzw. auf die Daten von internationalen Vergleichsstudien zurückgegriffen werden, wie dies z. B. bei der Festlegung der PISA-Kompetenzstufen oder den Referenzniveaus des gemeinsamen europäischen Referenzrahmens für Sprachen der Fall war (vgl. Abschnitt 3.2 bzw. OECD, 2004 sowie Trim u. a., 2001). Deshalb entspricht die Konstruktion des gestuften Kompetenzmodells in dieser Arbeit eher einer normativen Setzung, die sich in der Praxis erst als sinnvoll erweisen muss.

Eine Übersicht über das gesamte Kompetenzmodell gibt Tabelle 3.

| Stufen / Komponenten | **Stufe I**<br>***Die Schülerinnen und Schüler haben grundlegende Kompetenzen zu Algorithmen.***<br>***Die Schülerinnen und Schüler …*** | **Stufe II**<br>***Die Schülerinnen und Schüler haben vertiefte Kompetenzen zu Algorithmen.***<br>***Die Schülerinnen und Schüler …*** | **Stufe III**<br>***Die Schülerinnen und Schüler haben umfassendere Kompetenzen zu Algorithmen.***<br>***Die Schülerinnen und Schüler …*** |
|---|---|---|---|
| **A**<br>**Eigenschaften von Algorithmen** | ▷erklären den Algorithmusbegriff und die wesentlichen Eigenschaften von Algorithmen<br>▷überprüfen die wesentlichen Eigenschaften von Algorithmen in einfachen Fällen<br>▷nennen Probleme, die mithilfe von Algorithmen lösbar bzw. nicht lösbar sind | ▷erklären den Algorithmusbegriff und die wesentlichen Eigenschaften von Algorithmen an bekannten Beispielen<br>▷begründen anhand dieser Eigenschaften, ob gegebene Handlungsabläufe Algorithmen sind<br>▷nennen Probleme, die mithilfe von Algorithmen lösbar bzw. nicht lösbar sind | ▷erklären den Algorithmusbegriff und die wesentlichen Eigenschaften von Algorithmen an selbst konstruierten Beispielen<br>▷begründen anhand dieser Eigenschaften, ob gegebene Handlungsabläufe Algorithmen sind<br>▷nennen Probleme, die mithilfe von Algorithmen lösbar bzw. nicht lösbar sind |
| **B**<br>**Algorithmische Grundbausteine und Datentypen** | ▷erklären die algorithmischen Grundbausteine wie Variablen, Wertzuweisungen, Verzweigungen und Wiederholungen und wenden diese Erklärungen an<br>▷stellen die algorithmischen Grundbausteine als Pseudocode dar<br>▷verwenden einen numerischen Datentyp | ▷erklären die algorithmischen Grundbausteine wie Variablen, Wertzuweisungen, Verzweigungen und Wiederholungen und wenden diese Erklärungen an<br>▷stellen die algorithmischen Grundbausteine in verschiedenen Darstellungsformen dar<br>▷verwenden verschiedene Datentypen | ▷erklären die algorithmischen Grundbausteine wie Variablen, Wertzuweisungen, Verzweigungen, Wiederholungen und Unterprogramme mit Parametern und wenden diese Erklärungen an<br>▷stellen die algorithmischen Grundbausteine in verschiedenen Darstellungsformen dar und wechseln zwischen Darstellungsformen<br>▷verwenden verschiedene Datentypen |
| **C**<br>**Arbeit mit Algorithmen** | ▷lesen in Pseudocode gegebene einfache Algorithmen<br>▷prüfen schrittweise einfache Algorithmen mit gegebenen Beispielen<br>▷setzen gegebene einfache Algorithmen in Programme um<br>▷modifizieren und ergänzen einfache Algorithmen bzw. Programme nach Vorgaben | ▷analysieren die Funktionsweise und den Leistungsumfang gegebener Algorithmen<br>▷prüfen Algorithmen mit gegebenen Beispielen mithilfe von Durchlauftabellen (Schreibtischtest)<br>▷setzen gegebene Algorithmen in Programme um<br>▷modifizieren und ergänzen Algorithmen bzw. Programme nach Vorgaben | ▷analysieren die Funktionsweise und den Leistungsumfang gegebener komplexer Algorithmen<br>▷prüfen Algorithmen mithilfe von Durchlauftabellen (Schreibtischtest) und wählen dazu typische und untypische Beispiele selbst aus<br>▷setzen gegebene komplexe Algorithmen in Programme um<br>▷modifizieren und ergänzen komplexe Algorithmen bzw. Programme nach Vorgaben und nach selbst gesetzten Zielen<br>▷korrigieren gegebene fehlerhafte Algorithmen bzw. Programme |
| **D**<br>**Programmentwicklung** | ▷entwerfen einfache Programme skizzenhaft<br>▷implementieren einfache Programme mit einem Programmiersystem<br>▷testen einfache Programme anhand gegebener Eingaben auf ihre Grundfunktionalität | ▷fertigen einen schriftlichen Entwurf für Programme an<br>▷implementieren Programme mit einem Programmiersystem benutzungsfreundlich<br>▷testen Programme anhand gegebener Eingaben auf ihre Funktionalität<br>▷reflektieren über den Lösungsweg | ▷fertigen einen schriftlichen Entwurf für komplexe Programme an<br>▷implementieren komplexe Programme mit einem Programmiersystem benutzungsfreundlich<br>▷testen komplexe Programme anhand selbst gewählter Eingaben auf ihre Funktionalität<br>▷reflektieren über den Lösungsweg sowie über Vor- und Nachteile der Lösung<br>▷verbessern Programme eigenständig |

Tabelle 3: Kompetenzmodell Inhaltsbereich Algorithmen (vgl. Kohl und Fothe, 2007, S. 21)

Die Anzahl der Stufen wurde experimentell ermittelt; bei drei Stufen gelang es am besten, eine sachlich begründete und gleichzeitig lesbare Stufung der Kompetenzen vorzunehmen (vgl. Kohl und Fothe, 2007, S. 20). Es wurde stets darauf geachtet, dass die in den GI-Empfehlungen geforderten Kompetenzen (also die Mindeststandards) bereits in Stufe I des Kompetenzmodells umgesetzt wurden (vgl. Abschnitt 5.2). Die Stufen II und III stellen erweiterte Kompetenzen dar, die einen kompetenzorientierten Informatikunterricht auch über das Erfüllen der Bildungsstandards hinaus ermöglichen sollen (vgl. Kohl und Fothe, 2007, S. 20). Drei Kompetenzen werden auf diesen höheren Stufen zusätzlich zu den im Abschnitt 5.2 genannten gefordert. In der Komponente C sollen Schülerinnen und Schüler auf Stufe III als besondere Form der Modifikation fehlerhafte Algorithmen bzw. Programme korrigieren können. Außerdem wird in Komponente D auf Stufe III gefordert, dass Schülerinnen und Schüler in der Lage sind, Programme eigenständig zu verbessern. Durch diese beiden Kompetenzen wird der in den GI-Empfehlungen im Prozessbereich „Modellieren und Implementieren“ dargestellten Forderung entsprochen, dass Schülerinnen und Schüler Modellierungen und Implementierungen gegebenenfalls auch verbessern können (vgl. GI, 2008, S. 47). In der Komponente D werden auf Stufe II und III Reflexionen über den Lösungsweg bzw. über Vor- und Nachteile der Lösung gefordert. Damit werden Kompetenzen aus dem Prozessbereich „Begründen und Bewerten“ aufgegriffen. Denn die Schülerinnen und Schüler sollen so ihre Vorgehensweise bei der Modellierung und Implementierung begründen und die eigene Lösung anhand von Vor- und Nachteilen bewerten (vgl. GI, 2008, S. 48 f.).

Entsprechend der im Abschnitt 3.5 gewonnenen Erkenntnisse aus dem von Dorninger (2007) vorgestellten Kompetenzmodell zur angewandten Informatik wurde versucht die Kompetenzen werkzeugunabhängig und möglichst konkret zu formulieren. Typografische Festlegungen unterstützen die Lesbarkeit des Modells, so sind Kompetenzen der Stufe II bzw. III, die bereits auf Stufe I bzw. II genannt sind, grau gedruckt und nur Kompetenzen, die auf der jeweiligen Stufe neu hinzukommen, bzw. graduelle Unterschiede sind schwarz gedruckt (vgl. Kohl und Fothe, 2007, S. 20).

Ein wichtiger Bestandteil der Abstufung der Kompetenzen ist die Unterscheidung zwischen „einfachen Algorithmen“, „Algorithmen“ und „komplexen Algorithmen“, mit den folgenden Kriterien, die bei der Zuordnung von Algorithmen zu den drei Kompetenzstufen beachtet werden sollen (vgl. Kohl und Fothe, 2007, S. 21):

1. *Ein Algorithmus, der nur eine Verzweigung oder eine Wiederholung enthält, wird im Allgemeinen der Stufe I zugeordnet.*
2. *Ein Algorithmus, der mehrere, auch ineinander verschachtelte Verzweigungen*

*und Wiederholungen enthält, wird im Allgemeinen der Stufe II oder III zugeordnet.*

3. *Ein Algorithmus, der ein oder mehrere Unterprogramme mit Parametern enthält, wird im Allgemeinen der Stufe III zugeordnet.*

Diese Kriterien sollen nachfolgend kurz begründet werden:

**zu 1.** Auf Stufe I sollen keine komplexen algorithmischen Abläufe modelliert werden. Die erlernten algorithmischen Grundbausteine sollen lediglich in Programmen angewendet werden. Es sollen also „grundlegende“ Kompetenzen erworben werden. Hierfür reicht es aus, vereinzelt Wiederholungen und Verzweigungen anzuwenden. Deshalb sollen die im Kompetenzmodell als „einfach“ bezeichneten Algorithmen der oben genannten Forderung entsprechen.

**zu 2.** Das Entwerfen, Analysieren, Implementieren, Modifizieren und Prüfen von Algorithmen mit mehreren ineinander verschachtelten Verzweigungen und Wiederholungen erfordert von den Schülerinnen und Schülern mehr als nur „grundlegende“ Kompetenzen. Solche Algorithmen mit mehreren Kontrollstrukturen sind im Allgemeinen für Schülerinnen und Schüler der 8. bis 10. Jahrgangsstufe nicht einfach, denn die Funktionsweise der einzelnen algorithmischen Grundstrukturen sowie deren Zusammenwirken im Algorithmus bzw. Programm müssen verstanden werden. Hierzu sind „vertiefte“ bzw. „umfassendere“ Kompetenzen nötig.

**zu 3.** Unterprogramme mit Parametern zu erklären bzw. anzuwenden wird von den Schülerinnen und Schülern im Kompetenzmodell Algorithmen erst auf Stufe III gefordert (in der Komponente B). Die meisten Probleme, die von Schülerinnen und Schülern dieser Altersstufe bearbeitet werden, können auch ohne Unterprogramme gelöst werden. Wenn Unterprogramme mit Parametern erforderlich sind, so kann meist von komplexen Algorithmen bzw. Programmen gesprochen werden, denn es muss jeweils zwischen aktuellen und formalen Parametern unterschieden werden, außerdem muss der Parameterübergabe-Mechanismus beachtet werden.[60] Einige Lehrkräfte nutzen Unterprogramme bei der Einführung in die Programmierung von Beginn an zur Strukturierung von Programmen. Ein Beispiel wäre ein Programm, das in die drei Unterprogramme „Eingabe“, „Verarbeitung“ und „Ausgabe“ aufgeteilt ist. Dabei werden meist globale Variablen verwendet. Diese Art der Strukturierung ist mit der genannten Forderung nicht gemeint.

Durch die Kriterien zur Zuordnung von Algorithmen werden die abstrakten Begriffe

[60] Das Konzept der prozedurorientierten Programmierung erläutert Pomberger (1999, S. 517 ff.).

„einfache Algorithmen“, „Algorithmen“ und „komplexe Algorithmen“ fassbar und das Erstellen von Aufgaben zu den Stufen sowie das Einordnen von bestehenden Aufgaben in die Stufen wird vereinfacht.

Die Abstufungen einer jeden Kompetenz wurden in einem ausführlichen Diskussionsprozess entwickelt. Erfahrungen aus der im Kapitel 7 dargestellten Voruntersuchung sind in das Kompetenzmodell bereits mit eingeflossen. Auf die Unterschiede zwischen den Kompetenzen der Stufen I, II und III wird im Abschnitt 5.4 eingegangen (vgl. auch Anhang B).

## 5.4 Einordnung der Komplexität der Stufen

Eine Einordnung von konkreten Kompetenzen in eine relativ abstrakte Taxonomie erscheint schwierig und beinhaltet vermutlich auch immer subjektive Sichtweisen.[61] Trotzdem sollen die im entwickelten Modell geforderten Kompetenzen nachfolgend in die SOLO-Taxonomie eingeordnet werden, um so die Komplexität der verschiedenen Stufen analysieren zu können. Vorher werden die fünf Stufen der von Biggs und Collins (1982) entwickelten SOLO-Taxonomie („**S**tructure of the **O**bserved **L**earning **O**utcome“) vorgestellt.

Die SOLO-Taxonomie beschreibt Stufen beobachtbarer Lernergebnisse und zielt somit auf den Output (bzw. Outcome) von Bildung ab. Dabei bedeutet eine höhere Stufe eine höhere kognitive Komplexität (vgl. Braband und Dahl, 2007, S. 6). Die SOLO-Taxonomie wurde bereits von Lister u. a. (2006) sowie Mannila (2006) eingesetzt, um zu beschreiben, wie textueller Code von Anfängern verstanden wird. Braband (2007) nutzte die SOLO-Taxonomie, um die Kompetenzen in einen Universitätskurs zu „Concurrency“ zu strukturieren. Für eine Einführung in die SOLO-Taxonomie sei an dieser Stelle auf das von Brabrand und Andersen (2006) entwickelte Video „Teaching Teaching and Understanding Understanding“ verwiesen. Brabrand und Andersen (2006) beschreiben die fünf Stufen der SOLO-Taxonomie mit Bezug auf Biggs und Collins (1982) darin wie folgt:[62]

[61] Naps u. a. (2003, S. 144 ff.) versuchen auf Hochschulniveau, Kompetenzen zum Thema Algorithmen und Datenstrukturen in die Stufen der Bloomschen Taxonomie einzuordnen (vgl. Bloom, 1972). Sie stellen dabei fest: *[...] we must consider [...] that algorithm analysis is a complicated undertaking that has parts belonging to several levels in Bloom's taxonomy.*

[62] Die deutschen Untertitel des Videos von Lange und Axelsson (2006) dienten hier der Übersetzung.

**SOLO 1** *Auf der untersten Ebene 1 („pre-structural“) hat ein Student keinerlei Verständnis, benutzt irrelevante Informationen und verfehlt das Thema gänzlich.*

**SOLO 2** *Auf der Ebene 2 („uni-structural“) konzentriert sich der Student lediglich auf einen wichtigen Aspekt. Hier besitzt er die Fähigkeit eine Vorgehensweise nachzuvollziehen, etwas zu identifizieren und/oder zu rezitieren.*

**SOLO 3** *Eine Student auf der Ebene 3 („multi-structural“) kann mehrere verschiedene Aspekte in Betracht ziehen. Diese werden aber unabhängig voneinander behandelt. Er kann klassifizieren, kombinieren, aufzählen, etc.*

**SOLO 4** *Auf Ebene 4 („relational“) kann ein Student mehrere Teile zu einem kohärenten Ganzen zusammenfügen. Einzelheiten werden zu einem Fazit verbunden und die Bedeutung wird verstanden. Er kann Dinge zueinander in Beziehung setzen, vergleichen, analysieren, etc.*

**SOLO 5** *Auf der fünften und höchsten Ebene („extended abstract“) hat der Student die Fähigkeit, etwas über die ihm gegebene Informationen hinaus zu verallgemeinern und sogar neue Hypothesen oder Theorien aufzustellen, die dann hinterfragt werden können.*

Alle Kompetenzen des vorgestellten Modells wurden anhand der in Anhang B dargestellten Argumentationen in die Stufen der SOLO-Taxonomie eingeordnet.[63] In Tabelle 4 sind die Kompetenzen mit den zugeordneten Levels der SOLO-Taxonomie dargestellt. In Abbildung 20 wird veranschaulicht, wie oft auf Stufe I, II und III die unterschiedlichen SOLO-Levels zugeordnet wurden. Es zeigt sich, dass bei Kompetenzstufe II weniger Kompetenzen dem SOLO-Level 2 und mehr Kompetenzen dem SOLO-Level 4 zugeordnet wurden als bei Stufe I. Somit kann also eine gewisse Steigerung von Stufe I zu Stufe II festgestellt werden. In Kompetenzstufe III gibt es nur noch vereinzelt Kompetenzen, die den SOLO-Levels 2 und 3 zugeordnet wurden. Die meisten Kompetenzen entsprechen SOLO-Level 4, einige sogar SOLO-Level 5. Somit gibt es also auch eine Steigerung der Komplexität der Kompetenzen von Stufe II zu Stufe III. Insgesamt konnte festgestellt werden, dass bei einer höheren Kompetenzstufe mehr Kompetenzen auf einem höheren Level der SOLO-Taxonomie liegen. Dies ist ein Indikator dafür, dass die Kompetenzen auf einer höheren Stufe komplexer sind. Weitere theoretische Betrachtungen zu der Komplexität der Stufen des Kompetenzmodells erscheinen an dieser Stelle nicht sinnvoll. Die Festlegung der Stufen muss sich schlicht und einfach in der Praxis

[63] Ähnlich wie bei dem Versuch von Naps u. a. (2003), Kompetenzen in die Bloomsche Taxonomie einzuordnen, wären auch beim Einordnen in die SOLO-Taxonomie vereinzelt andere Argumentationen möglich gewesen. Nach Auffassung des Autors kann eine Einordnung vermutlich am treffsichersten erfolgen, wenn der erteilte Unterricht bekannt ist.

bewähren, d. h. Lehrpersonen müssen das Kompetenzmodell und die zugehörigen Aufgaben entsprechend des im Abschnitt 5.1 dargestellten Szenarios sinnvoll zur Strukturierung, Vorbereitung, Durchführung und Auswertung von Unterricht einsetzen können.

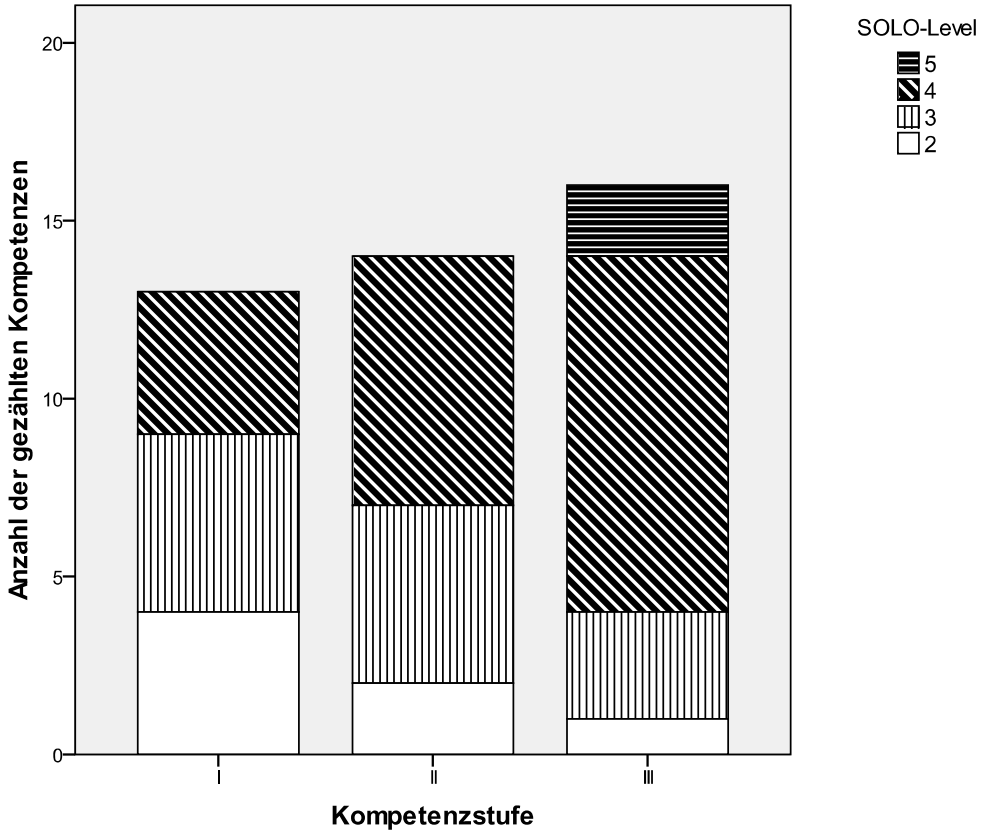



Abbildung 20: Anzahl der in die verschiedenen SOLO-Levels eingeordneten Kompetenzen je Stufe

## 5.5 Beziehungen zwischen Kompetenzmodell und GI-Empfehlungen

Das entwickelte Kompetenzmodell greift die in den GI-Empfehlungen geforderten Kompetenzen zum Inhaltsbereich „Algorithmen“ auf (vgl. Abschnitt 5.2). Wenn Bildungsstandards umgesetzt werden sollen, so reicht jedoch ein formales Abarbeiten eines einzelnen Inhaltsbereiches nicht aus (vgl. Friedrich und Puhlmann, 2007,

| Kompetenz | Stufe I | Stufe II | Stufe III |
|---|---|---|---|
| **Eigenschaften von Algorithmen** | | | |
| Den Algorithmusbegriff und die wesentlichen Eigenschaften von Algorithmen erklären | 2 | 2 | 3 |
| Die wesentlichen Eigenschaften von Algorithmen überprüfen | 3 | 4 | 4 |
| Probleme nennen, die mit Hilfe von Algorithmen lösbar bzw. nicht lösbar sind | 2 | 2 | 2 |
| **Algorithmische Grundbausteine und Datentypen** | | | |
| Algorithmische Grundbausteine erklären und anwenden | 3 | 3 | 3 |
| Algorithmische Grundbausteine darstellen | 2 | 3 | 4 |
| Datentypen verwenden | 2 | 3 | 3 |
| **Arbeit mit Algorithmen** | | | |
| Gegebene Algorithmen analysieren | 3 | 4 | 4 |
| Algorithmen prüfen | 3 | 3 | 4 |
| Algorithmen in Programme umsetzen | 4 | 4 | 4 |
| Algorithmen bzw. Programme modifizieren und ergänzen | 4 | 4 | 5 |
| Algorithmen bzw. Programme korrigieren | | | 4 |
| **Programmentwicklung** | | | |
| Programme entwerfen | 4 | 4 | 4 |
| Programme mit einem Programmiersystem implementieren | 4 | 4 | 4 |
| Programme auf Funktionalität testen | 3 | 3 | 4 |
| Über den Lösungsweg reflektieren | | 4 | 4 |
| Programme eigenständig verbessern | | | 5 |

Tabelle 4: Einordnung der Kompetenzen des Kompetenzmodells Algorithmen in die fünf Level der SOLO-Taxonomie

S. 31). Das entwickelte Kompetenzmodell beinhaltet vielfältige Verknüpfungen zu anderen Kompetenzbereichen der im Abschnitt 2.5 vorgestellten GI-Empfehlungen. In den in Anhang C dargestellten Tabellen sind in der linken Spalte Kompetenzen aus den GI-Empfehlungen angegeben, die von Schülerinnen und Schülern der Jahrgangsstufe 8 bis 10 in den jeweiligen Inhalts- und Prozessbereichen erwartet werden (vgl. GI, 2008, S. 14 ff.). Die mittlere Spalte zeigt jeweils die im Kompetenzmodell „Algorithmen“ geforderten Kompetenzen, die mit den Kompetenzen der Inhalts- und Prozessbereiche der GI-Empfehlungen in Verbindung stehen.[64] In der rechten Spalte der Tabellen werden die Beziehungen zwischen den in der linken und den in der mittleren Spalte angegebenen Kompetenzen erläutert.

In Tabelle 5 wurde jeweils mit einem Kreuz kenntlich gemacht, wenn die Kompetenzen einer Komponente mit den Inhalts- und Prozessbereichen in Verbindung stehen. Die Analyse zeigt, dass das entwickelte Kompetenzmodell nicht nur am Inhaltsbereich „Algorithmen“ der GI-Empfehlungen orientiert ist, sondern auch Bezüge zu „Information und Daten“, „Sprachen und Automaten“ sowie „Informatik, Mensch und Gesellschaft“ aufweist. Beziehungen zum Inhaltsbereich „Informatiksysteme“ konnten nicht festgestellt werden. Des Weiteren wurden die Verknüpfungen des Kompetenzmodells „Algorithmen“ zu Kompetenzen aller Prozessbereiche dargestellt. Diese Beziehungen zu den Inhalts- und Prozessbereichen der GI-Empfehlungen können helfen, den gesamten Informatikunterricht für eine Klasse, eine Schule oder auch ein Land zu konzipieren oder Aufgaben auf Grundlage des Kompetenzmodells zu entwickeln, in denen Kompetenzen verschiedener Inhalts- und Prozessbereiche kombiniert werden.
In den GI-Empfehlungen (2008, S. 23 ff.) wird Wert darauf gelegt, dass die Unterscheidung in Inhaltsbereiche nicht zur Unterteilung des Unterrichts in Themengebiete genutzt werden sollte, sondern dass die Kompetenzen in *beziehungsreichen Kontexten erworben werden. [...] Bei der Problemlösung werden also Inhalte aus mehreren Bereichen zugleich benötigt.* Die dargestellten Beziehungen zwischen dem Kompetenzmodell „Algorithmen“ und den Kompetenzen der anderen Inhaltsbereiche zeigen Möglichkeiten auf, im Unterricht solche beziehungsreichen Kontexte bereitzustellen.

[64] In Klammern sind jeweils die Komponente und die Stufe des Kompetenzmodells angegeben, in der die Kompetenz zu finden ist. Wenn eine Kompetenz in mehreren Stufen in unterschiedlicher Ausprägung angegeben werden könnte, wurde stets die Kompetenz der Stufe II aufgelistet.

| Inhalts- und Prozessbereiche der GI-Empfehlungen | Komponente A | Komponente B | Komponente C | Komponente D |
|---|---|---|---|---|
| Information und Daten | | X | X | X |
| Algorithmen | X | X | X | X |
| Sprachen und Automaten | | | X | X |
| Informatiksysteme | | | | |
| Informatik, Mensch und Gesellschaft | | | | X |
| Modellieren und Implementieren | | | X | X |
| Begründen und Bewerten | X | | X | X |
| Strukturieren und Vernetzen | | | | X |
| Kommunizieren und Kooperieren | | X | X | X |
| Darstellen und Interpretieren | | X | | X |

Tabelle 5: Beziehungen der in den einzelnene Komponenten angegebenen Kompetenzen zu den Inhalts- und Prozessbereichen der GI-Empfehlungen

## 5.6 Anforderungen an die Entwicklung eines Kompetenzmodells für einen Bereich der Informatik

Im Folgenden werden aus den Erfahrungen des dargestellten Entwicklungsprozesses gewonnene Anforderungen für die Entwicklung eines Kompetenzmodells für einen Bereich der Informatik vorgestellt. Dabei wird auch auf die im Kapitel 3 dargestellten Erkenntnisse wie z. B. die Unterteilung in Komponenten und Stufen eingegangen (vgl. Klieme u. a., 2003, S. 74). Die Anforderungen werden jeweils näher erläutert und an Beispielen des vorgestellten Kompetenzmodells veranschaulicht.

**1. Bei der Entwicklung des Kompetenzmodells relevante Forschungsergebnisse und die aktuelle Situation in Schulen beachten** Die Einbeziehung von Forschungsergebnissen kann entscheidende Hinweise für die Konstruktion von Kompetenzmodellen geben und führt außerdem zu einer wissenschaftlichen Fundierung des Kompetenzmodells. Wenn ein solches Modell Einfluss auf den Unterricht haben soll, ist es sinnvoll bereits während der Entwicklung die aktuelle Schulsituation zu beachten. Kompetenzmodelle, die aufgrund von Praxisferne unbeachtet bleiben, sind für den Unterricht letztlich wertlos.

In dieser Arbeit wurde allgemein auf Bildungsstandards, Kompetenzen, Kompetenzmodelle und Aufgaben sowie die entsprechenden Entwicklungen in der Informatik eingegangen (vgl. Kapitel 2 und 3). Anschließend wurde auf Grundlage der GI-Empfehlungen ein Kompetenzmodell entwickelt (vgl. Abschnitt 5.2 und 5.3). Dabei wurden Entscheidungen wie z. B. die Wahl des Themas „Algorithmen“ aufgrund von relevanten Forschungsergebnissen und der aktuellen Situation in Schulen begründet (vgl. Abschnitt 5.1). Außerdem erfolgten eine Einordnung der Komplexität der Stufen mit Hilfe der SOLO-Taxonomie sowie die Darstellung der Beziehungen des Kompetenzmodells zu den Inhalts- und Prozessbereichen der GI-Empfehlungen (vgl. Abschnitt 5.4 und 5.5).

**2. Das Kompetenzmodell in verschiedene Komponenten unterteilen** Durch das Einordnen der Kompetenzen in verschiedene Komponenten wird ein Kompetenzmodell systematisiert, die Diskussion darüber in der Wissenschaft, in einem Lehrerkollegium oder auch im Unterricht wird vereinfacht und die systematische Entwicklung von Aufgaben, insbesondere von Kompetenztests, wird unterstützt (vgl. Kohl und Fothe, 2007, S. 20).

Ohne die Unterteilung in Komponenten wären in Stufe III des entwickelten Kompetenzmodells 16 Kompetenzen unsystematisch in einer Spalte zusammengefasst

(vgl. Tabelle 3). In dieser Form wäre das Modell nur schwer handhabbar. Auch die Entwicklung eines Kompetenztests wurde durch das Einordnen der Kompetenzen in verschiedene Komponenten vereinfacht (vgl. hierzu auch Abschnitt 6.2).

**3. Kompetenzen verschiedener Stufen angeben** Um Schülerinnen und Schülern unterschiedlicher Leistungsniveaus kompetenzorientiert unterrichten zu können, ist es sinnvoll in einem Kompetenzmodell verschiedene Stufen unterschiedlicher Komplexität zu unterscheiden. Mit diesen Stufen sollte das Leistungsspektrum der Zielgruppe abgedeckt werden. Um eine ausreichende Differenzierung zu gewährleisten, sollte es zu jeder Stufe Aufgaben geben, die nur durch auf dieser Stufe neu erworbene Kompetenzen bearbeitet werden können (vgl. Klieme u. a., 2003, S. 76).

Im Kompetenzmodell wurden Kompetenzen auf drei Stufen angegeben. Eine Zuordnung von Kompetenzen zu den Stufen der SOLO-Taxonomie ergab, dass die Anzahl der Kompetenzen höherer Komplexität von Stufe I, über Stufe II hin zu Stufe III ansteigt. Beispielaufgaben, die die Unterschiede zwischen den drei Stufen verdeutlichen, werden im Kapitel 6 vorgestellt (zu weiteren Aufgaben vgl. Kohl, 2008a). Ob sich die Einteilung der Kompetenzen in die drei Stufen auch in der Praxis bewährt und inwieweit mit diesen Stufen das Leistungsspektrum der Zielgruppe abgedeckt wird, wird in den Untersuchungen dieser Arbeit analysiert (vgl. Kapitel 7 und 8).

**4. Die Kompetenzen verständlich und konkret formulieren, ohne unnötige Festlegungen zu treffen** Die Kompetenzen des Modells sollen für Lehrkräfte auch ohne die Untersetzung durch die Aufgaben ausreichend konkret und somit verständlich sein. Lehrpersonen müssen aber auch die Möglichkeit haben, in ihrem Unterricht spezifische Schwerpunkte zu setzen. Deshalb sind unnötige Festlegungen zu vermeiden. Insgesamt gilt es also, den richtigen Mittelweg zwischen zu konkreten und zu abstrakten Kompetenzbeschreibungen zu finden.

Kompetenzen wie z. B. „die Schülerinnen und Schüler erklären algorithmische Grundbausteine“, sind für ein Kompetenzmodell, das entsprechend der Anforderungen des vorgestellten Szenarios eingesetzt werden soll, nicht konkret genug und bedürfen weiterer Erläuterungen. Dem wurde durch die beispielhafte Angabe von algorithmischen Grundbausteinen entsprochen. An anderen Stellen wurden bewusst abstraktere Formulierungen gewählt. So wurde z. B. im entwickelten Kompetenzmodell auf Stufe II lediglich angegeben, dass verschiedene Darstellungsformen und Datentypen verwendet werden sollen. Welche das im Detail sind, kann die Lehrkraft selbst bestimmen. Eine Konkretisierung der Kompetenzen durch

spezielle Aufgaben ist möglich (vgl. Abschnitt 6.1). Ob der richtige Mittelweg zwischen zu konkreten und zu abstrakten Kompetenzformulierungen gefunden wurde, werden die Untersuchungen dieser Arbeit zeigen (vgl. Kapitel 7 und 8).

**5. Das Kompetenzmodell einheitlich, übersichtlich, kurz und handhabbar bereitstellen** Bei der Angabe von Kompetenzen sollte – soweit möglich – auf Wiederholungen verzichtet werden. Wenn alle Kompetenzen in einem einheitlichen Satzbau formuliert sind, kann ein typischer Satzanfang wie „Die Schülerinnen und Schüler ...“ in die oberste Zeile einer Stufe geschrieben werden. Die typografische Hervorhebung von Kompetenzen, die auf einer höheren Stufe hinzukommen, erhöht die Übersichtlichkeit. Die Darstellung des Kompetenzmodells auf mehreren Seiten vermindert die Übersichtlichkeit.

Im angegebenen Kompetenzmodell wurden alle Kompetenzen in einem einheitlichen Satzbau beschrieben. Kompetenzen der Stufe II, die bereits in Stufe I genannt sind und Kompetenzen der Stufe III, die bereits in Stufe II genannt sind, wurden im vorliegenden Modell grau gedruckt. Inwieweit diese typografische Hervorhebung neu hinzugekommener Kompetenzen sowie die komprimierte Darstellung des Modells auf einer Seite die Diskussion über das Kompetenzmodell und die Arbeit mit diesem vereinfachten, wird in den Untersuchungen dieser Arbeit analysiert (vgl. Kapitel 7 und 8).

**6. Im Kompetenzmodell nicht auf spezielle Software, Programmiersprachen oder Entwicklungsumgebungen eingehen** Der Umgang mit Technik – insbesondere als digitales Hilfsmittel – ist ein wesentlicher Bestandteil des Informatikunterrichts (vgl. GI, 2008, S. 8). Wenn allerdings in einem Kompetenzmodell spezielle Programmiersprachen, Entwicklungsumgebungen oder andere Softwareprodukte angegeben würden, so verminderte dies die Vielfalt der eingesetzten Werkzeuge im Unterricht und verkürzte evtl. die Lebensdauer des Kompetenzmodells aufgrund der Abhängigkeit vom verwendeten System. Die genannte Vielfalt der im Unterricht eingesetzten Werkzeuge bedeutet auch eine Variabilität im Informatikunterricht, deren Chancen und Möglichkeiten nicht durch unnötige Festlegungen zerstört werden sollten.

Im vorgestellten Kompetenzmodell wurde kein Programmiersystem angegeben, das zum Erwerben der Kompetenzen genutzt werden soll, und somit wurde eine Abhängigkeit von einer speziellen Software vermieden. Trotzdem ist im Kompetenzmodell nicht nur Konzept-, sondern auch Produktwissen gefordert.[65]

[65]Konzeptwissen umfasst die längerfristig gültigen, grundlegenden Zusammenhänge eines Sach-

Denn wenn Programme mit einem Programmiersystem implementiert werden sollen, so setzt das Kompetenzen im Umgang mit einem solchem System voraus. Ob dieses Programmiersystem Delphi, Greenfoot oder Puck sein soll, kann von der Lehrkraft entschieden werden.

**7. Parallel zum Kompetenzmodell zugehörige Aufgaben bereitstellen** Kompetenzmodelle sind zur Gestaltung kompetenzorientierten Unterrichts wenig hilfreich, wenn keine Aufgaben zur Verfügung stehen, mit denen die angegebenen Kompetenzen erworben und überprüft werden können. Deshalb ist es wichtig, nicht nur Aspekte eines Kompetenzmodells an einer einzelnen Aufgabe zu verdeutlichen, sondern parallel zum Kompetenzmodell eine Sammlung von zugehörigen Aufgaben bereitzustellen.

Zum vorgestellten Kompetenzmodell wurde parallel eine Sammlung von Aufgaben entwickelt, die im Kapitel 6 ausführlich vorgestellt wird (vgl. auch Kohl, 2008a).

**8. Für Verbreitung des Kompetenzmodells sorgen** Szenarien für den Einsatz des Kompetenzmodells sind vor der Entwicklung abzuklären. Eine wissenschaftliche Diskussion über ein Kompetenzmodell und der Einsatz eines solchen im Unterricht können nur stattfinden, wenn dieses Modell hinreichend verbreitet wurde. Hierfür gibt es verschiedene Möglichkeiten: Tagungen, Zeitschriftenartikel, Lehrbücher, Lehrerfortbildungen und Untersuchungsdesigns, bei denen eine größere Anzahl von Lehrkräften mit dem Kompetenzmodell in Kontakt kommt, sollen als Beispiele genannt werden.

Das entwickelte Modell wurde in zwei Untersuchungen erprobt (vgl. Kapitel 7 und 8). Außerdem wurde das Kompetenzmodell „Algorithmen“ in der Fachzeitschrift LOG IN veröffentlicht (vgl. Kohl und Fothe, 2007).

## 5.7 Zusammenfassung

In diesem Kapitel wurde entsprechend der Erkenntnisse aus Kapitel 3 ein Kompetenzmodell „Algorithmen“ für die Jahrgangsstufen 8 bis 10 entwickelt, das Informatiklehrkräften als Grundlage für kompetenzorientierten Unterricht dienen soll. Die Wahl des Themas wurde mit verschiedenen Argumenten begründet. Die Auswahl der zu erwerbenden Kompetenzen orientierte sich am Inhaltsbereich

gebietes; Produktwissen beinhaltet die Kenntnisse und Fähigkeiten, die zur Bedienung eines bestimmten Produkts nötig sind (vgl. Hartmann u. a., 2006, S. 23 ff.).

„Algorithmen“ der im Abschnitt 2.5 vorgestellten GI-Empfehlungen für Bildungsstandards Informatik. Die Kompetenzen wurden zur Systematisierung und zur begründeten Vorwegnahme der Struktur eines Kompetenztests vier Komponenten zugeordnet. Außerdem wurde das Kompetenzmodell in drei Stufen unterteilt. Die Analyse der einzelnen Kompetenzen mit Hilfe der SOLO-Taxonomie ergab, dass die Anzahl der Kompetenzen höherer Komplexität von Stufe I über Stufe II zu Stufe III ansteigt. Beziehungen des Kompetenzmodells zu anderen Inhalts- und Prozessbereichen der GI-Empfehlungen wurden aufgezeigt. Schließlich wurden in einem Reflexionsprozess acht Anforderungen für die Entwicklung eines Kompetenzmodells für einen Bereich der Informatik zusammengetragen.

# 6 Die Entwicklung von Aufgaben zum Kompetenzmodell

> *Im Informatikunterricht herrscht ein Mangel an interessanten und abwechslungsreichen Aufgaben.* (Schubert und Schwill, 2004, S. 137 f.)

Zur Beantwortung der ersten Forschungsfrage soll nachfolgend die Entwicklung von Aufgaben zu dem im Kapitel 5 vorgestellten Kompetenzmodell beschrieben werden. Dadurch wird auch versucht, dem von Schubert und Schwill (2004, S. 137 ff.) beschriebenen Mangel an Aufgaben im Informatikunterricht zu begegnen (vgl. hierzu auch Friedrich und Puhlmann, 2007, S. 32). Im Abschnitt 6.1 wird dargestellt, wie die in Vorbereitung auf die Untersuchungen entwickelten Aufgaben die geforderten Kompetenzen konkretisieren. Im Abschnitt 6.2 werden die Beispielaufgaben, zur Veranschaulichung und die Testaufgaben zur Überprüfung der Kompetenzen vorgestellt. Im Abschnitt 6.3 werden die Unterrichtsaufgaben, die von den Lehrpersonen im Rahmen der Hauptuntersuchung eingesetzt werden konnten, präsentiert. Im Abschnitt 6.4 wird exemplarisch eine Aufgabe mit Hilfe einer Bewertungsmatrix bezüglich der in ihr geforderten Kompetenzen analysiert. In einem Reflexionsprozess werden im Abschnitt 6.5 Anforderungen zur Entwicklung von Aufgaben für ein Kompetenzmodell im Informatikunterricht vorgestellt.

## 6.1 Konkretisierung der geforderten Kompetenzen

Nach Klieme u. a. (2003, S. 23) sagt das Erreichen einer Kompetenzstufe *etwas darüber aus, welche Handlungen und mentalen Operationen mit hoher Wahrscheinlichkeit korrekt ausgeführt werden können. Für die Umsetzung im Lehrplan und im Unterricht wie auch für die Bewertung von Schülerleistungen braucht man aber konkrete Beispiele und „Operationalisierungen", bis hin zu Messvorschriften in Gestalt von Testaufgaben.* Ziel der folgenden Festlegungen zu den im Kapitel 5 genannten Kompetenzen war es, für die durchzuführenden Untersuchungen einen Rahmen bereitzustellen, der es ermöglichte, die Kompetenzen durch konkrete Aufgaben zu überprüfen. Es wurde festgelegt, welche Datentypen, Darstellungsformen und Anweisungen im Unterricht dieser Untersuchungen thematisiert werden sollten. Somit war es möglich, dass alle beteiligten Lehrpersonen ein und denselben

Kompetenztest einsetzen konnten, in dem auch konkrete Algorithmen vorgegeben waren. Weil in der Untersuchung Puck verwendet werden sollte, orientieren sich die nachfolgend vorgestellten Festlegungen unter anderem an den Möglichkeiten dieser visuellen Programmiersprache (vgl. hierzu Kapitel 4).

- In Komponente A des Kompetenzmodells wird von den Schülerinnen und Schülern gefordert, wesentliche Eigenschaften von Algorithmen erklären und überprüfen zu können. Als wesentliche Eigenschaften von Algorithmen wurden in den Aufgaben Allgemeingültigkeit, Ausführbarkeit, Endlichkeit und Eindeutigkeit verstanden. Erläuterungen zu den vier Begriffen wurden im Arbeitsmaterial zur Untersuchung angegeben (vgl. Kohl, 2008a, S. 5).
- In Komponente B des Kompetenzmodells wird gefordert, dass die Schülerinnen und Schüler algorithmische Grundbausteine darstellen, und zwar als „Pseudocode“ auf Stufe I bzw. in verschiedenen Darstellungsformen auf Stufe II und III. Für Pseudocode gibt es keine allgemeinen Festlegungen. In allen entwickelten Aufgaben wurde eine einheitliche Form verwendet, in der Anweisungen in ganzen Sätzen natürlicher Sprache formuliert, Ausdrücke und Variablen stets kursiv geschrieben und Schleifenrümpfe, Entscheidungszweige sowie Unterprogramme stets eingerückt in einem Kasten dargestellt sind. Die Aufgaben der Komponenten B sind auf Stufe II und III so konstruiert, dass zur Lösung außer Pseudocode entweder Struktogramme oder Programmablaufpläne bekannt sein müssen.
- In Komponente B wird gefordert, dass die Schülerinnen und Schüler einen numerischen Datentyp (Stufe I) bzw. verschiedene Datentypen (Stufe II und III) verwenden. In vielen Programmiersprachen werden Variablen des Typs Integer in Zählscheifen genutzt. Als numerischer Datentyp wurde in den Aufgaben deshalb Integer gewählt. Auf Stufe II und III wurde, den Möglichkeiten der verwendeten Programmiersprache Puck entsprechend, Boolean als weiterer zu beherrschender Datentyp festgelegt. Damit werden logische Operationen, wie sie in Bedingungen von Alternativen und Schleifen verwendet werden, für die Schülerinnen und Schüler sinnvoll nutzbar. Somit müssen in Stufe II bzw. III arithmetische und logische Operationen gekannt und verwendet werden. Das entspricht einer der im Inhaltsbereich „Informationen und Daten“ der GI-Empfehlungen genannten Kompetenzen. Durch das Verwenden der Datentypen Integer und Boolean war es möglich, eine ausreichend große Menge von Aufgaben zu konstruieren (vgl. hierzu auch Abschnitt 4.4).
- In Komponente B kommen auf Stufe III zu den bis dahin verwendeten algorithmischen Grundbausteinen noch Unterprogramme mit Parametern hinzu.

In den entwickelten Aufgaben wurden hierfür Prozeduren mit Wert- und Referenzparametern verwendet. Diese Grundstrukturen können in verschiedenen Problemlösungen eingesetzt werden. Die genannten Parameterarten wurden im Arbeitsmaterial zur Untersuchung erläutert (vgl. Kohl, 2008a, S. 5).

- In den Komponenten C und D des entwickelten Modells sind Kompetenzen angegeben, die die Implementierung von gegebenen und selbst entwickelten Algorithmen in einem Programmiersystem enthalten. Grundsätzlich sind im Kompetenzmodell keine Vorgaben zum zu wählenden Programmiersystem gemacht. Um eine Abhängigkeit von Programmiersprachen und -systemen zu vermeiden, wurden auch die Aufgaben programmiersprachenunabhängig formuliert. Insbesondere sind Algorithmen in Pseudocode angegeben. Die Auswahl der in den Aufgaben verwendeten Anweisungen und Datentypen orientiert sich an der in den Untersuchungen eingesetzten visuellen Programmiersprache Puck. Die Aufgaben können aber auch mit verschiedenen anderen Systemen gelöst werden.

Durch diese Festlegungen und die im Abschnitt 5.3 vorgestellten Kriterien zur Einordnung von Algorithmen in die Stufen des Kompetenzmodells können nachfolgend konkrete Beispiel-, Unterrichts- und Testaufgaben konstruiert werden. Prinzipiell wären auch andere Konkretisierungen möglich gewesen.

## 6.2 Beispiel- und Testaufgaben

Mit Hilfe der Testaufgaben soll überprüft werden, inwieweit die im Kompetenzmodell „Algorithmen“ angegebenen Kompetenzen von den Lernenden erreicht wurden. Wie im Abschnitt 3.3 dargestellt, sollen die Testergebnisse also kriteriums- und nicht verteilungsorientiert interpretiert werden. Entsprechend der von Blum (2006, S. 18) vorgestellten Kriterien wurde versucht die Testaufgaben so zu konstruieren, dass diese für Schülerinnen und Schüler ohne externe Unterstützung in überschaubarer Zeit bearbeitbar sind und von den Lehrpersonen leicht korrigiert werden können (vgl. Abschnitt 3.3).
Außerdem wurden entsprechend der Darstellungen im Abschnitt 3.3 Beispielaufgaben bereitgestellt, die die Kompetenzen des Modells verdeutlichen und konkretisieren sowie Unterschiede zwischen den verschiedenen Stufen verdeutlichen (vgl. Klieme u. a., 2003, S. 124). Lehrende und Lernende können sich mit Hilfe der Beispielaufgaben darüber verständigen, was einfache bzw. komplexe Algorithmen sind. Sie können z. B. in Pseudocode gegebene Algorithmen verschiedener Stufen analysieren. Außerdem können Lehrpersonen sowie Schülerinnen und Schü-

ler anhand der Beispielaufgaben die erworbenen Kompetenzen einschätzen und entsprechende Konsequenzen ableiten. Denn die im Abschnitt 3.3 beschriebenen Vorteile niveau-angepasster Tests treten nur dann auf, wenn Schülerinnen und Schüler, die Kompetenzen einer bestimmten Stufe entwickelt haben, auch einen Kompetenztest dieser Stufe bearbeiten.

Die Beispiel- und Testaufgaben wurden unter Beachtung der Konkretisierungen aus Abschnitt 6.1 und der Kriterien zur Zuordnung von Algorithmen zu den Kompetenzstufen im Abschnitt 5.3 gezielt anhand der Kompetenzen des Modells entwickelt (vgl. Abschnitt 3.3 bzw. Klieme u. a., 2003, S. 124 f.). Es gibt je einen Satz Beispiel- und Testaufgaben für Stufe I, II und III. Dabei ist jeder Aufgabensatz in Aufgaben zu den vier Komponenten A, B, C, und D unterteilt. Es wurde versucht, die geforderten Kompetenzen möglichst genau durch die Aufgaben abzubilden.
Erste Untersuchungen hatten ergeben, dass bei Aufgaben, die in einem längeren Fließtext gegeben waren, von Schülerinnen und Schülern mitunter Teile nicht bearbeitet wurden.[66] Als Konsequenz auf diese Ergebnisse wurden die Aufgaben in überschaubare Teilaufgaben untergliedert und zu schriftlich zu bearbeitenden Aufgabenteilen wurde jeweils ein Antwortfeld bereitgestellt (vgl. hierzu Abschnitt 7.5).

Punktangaben wurden anhand von Musterlösungen festgelegt und sollten eine einheitliche Gewichtung der Teilaufgaben ermöglichen und Lehrenden sowie Lernenden eine Orientierung über den jeweiligen Umfang der verschiedenen Teilaufgaben und das Verhältnis zur gesamten Aufgabe geben. Eine Übersicht über die in den Beispiel- und Testaufgaben zu erreichenden Punkte gibt Tabelle 6. In den Aufgaben höherer Stufen wurden mehr Punkte vergeben, da komplexere Algorithmen mit erweiterten Kompetenzen bearbeitet werden mussten. Durch die unterschiedlichen Punktzahlen der verschiedenen Stufen ist es möglich, einfache Kriterien für das Erreichen der Kompetenzstufen zu definieren. Für die im Kapitel 8 dargestellte Hauptuntersuchung wurde festgelegt, dass zum Erreichen einer Stufe 75 % der auf dieser Stufe möglichen Punkte benötigt werden. Das folgende Beispiel verdeutlicht die sich daraus ergebenden Möglichkeiten der Bewertung:

**Beispiel:** Ein Schüler hat im Kompetenztest Stufe II bearbeitet und die Lehrerin hat den Test mit 25 der 48 möglichen Punkte bewertet. Das Erreichen der Stufe II kann ihm nicht bescheinigt werden, da sein Ergebnis unter 75 % (also 36 Punkten) liegt. Da er aber mehr als 18 Punkte erzielte (also mehr als 75 % der Stufe I), hat er Stufe I erreicht. Somit konnten dem Schüler die

[66] In der im Kapitel 7 dargestellten Voruntersuchung wurde eine erste Version der Beispiel- und Testaufgaben eingesetzt.

Kompetenzen der Stufe I bescheinigt werden, obwohl er einen Test der Stufe II wenig erfolgreich bearbeitet hat.

| Komponente | Stufe I | Stufe II | Stufe III |
|---|---|---|---|
| A | 4 | 8 | 12 |
| B | 4 | 8 | 12 |
| C | 8 | 16 | 24 |
| D | 8 | 16 | 24 |
| Gesamtpunktzahl | 24 | 48 | 72 |
| 75 % Kriterium | 18 | 36 | 54 |

Tabelle 6: Punkteverteilung zu Beispiel- und Testaufgaben

Alternativen zu dem beschriebenen Vorgehen wären, von allen Schülerinnen und Schülern die Kompetenztests aller Stufen bearbeiten zu lassen oder mit individuellen Hilfestellungen zu arbeiten (vgl. hierzu Abschnitt 3.5 bzw. Schlüter und Brinda, 2007). Die Punktverteilung und das 75 % Kriterium wurden nach den Erkenntnissen der im Kapitel 7 dargestellten Voruntersuchung erarbeitet. Die an der Hauptuntersuchung teilnehmenden Lehrpersonen sollten den Kompetenztest einsetzen und anschließend einschätzen (vgl. Kapitel 8). Nachfolgend werden exemplarisch zwei Test- und zwei Beispielaufgaben vorgestellt:

- Abbildung 21 zeigt eine Testaufgabe der Stufe I zur Komponente A.

  In der Aufgabe wird eine Handlungsvorschrift präsentiert. Es soll entschieden werden, ob diese Handlungsvorschrift die Eigenschaften „Allgemeingültigkeit“ und „Eindeutigkeit“ erfüllt. Außerdem soll der Algorithmusbegriff erklärt werden und es soll ein Beispiel für ein Problem angegeben werden, das mithilfe von Algorithmen lösbar ist.

- Abbildung 22 zeigt eine Testaufgabe der Stufe II zur Komponente D.

  In der Aufgabe wird eine Übersicht über die Preise verschiedener Telefonanbieter (Vorvorwahl) gegeben. Ein Programm, welches zur Eingabe von Art der Nummer und Tageszeit den günstigsten Anbieter mit Nummer ausgibt soll entworfen, implementiert und getestet werden. Entsprechend der in Komponente D auf Stufe II geforderten Kompetenz zur Reflexion soll außerdem über den Lösungsweg reflektiert werden.

- Abbildung 23 zeigt eine Beispielaufgabe der Stufe III zur Komponente B.

  In der Aufgabe sollen zwei Parameterarten mit Funktionsweisen der Parameterübergabe sowie zwei Datentypen mit möglichen Werten und Operationen

angegeben werden. Außerdem sollen algorithmische Grundbausteine, einer kurzen Anweisungsfolge in eine jeweils andere Darstellungsform überführt werden.

- Abbildung 24 zeigt eine Beispielaufgabe der Stufe II zur Komponente C.

  In der Aufgabe ist ein in Pseudocode formulierter Algorithmus für eine Kasse in einem Lebensmittelladen gegeben. Die Funktionsweise des Algorithmus soll anhand einer Durchlauftabelle analysiert und mit eigenen Worten wiedergegeben werden. Anschließend ist der Algorithmus umzusetzen und zu modifizieren.

Musterlösungen zu den vier vorgestellten Aufgaben sind im Anhang D dargestellt. Alle anderen Beispiel- und Testaufgaben, die den Lehrkräften in der Hauptuntersuchung dieser Arbeit bereitgestellt wurden, sind im Arbeitsmaterial zur Untersuchung zusammengetragen (vgl. Kohl, 2008a, S. 8 ff.).

**Kompetenztest** **Stufe I** **Komponente A**

**Handlungsvorschrift: Eine Erfindung zu Geld machen**

1. Machen Sie eine Erfindung.
2. Recherchieren Sie, ob es diese Erfindung schon gibt.
3. Wenn es die Erfindung schon gibt, gehen Sie zu 1.
4. Reichen Sie einen Antrag für ein Patent über die neue Erfindung beim Patentamt ein.
5. Bezahlen Sie die Gebühren.
6. Wenn Sie eine zahlungskräftige Firma finden, die Interesse an der Erfindung hat, dann verkaufen Sie das Patent an diese Firma.
7. Wenn Sie keine zahlungskräftige Firma finden, die Interesse an der Erfindung hat, dann gründen Sie eine eigene Firma und entwickeln Sie ein Produkt, das sich gut verkauft.

Erfüllt die gegebene Handlungsvorschrift die Eigenschaft Allgemeinheit?

O JA

O NEIN

Erfüllt die gegebene Handlungsvorschrift die Eigenschaft Eindeutigkeit?

O JA

O NEIN

2 Punkte

Erklären Sie den Begriff Algorithmus.

1 Punkt

Nennen Sie ein Beispiel für ein Problem, das mithilfe von Algorithmen lösbar ist.

1 Punkt

Abbildung 21: Eine Testaufgabe Stufe I Komponente A

**Kompetenztest** **Stufe II** **Komponente D**

Marco hat im letzten Monat zu viel telefoniert. Seine Eltern haben jetzt beschlossen, ihm einen festen Betrag für Telefonate bereitzustellen. Marco will nun günstiger telefonieren. Er hat die Minutenpreise von vier unterschiedlichen Anbietern und die zugehörigen Telefonnummern herausgesucht.

| | Hauptzeit Festnetz | Nebenzeit Festnetz | Hauptzeit Mobil | Nebenzeit Mobil |
|---|---|---|---|---|
| Telefonus (01234) | 0,03 | 0,02 | 0,38 | 0,30 |
| Mobilus (09876) | 0,07 | 0,05 | 0,25 | 0,20 |
| Genialion (0547490) | 0,04 | 0,01 | 0,32 | 0,27 |
| Billigus (0150705) | 0,05 | 0,03 | 0,27 | 0,18 |

Kreuzen Sie den jeweils günstigsten Anbieter in jeder der vier Spalten an.

Fertigen Sie einen schriftlichen Entwurf für ein Programm an, bei dem der Benutzer zwischen Haupt- und Nebenzeit sowie zwischen Festnetz- und Mobilnummer auswählen kann. Anschließend soll der günstigste der vier gegebenen Anbieter mit Nummer ausgegeben werden.

5 Punkte

Implementieren Sie das entworfene Programm benutzungsfreundlich. 6 Punkte

Testen Sie Ihr Programm mit folgenden Eingaben:

A) ein Telefonat zu einer Festnetznummer in der Nebenzeit
B) ein Telefonat zu einer Mobilfunknummer in der Hauptzeit

Notieren Sie alle Eingaben und Ausgaben Ihres Programms bei beiden Testfällen.

4 Punkte

Reflektieren Sie über Ihren Lösungsweg. Welche Schwierigkeiten traten beim Entwurf und bei der Implementierung auf? Wie haben Sie diese Probleme gelöst?

1 Punkt

Abbildung 22: Eine Testaufgabe Stufe II Komponente D

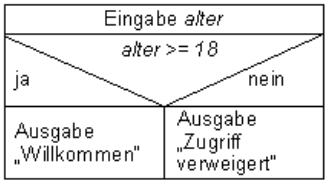
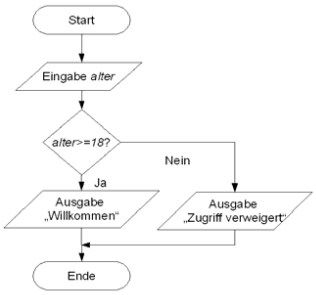
**Beispielaufgaben** **Stufe III** **Komponente B**

Nennen Sie zwei Ihnen bekannte Parameterarten und erklären Sie die Funktionsweise der Parameterübergabe.

| Parameterart | Funktionsweise der Parameterübergabe |
|---|---|
| | |
| | |

4 Punkte

Überführen Sie die gegebenen Beispiele in **eine** andere Darstellungsform. 4 Punkte

| Pseudocode | Struktogramm | Programmablaufplan |
|---|---|---|
| Weise der Variablen *a* den Wert von *0* zu.<br>Solange wie *(a < 5)* wahr ist, mache Folgendes:<br>Weise der Variablen *a* den Wert von *a+1* zu.<br>Weise der Variablen *b* den Wert von *a*a* zu.<br>Gib den Wert von *b* aus.<br>Gib den Text „Programmende" aus. | | |
| |  |  |

Geben Sie für zwei Datentypen jeweils zwei Werte und zwei Operationen an.

| Datentyp | Werte | Operationen |
|---|---|---|
| | | |
| | | |

4 Punkte

Abbildung 23: Eine Beispielaufgabe Stufe III Komponente B

**Beispielaufgaben** **Stufe II** **Komponente C**

Die Großeltern von Stefan besitzen einen kleinen Lebensmittelladen. Sie haben vor kurzem einen Computer gekauft. Sie wünschen sich nun von Stefan ein Programm, das die in die Jahre gekommene Kasse ersetzt. Dabei soll für alle eingekauften Waren eines Kunden jeweils die Menge und der Preis in Cent eingegeben werden. Am Ende soll der Gesamtpreis ausgegeben werden.
Stefan hat sich den folgenden Algorithmus überlegt:

ALGORITHMUS Kasse

Lege die Variablen *menge*, *preis* und *summe* vom Typ Integer an.
Weise der Variablen *summe* den Wert *0* zu.

Mache Folgendes so lange, bis *(menge = 0)* wahr ist:

Fordere den Benutzer mit: "Wie viele Artikel (0 zum Beenden)?" auf, eine Zahl einzugeben. Speichere die eingegebene Zahl in die Variable *menge*.

Wenn die Bedingung *(menge # 0)* wahr ist, mache Folgendes:

Fordere den Benutzer mit: "Preis in Cent:" auf, eine Zahl einzugeben.
Speichere die eingegebene Zahl in die Variable *preis*.
Weise der Variablen *summe* den Wert von *summe+menge*preis* zu.

Gib "Der Kunde muss " und danach den Wert von *summe* aus.
Gib den Text " Cent zahlen." aus.

Lesen Sie den Algorithmus und geben Sie seine Funktionsweise mit eigenen Worten wieder.

3 Punkte

Frau Meier kauft :
zwei Becher Joghurt für je 50 Cent,
vier Kiwis für je 25 Cent
und eine Mango für 99 Cent.

Überprüfen Sie den gegebenen Algorithmus für den Einkauf von Frau Meier mithilfe einer Durchlauftabelle.

| menge | preis | summe |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

3 Punkte

Setzen Sie den Algorithmus in ein Programm um. 6 Punkte

Verändern Sie das Programm so, dass die Summe in Euro und Cent ausgegeben wird. 2 Punkte

Außerdem soll nach der Berechnung der Summe noch eingegeben werden, mit welchem Betrag der Kunde zahlt. Anschließend soll ausgegeben werden, wie viel Wechselgeld der Kassierer zurückzugeben hat. 2 Punkte

Abbildung 24: Eine Beispielaufgabe Stufe II Komponente C

## 6.3 Unterrichtsaufgaben

In Vorbereitung auf die Untersuchungen dieser Arbeit wurden Aufgaben zum Thema „Algorithmen“ entwickelt, die mit Puck bearbeitet werden können. Die in der Voruntersuchung eingesetzten Unterrichtsaufgaben wurden aufgrund der gesammelten Erfahrungen in die Komponenten und Stufen des Kompetenzmodells eingeordnet, teilweise modifiziert und ergänzt (vgl. hierzu Abschnitt 7.4).[67] Die folgenden Darstellungen beziehen sich auf die dadurch entstandene Sammlung von 70 Unterrichtsaufgaben, die im Arbeitsmaterial zur Hauptuntersuchung nummeriert dargestellt sind (vgl. Kohl, 2008a, S. 18 ff.). Diese decken nicht immer alle jeweils geforderten Kompetenzen einer Komponente des Kompetenzmodells ab. Das ist auch nicht erforderlich, da im Unterricht im Allgemeinen eine größere Anzahl an Aufgaben in verschiedener Weise eingesetzt wird, sei es als Einführung in ein neues Thema, als Übung zur Festigung, als Zusatzaufgabe zur Differenzierung, als Hausaufgabe, zur Leistungsüberprüfung oder in anderer Funktion. Das Ziel bestand darin, ein möglichst breites Spektrum an Unterrichtsaufgaben bereitzustellen. Auch die im Abschnitt 6.2 vorgestellten Beispielaufgaben können im Unterricht eingesetzt werden (vgl. hierzu Abschnitt 3.4).

Girmes (2003, S. 6) stellt fest, dass schulische Aufgaben von den Lernenden nur angenommen werden, wenn sie einen Bezug zu ihrer Welt haben (vgl. auch Brinda und Mägdefrau, 2008, S. 16). Breier und Hilger (2008, S. 119) erläutern an einem Beispiel das Unterrichtskonzept „Informatik im Kontext“, bei dem der gesamte Unterricht entlang einer Abfolge sinnstiftender Kontexte organisiert ist. Und auch in den „Grundsätzen und Standards für die Informatik in der Schule“ wird von der GI (2008, S. 6) empfohlen, von der Erfahrungswelt der Schülerinnen und Schüler auszugehen, denn *Lebensverbundenheit und lebensweltlicher Bezug sind unverzichtbar für einen guten (und erfolgreichen) Informatikunterricht [...].*[68] Dementsprechend beginnen die Aufgaben 13, 20, 21, 23, 43, 46 und 61 der Sammlung mit der Darstellung einer Kontextsituation, die für Schülerinnen und Schüler

[67] Im Rahmen dieser Arbeit wurden die Unterrichtsaufgaben jeweils genau einer Stufe und einer Komponente des Kompetenzmodells zugeordnet. Vermutlich kann es auch sinnvoll sein, Aufgaben zu konstruieren, die mehreren Komponenten bzw. mehreren Stufen zuzuordnen sind.

[68] Im Rahmen der Entwicklung der GI-Empfehlungen sind verschiedene Aufgaben vorgestellt worden, die mit der Angabe einer den Schülerinnen und Schülern bekannten Kontextsituation beginnen. Die von Puhlmann (2005, S. 30 f.) präsentierten Aufgaben beziehen sich beispielsweise auf den Einkauf in einem Supermarkt. Dieser Kontext sollte nahezu allen Schülerinnen und Schülern bekannt sein und somit ein gewisses Interesse wecken. Eine weitere von Puhlmann und Friedrich (2007, S 17 f.) vorgestellte Aufgabe greift ein jugendgemäßes Thema, nämlich ein Bowlingspiel, auf.

nachvollziehbar sein sollte. Des Weiteren beschäftigen sich die Aufgaben 9, 30, 45, 51, 62 und 69 mit Themen, die im Allgemeinen das Interesse von Schülern der 8. bis 10. Jahrgangsstufe wecken.
In den GI-Empfehlungen (2008, S. 10) wird dargestellt, dass ein wesentliches Kennzeichen der Informatik als Wissenschaft ihre Interdisziplinarität ist, dass Informatik per se fachübergreifend und fächerverbindend ist und dass deshalb Interdisziplinarität ein Grundsatz der Unterrichtsgestaltung ist. Dementsprechend wurden auch fächerübergreifende Aufgaben entwickelt. Eine Übersicht über die Beziehungen der Unterrichtsaufgaben zu anderen Fächern gibt Tabelle 7.[69]

Schwarz u. a. (2007) stellen fest, dass freiere Aufgabenstellungen motivierender sind und Aufgaben, die der Kreativität der Schülerinnen und Schüler mehr Raum geben, besser ankommen als Aufgaben, die einschränken. Für das Hervorbringen kreativer Leistungen im Informatikunterricht empfiehlt Romeike (2007b, S. 66) offene Aufgaben zu stellen, mit denen Schülerinnen und Schüler erfahren können, dass sie Informatiksysteme selbst gestalten können.[70] Lankes u. a. (2006, S. 27 f.) bezeichnen Aufgaben als geschlossen, *wenn die betreffenden Fragen bzw. geforderten Ziele explizit formuliert sind und die anzuwendenden Methoden und Hilfsmittel durch die Aufgabenstellung nahe gelegt werden. Alle Aufgaben, die eine der beiden genannten Bedingungen oder beide nicht erfüllen heißen offen.* In den Aufgaben 21, 26, 48, 54, 56, 62, 64 und 69 sind zumindest Teilaufgaben offen formuliert. Die Aufgaben 21, 26, 30 und 48 können nach Auffassung des Autors zur Kreativität im Informatikunterricht anregen.
Im Bereich der Programmierung gibt es oft unterschiedliche Vorkenntnisse und Lerntempi bei den Lernenden (vgl. Abschnitt 4.4 bzw. Baldwin und Kujis, 2004, S. 285; Perkins u. a. 1989, S. 261 sowie Schubert und Schwill, 2004, S. 277). Differenzierung kann bereits bei der Entwicklung von Unterrichtsaufgaben vorbereitet werden.[71] Die Aufgaben 4, 10, 14, 16, 21, 38, 39, 40, 45, 60, 62 und 70 der Sammlung

[69] Trotz der im Abschnitt 4.3 dargestellten Argumentation zur teilweise geringen Motivation von Aufgaben aus dem Bereich der Mathematik, wurden die 13 Unterrichtsaufgaben, die Bezüge zu diesem Fach haben, nicht aus der Sammlung entfernt, um eine Vielfalt im Unterricht zu ermöglichen.

[70] Weitere Kriterien kreativen Informatikunterrichts wurden von Romeike (2007b) vorgestellt und können bei der Erstellung von Aufgaben nützlich sein.

[71] Humbert (2005, S. 75)definiert schulische Differenzierung wie folgt: *Schulische Differenzierung wird mit dem Ziel vorgenommen den individuellen Kompetenzen, Interessen und dem objektiven Bedarf der Schülerinnen und Schüler Rechnung zu tragen. Sie wird umgesetzt, in dem die Schülerinnen nach ausgewählten Kriterien in Lerngruppen [...] eingeteilt werden.* Humbert (2005, S. 75) unterscheidet zwischen äußerer Differenzierung, bei der die Lernenden in getrennten Gruppen unterrichtet werden und innerer Differenzierung (oder Binnendifferenzierung) bei der die Lernenden innerhalb des Unterrichts in bestimmter Weise gruppiert werden. Brinda und Mägdefrau (2008, S. 16 f.) geben eine Übersicht über verschiedene

| Bezugsfach | Nummer der Unterrichtsaufgaben mit Beziehungen zum jeweiligen Fach (vgl. Kohl, 2008a, S. 18 ff.) |
|---|---|
| Mathematik | 7, 9, 16, 24, 28, 38, 40, 42, 44, 50, 60, 69, 70 |
| Musik | 18, 29, 30, 31, 65, 66, 67 |
| Deutsch | 26, 43 |
| Geschichte | 46 |

Tabelle 7: Beziehungen der Unterrichtsaufgaben zu anderen Fächern

können neben den oben genannten offenen Aufgaben zur Binnendifferenzierung im Unterricht eingesetzt werden.

Die Unterrichtsaufgaben wurde den Lehrerinnen und Lehrern, die an der Hauptuntersuchung teilnahmen, sowohl in gedruckter als auch in digitaler Form übergeben, so dass sie gegebenenfalls an die speziellen Bedürfnisse des jeweiligen Unterrichts angepasst werden konnten (vgl. Abschnitt 8.1). Den Lehrkräften sollte die Möglichkeit gegeben werden, Teilaufgaben zu entfernen und hinzuzufügen sowie Aufgabenblätter mit Aufgaben unterschiedlicher Stufen selbst zu erstellen und zur Differenzierung einzusetzen. Dafür wurde die digitale Version der Aufgabensammlung in einem Dateiformat übergeben, dass leicht bearbeitet werden konnte und auch Möglichkeiten zur Formatierung gab.

Nachfolgend werden exemplarisch zwei Unterrichtsaufgaben vorgestellt:

- Abbildung 25 zeigt eine Unterrichtsaufgabe der Stufe I zur Komponente C.

  In der Aufgabe ist ein in Pseudocode formulierter Algorithmus gegeben, mit dessen Hilfe das neue Gehalt eines Mitarbeiters nach einer Gehaltserhöhung berechnet wird. Für zwei gegebene Wert soll die Ausgabe des Programms analysiert werden. Anschließend ist der Algorithmus umzusetzen, zu testen und so zu modifizieren, dass die entsprechende Gehaltserhöhung auch mit ausgegeben wird.

- Abbildung 26 zeigt eine Unterrichtsaufgabe der Stufe II zur Komponente D.

  In der Aufgabe soll ein Programm entworfen, umgesetzt und getestet werden, das zur Eingabe des Alters und des Geschlechts eines Kindes passende Geschenkvorschläge macht.

Eine Musterlösung zu der in Abbildung 25 dargestellten Aufgabe ist im Anhang D dargestellt. Ein programmiersprachliche Lösung zu der Aufgabe in Abbildung 26 findet sich in Abbildung 4 auf Seite 55.

Möglichkeiten der Differenzierung.

**Unterrichtsaufgabe** **Stufe I** **Komponente C**

***Aufgabe (Gehaltserhöhung)***

Die Softwarefirma IT-Triple konnte ihre Effizienz durch den Umstieg auf eine visuelle Programmiersprache steigern. Der Chef möchte deshalb seine Mitarbeiter belohnen. Die Gehälter aller Mitarbeiter sollen um 4 %, mindestens aber um 80 € im Monat erhöht werden.

Der Algorithmus zur Berechnung dieser Gehaltserhöhung ist gegeben:

ALGORITHMUS Gehaltserhoehung

Lege die Variable *altesgehalt* vom Typ Integer an.
Fordere den Benutzer mit "Geben Sie bitte ihr altes Gehalt ein." auf, eine Zahl einzugeben.
Speichere die eingegebene Zahl in die Variable *altesgehalt*.

Wenn die Bedingung *((altesgehalt*104 DIV 100)-altesgehalt > 80)* wahr ist, mache Folgendes:
Gib "Neues Gehalt:" und danach den Wert von (*altesgehalt*104 DIV 100*) aus.
Wenn die Bedingung (*(altesgehalt*104 DIV 100)-altesgehalt > 80*) falsch ist, mache Folgendes:
Gib "Neues Gehalt:" und danach den Wert von *altesgehalt+80* aus.

a) Welche Ausgaben liefert der Algorithmus für das Gehalt 1000 € und für das Gehalt 10000 €?

Ausgabe:

b) Lesen Sie den Algorithmus, übertragen Sie ihn in ein Programm und testen Sie es.

c) Finden Sie durch mehrfaches Testen heraus, ab welcher Gehaltsgrenze mehr als 80 € Gehaltserhöhung fällig ist.

d) Ergänzen Sie das Programm so, dass jeweils die Gehaltserhöhung mit ausgegeben wird.

Abbildung 25: Eine Unterrichtsaufgabe Stufe I Komponente C

**Unterrichtsaufgabe** **Stufe II** **Komponente D**

***Aufgabe (Kindergeburtstag)***

Ein Programm zur Unterstützung von Großeltern bei Kindergeburtstagen soll entwickelt werden. Zur Eingabe des Alters und des Geschlechts eines Kindes soll das Programm einen passenden Geschenkvorschlag ausgeben.
Entwerfen Sie schriftlich einen Algorithmus für dieses Problem.
Setzen Sie das gewünschte Programm benutzungsfreundlich um, das für Kinder der Alterspannen 0-6, 7-12 und 13-18 beider Geschlechter Geschenkvorschläge macht!
Testen Sie das Programm für Mädchen und Jungen aller drei Altersstufen.

Abbildung 26: Eine Unterrichtsaufgabe Stufe II Komponente D

## 6.4 Analyse einer Aufgabe mit einer Bewertungsmatrix

Eine Möglichkeit eine Aufgabe genauer zu analysieren ist die Angabe der Querbezüge zwischen den Kompetenzen der Inhalts- und Prozessbereiche der GI-Empfehlungen. Diese Analyse kann mithilfe der von Puhlmann und Friedrich (2007, S. 17) vorgestellten Bewertungsmatrix erfolgen: *Die Matrix enthält eine Zelle für jede Kombination eines Inhalts- mit einem Prozessbereich. Dies entspricht der Vorstellung, dass Inhalts- und Prozesskompetenzen nicht getrennt voneinander erworben werden, sondern dass eine Kompetenz darin besteht, mit den Inhalten in einer Arbeitsweise umzugehen, die in den Prozessbereichen beschrieben ist* (vgl. Puhlmann und Friedrich, 2007, S. 17). Nachfolgend werden exemplarisch für eine Aufgabe die Beziehungen zwischen den Kompetenzen der Inhalts- und Prozessbereiche der GI-Empfehlungen in einer Bewertungsmatrix eingeordnet:

Abbildung 27 zeigt die Unterrichtsaufgabe „Binäruhr“, die den Lehrkräften in der Hauptuntersuchung dieser Arbeit zur Verfügung gestellt wurde (vgl. Kohl, 2008a, S. 64 f.). Die Aufgabe ist der Komponente C auf Stufe III zugeordnet. Ein Algorithmus zur Umsetzung einer Binäruhr ist als Pseudocode gegeben. Nachdem sich die Schülerinnen und Schüler durch eine Internetrecherche über Binäruhren informiert haben (a), sollen sie den gegebenen Algorithmus mithilfe von Durchlauftabellen und einer grafischen Veranschaulichung der Ausgabe analysieren (b und c). Anschließend sollen sie den gegebenen Pseudocode in ein Programm umsetzen und dieses testen (d). Weiterhin soll das Programm, welches Sekunden binär ausgibt, um die binäre Darstellung von Minuten erweitert werden (e). In der offen gestellten Teilaufgabe (f) soll das Programm in irgendeiner Weise verbessert oder erweitert werden.[72]

Nachfolgend wird für die Teilaufgaben jeweils untersucht, welche Kompetenzen der Inhalts- und Prozessbereiche der GI-Empfehlungen gefordert sind und wie diese miteinander in Beziehung stehen:

- In Teilaufgabe a verwenden die Schülerinnen und Schüler einen Internetdienst um sich eine Anwendung bzw. ein Informatiksystem selbstständig zu erschließen (vgl. Inhaltsbereich „Informatiksysteme“ GI, 2008, S. 39 f.). Dabei ist nicht angegeben, dass die gefundenen Informationen in irgendeiner Weise zusammengefasst werden sollen. Beim Vergleich der Lösungen im Unterricht wäre es möglich, die Schülerinnen und Schüler das Konzept einer Binäruhr mündlich oder schriftlich darstellen zu lassen (vgl. Prozessbereich

[72] Es wäre möglich, einen Countdown zu implementieren, vom Benutzer eine Startuhrzeit abzufragen oder in einer Art „Quiz“ verschiedene (zufällige) binär dargestellte Zeiten vom Benutzer als Zahlen abzufragen.

Abbildung 27: Eine Unterrichtsaufgabe Stufe III Komponente C

„Kommunizieren und Kooperieren“ GI, 2008, S. 56).

- In den Teilaufgaben b und c lesen die Schülerinnen und Schüler den formal dargestellten Algorithmus und ermitteln das Ergebnis des algorithmischen Ablaufs mit Hilfe von Durchlauftabellen (vgl. Inhaltsbereich „Algorithmen“ GI, 2008, S. 33). Dabei verwenden sie unter anderem auch arithmetische Operationen auf dem Datentyp Zahl (vgl. Inhaltsbereich „Informationen und Daten“ GI, 2008, S. 29 f.). Im Rahmen der Analyse des Algorithmus wird ein angemessenes Modell erarbeitet, das auch für die weiteren Teilaufgaben nützlich ist (vgl. Prozessbereich „Modellieren und Implementieren“ GI, 2008, S. 47). In der Teilaufgabe c nutzen die Schülerinnen und Schüler Grafiken, um sich den gegebenen informatischen Sachverhalt selbstständig zu erarbeiten (vgl. Prozessbereich „Darstellen und Interpretieren“ GI, 2008, S. 56).
- In der Teilaufgabe d muss der gegebene Algorithmus in ein lauffähiges Programm umgesetzt werden (vgl. Inhaltsbereich „Algorithmen“ GI, 2008, S. 33). In den Teilaufgaben e und f wird das so entstandene Programm modifiziert (vgl. Inhaltsbereich „Algorithmen“ GI, 2008, S. 33). Die Schülerinnen und Schüler geben die Problemlösung dabei jeweils in einer Programmiersprache an (vgl. Inhaltsbereich „Sprachen und Automaten GI, 2008, S. 36). Bei der Implementierung verwenden sie die algorithmischen Grundbausteine (vgl. Prozessbereich „Modellieren und Implementieren“ GI, 2008, S. 47). Bei der Erweiterung der Binäruhr in Teilaufgabe f planen die Schülerinnen und Schüler einen neuen Arbeitsablauf bzw. eine neue Handlungsfolge und nutzen dabei Analogien zu bekannten Inhalten und Vorgehensweisen (vgl. Prozessbereich „Strukturieren und Vernetzen“ GI, 2008, S. 51 f.).

Bei der Analyse der Aufgabe „Binäruhr“ wurde festgestellt, dass Kompetenzen aus vier Inhalts- und vier Prozessbereichen gefordert sind. Die Kombinationen der Kompetenzen aus den Inhalts- und Prozessbereichen sind in Tabelle 8 in Form einer Bewertungsmatrix zusammengefasst. Entsprechend der Darstellungen von Puhlmann und Friedrich (2007, S. 17 f.) sind in Klammern jeweils die betroffenen Teilaufgaben angegeben. Außerdem ist die im Zentrum der Teilaufgabe stehende Kombination von Inhalts- und Prozesskompetenz durch zwei Kreuze markiert. Den in Komponente C des Kompetenzmodells geforderten Kompetenzen entsprechend, ist der größte Teil der Aufgabe dem Inhaltsbereich „Algorithmen“ und der Prozesskompetenz „Modellieren und Implementieren“ zuzuordnen.

Da sich die entwickelten Aufgaben auf die im Kompetenzmodell geforderten Kompetenzen beziehen, weisen sie häufig ähnliche Beziehungen mit den Inhalts- und Prozessbereichen der GI-Empfehlungen auf, wie die in der jeweiligen Komponente und Stufe geforderten Kompetenzen des Modells (vgl. Abschnitt 5.5 bzw.

| | Modellieren und Implementieren | Begründen und Bewerten | Strukturieren und Vernetzen | Kommunizieren und Kooperieren | Darstellen und Interpretieren | sonst |
|---|---|---|---|---|---|---|
| Informationen und Daten | $X_{(b, c, d, e, f)}$ | | | | | |
| Algorithmen | $XX_{(b, c, d, e, f)}$ | | $X_{(f)}$ | | $X_{(c)}$ | |
| Sprachen und Automaten | $X_{(d, e, f)}$ | | | | | |
| Informatiksysteme | | | | $X_{(a)}$ | | $XX_{(a)}$ |
| Informatik, Mensch und Gesellschaft | | | | | | |
| sonst | | | | | | |

Tabelle 8: Bewertungsmatrix für die in Abbildung 27 dargestellte Aufgabe „Binäruhr“

Anhang C). Teilaufgabe (a) zeigt, dass – zumindest in den Unterrichtsaufgaben – vereinzelt zusätzliche Kompetenzen gefordert sein können, die nicht mit den Kompetenzen des Kompetenzmodells „Algorithmen“ in Verbindung stehen. Die Arbeitsweisen (Prozesskompetenzen) mit denen die Inhalte (Inhaltskompetenzen) erworben werden, können sich in den einzelnen Aufgaben unterscheiden. Für eine detaillierte Analyse der in einer Aufgabe geforderten Kompetenzen erscheint die von Puhlmann und Friedrich (2007, S. 17) vorgestellte Bewertungsmatrix geeignet. Wenn zu verschiedenen Inhaltsbereichen Aufgaben existieren, die verschiedene Prozesskompetenzen erfordern, so kann die Bewertungsmatrix bei der Auswahl von Aufgaben für den Unterricht durchaus hilfreich sein um gezielt Kompetenzen zu fördern.

## 6.5 Anforderungen an die Entwicklung von Aufgaben zu einem Kompetenzmodell

Im Folgenden werden aus den Erfahrungen des dargestellten Entwicklungsprozesses gewonnene Anforderungen an die Entwicklung von Aufgaben zu einem Kompetenzmodell in der Informatik vorgestellt. Dabei wird auch auf die im Abschnitt 3.3 dargestellten Erkenntnisse eingegangen. Die Anforderungen werden jeweils näher erläutert und an Beispielen veranschaulicht.

**1. Zwischen Beispiel-, Unterrichts- und Testaufgaben unterscheiden** Zum Herausbilden und zum Überprüfen von Kompetenzen werden unterschiedliche Aufgaben benötigt (vgl. Abschnitt 3.3 bzw. Herper, 2005, S. 74 oder Leuders, 2006, S. 81). Es sollte also zwischen Testaufgaben und Unterrichtsaufgaben unterschieden werden. Zusätzlich entwickelte Beispielaufgaben verdeutlichen, was in den Komponenten und Stufen des Kompetenzmodells gefordert wird und wie ein Kompetenztest aufgebaut sein kann (vgl. Abschnitt 3.3 bzw. Klieme u. a., 2003, S. 121 f.). Durch den Einsatz solcher Beispielaufgaben im Unterricht wird eine Einschätzung der erworbenen Kompetenzen ermöglicht und die bewusste, eigenständige Wahl der Stufe in einem Kompetenztest kann vorbereitet werden.

Bei der Entwicklung wurde zwischen Beispiel-, Unterrichts- und Testaufgaben unterschieden. Die Abbildungen 23 und 24 zeigen Beispielaufgaben. Unterrichtsaufgaben sind in den Abbildungen 25, 26 und 27 dargestellt. Die Abbildungen 21 und 22 zeigen Testaufgaben. Die Sammlung von zwölf Beispiel-, 70 Unterrichts- und zwölf Testaufgaben ist im Arbeitsmaterial zur Hauptuntersuchung verfügbar (vgl. Kohl, 2008a, S. 8 ff.).

**2. Beispiel- und Testaufgaben anhand der im Kompetenzmodell geforderten Kompetenzen konstruieren und in klare, überschaubare Teilaufgaben untergliedern** Beispiel- und Testaufgaben sollten sich möglichst genau auf die in einem Modell angegebenen Kompetenzen beziehen. Das Unterteilen der Aufgaben in überschaubare Teilaufgaben ermöglicht eine präzise Erfassung bzw. Verdeutlichung einzelner Kompetenzen und trägt gleichzeitig zur Strukturierung der Aufgaben bei. Testaufgaben sollten weiterhin so formuliert sein, dass allen Beteiligten klar ist, was zur richtigen Lösung zu tun ist. Konkrete Punktangaben für Teilaufgaben und ein Erwartungshorizont bzw. eine Musterlösung können bei der Korrektur hilfreich sein.

In Komponente B sind auf Stufe III folgende drei Kompetenzen angegeben:

Schülerinnen und Schüler

1. erklären die algorithmischen Grundbausteine wie Variablen, Wertzuweisungen, Verzweigungen, Wiederholungen und Unterprogramme mit Parametern und wenden diese Erklärungen an
2. stellen die algorithmischen Grundbausteine in verschiedenen Darstellungsformen dar und wechseln zwischen Darstellungsformen
3. verwenden verschiedene Datentypen

Die in Abbildung 23 dargestellte Beispielaufgabe zur Komponente B auf Stufe III geht in den drei Teilaufgaben auf genau diese drei Kompetenzen ein:

1. Nennen Sie zwei Ihnen bekannte Parameterarten und erklären Sie die Funktionsweise der Parameterübergabe.
2. Überführen Sie die gegebenen Beispiele in eine andere Darstellungsform.
3. Geben Sie für zwei Datentypen jeweils zwei Werte und zwei Operationen an.

Die Beispiel- und Testaufgaben wurden anhand der Kompetenzen des Modells konstruiert und in entsprechende Teilaufgaben untergliedert.[73] Außerdem wurde jeder Aufgabe eine Punktzahl zugeordnet. Im Rahmen der Voruntersuchung war ein Erwartungshorizont bei der Korrektur hilfreich.

**3. Vielfältige, abwechslungsreiche Unterrichtsaufgaben zusammenstellen** Eine Sammlung, die motivierende, fächerübergreifende, offene, kreativitätsfördernde und differenzierende Aufgaben enthält, kann in verschiedenen Unterrichtsphasen eingesetzt werden, sei es für eine Gruppenarbeit, eine Leistungsüberprüfung, eine Hausaufgabe, eine Einführung in ein neues Thema oder anderes. Außerdem kann eine vielfältige und abwechslungsreiche Aufgabensammlung verschiedene Lehrer- und Lernertypen ansprechen und einen Beitrag zur Weiterentwicklung der Aufgabenkultur im Informatikunterricht leisten.

Die entwickelte Sammlung enthält 70 Unterrichtsaufgaben zu verschiedenen Themen (vgl. Kohl, 2008a, S. 18 ff.). Im Abschnitt 6.3 wurden Beispiele für motivierende, fächerübergreifende, offene, kreativitätsfördernde und differenzierende Aufgaben dieser Sammlung angegeben.

[73] Da auch andere Kriterien wie z. B. Bearbeitungsdauer von Bedeutung waren, konnten bei einzelnen Aufgaben nicht alle Kompetenzen abgebildet werden.

**4. Die Unterrichtsaufgaben in einem digitalen, einfach modifizierbaren Format bereitstellen** Nachfolgend wird davon ausgegangen, dass zumindest einige der bereitgestellten Unterrichtsaufgaben je nach Einsatz und Zielstellung modifiziert werden. Darum ist es sinnvoll, Lehrpersonen die entwickelten Aufgaben in digitaler Form zur Verfügung zu stellen. Der Dateityp, in dem die digitale Version der Aufgabensammlung gespeichert ist, sollte von allen Lehrerinnen und Lehrern einfach zu bearbeiten sein, denn so können die Lehrkräfte die Aufgaben leicht an verschiedene Situationen anpassen bzw. zu Arbeitsblättern zusammenfassen. Hierfür ist auch eine Unterteilung in kleinschrittige Teilaufgaben, die gegebenenfalls leicht entfernt werden können, sinnvoll. Die digitale Bereitstellung von Aufgaben schließt eine parallele Weitergabe in gedruckter Form nicht aus.

Wie bereits im Abschnitt 6.3 erläutert wurde, wurden die Unterrichtsaufgaben in der Hauptuntersuchung dieser Arbeit parallel in gedruckter und digitaler Form an die Lehrerinnen und Lehrer verteilt (vgl. auch Abschnitt 8.1). Dabei wurde ein Dateiformat gewählt, das Veränderungen und Umformatierungen ermöglichte. Die Unterrichtsaufgaben wurden entweder direkt in Teilaufgaben unterteilt oder es wurden einzelne Sätze zu den geforderten Kompetenzen formuliert, um ein leichtes Modifizieren zu ermöglichen (vgl. hierzu die Abbildungen 25, 26 und 27).

**5. In den Aufgaben nicht auf spezielle Software, Programmiersprachen oder Entwicklungsumgebungen eingehen** Bei der Entwicklung von Aufgaben gilt es, die Vielfalt der im Unterricht eingesetzten Systeme zu unterstützen, um so den Informatikunterricht nicht unnötig einzuschränken. Wenn in einer Aufgabe spezifische Angaben zu Software, Programmiersprachen oder Entwicklungsumgebungen gemacht werden, so wird diese auch größtenteils von den Lehrkräften eingesetzt werden, die das entsprechende System verwenden. Dadurch wird die Verbreitung der Aufgabensammlung eingeschränkt. Bei der Entwicklung von Testaufgaben für größere Schülerpopulationen ist die Alternative zu einer systemunabhängigen Formulierung eine parallele Entwicklung von Aufgaben für alle verwendeten Systeme.

Algorithmen sind in den hier vorgestellten Aufgaben stets in programmiersprachenunabhängigem Pseudocode angegeben (vgl. Kohl, 2008a, S. 8 ff.). Trotzdem können die im Modell geforderten Kompetenzen überprüft werden. Als Beispiel sei auf die in den Abbildungen 24, 25 und 27 dargestellten Aufgaben verwiesen. Diese können problemlos unter Verwendung unterschiedlicher Programmiersprachen bzw. Entwicklungsumgebungen gelöst werden.

**6. Aufgaben vor einem größeren Einsatz erproben** Bevor Aufgaben einer umfangreichen Untersuchung unterzogen werden, sollten sie in kleinem Rahmen erprobt werden. So können Probleme, die beim Lösen der Aufgaben auftreten, frühzeitig erkannt und beseitigt werden. Außerdem ist es so möglich, spezielle Kriterien wie Bearbeitungsdauer oder leichte Korrigierbarkeit zu überprüfen. Die Personen, die die Aufgaben erproben, sollten möglichst der Zielgruppe entsprechen, bei der die Aufgaben eingesetzt werden.

Die Aufgaben wurden in einer Voruntersuchung mit zwei 8. Klassen und vier 10. Klassen im Unterricht erprobt (vgl. Kapitel 7). Eine Einschätzung der Beispiel- und Unterrichtsaufgaben erfolgte durch die Lehrerinnen und Lehrer sowie durch Beobachtungen des Autors bei Unterrichtsbesuchen (vgl. Abschnitt 7.4). Der Kompetenztest wurde in den sechs Klassen bearbeitet. Die Ergebnisse wurden ausgewertet, dabei wurden typische Fehler und interessante Lösungen für jede Teilaufgabe protokolliert (vgl. Abschnitt 7.5). Aufgrund der gewonnenen Erfahrungen konnten einige Aufgaben vor der Hauptuntersuchung weiterentwickelt werden.

## 6.6 Zusammenfassung

In diesem Kapitel wurden zunächst dem Untersuchungsdesign entsprechende Festlegungen zu den im Modell offen formulierten Kompetenzen getroffen, die es ermöglichten, einen Kompetenztest und weitere Aufgaben zu erstellen, die in der Untersuchung von allen Lehrpersonen eingesetzt werden konnten. Anschließend wurden Beispielaufgaben zum Verdeutlichen, Unterrichtsaufgaben zum Erwerben und Testaufgaben zum Überprüfen von Kompetenzen beschrieben und entsprechende Beispiele angegeben. Zur genaueren Betrachtung der Beziehungen zwischen Inhalts- und Prozesskompetenzen wurde eine Aufgabe exemplarisch mithilfe einer Bewertungsmatrix analysiert. Schließlich wurden in einer Reflexion Anforderungen für die Entwicklung von Aufgaben für ein Kompetenzmodell im Informatikunterricht zusammengetragen.

# 7 Voruntersuchung an sechs Thüringer Schulen

*Those that Hobgoblin call you and sweet Puck,*
*You do their work, and they shall have good luck.*
(William Shakespeare, A Midsummer Night's Dream, S. 26)

Im Folgenden wird die Voruntersuchung an sechs Thüringer Schulen im Schuljahr 2006/2007 beschrieben, in der erste Versionen des Kompetenzmodells und der zugehörigen Aufgaben unter Verwendung von Puck eingesetzt wurden. Ziel dieser explorativen Untersuchung war es, herauszufinden, ob mithilfe der bereitgestellten Materialien, entsprechend der Darlegungen im Abschnitt 3.4, kompetenzorientierter Informatikunterricht umgesetzt werden kann. Aufgrund der Ergebnisse sollten die Materialien weiter entwickelt, erste Antworten zu den Forschungsfragen zwei und drei gewonnen und Konsequenzen für die Durchführung der Hauptuntersuchung abgeleitet werden. Diesen Zielstellungen entsprechend wurden verschiedene Forschungsmethoden in einem Untersuchungsdesign kombiniert, das im Abschnitt 7.1 vorgestellt wird. Nach dem Beschreiben der Stichprobe und des durchgeführten Unterrichts im Abschnitt 7.2, wird in den Abschnitten 7.3 bis 7.6 auf die Erfahrungen mit dem Kompetenzmodell, den Beispiel- und Unterrichtsaufgaben, dem Kompetenztest und der visuellen Programmiersprache Puck eingegangen. Die Ergebnisse dieser Voruntersuchung werden im Abschnitt 7.7 eingeordnet.

## 7.1 Untersuchungsdesign

Insgesamt beteiligten sich sechs Schulen aus Thüringen mit je einer Lehrkraft und einer Klasse bzw. Arbeitsgemeinschaft an der Voruntersuchung. Schulen und Lehrpersonen werden nachfolgend anonymisiert mit A, B, C, D, E und F bezeichnet.

### Vorbereitungsveranstaltungen

Um die Lehrenden auf den Unterricht mit dem entwickelten Materialien vorzubereiten, wurden vom Autor im Zeitraum Mai bis September 2006 drei Veranstaltungen an der Universität Jena durchgeführt. Beim ersten Zusammentreffen wurde die

Untersuchung erläutert, das Programmiersystem Puck wurde vorgestellt und in einer gemeinsamen Arbeitsphase ausprobiert. In der zweiten Veranstaltung wurden das bereitgestellte Kompetenzmodell, zugehörige Beispielaufgaben und einige spezielle Features des Puck-Systems präsentiert. Beim dritten Zusammentreffen wurden den Lehrpersonen die Unterrichtsaufgaben übergeben und anschließend unter Verwendung von Puck ausprobiert. Zwischen der zweiten und der dritten Veranstaltung besuchte der Autor alle Schulen, um bei der Installation von Puck aufgetretene Fragen zu klären und ein erstes leitfadenorientiertes Interview mit den Lehrpersonen durchzuführen.

### Verwendete Materialien

In der Untersuchung wurde eine erste Version des im Kapitel 5 vorgestellten Kompetenzmodells eingesetzt. Das Modell hatte zu diesem Zeitpunkt noch einen Umfang von zwei Seiten und war untergliedert in die Komponenten „Entscheiden“, „Grundbausteine erklären“, „Analysieren“, „Modifizieren“ und „Entwickeln“. Die im Abschnitt 5.3 beschriebenen typografischen Hervorhebungen der auf einer Stufe neu hinzugekommenen Kompetenzen fehlten noch. Außerdem erhielten die Lehrpersonen eine Sammlung von 53 programmiersprachenunabhängig formulierten Unterrichtsaufgaben zum Analysieren, Modifizieren und Entwickeln von Algorithmen. Diese waren noch nicht in die Komponenten und Stufen des Kompetenzmodells eingeordnet, sondern anhand der in der Musterlösung verwendeten algorithmischen Grundbausteine in Gruppen eingeteilt. Des Weiteren erhielten die Lehrpersonen zu Beginn der Untersuchung Beispielaufgaben und eine schriftliche Einführung in die Programmierung mit Puck (vgl. Kohl, 2006a). Zum Ende der Untersuchung wurden außerdem Testaufgaben bereitgestellt. Den drei Stufen entsprechend wurden jeweils drei Beispiel- und Testaufgaben zum „Entscheiden“, „Grundbausteine erklären“, „Analysieren und Modifizieren“ sowie zum „Entwickeln“ bereitgestellt, die inhaltlich größtenteils den im Abschnitt 6.2 vorgestellten Aufgaben entsprachen (vgl. Kohl, 2008a, S. 8 ff.). Sie waren zu diesem Zeitpunkt noch platzsparend in einem zusammenhängenden Fließtext formuliert. Dadurch war es möglich, die Aufgaben zu allen Komponenten einer Stufe auf einem Aufgabenblatt zusammenzufassen. Für die Durchführung des Kompetenztests wurde eine Bearbeitungszeit von 90 Minuten empfohlen.

### Forschungsmethodik

**Interviews am Anfang des Halbjahres** Die Lehrkräfte wurden vor Beginn des Halbjahres zu Puck, dem Kompetenzmodell und ihren bisherigen Erfahrun-

gen beim Einstieg in die Programmierung befragt.[74]

**Schülerfragebogen** Das Vorwissen der Schülerinnen und Schüler wurde mit einem Fragebogen erhoben, um eventuell bestehende Einflüsse dieses Vorwissens auf den Unterricht und die Ergebnisse des Kompetenztests beschreiben und einordnen zu können.[75]

**Unterrichtsbeobachtung** Der Unterricht an allen Schulen wurde zumindest teilweise vom Autor der Arbeit beobachtet und protokolliert.[76]

**Auswertung des Kompetenztests** Die Ergebnisse der Schülerinnen und Schüler beim Kompetenztest wurden vom Autor der Arbeit ausgewertet und analysiert.

**Interviews am Ende des Halbjahres** Nachdem der Unterricht mit den vorgegebenen Materialien abgeschlossen war, wurden die Lehrkräfte einzeln an der Universität Jena befragt. Den Interviews lag ein Leitfaden mit 30 Fragen zugrunde, der durch die Kategorien „Fragen zum Unterricht“, „Fragen zu Kompetenzmodell und Aufgabensammlung“ und „Fragen zu Puck“ strukturiert wurde.[77]

Durch die Kombination der verschiedenen Forschungsmethoden war es möglich, den durchgeführten Unterricht unter verschiedenen Aspekten zu untersuchen. Die Ergebnisse werden nachfolgend vorgestellt.

[74] Bei dem Lehrer an Schule A konnte das Interview aus äußeren Gründen nicht stattfinden.

[75] Im Fragebogen wurden folgende Daten der Schülerinnen und Schüler erhoben: Schule, Name, Geschlecht, Alter, Möglichkeit der Computernutzung zu Hause, durchschnittliche Dauer der Computernutzung pro Woche insgesamt, durchschnittliche Dauer der Computernutzung pro Woche im Rahmen des Unterrichts, Einschätzung der eigenen Fähigkeiten im Umgang mit dem Computer, häufig benutzte Computeranwendungen, Erwartungen an das Fach Informatik, Erfahrungen mit Programmiersprachen. In Schule B wurde der Fragebogen aus organisatorischen Gründen nicht bearbeitet.

[76] Während in den Schule B, C und F mehr als die Hälfte der Unterrichtsstunden beobachtet wurden, konnte in den Schulen A, D und E aus organisatorischen Gründen nur jeweils ca. ein Viertel der Unterrichtszeit hospitiert werden.

[77] Das Leitfadeninterview wird von Bortz und Döring (2006, S. 314) wie folgt beschrieben: *Das Leitfadeninterview ist die gängigste Form qualitativer Befragung. Durch den Leitfaden und die darin angesprochenen Themen erhält man ein Gerüst für Datenerhebung und Datenanalyse, das Ergebnisse unterschiedlicher Interviews vergleichbar macht. Dennoch lässt es genügend Spielraum, spontan aus der Interviewsituation heraus neue Fragen und Themen einzubeziehen oder bei der Interviewauswertung auch Themen herauszufiltern, die bei der Leitfadenkonzeption nicht antizipiert wurden.*

## 7.2 Beschreibung der Stichprobe und des durchgeführten Unterrichts

Eine Übersicht über die Untersuchung mit den beteiligten Schulen und der jeweiligen Jahrgangsstufe, Unterrichtsform und Schüleranzahl gibt Tabelle 9.[78] Die dort dargestellten Unterrichtsformen legen die Vermutung nahe, dass die Schülerinnen und Schüler größtenteils freiwillig am Informatikunterricht teilnahmen bzw. sich zumindest aktiv für diesen entschieden hatten.[79] Die in Tabelle 10 dargestellten Angaben der Lehrkräfte zu ihrer Berufserfahrung zeigen, dass nicht nur Berufsanfänger bzw. nicht nur erfahrene Lehrende an der Untersuchung teilgenommen haben.

Am Unterricht nahmen insgesamt 20 Schülerinnen und 54 Schüler teil. Nachfolgend werden die Angaben der 20 Schülerinnen und 34 Schüler betrachtet, die den zu Beginn des ersten Halbjahres ausgeteilten Fragebogen vollständig ausfüllten.[80]
Die Lernenden waren zwischen 13 und 17 Jahren alt (MEAN=14,7 (SD=1,3)). Alle hatten die Möglichkeit, zu Hause einen Computer zu nutzen und nahmen diese auch war.[81] Der Großteil der Schülerinnen und Schüler erwartete im Informatik-

[78] Die Schulen A, D und E gehören zur Gruppe der Thüringer Medienschulen, die vorbildlich mit Medientechnik ausgestattet sind und die bereit sind, Ergebnisse ihrer medienpädagogischen Profilierung anderen Thüringer Schulen zugänglich zu machen (vgl. Thüringer Kultusministerium, 2008).

[79] Ca. die Hälfte der Lernenden besuchte Schule C, dort gehört der Informatikunterricht zum Profilzweig Wirtschaft-Informatik. Das bedeutet, dass die Schülerinnen und Schüler anstatt einer dritten Fremdsprache pro Woche zwei Stunden Informatik, zwei Stunden Wirtschaft/Recht und eine Stunde Mathematik zusätzlich zum normalen Unterricht haben. An Schule F konnte Informatik als eines von mehreren Wahlpflichtfächern gewählt werden. An den anderen vier Schulen wurde der Unterricht als zusätzliches Wahlfach bzw. als Arbeitsgemeinschaft angeboten. Einige Schülerinnen und Schüler besuchten den Unterricht im Wahlfach oder in der Arbeitsgemeinschaft nicht bis zum Ende des Halbjahres. Insgesamt war der Anteil der motivierten und am Fach Informatik interessierten Schülerinnen und Schülern bei der Durchführung des Kompetenztests vermutlich recht hoch.

[80] Von den 13 Schülern an Schule B wurde der Fragebogen aus organisatorischen Gründen nicht ausgefüllt. An den anderen Schulen waren insgesamt sieben Schülerinnen und Schüler in der entsprechenden Unterrichtsstunde nicht anwesend.
Bei der Auswertung des Schülerfragebogens werden nachfolgend beim Vorliegen von Intervallskalenniveau zur Darstellung der deskriptiven Statistik Mittelwert (MEAN) und Standardabweichung (SD) angegeben.

[81] Bei der Dauer der durchschnittlichen Computernutzung pro Woche konnte jeweils ca. ein Viertel der Schülerinnen und Schüler den Gruppen 1-5 Stunden, 6-10 Stunden, 11-15 Stunden und mehr als 16 Stunden zugeordnet werden (MEAN=15,3 (SD=13,2)). Vier Schüler gaben an, durchschnittlich pro Woche mehr als 48 Stunden am Computer zu verbringen. Es ist offen, ob Schülerinnen und Schüler diesen Alters die tatsächliche Dauer der Computernutzung richtig einschätzen können.

unterricht das Programmieren zu erlernen, weiterhin wurde von ca. einem Drittel der Ausbau der Fähigkeiten im Umgang mit dem PC erwartet.[82] Die Mehrzahl der Lernenden schätzte die eigenen Fähigkeiten im Umgang mit dem PC als „durchschnittlich“ bzw. „gut“ ein.[83] 24 der 54 Schülerinnen und Schüler gaben an, schon einmal mit einer Programmiersprache gearbeitet zu haben. Sechs dieser 24 nannten lediglich die Auszeichnungssprache HTML als Programmiersprache. Von den restlichen 18 wurden unterschiedliche Sprachen genannt.[84] Die Angaben zu den mit der Programmiersprache gelösten Aufgaben legten allerdings lediglich bei drei Schülern die Vermutung nahe, dass bereits relevante Programmierkenntnisse vorhanden waren.
Die Daten zeigen, dass alle Schülerinnen und Schülern schon Erfahrungen im Umgang mit dem Computer gemacht hatten. Die überwiegend positive Selbsteinschätzung der Fähigkeiten im Umgang mit dem Computer und die Anzahl der

Die Computernutzung im Rahmen des Unterrichts machte nur einen kleinen Teil der durchschnittlichen Nutzungsdauer aus (MEAN=1,7 (SD=1,8)).
5,8 % der Schülerinnen und Schüler gaben an, den Computer gar nicht im Unterricht zu nutzen. 46,2 % gaben an, den Computer durchschnittlich in bis zu einer Stunde pro Woche im Unterricht zu nutzen. 32,7 % gaben an, den Computer durchschnittlich in mehr als einer und bis zu zwei Stunden pro Woche im Unterricht zu nutzen. 15,4 % gaben an, den Computer durchschnittlich in mehr als zwei Stunden pro Woche im Unterrichts zu nutzen.
Folgende Angaben machten die Schülerinnen und Schüler zur Frage, welche Anwendungen sie häufig nutzten (Häufigkeit der Nennung in Klammern; das Nennen von mehrern Anwendungen war möglich): Internet (31 mal); Bildbearbeitungs- und Grafikprogramme (5 mal); Medienwiedergabeprogramme (13 mal); Office-Programme (12 mal); Spiele (15 mal); Windows (16 mal); sonstige Anwendungen (9 mal)

[82] 63,1 % der Schülerinnen und Schüler erwarteten, im Unterricht das Programmieren zu erlernen.
14,8 % erwarteten den Umgang mit dem PC zu erlernen.
13,0 % erwarteten beides.
11,1 % gaben keine Erwartungen an das Fach an.

[83] Keiner der Befragten gab an, „sehr geringe“ Fähigkeiten im Umgang mit dem Computer zu haben.
3,7 % der Schülerinnen und Schüler schätzten ihre Fähigkeiten im Umgang mit dem Computer als „gering“ ein.
50,0 % schätzten ihre Fähigkeiten im Umgang mit dem Computer als „durchschnittlich“ ein.
38,9 % schätzten ihre Fähigkeiten im Umgang mit dem Computer als „gut“ ein.
7,4 % schätzten ihre Fähigkeiten im Umgang mit dem Computer als „sehr gut“ ein.

[84] Folgende Angaben machten die Schülerinnen und Schüler zur Frage, mit welcher Programmiersprache sie schon gearbeitet haben (Häufigkeit der Nennung in Klammern; das Nennen von mehrern Sprachen war möglich):
C++ (2 mal); Delphi (1 mal); HTML (9 mal); Java (3 mal); Kara (5 mal); Logo (4 mal); Oberon (1 mal); Python (1 mal); Turbo Pascal (1 mal); Visual Basic (3 mal); sonstige (2 mal)
An den Schulen D, E und F gaben jeweils mehr als die Hälfte der Schülerinnen und Schüler, die den Fragebogen ausfüllten, an, bereits mit einer Programmiersprache gearbeitet zu haben.

| Schule | Schulform | Jahr-gangs-stufe | Unterrichtsform | Anzahl der Schülerin-nen und Schüler |
|---|---|---|---|---|
| A | Regelschule | 10 | Wahlfach Informatik | 7 |
| B | Gesamtschule | 10 | Arbeitsgemeinschaft Informatik | 13 |
| C | Gymnasium | 10 | Informatikunterricht | 27 |
| D | Gymnasium | 10 | Wahlfach Informatik | 5 |
| E | Gymnasium | 8 | Wahlfach Informatik | 12 |
| F | Gymnasium mit math./naturwiss. Spezialklassen | 8 | wahlobligatorischer Unterricht | 10 |

Tabelle 9: Übersicht über den Unterricht an den sechs Schulen in der Voruntersuchung

| Lehrerin / Lehrer | Jahre im Schuldienst | Jahre Unterrichts-erfahrung Informatik | Informatikabschluss |
|---|---|---|---|
| Lehrer A | 28 | 6 | keine Angabe |
| Lehrerin B | 21 | 14 | Unterrichtserlaubnis |
| Lehrer C | 1 | 1 | abgeschlossenes Hochschulstudium & 2. Staatsexamen |
| Lehrer D | 28 | 16 | Unterrichtserlaubnis |
| Lehrer E | 37 | 16 | Postgraduales Studium |
| Lehrerin F | 7 | 7 | abgeschlossenes Hochschulstudium & 2. Staatsexamen |

Tabelle 10: Übersicht zur Berufserfahrung der Lehrpersonen in der Voruntersuchung

Lernenden mit Programmiererfahrung sind weitere Indizien dafür, dass größtenteils an der Informatik interessierte Schülerinnen und Schüler am untersuchten Unterricht teilgenommen haben.

Im ersten Schulhalbjahr 2006/2007 wurde von fünf Lehrpersonen (A, B, C, D, F) mit den bereitgestellten Materialien jeweils eine Doppelstunde pro Woche unterrichtet.[85] In Schule E wurden die Materialien ein dreiviertel Schuljahr lang in einer Stunde pro Woche eingesetzt. Die Lehrkräfte C, D, E und F nannten zu Beginn des Halbjahres als Ziel des Unterrichts, dass ein Großteil der Lernenden, Stufe II erreicht. Lehrerin B gab aufgrund der speziellen Ausrichtung ihrer Schule Stufe III als Ziel für ihre Schülerinnen und Schüler an.

## 7.3 Ergebnisse zum Kompetenzmodell

Aufgrund der Angaben der Lehrpersonen in den Interviews konnten fünf verschiedene Nutzungsvarianten bezüglich der Unterrichtsplanung anhand des Kompetenzmodells festgestellt werden:

**Kein Einsatz:** (Lehrer D) Die Lehrperson verwendete das Kompetenzmodell nicht zur Unterrichtsplanung.

**Einsatz am Anfang und am Ende des Schulhalbjahres** (Lehrkräfte A, E) Die Lehrperson orientierte sich am Anfang des Schulhalbjahres bei der groben Unterrichtsplanung am Kompetenzmodell. Am Ende des Schulhalbjahres wurde das Kompetenzmodell bei der Vorbereitung auf den Kompetenztest genutzt.

**Einsatz bei der Planung größerer Unterrichtssequenzen** (Lehrerin F) Die Lehrperson orientierte sich bei der Planung von größeren Unterrichtssequenzen, die aus mehreren Stunden bestanden, am Kompetenzmodell.

**Einsatz bei der Planung einzelner Unterrichtsstunden** (Lehrer C) Die Lehrperson orientierte sich bei der Planung einzelner Unterrichtsstunden am Kompetenzmodell.

**Einsatz bei der Planung einer jeden Unterrichtsstunde** (Lehrerin B) Die Lehrperson orientierte sich bei der Planung einer jeden Unterrichtsstunde am Kompetenzmodell.

[85] Nach ca. der Hälfte des Halbjahres konnte Lehrerin F aufgrund äußerer Umstände nicht weiter unterrichten. Der Unterricht wurde deshalb in Absprache mit der Schulleitung vom Autor der Arbeit fortgeführt. Die folgenden Darstellungen beziehen sich im Allgemeinen auf den Unterricht der Lehrerin. Lediglich der Einsatz der Beispielaufgaben und des Kompetenztests an Schule F beziehen sich auf den vom Autor durchgeführten Unterricht.

Anhand dieser festgestellten Nutzungsvarianten wurden die Lehrpersonen in der Hauptuntersuchung zum Einsatz des Kompetenzmodells befragt (vgl. Abschnitt 8.3). Im Unterricht wurden lediglich von Lehrer E auszugsweise Kompetenzen des Modells vorgestellt. Von den anderen Lehrkräften wurden unterschiedliche Gründe angegeben, warum das Kompetenzmodell den Lernenden nicht präsentiert wurde.[86] Ob ein Vorstellen dieses Modells im Unterricht der Sekundarstufe I sinnvoll ist, kann aufgrund der geringen Erfahrungen nicht beurteilt werden.

Nach dem Unterricht gaben die Lehrer A und D an, aufgrund der geringen Beachtung des bereitgestellten Kompetenzmodells im Laufe der Untersuchung zu diesem keine Einschätzung abgeben zu können. Die anderen Lehrkräfte (B, C, E und F) bewerteten das Kompetenzmodell „Algorithmen“ insgesamt positiv.[87] Negative Äußerungen zum Kompetenzmodell gab es nicht. Trotzdem wird vom Autor der Arbeit vermutet, dass die unübersichtliche Darstellung des Kompetenzmodells auf zwei Seiten ein Grund dafür war, dass das Kompetenzmodell von einigen Lehrpersonen gar nicht bzw. nur am Anfang und am Ende des Schulhalbjahres beachtet wurde. Durch die Zusammenfassung der Komponenten „Analysieren“ und „Modifizieren“ zur Komponente „Arbeit mit Algorithmen“ sowie durch verbesserte Formulierungen war es nach dieser Untersuchung möglich, das Kompetenzmodell auf einer Seite darzustellen. Außerdem trugen typografische Hervorhebungen der auf den höheren Stufen neu hinzugekommenen Kompetenzen zur Übersichtlichkeit bei.

Nach dem erteilten Unterricht wurde die Verteilung der Kompetenzstufen von allen Lehrpersonen als gut eingeschätzt. Regelschullehrer A schätzte ein, dass Stufe I des Kompetenzmodells auch für Regelschülerinnen und -schüler hinreichend einfach sei. Nach der Aussage der Lehrpersonen B und C könnten mehr als drei Kompetenzstufen zu Verwirrung führen. Diesen Aussagen entsprechend gab es keinen Grund, größere Veränderungen an der Einteilung der Kompetenzstufen vorzunehmen.

[86] Als Gründe wurden angegeben, dass das Kompetenzmodell für Regelschüler zu theoretisch wäre (Lehrer A), dass ein Vorstellen des Modells erst in der Sekundarstufe II sinnvoll sei (Lehrerin B), dass das Kompetenzmodell für Lernende nicht überschaubar sei und somit ein Vorstellen keinen Sinn machen würde (Lehrer C), dass das Erläutern des Modells in einem Unterricht ohne Noten nur eine Belastung für die Schülerinnen und Schüler wäre (Lehrer D), dass das Kompetenzmodell als methodisches Werkzeug der Lehrkraft den Schülerinnen und Schülern nicht vorgestellt werden müsste (Lehrerin F).

[87] Nach der Untersuchung wurde das Kompetenzmodell durch die Lehrpersonen als *sehr gut handhabbar (B), praktikabel (B), ausgereift (C), gut strukturiert (E), durchdacht (E)* und *O.K. (F)* bezeichnet. Keine der Lehrpersonen gab Probleme oder Verbesserungsvorschläge an.

Lehrerin B sah es als problematisch an, dass einem Außenstehenden, der eine Kompetenzbewertung eines Schülers sehe, nicht sofort klar sei, ob Stufe I oder Stufe III umfassendere Kompetenzen beinhalte. Nach der Voruntersuchung wurde deshalb die Komplexität der Stufen mit den Begriffen „grundlegende“, „vertiefte“ und „umfassendere“ Kompetenzen auch Außenstehenden verdeutlicht. Lehrer C forderte mehr Kompetenzen zur Reflexion auf höheren Stufen. Dementsprechend wurden auf Stufe II und III noch Kompetenzen zum Reflektieren hinzugefügt (vgl. Abschnitt 5.3).

Bei der Frage nach einer allgemeinen Einschätzung des Unterrichtens nach einem vorgegeben Kompetenzmodell konnte festgestellt werden, dass die Lehrkräfte ein solches Modell hauptsächlich als *Leitfaden* (A, B) und *Grundkonzept* (F) für den Unterricht bewerteten. Im Gegensatz zum Unterrichten nach einem Lehrplan wurde von den Lehrkräften B, C, D und F besonders das Unterteilen in unterschiedliche Stufen als positiv hervorgehoben.

## 7.4 Ergebnisse zu den Beispiel- und Unterrichtsaufgaben

Die Untersuchung ergab, dass die Unterrichtsaufgaben von allen sechs Lehrpersonen in nahezu jeder Stunde verwendet wurden. Die Lehrkräfte B, C, E und F wählten die Aufgaben entsprechend der angestrebten Kompetenzen aus. Auch die Beispielaufgaben konnten im Unterricht aller Schulen sinnvoll zum Verdeutlichen der Kompetenzen eingesetzt werden. An den Schulen B, C und F konnten die Schülerinnen und Schüler dadurch die im Test zu bearbeitende Stufe eigenständig wählen (vgl. Abschnitt 7.5). Dieser häufige Einsatz spricht für die Akzeptanz der Beispiel- und Unterrichtsaufgaben durch die Lehrerinnen und Lehrer.

Die einzelnen Unterrichtsaufgaben sollten, sofern sie eingesetzt wurden, nach der Untersuchung von den Lehrkräften beurteilt werden. Insgesamt wurden 73 Bewertungen zu insgesamt 38 der 53 in der Voruntersuchung verwendeten Unterrichtsaufgaben abgegeben. 69 dieser 73 Einschätzungen waren positiv. Die im Arbeitsmaterial zur Hauptuntersuchung mit den Nummern 7, 9, 15, 16, 17, 31, 51, 52 und 63 bezeichneten Aufgaben wurden jeweils von mindestens drei Lehrpersonen im Unterricht erprobt und durchgehend positiv eingeschätzt (zu den Aufgaben vgl. Kohl, 2008a, S. 18 ff.). Die vier negativen Bewertungen betrafen Aufgaben mit Bezügen zur Mathematik. Lehrer A, der drei dieser vier negativen Einschätzungen abgegeben hatte, stellte dar, dass die meisten Aufgaben mit Bezügen zur Mathematik für seine Schülerinnen und Schüler nicht geeignet waren. An der Regelschule A wurden von ihm *Aufgaben mit Grafik, Animationen und Musik* bevorzugt, *bei denen sofort sichtbar [bzw. hörbar] ist, ob etwas richtig*

*gelöst wurde*. Von den anderen Lehrpersonen wurden Aufgaben mit Bezügen zur Mathematik positiv bewertet. Lehrer E gab zu bedenken, dass einige Aufgaben evtl. zu schwierig für Schülerinnen und Schüler der Jahrgangsstufe 8 sein könnten. Da insgesamt die Vielfältigkeit der bereitgestellten Aufgaben positiv beurteilt wurde, wurden keine Unterrichtsaufgaben aus der Sammlung entfernt.

Durch die Analyse der Interviews im Zusammenhang mit dem beobachteten Unterricht konnten zwei verschiedene Varianten von Differenzierung im Unterricht festgestellt werden:

**Vereinzelte Differenzierung für Leistungsstärkere** (Lehrkräfte A, D, E) Im Unterricht wurde nur vereinzelt differenziert gearbeitet. Leistungsstärkere Lernende bekamen im Unterricht teilweise Zusatzaufgaben gestellt oder halfen anderen Schülerinnen und Schülern.

**Häufige Differenzierung durch Anzahl und/oder Schwierigkeit der Aufgaben** (Lehrkräfte B, C, F) In den meisten Unterrichtsstunden wurde differenziert gearbeitet. Dabei stellte die Lehrperson im Unterricht eine Menge von Aufgaben zur Verfügung, aus denen die Lernenden auswählen konnten. Die Differenzierung erfolgte durch die Anzahl und/oder die Schwierigkeit der gelösten Aufgaben.

Dieses Ergebnis der Voruntersuchung wurde zum Anlass genommen, um die Unterrichtsaufgaben so zu modifizieren, dass diese noch besser zur Differenzierung eingesetzt werden können.
Einige Aufgaben wurden nach der Voruntersuchung in Teilaufgaben untergliedert, um so auch bei einzelnen Aufgaben Möglichkeiten zur Differenzierung zu geben. Lehrer C gab an, dass er es als günstig empfinden würde, wenn bei den Unterrichtsaufgaben die Kompetenzstufen mit angegeben wären. Dementsprechend wurden die Unterrichtsaufgaben nach der Voruntersuchung jeweils einer Stufe (und einer Komponente) des Kompetenzmodells zugeordnet. Dadurch ist das Ausrichten des Unterrichts an den Kompetenzen des Modells durch die Auswahl entsprechender Aufgaben vermutlich noch besser möglich. Das Bereitstellen von Aufgaben unterschiedlicher Stufen kann auch zur Differenzierung genutzt werden. Weitere gezielte Hinweise zu einzelnen Aufgaben dienten der Weiterentwicklung derselben.

## 7.5 Ergebnisse zum Kompetenztest

### Durchführung und Bewertung des Kompetenztests

Die Testaufgaben wurden den Lehrpersonen erst am Tag des Kompetenztests übergeben, um ein „Teaching to the Test“ zu vermeiden. Der Kompetenztest wurde

– wie empfohlen – an allen sechs Schulen in 90 Minuten bearbeitet. Probleme mit dieser Zeitvorgabe gab es nach den Angaben der Lehrkräfte und den Beobachtungen des Autors nicht. Teilweise wurden von den Schülerinnen und Schülern sogar die Aufgaben zweier Stufen gelöst. Beim Einsatz des Kompetenztests konnten vier verschiedene Varianten zur Auswahl der Kompetenzstufen festgestellt werden:

1. **Variante** (Schulen A, E): Die Lehrperson legte eine Stufe fest, die von der ganzen Klasse einheitlich bearbeitet wurde.
2. **Variante** (Schule D): Die Lehrperson gab individuelle Empfehlungen für jede Schülerin und jeden Schüler, welche Stufe bearbeitet werden sollte.
3. **Variante:** (Schulen B, C) Jede Schülerin und jeder Schüler legte für sich fest, welche Stufe bearbeitet wurde.
4. **Variante:** (Schule F) Jede Schülerin und jeder Schüler legte für sich für jede der Komponenten die Stufe fest, die jeweils bearbeitet wurde.

Diese vier Varianten wurden für die folgende Untersuchung systematisch zusammengefasst und den Lehrpersonen mit dem Kompetenztest zur Verfügung gestellt (vgl. Abschnitt 8.1). Alle sechs Lehrpersonen stellten es den Schülerinnen und Schülern, die vorzeitig die Aufgaben einer Stufe gelöst hatten, frei, die Aufgaben weiterer Stufen zu bearbeiten. Die Lösungen der Lernenden wurden vor Ort von den Lehrkräften bewertet.

Der Autor dieser Arbeit erhielt die gespeicherten Dateien sowie Kopien der schriftlichen Aufzeichnungen beim Kompetenztest und wertete die Ergebnisse der Schülerinnen und Schüler aus. Die Punktverteilungen und die Bewertungskriterien waren dabei einheitlich, entsprachen in der Voruntersuchung aber noch nicht den im Abschnitt 6.2 vorgestellten.[88] Trotzdem war es für Schülerinnen und Schüler, die Testaufgaben der Stufe II oder III bearbeiteten, bereits möglich, auch die jeweils niedrigeren Stufen erreichen zu können.

### Ergebnisse zu den Testaufgaben

Bei kompakt formulierten Aufgaben, bei denen mehrere Arbeitsschritte in einem kurzen Fließtext gefordert waren, wurden bisweilen einzelne Teilaufgaben von Schülerinnen und Schülern nicht bearbeitet. Ein Beispiel soll dies verdeutlichen:

[88] In der Voruntersuchung waren für das Erreichen einer Stufe folgende Bedingungen zu erfüllen:
- Es mussten zwei Drittel der in der jeweiligen Stufe möglichen Punkte erreicht werden und
- in wenigstens drei der fünf Komponenten musste auf der angestrebten oder einer höheren Stufe mindestens die Hälfte der möglichen Punkte erreicht werden.

In der Testaufgabe zum „Analysieren und Modifizieren“ auf Stufe II war folgendes gefordert:
*Analysieren Sie den gegebenen Algorithmus in Stichpunkten, übertragen Sie ihn in ein Programm und testen Sie dieses.*
Neun Schülerinnen und Schüler erhielten Punkte für die Analyse des Algorithmus. 23 Schülerinnen und Schüler erhielten Punkte für das Übertragen des Algorithmus in ein Programm.

Aufgrund der Erfahrungen aus den Unterrichtsbeobachtungen werden für die unterschiedlichen Lösungshäufigkeiten nachfolgend drei Erklärungsversuche genannt und teilweise Konsequenzen abgeleitet:

- Die erste Teilaufgabe wurde von den Schülerinnen und Schülern „überlesen“ (zur Förderung von Lesekompetenz vgl. Artelt u. a., 2007).
  Um diesem Phänomen im Kompetenztest der Hauptuntersuchung entgegenzuwirken, wurden nach der Voruntersuchung an den Testaufgaben die folgenden Veränderungen vorgenommen.
  Teilaufgaben wurden explizit als solche gekennzeichnet. Zu jeder schriftlich zu beantwortenden Teilaufgabe wurde ein Antwortfeld im Aufgabenblatt bereitgestellt. Dadurch war es allerdings nicht mehr möglich, den Kompetenztest einer Stufe auf einer Seite darzustellen.

- Den Schülerinnen und Schülern fehlten die nötigen Fähigkeiten und Fertigkeiten zur Analyse des Algorithmus.
  Im Unterricht stand größtenteils die praktische Arbeit am Computer im Vordergrund. Schriftliche Analysen von Algorithmen wurden an allen sechs Schulen nur vereinzelt im Unterricht gefordert und geübt. Um die geforderten Kompetenzen angemessen zu verdeutlichen wurden nach der Voruntersuchung die oben genannten Modifikationen auch an den Beispielaufgaben vorgenommen.

- Den Schülerinnen und Schülern fehlte die nötige Motivation zur Analyse des Algorithmus.
  Bei der Beobachtung des Unterrichts war aufgefallen, dass die Funktionsweise eines Algorithmus von den Lernenden nur „ungern“ in eigenen Worten notiert wurde. Wenn die Schülerinnen und Schüler die Aufgaben trotz vorhandener Fähigkeiten und Fertigkeiten wissentlich nicht gelöst haben, so kann dies als Indiz dafür gewertet werden, dass motivationale, volitionale und soziale Bereitschaften und Fähigkeiten fehlten, um die Kompetenz nachzuweisen. Dies spricht für die Verwendung des im Abschnitt 3.1 vorgestellten Kompetenzbegriffs. Konsequenzen wurden aus diesem Erklärungsversuch nicht abgeleitet.

Bei den Schülerlösungen zu den Testaufgaben der Komponenten „Entscheiden“ und „Grundbausteine erklären“ gab es häufig Punktabzüge, weil Erklärungen bzw. Begründungen unzureichend oder unter falscher Verwendung von Fachbegriffen abgegeben wurden. Eventuell fehlten Unterrichtsaufgaben, mit denen entsprechende Kompetenzen erworben werden konnten. Solche Aufgaben wurden in Vorbereitung auf die Hauptuntersuchung bereitgestellt.

Von den sechs Lehrkräften wurden die Aufgaben des Kompetenztests insgesamt als angemessen eingeschätzt. Einen von außen bereitgestellten Kompetenztest empfanden alle Lehrpersonen als hilfreich, um über den eigenen Unterricht zu reflektieren. Dies entspricht den Ergebnissen von Fothe und Ludwig (2008, S. 12), die festgestellt hatten, *dass extern bereitgestellte Tests nicht nur „im Prinzip“ eine gute Sache sind, sondern dass sie eine begründete Chance haben, in der Schulpraxis auch wirklich gewinnbringend eingesetzt zu werden.* Die Ergebnisse der Schülerinnen und Schüler entsprachen größtenteils den Erwartungen der Lehrpersonen.[89]

Außer den genannten Aspekten sind bei der Auswertung des Kompetenztests bei den einzelnen Aufgaben keine Fehler festgestellt worden, die so häufig auftraten, dass von einem systematischen Fehler gesprochen werden könnte. Genauere Aussagen zu der Bearbeitung der Testaufgaben sind aufgrund der gewählten Forschungsmethodik nicht möglich. Hier würde es sich bei weiteren Untersuchungen anbieten, einzelne Schülerinnen und Schüler bei der Bearbeitung des Tests „laut denken“ zu lassen oder sie anschließend zu interviewen, um Gründe für das schlechte Abschneiden bei einzelnen Teilaufgaben herauszufinden.[90]

### Ergebnisse der Lernenden beim Kompetenztest

Insgesamt konnten die Kompetenztests von 55 Lernenden vollständig ausgewertet werden. Die Ergebnisse werden im Folgenden vorgestellt:
65,5 % der Schülerinnen und Schüler bearbeiteten die Aufgaben der Kompetenzstufen II oder III. 89,1 % haben Stufe I oder höher, also die Mindeststandards, erreicht. Ca. ein Drittel der Schülerinnen und Schüler erlangte Kompetenzen, die über die Mindeststandards hinausgehen (also Stufe II oder III). Eine Übersicht

[89] Lediglich Lehrer E bemerkte, dass in den Komponenten „Entscheiden“ und „Grundbausteine erklären“ weniger Punkte als erwartet erreicht wurden.

[90] Kujath (2006, S. 52 f.) setzte die Methode des „lauten Denkens“ bei der Untersuchung der Lösungen von Aufgaben im Informatikunterricht ein (zur Methode vgl. Bortz und Döring, 2006, S. 325).
Erste Erfahrungen zu Schülerinterviews, bei denen die Ergebnisse allerdings zur Reflexion über Unterricht dienten, wurden von Fothe u. a. (2006, S. 57 ff.) vorgestellt.

über die insgesamt von den Lernenden erreichten Stufen im Test gibt Abbildung 28.

Eine Untersuchung hinsichtlich der Zusammenhänge der von den Schülerinnen und Schülern im Test erreichten Kompetenzstufe mit Geschlecht, Alter, Selbsteinschätzung der Fähigkeiten im Umgang mit dem Computer, bzw. Dauer der Computernutzung (zu Hause und in der Schule) ergaben keine signifikanten Ergebnisse.[91] Schülerinnen und Schüler, die schon Erfahrungen mit einer Programmiersprache gemacht hatten, erreichten – wie erwartet – durchschnittlich eine höhere Stufe im Kompetenztest. Abbildung 29 verdeutlicht diesen Zusammenhang in einem Diagramm.[92] Dass keiner der Lernenden mit Vorkenntnissen Stufe III erlangte, deutet darauf hin, dass die auf Stufe III angegebenen Kompetenzen auch für diese Schülerinnen und Schüler herausfordernd sind.

In den Abbildungen 30 und 31 sind die Testergebnisse der Lernenden zu den Komponenten „Grundbausteine erklären“ und „Analysieren“ dargestellt.[93] Bei der Komponenten „Grundbausteine erklären“ konnten lediglich 60,0 % der Lernenden Kompetenzen der Stufe I, II oder III nachweisen. Ein möglicher Grund dafür ist, dass entsprechende Unterrichtsaufgaben zum Erwerben der Kompetenzen dieser Komponente fehlten. Die oben bereits erwähnte Entwicklung solcher Aufgaben im Anschluss an diese Voruntersuchung kann diesem Mangel entgegenwirken.
Auch bei der Komponente „Modifizieren“ erreichten 40,0 % der Schülerinnen und Schüler keine Stufe. Dies kann evtl. auf die oben beschriebene Problematik der in einem Fließtext formulierten Aufgaben zurückgeführt werden. Dem wird mit den genannten Konsequenzen entgegengewirkt.
Die Verteilung der durch die Schülerinnen und Schüler erreichten Stufen der Komponenten „Entscheiden“, „Analysieren“ und „Entwickeln“ sind ähnlich ausgeprägt wie die Verteilung der Stufen beim gesamten Kompetenztest in Abbildung 28.

Soweit es die Ergebnisse dieser kleinen Stichprobe zulassen, erscheinen die Testaufgaben prinizipiell als geeignet, um zwischen unterschiedlichen Kompetenzstufen zu unterscheiden. Bei allen Komponenten konnten, wie beim Kompetenztest insgesamt, Schülerinnen und Schüler zu jeder Stufe zugeordnet werden.

Aufgrund der insgesamt positiven Erfahrungen bei der Arbeit mit dem Kompetenztest in dieser Voruntersuchung wurden in Vorbereitung auf die Hauptuntersuchung

[91] Zur Erläuterung der statistischen Analysen vgl. Abschnitt 8.1.

[92] Es konnten lediglich die Ergebnisse der Schülerinnen und Schüler untersucht werden, die den Fragebogen ausgefüllt und den Kompetenztest bearbeitet hatten.

[93] Um bei einer Komponente eine Stufe zu erlangen, musste mindestens diese angestrebte Stufe bearbeitet worden sein, und es waren mindestens die Hälfte der auf dieser Stufe möglichen Punkte nötig.

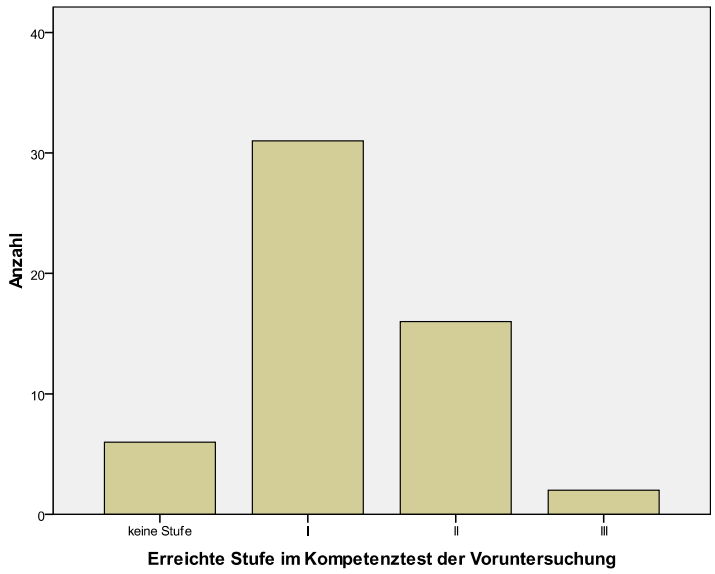



Abbildung 28: Gesamtübersicht über die Ergebnisse der Lernenden beim Kompetenztest in der Voruntersuchung (N=55)

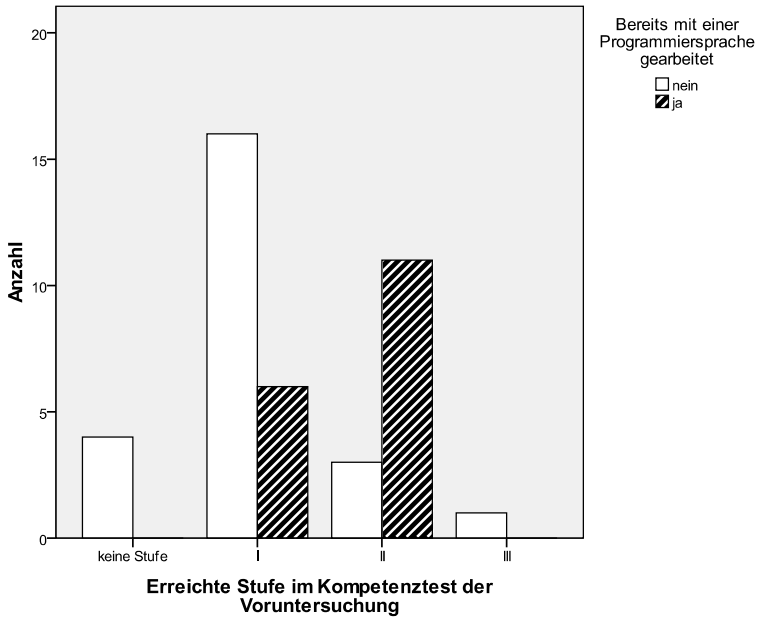



Abbildung 29: Zusammenhang zwischen Erfahrungen mit einer Programmiersprache und der erreichten Stufe im Kompetenztest (N=41)

außer den genannten Aspekten keine größeren Veränderungen an den Testaufgaben vorgenommen.

## 7.6 Ergebnisse zur visuellen Programmiersprache Puck

Bei der Einführung in die Programmierung mit Puck in den Vorbereitungsveranstaltungen arbeitete zu Beginn jeweils eine Lehrperson an einem zentralen Computer, dessen Bildschirminhalt für die restlichen Teilnehmer projiziert wurde. Dieses in verschiedenen Fortbildungen erprobte methodische Vorgehen wurde auch im Unterricht an den Schulen eingesetzt und hat sich nach Auffassung des Autors bewährt. Die Lehrkräfte fühlten sich durch die Veranstaltungen an der Universität Jena gut auf den Unterricht mit Puck vorbereitet. Insgesamt wurde die visuelle Programmiersprache von den Lehrerinnen und Lehrern auch positiv eingeschätzt. Insbesondere die verminderte Fehlerproblematik ist nach Meinung aller Lehrpersonen günstig für den Anfangsunterricht und führt zu keinen anderen Problemen. Die Lehrkräfte B, C, D, E und F gaben an, dass mehr Zeit für die individuelle Förderung einzelner Schülerinnen und Schüler blieb. Durch die Arbeit mit Puck sind die Lernenden nach Ansicht aller sechs Lehrpersonen auf die Arbeit mit einer textuellen Sprache vorbereitet worden, auch wenn nicht direkt am textuellen Code eines Programms gearbeitet wurde. Alle Lehrkräfte wollten die visuelle Programmiersprache im nächsten Schuljahr wieder einsetzen. Einige Features des Puck-Systems wurden speziell für den Einsatz im Informatikunterricht entwickelt (vgl. Kohl u. a., 2007, S. 45 ff. sowie Kohl, 2007b, S. 192 ff.). Die Häufigkeit des Einsatzes dieser Features und die Beurteilung derselben durch die Lehrpersonen sollen im Folgenden vorgestellt werden:

- Alle Lehrkräfte äußerten sich positiv über die ausführliche Hilfe des Puck-Systems.
- Das Vorgeben der von Schülerinnen und Schülern verwendbaren Bausteine durch Einstellungen der Lehrkraft in den Optionen wurde von den Lehrpersonen A, B, D und E am Anfang und von der Lehrkraft C im gesamten Unterricht genutzt. Lehrerin F stellte ihren Schülerinnen und Schülern in einem entdeckenden Unterricht die Wahl der Bausteine frei. Die Möglichkeit die verwendbaren Bausteine vorzugeben, wurde von allen sechs Lehrkräften als sehr nützlich eingeschätzt.
- Die Möglichkeit, Pseudocode zu einem Programm zu generieren, wurde von den Lehrern C und E, wie bei der Entwicklung intendiert, zur Vorbereitung des Unterrichts genutzt. Lehrer D ermöglichte seiner Schülerin und seinen Schülern das Generieren von Pseudocode zu im Unterricht entwickelten


Abbildung 30: Ergebnisse der Lernenden zu den Aufgaben der Komponente „Grundbausteine erlären“ beim Kompetenztest in der Voruntersuchung (N=55)

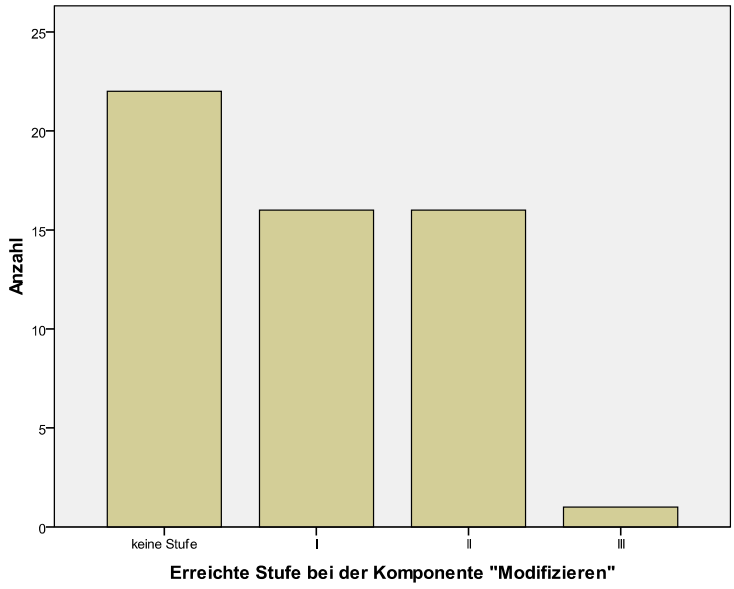



Abbildung 31: Ergebnisse der Lernenden zu den Aufgaben der Komponente „Modifizieren“ beim Kompetenztest in der Voruntersuchung (N=55)

Programmen. Er beobachtete, dass die Schülerin und die Schüler diese symbolische Repräsentation des Programms häufig nutzten, und sah diese weitere Darstellungsmöglichkeit als sinnvolles Mittel bei der Analyse von Programmen an (zu Repräsentationsformen vgl. Abschnitt 4.1).

- Lehrerin F setzte die Komplexitätsberechnung wie intendiert zum Vergleich von Schülerlösungen ein und empfahl ihren Schülerinnen und Schülern, möglichst Programme mit geringer Komplexität zu entwickeln. Außerdem wurde die Komplexitätsberechnung im Unterricht von Lehrer A und C testweise ausprobiert, aber nicht langfristig eingesetzt.
- Lehrer A nutzte einige Male die Möglichkeit, Log-Files der Schülerinnen und Schüler zu betrachten.
- Im Untersuchungszeitraum wurde das Generieren von textuellem Code von keiner Lehrkraft genutzt. Die Lehrpersonen B und C gaben an, die Generierung von Quelltext einer textuellen Programmiersprache im zweiten Schulhalbjahr nutzen zu wollen.

Die Puck-Hilfe und das Vorgeben der verwendbaren Bausteine wurden häufig, die restlichen Features nur vereinzelt eingesetzt. Aufgrund der überwiegend positiven Einschätzungen wurden für die Hauptuntersuchung keine Veränderungen an der visuellen Programmiersprache vorgenommen. Eine am Anfang der Voruntersuchung eingerichtete Online-Diskussionsgruppe wurde im Laufe des Untersuchungszeitraums nur vereinzelt genutzt.[94] Um Lehrenden sowie Lernenden eine bessere Plattform zum Austausch von Informationen rund um Puck zu bieten, wurde nach Abschluss der Voruntersuchung die Homepage `www.ipuck.de` entwickelt.

## 7.7 Einordnung der Ergebnisse

Aufgrund der in den Abschnitten 7.3 bis 7.6 dargestellten Ergebnisse kann festgestellt werden, dass zumindest von den Lehrkräften B, C und F die entwickelten Materialien wie in den Erläuterungen im Abschnitt 3.4 intendiert, zur Strukturierung, Vorbereitung, Durchführung und Auswertung von kompetenzorientiertem Informatikunterricht eingesetzt wurden. Weiterhin konnten durch diese Voruntersuchung sinnvolle Hinweise zur Weiterentwicklung von Kompetenzmodell und zugehörigen Aufgaben und zur Vorbereitung der Hauptuntersuchung gewonnen werden. Der Einsatz der visuellen Programmiersprache Puck wurde positiv beurteilt. Die Ergebnisse deuten darauf hin, dass Puck geeignet ist, um Schülerinnen

[94] Insgesamt wurden lediglich 5 Nachrichten von den Lehrpersonen gepostet. Diese betrafen hauptsächlich technische Probleme.

und Schülern der Jahrgangsstufen 8 bis 10 eine Einführung in die Programmierung zu geben.
Aufgrund des explorativen Charakters der Untersuchung sind die Erkenntnisse aber nicht ohne Weiteres auf andere Situationen übertragbar. Nachfolgend sollen zur Relativierung einige Punkte genannt werden, die die Verallgemeinerbarkeit der Ergebnisse auf den Informatikunterricht in deutschsprachigen Ländern einschränken.

- An der Untersuchung nahmen eine Regelschule, eine integrierte Gesamtschule und vier Gymnasien, davon eines mit mathematisch naturwissenschaftlichen Spezialklassen, teil. Damit ist ein relativ großer Anteil der Lehrenden und Lernenden der Schulart Gymnasium zuzuordnen. Alle Schulen befinden sich in Thüringen. Dies schränkt die Verallgemeinerbarkeit der Ergebnisse auf allgemeinbildende Schulen ein.

- Informatikunterricht in den Jahrgangsstufen 8 bis 10 ist in Thüringen nur wenig verbreitet und die Anzahl der in Frage kommenden Lehrkräfte war relativ gering. Trotzdem haben sich zwei Lehrerinnen und vier Lehrer freiwillig bereit erklärt, an der Untersuchung teilzunehmen. Diese freiwillige Bereitschaft deutet darauf hin, dass es sich um engagierte Lehrkräfte handelt. Fraglich ist, ob die Materialien im Unterricht weniger engagierter Lehrkräfte ähnlich eingesetzt bzw. eingeschätzt worden wären.

- Die Lehrkräfte unterrichteten größtenteils zum ersten Mal auf Grundlage eines Kompetenzmodells und mit der visuellen Programmiersprache Puck und konnten somit nicht auf entsprechende Erfahrungen zurückgreifen. Fragen konnten allerdings durch die häufigen Schulbesuche des Autors meist zeitnah beantwortet werden. Es bleibt offen, ob Lehrpersonen auch ohne die Beantwortung auftretender Fragen, ähnlich mit den bereitgestellten Materialien umgegangen wären bzw. wie sich Erfahrungen im Umgang mit Kompetenzmodellen und visuellen Programmiersprachen auf den Unterricht auswirken.

- Durch die Arbeit mit Puck wurden Syntaxfehler verhindert, somit blieb häufig mehr Zeit für die individuelle Förderung einzelner Schülerinnen und Schüler (vgl. Abschnitt 7.6). Ob ähnliche Ergebnisse auch beim Verwenden eines anderen Programmiersystems erreicht worden wären, ist unklar.

- Die Schülerinnen und Schüler nahmen größtenteils freiwillig am Informatikunterricht teil oder hatten sich zumindest aktiv für dieses Fach entschieden (vgl. Abschnitt 7.2). Dies schränkt die Verallgemeinerbarkeit auf Informatikunterricht, der als Pflichtfach erteilt wird, ein.

- Es kann nicht ausgeschlossen werden, dass das beobachtende Teilnehmen des Autors am Unterricht Auswirkungen auf denselben hatte.
- Aufgrund der gewählten Forschungsmethodik kann nicht eingeschätzt werden, inwieweit beim eingesetzten Fragebogen und bei den leitfadenorientierten Interviews Effekte sozialer Erwünschtheit auftraten.

## 7.8 Zusammenfassung

In der beschriebenen Voruntersuchung setzten sechs Lehrpersonen Puck und erste Versionen von Kompetenzmodell und Aufgaben ein Halbjahr lang in den Jahrgangsstufen 8 und 10 ein. Durch die Kombination verschiedener Forschungsmethoden konnte der Einsatz der bereitgestellten Materialien und des Werkzeugs Puck im Unterricht untersucht werden. Generell wurde das Unterrichten nach einem vorgegebenen Kompetenzmodell als positiv eingeschätzt. Die Unterrichtsaufgaben wurden von allen Lehrkräften häufig zur Vorbereitung genutzt. Aufgrund der gewonnenen Erfahrungen wurden nach der Untersuchung weitere Aufgaben entwickelt und die existierenden Aufgaben wurden in die Komponenten und Stufen des Kompetenzmodells eingeordnet. Die Ergebnisse des Kompetenztests und die Einschätzungen der Lehrpersonen deuten darauf hin, dass die Testaufgaben geeignet sind, die im Modell angegebenen Kompetenzen zu erfassen und dass die Stufen wirklich unterschiedliche Kompetenzniveaus beschreiben. Das Puck-System wurde von den Lehrpersonen weitgehend als positiv eingeschätzt. Die verminderte Fehlerproblematik ist nach Meinung aller Lehrkräfte günstig für den Anfangsunterricht und führt zu keinen anderen Problemen. Auch wenn die dargestellten Ergebnisse nicht ohne Weiteres auf andere Situationen übertragbar sind so konnten doch erste Erkenntnisse zu kompetenzorientiertem Informatikunterricht zusammengetragen werden, die auch zur Vorbereitung der Hauptuntersuchung genutzt werden konnten.

# 8 Hauptuntersuchung

> *Puck ist genau das, was ich seit einiger Zeit suche, eine Möglichkeit, Schülern der Sekundarstufe I im Rahmen des Informatikunterrichts meiner 10. Klassen neben meinen anderen Themen mit einem einfachen graphischen System eine Grundvorstellung von Programmstrukturen vermitteln zu können, ohne eine komplexe Programmiersprache lernen zu müssen.*
> (Ein Informatiklehrer über Puck, Auszug aus einer E-Mail, Mai 2008)

Um Antworten auf die im Kapitel 1 vorgestellte zweite und dritte Forschungsfrage zu erhalten, wurde untersucht, wie das Kompetenzmodell „Algorithmen“, die zugehörigen Aufgaben und das Programmiersystem Puck durch eine größere Zahl von Lehrkräften im Unterricht eingesetzt wurden. Zum Ende des Halbjahres wurde den Lehrpersonen ein Kompetenztest zur Verfügung gestellt, mit dem die von den Schülerinnen und Schülern erworbenen Kompetenzen erfasst werden sollten. Anschließend wurden die Lehrerinnen und Lehrer gebeten, in einem Online-Fragebogen die bereitgestellten Materialien und den durchgeführten Unterricht einzuschätzen. Das gesamte Untersuchungsdesign wird im Abschnitt 8.1 vorgestellt. Im Abschnitt 8.2 wird die untersuchte Stichprobe sowie der durchgeführte Unterricht beschrieben. Anschließend werden die Ergebnisse der Befragung in den Abschnitten 8.3 bis 8.6 dargestellt, dabei wird auf das Kompetenzmodell, die Beispiel- und Unterrichtsaufgaben, auf den Kompetenztest sowie auf die visuelle Programmiersprache Puck eingegangen. Eine Einordnung der Ergebnisse erfolgt im Abschnitt 8.7.

## 8.1 Untersuchungsdesign

**Untersuchungsszenario** Die Lehrpersonen, die sich an der Untersuchung beteiligten, sollten das Kompetenzmodell, die zugehörigen Aufgaben und Puck nach Möglichkeit ein Halbjahr lang während zwei Unterrichtsstunden pro Woche in der Jahrgangsstufe 8, 9 oder 10 einsetzen. Vorbereitungsveranstaltungen wurden aufgrund der räumlichen Verteilung der Untersuchungsteilnehmer nicht durchgeführt. Zur Teilnahme waren Schulen im deutschsprachigen Raum eingeladen, in denen das Fach Informatik in einer der genannten Jahrgangsstufen unterrichtet

wird. Informatiklehrerinnen und -lehrer wurden in einem Artikel in der Fachzeitschrift LOG IN, in dem das Puck-System vorgestellt wurde, eingeladen, an der Untersuchung teilzunehmen (vgl. Kohl u. a., 2007). Außerdem wurden ca. 10 % der weiterführenden Schulen aller deutschsprachigen Länder, Bundesländer und Kantone (Deutschland, Österreich, Schweiz) direkt per E-Mail über das Vorhaben informiert und zur Teilnahme eingeladen. Folgendes Szenario wurde in der per E-Mail verschickten Einladung beschrieben:

*Für August/September 2007 bis Februar 2008 laden wir Sie ein, das Thema „Algorithmen“ in einer der Klassenstufen 8-10 mithilfe des Puck-Systems und der Aufgabensammlung nach dem gegebenen Kompetenzmodell zu unterrichten. Alle Lehrerinnen und Lehrer, die an der Untersuchung teilnehmen möchten, bekommen im Juli 2007 einen Ordner mit Material zugeschickt. Anschließend unterrichten sie das Themengebiet „Algorithmen“ nach dem gegebenen Kompetenzmodell im ersten Halbjahr des Schuljahres 2007/2008 (zwei Unterrichtsstunden pro Woche). Wir werden Fragen gegebenenfalls per E-Mail oder telefonisch beantworten [...]. Am Ende des Halbjahres wird ein Kompetenztest zur Verfügung gestellt, mit dem überprüft werden kann, welche Kompetenzstufen die Schülerinnen und Schüler erreicht haben. Der Kompetenztest soll von den teilnehmenden Lehrerinnen und Lehrern selbstständig durchgeführt und bewertet werden. [...] In der letzten Woche des Halbjahres werden die Lehrerinnen und Lehrer dann durch [...] einen Fragebogen zu ihren Erfahrungen mit der visuellen Programmiersprache und dem Unterrichten nach einem Kompetenzmodell befragt.*

107 Informatiklehrerinnen und -lehrer aus insgesamt 57 Schulen in Deutschland, Österreich und der Schweiz meldeten sich mit einem vom Schulleiter unterschriebenen Fax zu der Untersuchung an. 23 Lehrkräfte gaben im Laufe des Halbjahres an, die Materialien aus organisatorischen Gründen nicht im Unterricht einsetzen zu können. Der häufigste Grund dafür war, dass der entsprechende Unterricht nicht zustande gekommen ist.

**Verwendete Materialien und E-Mail-Verkehr** Den Lehrpersonen, die sich für die Untersuchung angemeldet hatten, wurde auf dem Postweg ein Arbeitsmaterial zur Untersuchung und eine Einführung in die Programmierung mit Puck zugesandt (vgl. Kohl, 2006a und 2008a). Das Arbeitsmaterial enthielt das im Kapitel 5 vorgestellte Kompetenzmodell „Algorithmen“ sowie die im Kapitel 6 präsentierten Beispiel- und Unterrichtsaufgaben (vgl. Kohl, 2008a, S. 3 ff.). Außerdem wurden die bereitgestellten Materialien und das Puck-System sowie die im Abschnitt 6.1 vorgestellten Konkretisierungen der im Modell geforderten Kompetenzen beschrieben (vgl. Kohl, 2008a, S. 4 ff.). Zu Beginn des Schuljahres erhielten die

Lehrpersonen das Arbeitsmaterial zusätzlich in digitaler Form, um das selbstständige Zusammenstellen von Arbeitsblättern zu ermöglichen. Weiterhin wurden den Lehrerinnen und Lehrern Musterlösungen mit zugehörigen Puck-Programmen zu allen Unterrichtsaufgaben zur Verfügung gestellt. Auf die Möglichkeit, das Puck-System und zugehörige Materialien aus dem Internet herunterzuladen, wurde hingewiesen.

Nach der Anmeldung wurde mit den Lehrerinnen und Lehrern per E-Mail kommuniziert. So konnten individuelle Fragen meist recht schnell beantwortet werden. Insgesamt wandten sich 37 Lehrkräfte in ca. 100 E-Mails an den Autor. Die meisten E-Mails betrafen technische Anfragen zur Arbeit mit dem Puck-System oder organisatorische Fragen und konnten schnell geklärt werden. Einige Erweiterungsvorschläge zum Puck-System wurden geäußert. Bemerkenswert ist, dass in keiner E-Mail Fragen zum Kompetenzmodell und den zugehörigen Aufgaben gestellt wurden. Gründe hierfür sind nicht bekannt.
Die folgenden E-Mails wurden jeweils an alle Lehrkräfte gesendet:

- Bestätigung der Anmeldung zur Untersuchung mit Hinweis auf Arbeitsmaterial zu Puck im Internet (zu Puck-Arbeitsmaterial vgl. Kohl, 2006b sowie Kohl u. a., 2007)
- Übermittlung des digitalen Arbeitsmaterials zur Untersuchung inkl. Musterlösungen zu den Unterrichtsaufgaben (vgl. Kohl, 2008a, S. 3 ff.)
- Hinweis auf das Ende der Untersuchung und den durchzuführenden Kompetenztest
- Übermittlung des Kompetenztests zur Untersuchung (vgl. Kohl, 2008a, S. 70 ff.)
- erste Aufforderung den Fragebogen auszufüllen (inkl. Link zur Befragung)
- zweite Aufforderung den Fragebogen auszufüllen (inkl. Link zur Befragung)
- Dank für die Teilnahme an der Untersuchung

**Fragebogen** Zur Erhebung der Erfahrungen der Lehrpersonen wurde ein Fragebogen eingesetzt. Es wurde davon ausgegangen, dass die beteiligten Informatiklehrerinnen und -lehrer Zugang zum Internet haben. Dadurch war es möglich, den Fragebogen online bereitzustellen, was die Durchführung der Befragung erheblich vereinfachte:

- Druck- und Versandkosten konnten eingespart werden. Die Lehrpersonen konnten per E-Mail an das Ausfüllen des Fragebogens erinnert werden.

- Hürden bei der schriftlichen Beantwortung und Rücksendung, die die Rücklaufquote hätten negativ beeinflussen können, wurden vermindert.
- Da kein manuelles Erfassen der Antworten für die elektronische Auswertung nötig war, wurden Übertragungsfehler vermieden.
- Der Einsatz von Filterfragen, von deren Beantwortung es abhing, welche weiteren Fragen gestellt wurden, konnte problemlos realisiert werden (zu Filterfragen vgl. Bortz und Döring, 2006, S. 244).[95]

Der Fragebogen war von Anfang Februar bis Mitte April 2008 über die Internetseite `www.befrager.de` auf einer speziellen URL verfügbar. Er wurde von 40 der 84 an der Untersuchung beteiligten Lehrpersonen vollständig ausgefüllt. Es wurden keine Probleme bei der Bearbeitung festgestellt. Gründe, warum der Fragebogen von 44 Untersuchungsteilnehmern nicht bzw. nicht vollständig ausgefüllt wurde, sind nicht bekannt.
Der in Anhang E vollständig dargestellte Fragebogen ist in die folgenden sieben Bereiche aufgeteilt:

1. Einleitung
2. Fragen zur Lehrkraft und zum Unterricht
3. Fragen zum Kompetenzmodell
4. Fragen zu den Unterrichtsaufgaben
5. Fragen zum Puck-System
6. Fragen zum Kompetenztest
7. Allgemeine Fragen zu kompetenzorientiertem Unterricht

In der Einleitung wurden Informationen und Hinweise zur Bearbeitung des Fragebogens bereitgestellt. Es wurde auf die Anonymität der Angaben hingewiesen und die Bereiche 2 bis 7 wurden stichpunktartig vorgestellt. Die Dauer der Beantwortung des Fragebogens wurde mit ca. 20 Minuten angegeben. Die Fragen zur Lehrkraft und zum Unterricht dienen einerseits der Beschreibung der Stichprobe und des durchgeführten Unterrichts, sie ermöglichen aber andererseits auch eine differenzierte Datenanalyse. So können zum Beispiel die Antworten der Lehrkräfte, die die Materialien nur in einem kurzen Zeitraum eingesetzt haben, mit denen der Lehrpersonen, die die Materialien länger einsetzten, verglichen werden.
Die oben genannten Bereiche 3 bis 6 stellen den Kern des Fragebogens dar. Die

[95] Mit Hilfe von Filterfragen konnten einzelne Fragebogenbereiche für bestimmte Lehrpersonen gezielt ein- und ausgeblendet werden. So wurden zum Beispiel nur Lehrpersonen, die den Kompetenztest im Unterricht eingesetzt hatten, zu diesem befragt.

Angaben der Lehrpersonen zum bereitgestellten Kompetenzmodell, zu den zugehörigen Aufgaben, zur visuellen Programmiersprache Puck und zum Kompetenztest wurden konzipiert, um die Forschungsfragen zwei und drei zu operationalisieren. Anschließend wurden im Bereich 7 noch Fragen gestellt, die zu einer allgemeineren Einschätzung von kompetenzorientiertem Informatikunterricht nützlich sind.

**Auswertung des Fragebogens** Im Rahmen der Auswertung wird davon ausgegangen, dass Lehrkräfte, als Experten für Unterricht, über diesen reflektieren sowie die Leistungen ihrer Schülerinnen und Schüler angemessen einschätzen können. Die erhobenen Daten der 40 vollständig ausgefüllten Fragebögen werden in den folgenden Abschnitten vorgestellt und sind im Anhang F geordnet dargestellt. Bei geschlossenen Fragen wird jeweils angegeben, wie häufig die jeweils vorgegebenen Antwortmöglichkeiten gewählt wurden. Bei den Prozentangaben muss beachtet werden, dass entsprechend der kleinen Stichprobengröße mehrere Prozentpunkte für eine einzelne Person stehen. Bei den Antworten zu den offenen Fragen wird jeweils angegeben, welche Lehrkraft diese Angabe machte (Bezeichnung mit L1 bis L40), um so die Antworten zu verschiedenen offen formulierten Fragen miteinander in Beziehung setzen zu können.[96]
Bei der Auswertung des Fragebogens wurden statistische Standardmethoden eingesetzt. Die Bewertung statistischer Hypothesen erfolgt anhand des p-Wertes. Dabei wird bei einem p-Wert von $p<0{,}05$ von einem signifikanten und bei $p<0{,}01$ von einem sehr signifikanten Ergebnis gesprochen (vgl. Bortz und Döring, 2006, S. 494). Ergebnisse mit einem p-Wert von $p\geqq 0{,}05$ werden als nicht signifikant (n. s.) bezeichnet. Außerdem wird in Abhängigkeit vom jeweiligen Skalenniveau die Stärke einer Beziehung mit dem Korrelationskoeffizienten r angegeben. Bei $|r|\approx 0{,}1$ wird von einem schwachen, bei $|r|\approx 0{,}3$ von einem mittleren und bei $|r|\approx 0{,}5$ von einem starken Zusammenhang gesprochen (vgl. Sedlmeier und Renkewitz, 2008, S. 221). Für dichotome nominalskalierte und ordinalskalierte Daten wird Spearmans Rangkorrelationskoeffizient angegeben, bei intervallskalierten Daten wird der Pearsonsche Produkt-Moment-Korrelations-Koeffizient berechnet (vgl. Bühl und Zöfel, 2000, S. 241 ff.). Beim Vorliegen von Intervallskalenniveau werden zur Darstellung der deskriptiven Statistik Mittelwert (MEAN) und Standardabweichung (SD) angegeben.

[96] Die Aussagen wurden im Originallaut wiedergegeben (lediglich Rechtschreibfehler wurden korrigiert).

## 8.2 Beschreibung der Stichprobe und des durchgeführten Unterrichts

**Beschreibung der Stichprobe** Der Fragebogen wurde von 39 Lehrpersonen aus elf Ländern Deutschlands und von einem Lehrer aus der Schweiz beantwortet. Acht Lehrkräfte arbeiteten in Thüringen, elf Lehrpersonen arbeiteten in Hessen, aus den anderen Bundesländern beantworteten jeweils bis zu fünf Lehrpersonen den Fragebogen. Eine Verzerrung der Ergebnisse durch die größere Anzahl der Lehrerinnen und Lehrer aus Thüringen und Hessen konnte statistisch nicht belegt werden. Alle Teilnehmer haben den Fragebogen differenziert beantwortet, es gab keine Lehrperson, die alles positiv bzw. alles negativ eingeschätzt hat. Dies spricht für eine sorgfältige Bearbeitung des Fragebogens.
In Abbildung 32 sind Alterstruktur und die Verteilung zwischen Informatiklehrerinnen und Informatiklehrern dargestellt. Ein Vergleich mit der Gesamtpopulation der Informatiklehrkräfte wurde nicht durchgeführt, da auf entsprechende Vergleichsdaten nicht zugegriffen werden konnte. Die statistischen Angaben sind den von Fothe u. a. (2006, S. 60) vorgestellten Ergebnissen ähnlich. Es besteht kein Grund zu der Vermutung, dass männliche, weibliche, jüngere oder ältere Lehrpersonen überrepräsentiert waren.
Ca. zwei Drittel der Lehrpersonen besitzen eine Lehrbefähigung für Informatik, über die Qualifikation der restlichen Untersuchungsteilnehmer ist nichts bekannt. Ein Lehrer befand sich während der Untersuchung im Referendariat. 92,5 % der Lehrkräfte unterrichteten Informatik bereits seit mindestens 6 Jahren, 57,5 % seit mindestens 10 Jahren. Somit verfügte der Großteil der Lehrkräfte über eine langjährige Erfahrung im Fach.

**Beschreibung des durchgeführten Unterrichts** Drei Lehrpersonen setzten die Materialien bei Hauptschülern, acht bei Realschülern, 29 bei Gymnasiasten ein.[97] Gründe für die verhältnismäßig geringe Beteiligung von Real- und Hauptschullehrern sind nicht bekannt. Der Unterricht wurde in 25,0 % der Fälle in der 8. Jahrgangsstufe, in 45,0 % der Fälle in der 9. Jahrgangsstufe und in 22,5 % der Fälle in der 10. Jahrgangsstufe durchgeführt. Je einmal wurden nach den bereitgestellten Materialien in der 7. und in der 11. Jahrgangsstufe unterrichtet.[98]
Wie Abbildung 33 zeigt, gibt es große Unterschiede bei der Anzahl der Unterrichtsstunden, die im Rahmen der Untersuchung erteilt wurden (MEAN=18.1 (SD=8.8)). Während eine Lehrperson nur zwei Stunden unterrichtete, setzte eine

[97] Ein Gymnasiallehrer machte keine Angabe zum Schulzweig der Schülerinnen und Schüler, es wird davon ausgegangen, dass er Gymnasiasten unterrichtete.
[98] Eine Lehrerin machte keine Angabe zur unterrichteten Jahrgangsstufe.

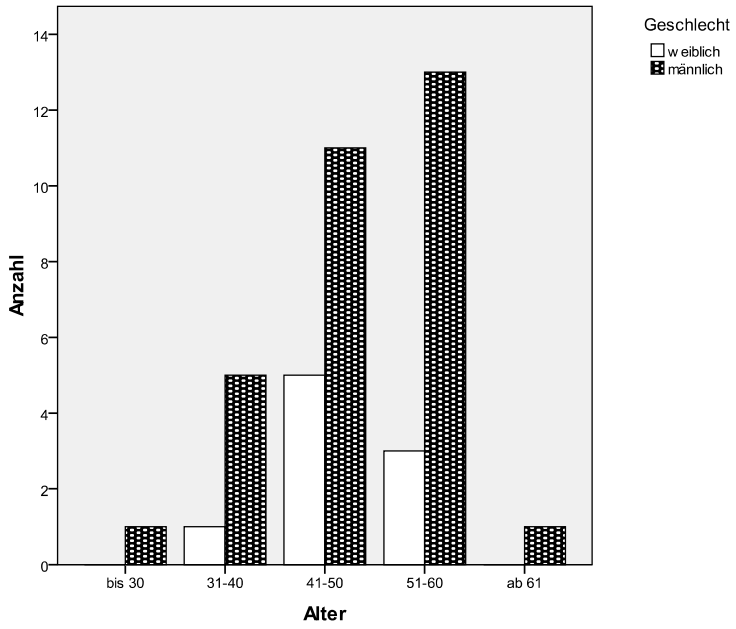



Abbildung 32: Verteilung von Geschlecht und Alter bei der untersuchten Stichprobe in der Hauptuntersuchung (N=40)

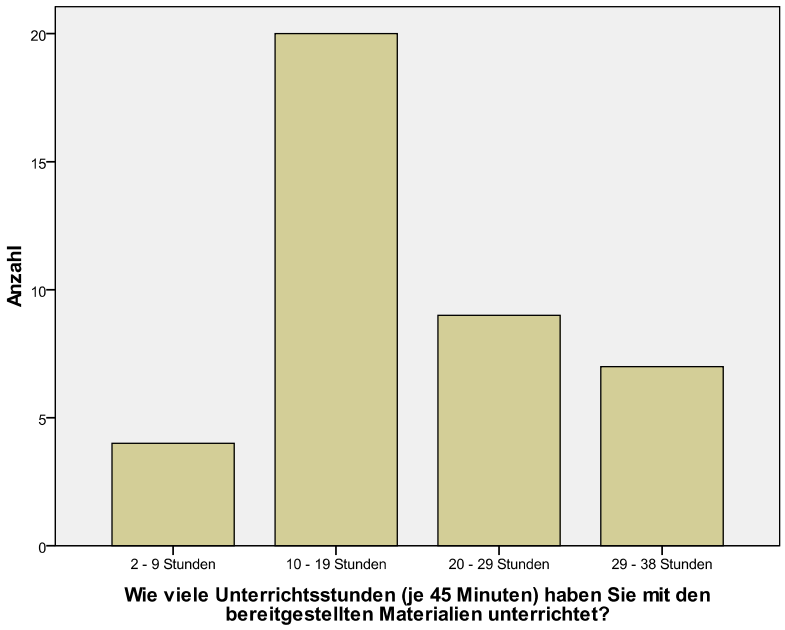



Abbildung 33: Anzahl der Unterrichtsstunden, in denen nach den bereitgestellten Materialien unterrichtet wurde (Darstellung in Gruppen, N=40)

andere die Materialien in 38 Unterrichtsstunden ein. Im Untersuchungsszenario war angegeben, dass die bereitgestellten Materialien ein Halbjahr lang zwei Stunden pro Woche eingesetzt werden sollten. Geht man von 20 Unterrichtswochen pro Halbjahr aus, so wären das 40 Unterrichtsstunden. Ein Großteil der Lehrpersonen hat die Materialien in weniger als 20 Stunden, also weniger als die Hälfte der empfohlenen Zeit eingesetzt. Dies ist insbesondere bei der Beurteilung des verwendeten Tests problematisch, da in der geringen Unterrichtszeit vermutlich nicht alle nötigen Kompetenzen erworben werden konnten. An einzelnen Stellen der Auswertung wird deshalb zwischen den Lehrkräften, die die Materialien in weniger als 20 Unterrichtstunden und jenen, die die Materialien in mindestens 20 Unterrichtsstunden eingesetzt haben, unterschieden. 70 % der Lehrpersonen benoteten den durchgeführten Unterricht. Auch die Benotung des Unterrichts bzw. des Kompetenztests kann sich auf dessen Ergebnisse auswirken. Dementsprechend wird bei den detaillierteren Analysen teilweise zwischen Lehrpersonen, die den Unterricht bzw. den Kompetenztest benoteten und solchen, die dies nicht taten, unterschieden.

## 8.3 Ergebnisse zum Kompetenzmodell

**Ausrichten des Unterrichts am Kompetenzmodell** Den Darstellungen im Abschnitt 3.4 entsprechend ist es sinnvoll, Ziele eines kompetenzorientierten Unterrichts anhand der im Modell geforderten Kompetenzen auszuwählen. Eine Übersicht zur Häufigkeit der Ausrichtung des Unterrichts am Kompetenzmodell in der Hauptuntersuchung gibt Abbildung 34. Ein Großteil der Lehrpersonen hat den Unterricht oft (60,0 %) bzw. immer (12,5 %) am Kompetenzmodell ausgerichtet. Die restlichen Lehrerinnen und Lehrer richteten den Unterricht gelegentlich (17,5 %) bzw. selten (10,0 %) am Kompetenzmodell aus. Als Gründe für das seltene Ausrichten des Unterrichts am Kompetenzmodell wurden je einmal Zeitgründe (L3), die Komplexität des Kompetenzmodells (L10), schulorganisatorische Gründe (L28) und eine Ausrichtung des Unterrichts an Schülerwünschen unter Beachtung des Lehrplans (L37) angegeben.
Zwei Lehrer (L3, L10) sahen das Überführen des Kompetenzmodells in Unterricht als problematisch an und forderten eine stärkere Strukturierung der Inhalte. Zwei Lehrkräfte (L5, L11) gaben an, dass das Überführen des Kompetenzmodells in Unterricht gut möglich war.
Kompetenzmodelle sollen nach Klieme u. a. (2003, S. 71) beschreiben, welche Lernergebnisse von Schülerinnen und Schülern bestimmter Altersstufen in einem Fach erwartet werden können. Evtl. kann ein Lehrplan, ein didaktisches System, ein auf der Grundlage von Bildungsstandards entwickeltes Unterrichtsszenario oder

etwas Ähnliches bei der Strukturierung der Inhalte behilflich sein (zum Begriff des didaktischen Systems vgl. Brinda, 2004; zu Unterrichtsszenarien auf der Grundlage von Bildungsstandards vgl. Fothe, 2008, S. 113).
Die Lehrpersonen wurden entsprechend der im Abschnitt 7.3 ermittelten Nutzungsvarianten nach der Häufigkeit des Einsatzes des Kompetenzmodells zur Unterrichtsplanung befragt. Weiterhin wurde gefragt, wie häufig das Kompetenzmodell zur Einschätzung des Unterrichts eingesetzt wurde. Nachfolgend wird dargestellt, wie viele Lehrpersonen verschiedenen Nutzungsgruppen zugeordnet werden konnten:

- Ein Lehrer (2,5 %) setzte das Kompetenzmodell nicht zur Unterrichtsplanung oder -einschätzung ein.
- 7,5 % der Lehrpersonen setzten das Kompetenzmodell lediglich zur Planung und/oder Einschätzung des Unterrichts am Anfang bzw. am Ende des Schulhalbjahres ein.
- 35,0 % der Lehrpersonen setzten das Kompetenzmodell zur Planung und/oder Einschätzung von größeren Unterrichtssequenzen, aber nicht zur Planung oder Einschätzung einzelner Unterrichtsstunden ein.
- 47,5 % der Lehrpersonen setzten das Kompetenzmodell zur Planung und/oder Einschätzung einzelner Unterrichtsstunden ein.
- 7,5 % der Lehrpersonen setzten das Kompetenzmodell zur Planung und/oder Einschätzung jeder Unterrichtsstunde ein.

Insgesamt konnte festgestellt werden, dass das entwickelte Kompetenzmodell von 97,5 % der Lehrpersonen in irgendeiner Weise zur Planung und von 90,0 % in irgendeiner Weise zur Einschätzung von Unterricht eingesetzt wurde.
Aus dem verhältnismäßig häufigen Einsatz des Modells durch einen Großteil der Lehrkräfte wird geschlussfolgert, dass das entwickelte Kompetenzmodell „Algorithmen“ als Grundlage von kompetenzorientierten Informatikunterricht geeignet ist.

**Einschätzung der Übersichtlichkeit und Einteilung des Kompetenzmodells**
Wie in Abbildung 35 dargestellt ist, bezeichneten 85,0 % der Lehrkräfte das Kompetenzmodell als übersichtlich oder sehr übersichtlich. Lediglich zwei Lehrer gaben Gründe für ihre negative Einschätzung der Übersichtlichkeit an. Ein Lehrer (L10), der das Kompetenzmodell nicht zur Unterrichtsplanung oder Einschätzung eingesetzt hatte, empfand die Einteilung als zu kompliziert und zu komplex. Ein anderer Lehrer (L27) gab an, dass die Schwierigkeitsstufen von den Schülerinnen und Schülern teilweise anders empfunden wurden. Aufgrund dieser unspezifischen

Angaben und der großen Anzahl der positiven Einschätzungen der Übersichtlichkeit wurden keine Konsequenzen für die Weiterentwicklung bzw. Vereinfachung des Kompetenzmodells abgeleitet. 87,5 % der Lehrkräfte hielten die Einteilung des Kompetenzmodells in verschiedene Stufen für sinnvoll. 62,5 % hielten die Einteilung des Kompetenzmodells in verschiedene Komponenten für sinnvoll. Die dargestellten Einschätzungen werden so interpretiert, dass die Mehrzahl der Lehrkräfte mit der Komplexität des Kompetenzmodells umgehen konnte. Es wird vermutet, dass die im Kapitel 7 vorgestellten Modifikationen nach der ersten Untersuchung zu einer besseren Übersichtlichkeit und somit auch zum häufigeren Einsatz des Modells zur Planung und Einschätzung von Unterricht beigetragen haben.

**Vorstellen des Kompetenzmodells im Unterricht** 47,5 % der Lehrkräfte stellten den Schülerinnen und Schülern das Kompetenzmodell im Unterricht kurz vor, zwei Lehrer (5,0 %) taten dies sogar ausführlich. Ein vermuteter Zusammenhang, dass hauptsächlich die Lehrpersonen, die den Kompetenztest einsetzten, auch das Kompetenzmodell im Unterricht vorstellten, konnte statistisch nicht bestätigt werden (r=-0,064; n. s.). Inwieweit das Vorstellen des Kompetenzmodells im Unterricht nützlich ist, kann eine Fragestellung für weitere Forschungsarbeiten sein. Bei der Entwicklung von Kompetenzmodellen sollte aufgrund der dargestellten Ergebnisse zumindest die Möglichkeit, dass das Modell im Unterricht präsentiert werden kann, in Betracht gezogen werden.

**Anmerkungen zum Kompetenzmodell** Ein Gymnasiallehrer (L35) bemerkte, dass das Behandeln des Algorithmusbegriffs sowie das Darstellen von Algorithmen als Pseudocode, Struktogramm oder Programmablaufplan, in der 8. Jahrgangsstufe nicht einfach ist, weil den Schülerinnen und Schülern die Abstraktion schwer fällt. Es kann darauf verwiesen werden, dass in den Empfehlungen der GI (2008, S. 33) zu Bildungsstandards Informatik das fachlich korrekte Verwenden eines einfachen Algorithmusbegriffs und der Eigenschaften von Algorithmen sowie das formale Darstellen von Algorithmen zum Ende der 10. Jahrgangsstufe als Mindeststandards gefordert sind und diese Kompetenzen deshalb ins Kompetenzmodell aufgenommen wurden (vgl. Abschnitt 5.2).
Sieben Lehrpersonen (L2, L5, L11, L12, L13, L33, L36) gaben in verschiedener Art und Weise positives Feedback zum Kompetenzmodell. Aufgrund des überwiegend positiven Feedbacks, der geringen Anzahl der kritischen Bemerkungen und der fehlenden Verbesserungsvorschläge wurde das Kompetenzmodell nach der Auswertung des Fragebogens nicht verändert.

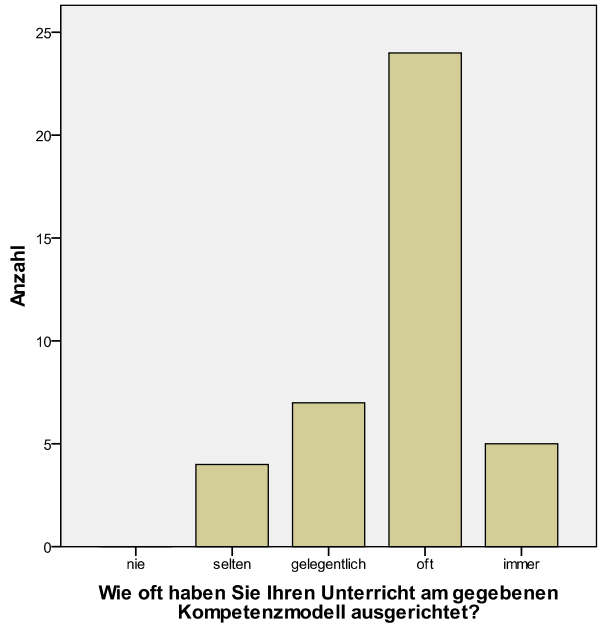



Abbildung 34: Einsatz des Kompetenzmodells in der Hauptuntersuchung (N=40)

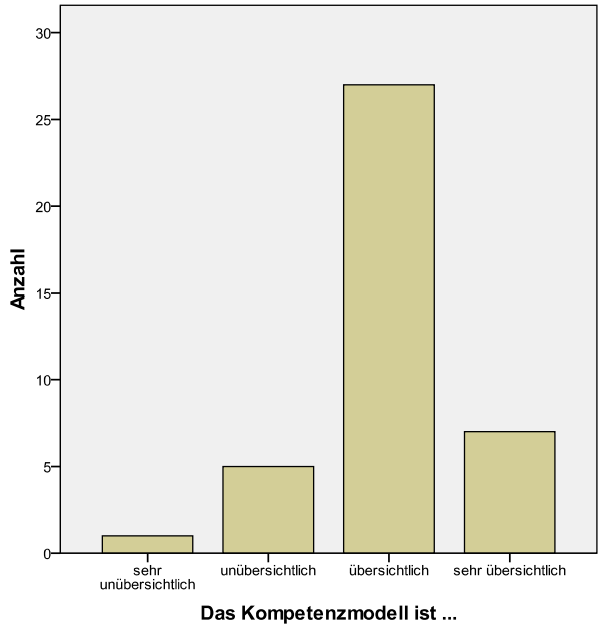



Abbildung 35: Einschätzung der Übersichtlichkeit des Kompetenzmodells in der Hauptuntersuchung (N=40)

**Allgemeine Einschätzung von Kompetenzmodellen** Generell wurde die Einteilung von Kompetenzmodellen in verschiedene Komponenten und Stufen von der Mehrzahl der Lehrpersonen als nützlich eingeschätzt. 90,0 % beurteilten die Einteilung von Kompetenzmodellen in verschiedene Komponenten als sinnvoll. Außer einem Lehrer (2,5 %), der keine Einschätzung zu dieser Frage abgab, beurteilten alle Befragten das Einteilen in verschiedene Stufen als sinnvoll. Damit werden die im Abschnitt 5.6 genannten Empfehlungen, Kompetenzmodelle in Komponenten und Stufen zu unterteilen, durch die Ergebnisse dieser Befragung untermauert.
60,0 % der Lehrpersonen sprachen sich für Fortbildungen zum Einsatz von Kompetenzmodellen im Informatikunterricht aus. 65,0 % der Lehrpersonen gaben an, dass sie Kompetenzmodelle auch in anderen Bereichen des Informatikunterrichts einsetzen würden. Ein Interesse der Mehrzahl der Lehrkräfte an kompetenzorientiertem Unterricht konnte also festgestellt werden. Es stellt sich die Frage wie Lehrerinnen und Lehrer bei ihrem pädagogischem Handeln weiter unterstützt werden können. Fothe (2008, S. 113) sieht es als wesentlich an, Unterrichtsszenarien auf der Grundlage von Bildungsstandards exemplarisch zu erarbeiten und zu erproben. Diese können in Fortbildungen, Lehrbüchern und Fachzeitschriften dargestellt werden. Aber auch Internetplattformen bieten prinzipiell die Möglichkeit der Weitergabe erfolgreich erprobter Konzepte (vgl. z. B. das Wiki der Zentrale für Unterrichtsmedien im Internet e.V., 2008).

## 8.4 Ergebnisse zu den Beispiel- und Unterrichtsaufgaben

**Einsatz der Aufgaben** Wie Abbildung 36 zeigt, wurden die bereitgestellten Aufgaben von einem Großteil der Lehrpersonen oft (60,0 %) bzw. immer (15,0 %) zur Unterrichtsvorbereitung eingesetzt. 20,0 % der Lehrpersonen setzten die Aufgaben gelegentlich, jeweils eine Lehrkraft setzte die Aufgaben selten (2,5 %) bzw. nie (2,5 %) ein. Dies spricht für eine generelle Akzeptanz der Beispiel- und Unterrichtsaufgaben durch die Lehrpersonen. Hätten sich die Aufgaben nicht bewährt, wären sie nicht so häufig eingesetzt worden.
Aufgaben die unterschiedlichen Kompetenzstufen zugeordnet sind, wurden von 80,0 % der Lehrerinnen und Lehrer zur Differenzierung eingesetzt (15,0 % oft; 30,0 % gelegentlich; 35,0 % selten). Im Unterricht der Lehrkräfte, bei denen Schülerinnen und Schüler im Test Kompetenzen der Stufen II bzw. III nachweisen konnten, wurde wie erwartet häufiger mit Hilfe von Aufgaben, die unterschiedlichen Stufen zugeordnet waren, differenziert (r=0,532; p=0,009). Es wird geschlussfolgert, dass besonders dann, wenn von Lernenden Kompetenzen auf verschiedenen Stufen erworben wurden, auch das Bereitstellen von Aufgaben, die unterschiedlichen Stufen zugeordnet sind, sehr nützlich zur Differenzierung im Unterricht war.

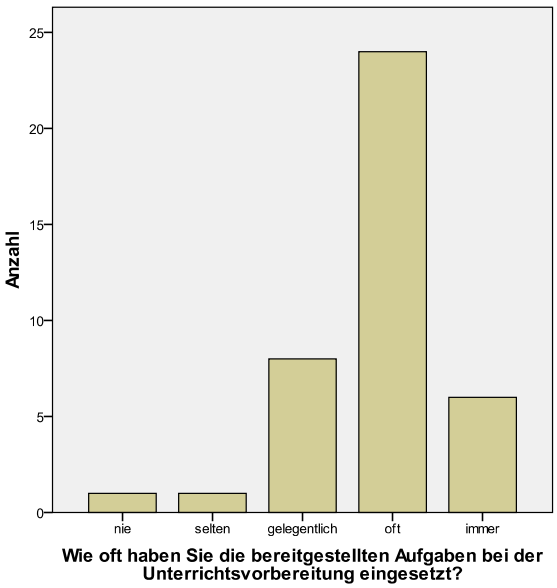



Abbildung 36: Einsatz der bereitgestellten Aufgaben zur Unterrichtsvorbereitung (N=40)

**Einschätzung der Einordnung der Aufgaben in die Komponenten und Stufen des Kompetenzmodells** Aufgrund der Ergebnisse der Voruntersuchung war zu den Unterrichtsaufgaben angegeben, welche Kompetenzen mit ihnen erworben werden können (vgl. Abschnitt 7.4). 82,5 % der Befragten stimmten der Aussage zu, dass diese Einordnung der Aufgaben in die Komponenten und Stufen des Kompetenzmodells eine wichtige Hilfe war. Dies sollte auch bei den Entwicklungen weiterer Kompetenzmodelle und zugehöriger Aufgaben aufgegriffen werden, indem jeweils die zum Lösen einer Aufgabe benötigten Kompetenzen angegeben werden. 70,0 % der Lehrkräfte stimmten der Aussage zu, dass die geforderten Kompetenzen durch die Aufgaben angemessen abgebildet wurden. 22,5 % der Lehrerinnen und Lehrer waren unentschlossen. Auf die Erläuterungen von drei Lehrpersonen (7,5 %), die der genannten Aussage nicht zustimmten, wird im Folgenden eingegangen:

- Ein Lehrer (L10) gab an, dass für das Aufteilen eines Programms in Prozeduren Aufgaben fehlten.
  Algorithmen die ein oder mehrere Unterprogramme mit Parametern enthalten, wurden entsprechend der Argumentation im Abschnitt 5.3 im Allgemeinen der Stufe III zugeordnet. Neun der 14 Unterrichtsaufgaben, die den Komponenten B, C und D auf Stufe III zugeordnet waren, enthielten explizit den Hinweis bzw. die Anforderung, Prozeduren zu verwenden (vgl.

Kohl, 2008a, S. 61 ff.). Das Erstellen weiterer Aufgaben wäre sinnvoll.

- Eine Lehrerin (L29), die die Materialien in einer Hauptschulklasse einsetzte, merkte an, dass Schülerinnen und Schüler Probleme mit den gegebenen Handlungsvorschriften und mit mathematischen Aufgaben hatten.
  Dass mathematische Aufgaben bei der Einführung in die Programmierung problematisch sein können, wurde bereits im Abschnitt 4.3 beschrieben. Dementsprechend wurden – wie im Abschnitt 6.3 dargestellt – auch ausreichend andere Aufgaben zur Verfügung gestellt. Welche Probleme mit den gegebenen Handlungsvorschriften auftraten, kann aufgrund der Aussage der Lehrerin nicht nachvollzogen werden. Deshalb werden keine Konsequenzen abgeleitet.
- Ein Lehrer (L32) gab an, dass grundlegende Kompetenzen zum Algorithmusbegriff durch die Aufgaben nicht angemessen abgebildet wurden. Er stellte dar, dass die Eigenschaft der Terminiertheit fehlte.
  Die Eigenschaft Terminiertheit wurde in den Materialien explizit nicht als verpflichtende Eigenschaft eines Algorithmus angegeben.[99] Die Aussage dieses Lehrers deutet darauf hin, dass es günstig war, den Lehrpersonen keinen expliziten Erwartungshorizont zu den Aufgaben zur Verfügung zu stellen, da die einzelnen Lehrenden in ihrem Unterricht spezifische Schwerpunkte setzen.

Die dargestellten Ergebnisse lassen zusammen mit der Diskussion zu den kritischen Anmerkungen den Schluss zu, dass die geforderten Kompetenzen durch die bereitgestellten Aufgaben hinreichend abgebildet werden.

**Zusammenhang zwischen dem Ausrichten des Unterrichts am Kompetenzmodell, dem Einsatz der Aufgaben und der Einordnung der Aufgaben in die Komponenten und Stufen des Kompetenzmodells** Es konnte eine starker Zusammenhang zwischen der Häufigkeit des Ausrichten des Unterrichts am Kompetenzmodell und der Häufigkeit des Einsatzes der Aufgaben im Unterricht festgestellt werden ($r=0{,}531$; $p<0{,}001$). Außerdem ergab die statistische Auswertung der Befragung einen starken Zusammenhang zwischen der Häufigkeit des Ausrichtens des Unterrichts am Kompetenzmodell und der Zustimmung zu der Aussage, dass die Einordnung der Aufgaben in die Komponenten und Stufen des Kompetenzmodells eine wichtige Hilfe war ($r=0{,}564$; $p<0{,}001$). Weiterhin konnte

[99] Ein Herzschrittmacher ist ein Beispiel für ein Gerät, das mit einem nicht terminierenden Algorithmus arbeitet. Pomberger (1999, S. 518) erläutert in diesem Zusammenhang Folgendes: *Ein Algorithmus heißt abbrechend, wenn er die gesuchten Größen nach endlich vielen Schritten liefert, andernfalls heißt der Algorithmus nicht abbrechend.*

ein mittlerer Zusammenhang zwischen der Häufigkeit des Einsatzes der Aufgaben im Unterricht und der Zustimmung zu der Aussage, dass die Einordnung der Aufgaben in die Komponenten und Stufen des Kompetenzmodells eine wichtige Hilfe war, festgestellt werden (r=0,389; p=0,013).
Diese signifikanten Zusammenhänge werden als Beleg dafür interpretiert, dass es wichtig war, zu einem Kompetenzmodell auch Aufgaben bereitzustellen, die in die Komponenten und Stufen des Modells eingeordnet sind. Dies bekräftigt die im Abschnitt 5.6 genannte Forderung, zu einem Kompetenzmodell entsprechende Aufgaben zur Verfügung zu stellen.

**Anmerkungen zu den Aufgaben** Weitere Anmerkungen zur Aufgabensammlung werden nachfolgend vorgestellt und diskutiert:

- Ein Lehrer (L14), der die Materialien in der 7. Jahrgangsstufe eingesetzt hatte, forderte mehr und leichtere Unterrichtsaufgaben.
  Da die Aufgabensammlung für die Jahrgangsstufen 8 bis 10 konzipiert ist, wird auf diese Forderung nicht weiter eingegangen.

- Eine Lehrerin (L11) bemerkte, dass es Probleme mit den Aufgaben lediglich dann gab, wenn die Schülerinnen und Schüler die Aufgabenstellungen nicht richtig gelesen hatten.
  Diese Aussage spricht für die nach der Voruntersuchung vorgenommene Aufteilung der Testaufgaben in einzelne Teilaufgaben (vgl. Abschnitt 7.5). Zur Entwicklung der Kompetenz „komplexere Aufgabenstellungen richtig zu lesen“ kann auf die Expertise zur Förderung von Lesekompetenz von Artelt u. a. (2007) verwiesen werden.

- Eine Lehrerin (L12), beanstandete, dass die Aufgaben „von Anfang an zu anspruchsvoll“ waren. Sie verdeutlichte ihre Aussage an der „ersten Aufgabe“, bei der bereits vorausgesetzt wurde, dass die Schülerinnen und Schüler Schleifen verstehen können.
  Die Lehrerin hatte die Materialien lediglich zwei Stunden im Unterricht eingesetzt. Mit der „ersten Aufgabe“ ist vermutlich die Beispielaufgabe zur Komponente C Stufe I gemeint (vgl. Kohl, 2008a, S. 9). Im Material gibt es genügend Unterrichtsaufgaben zur Einführung, die ohne das Verwenden von Kontrollstrukturen gelöst werden können (vgl. z. B. Kohl, 2008a, S. 26). Evtl. sollte bei weiteren Aufgabensammlungen explizit angegeben werden, ob die Reihenfolge der Aufgaben auch für den Einsatz im Unterricht geeignet ist.

- Ein Lehrer (L19) nannte den Verbesserungsvorschlag, dass zu einzelnen Aufgaben Zusatzaufgaben höherer Kompetenzstufen formuliert werden könnten.

Bei der Erarbeitung weiterer Aufgaben sollte geprüft werden, inwieweit sich dieser Vorschlag umsetzen lässt.

- Ein Lehrer (L22) gab an, dass die Unterrichtsaufgabe „Summieren“ fehlerhaft war, nannte den Fehler selbst aber nicht (vgl. zur Aufgabe Kohl, 2008a, S. 43).
  Bei einer im Anschluss an die Untersuchung durchgeführten ausführlichen Analyse der Aufgabe konnte kein Fehler in der Aufgabenstellung festgestellt werden.
- Ein Lehrer (L24) erweiterte die Aufgabe „Rom“ um die verkürzte Schreibweise römischer Zahlen (zur Aufgabe vgl. Kohl, 2008a, S. 51).
  Diese Zusatzaufgabe erscheint sinnvoll.
- Positives Feedback für die Aufgabensammlung wurde von sechs Lehrpersonen (L3, L5, L10, L27, L35, L36) formuliert, dabei wurden insbesondere Aufgaben mit grafischen Ausgaben hervorgehoben.

Insgesamt werden die dargestellten Ergebnisse so interpretiert, dass ein Großteil der Lehrpersonen die bereitgestellten Aufgaben sinnvoll als Grundlage von kompetenzorientiertem Unterricht einsetzen konnte.

## 8.5 Ergebnisse zum Kompetenztest

**Einsatz des Kompetenztest** 57,5 % der Lehrkräfte gaben an, den Kompetenztest im Unterricht eingesetzt zu haben. Im Folgenden werden Gründe, warum der Test nicht eingesetzt wurde, analysiert:

- Ein Lehrer (L10) hatte Probleme mit der Einteilung der Kompetenzen und empfand den Test als zu lang. Er gab an, dass der Test viel kürzer und übersichtlicher sein sollte.
  Da aus den Angaben des Lehrers nicht hervorgeht, welche Probleme er bei der Einteilung der Kompetenzen hatte, soll im Folgenden nur auf die Länge des Tests, also auf den Umfang der gegebenen Aufgabenstellungen eingegangen werden. In der Voruntersuchung waren von Schülerinnen und Schülern einzelne Aufgabenteile überlesen worden. Daraufhin waren die Testaufgaben in Teilaufgaben untergliedert worden. Weiterhin waren für schriftlich zu bearbeitende Aufgabenstellungen Antwortfelder bereitgestellt worden (vgl. Abschnitt 7.5). Hierdurch erhöhte sich der Platzbedarf des Kompetenztests einer Stufe auf drei bis vier Seiten. Dass es in der hier dargestellten Hauptuntersuchung beim Kompetenztest keine Anmerkungen

zu überlesenen Teilaufgaben gab, wird als positives Ergebnis dieser Veränderung interpretiert. Das Ziel weiterer Untersuchungen könnte es sein, einen kürzeren Kompetenztest zu konstruieren, der es weiterhin ermöglicht, die Kompetenzen verschiedener Stufen zu erfassen.

- Zwei Lehrpersonen (L5, L22) gaben an, dass sie den Test in ihrem Unterricht nicht einsetzen konnten, weil nicht für jeden Lernenden ein Computer zur Verfügung stand. Beide schlugen vor, den Test in zwei Bereiche zu unterteilen, von denen einer mit und einer ohne Computer bearbeitet werden soll. Damit wäre es möglich, eine Klasse in zwei Gruppen aufzuteilen. Eine Gruppe könnte dann am Computer arbeiten, eine zweite Gruppe könnte ohne Computer arbeiten und nach der Hälfte der Zeit könnten die Gruppen die Arbeitsplätze tauschen. Somit könnte der Test auch problemlos in Klassen geschrieben werden, in denen nicht für jeden Lernenden ein Computer zur Verfügung steht.
  Prinzipiell gibt es bereits eine solche Zweiteilung im Kompetenztest, denn für die Bearbeitung der Aufgaben der Komponenten A und B ist kein Computer erforderlich. Allerdings stellt sich die Frage, ob für die Bearbeitung dieser Aufgaben genau so viel Zeit benötigt wird, wie für das Bearbeiten der Aufgaben zu den Komponenten C und D.[100] Bei der Entwicklung von weiteren Kompetenztests sollte überprüft werden, inwieweit das vorgeschlagene Unterteilen in zwei Testbereiche möglich ist.

- Fünf Lehrpersonen (L18, L20, L21, L24, L39) gaben an, dass sie den Test aus Zeitgründen nicht eingesetzt haben. Ein Lehrer (L4) und eine Lehrerin (L12) setzten den Test aufgrund geringer Lernfortschritte der Schülerinnen und Schüler nicht im Unterricht ein. Drei Lehrkräfte (L15, L23, L28) gaben an, dass sie den Test nach dem Ausfüllen des Fragebogens noch einsetzen werden. Die zehn genannten Lehrkräfte hatten die Materialien jeweils in weniger als 25 Unterrichtsstunden eingesetzt.
  Aufgrund der aktuellen Lage des Informatikunterrichts in den Ländern konnten nicht alle Lehrpersonen die Materialien, wie im Untersuchungsszenario gefordert, in zwei Stunden pro Woche während eines Halbjahres einsetzen. Aus den Aussagen, dass der Kompetenztest aus Zeitgründen oder aufgrund geringer Lernfortschritte nicht eingesetzt wurde bzw. dass er noch eingesetzt werden soll, wurden keine Konsequenzen abgeleitet. Die Vermutung, dass hauptsächlich die Lehrpersonen, die die Materialien über einen längeren Zeit-

[100] Eventuell können Schülerinnen und Schüler bei der Arbeit ohne Computer auch einige Aufgabenteile zu den Komponenten C und D lösen. Es muss aber sichergestellt werden, dass die Reihenfolge der Bearbeitung der Aufgaben, die mit bzw. ohne Computer zu lösen sind, keine Vor- bzw. Nachteile mit sich bringt.

raum im Unterricht eingesetzt haben, den Test nutzten, konnte statistisch nicht belegt werden (r=-0,090; n. s.).

Der Kompetenztest wurde größtenteils dann eingesetzt, wenn der Unterricht auch benotet wurde (r=0,430; p=0,006). Obwohl durch einen unbenoteten Test Versagensängste vermieden werden können, setzten lediglich drei Lehrpersonen (7,5 %) den Kompetenztest im unbenoteten Unterricht ein. Evtl. war es im benoteten Unterricht einfacher, eine Testsituation herzustellen.

**Durchführung und Bewertung des Kompetenztests** Aufgrund der positiven Erfahrungen aus der Voruntersuchung wurden für den Kompetenztest 90 Minuten Bearbeitungszeit empfohlen. Dies wurde von elf Lehrpersonen umgesetzt. Acht Lehrpersonen ließen ihren Schülerinnen und Schülern zwischen 45 und 80 Minuten Zeit zur Bearbeitung des Tests.[101] Es gab keine negativen Bemerkungen zur Bearbeitungsdauer. Dies spricht dafür, dass die Testaufgaben der im Abschnitt 3.3 dargestellten Bedingung „individuelle Bearbeitbarkeit in überschaubarer Zeit“ genügen (vgl. Blum, 2006, S. 18). Dementsprechend wird die Empfehlung, den Test in 90 Minuten bearbeiten zu lassen, beibehalten.
Entsprechend den im Abschnitt 7.5 vorgestellten Erkenntnissen aus der Voruntersuchung wurden für die Auswahl der Aufgaben folgende vier Varianten genannt (vgl. Kohl, 2008a, S. 70):

1. ***Variante:*** *Die Lehrperson legt eine Stufe fest, die von der ganzen Klasse einheitlich bearbeitet werden soll.*
2. ***Variante:*** *Die Lehrperson legt individuell für jede Schülerin und jeden Schüler (oder für Schülergruppen) fest, welche Stufe bearbeitet werden soll.*
3. ***Variante:*** *Jede Schülerin und jeder Schüler legt für sich fest, welche Stufe bearbeitet werden soll.*
4. ***Variante:*** *Jede Schülerin und jeder Schüler legt für sich für jede der Komponenten die Stufe fest, die bearbeitet werden soll.*

Des Weiteren wurde aufgrund der Erfahrungen der Voruntersuchung auf die Möglichkeit hingewiesen, Schülerinnen und Schülern, die die Aufgaben einer Stufe vor Ablauf der Zeit vollständig bearbeitet haben, die Aufgaben einer weiteren Stufe zur Verfügung zu stellen (vgl. Kohl, 2008a, S. 70). Nachfolgend wird analysiert,

[101] Alle Lehrpersonen, die den Lernenden weniger als 90 Minuten Zeit zur Verfügung gestellt hatten, legten die im Test zu bearbeitende Stufe einheitlich fest. Vier Lehrkräfte machten keine Angabe zur Bearbeitungsdauer des Tests.

wie häufig die oben genannten Varianten zur Wahl der Stufen des Kompetenztests genutzt wurden:[102]

- 19 Lehrpersonen legten eine Stufe fest, die von der ganzen Klasse einheitlich bearbeitet wurde. Zwölfmal wurde von allen Schülerinnen und Schülern Stufe I des Kompetenztests bearbeitet, Stufe II wurde viermal bearbeitet, Stufe III wurde dreimal bearbeitet.
- Keine Lehrkraft hat individuell für jede Schülerin und jeden Schüler (oder für Schülergruppen) festgelegt, welche Stufe bearbeitet werden soll.
- Bei einer Lehrerin (L29) haben die Schülerinnen und Schüler selbstständig eine zu bearbeitende Stufe gewählt.
- Zwei Lehrpersonen (L26, L27) stellten ihren Schülerinnen und Schülern frei, bei jeder Komponente eine Stufe zu wählen.

Dass den Schülerinnen und Schülern zumindest von drei Lehrpersonen die Möglichkeit gegeben wurde, die zu bearbeitende Stufe selbst zu wählen, wird als Indiz für die Durchführbarkeit der beiden letztgenannten Varianten gedeutet.[103] Da von den meisten Lehrpersonen einheitlich eine von allen Lernenden zu bearbeitende Stufe festgelegt wurde, wäre es sinnvoll, einen einzigen Kompetenztests zu entwickeln, mit denen die Kompetenzen mehrerer Stufen überprüft werden können. Ob das beim Inhaltsbereich „Algorithmen“ möglich ist und wie ein solcher Kompetenztest strukturiert werden könnte, ist offen.[104] Nur acht der 23 Lehrkräfte die den Kompetenztest eingesetzt haben, hatten zumindest 20 Stunden nach den Materialien unterrichtet. Zehn der zwölf Lehrpersonen, die Stufe I des Tests bearbeiten ließen, hatten die Materialien weniger als 20 Stunden im Unterricht eingesetzt. Ein großer Teil der Schülerinnen und Schüler hatte somit aufgrund des geringen Stundenumfangs lediglich die Möglichkeit, Stufe I zu erreichen. Dies muss bei der Analyse der Ergebnisse beachtet werden.

Zu jeder Teilaufgabe des Kompetenztests waren die zu erreichenden Punkte angegeben (vgl. Kohl, 2008a, S. 71 ff.). Insgesamt wurde so die in Tabelle 6 auf Seite 111 beschriebene Punktverteilung umgesetzt. Zur Bewertung des Kompetenztests wurden folgende Hinweise gegeben (vgl. Kohl, 2008a, S. 70):

[102] Ein Lehrer (L16) machte keine Angabe zur Wahl der Stufen. Er gab an, dass er den Test nicht komplett übernehmen konnte, da er einige Aufgaben im Unterricht behandelt hatte, keine Computer verwenden konnte und nur 60 Minuten zur Verfügung hatte.

[103] Nach Auffassung des Autors erscheint es sinnvoll, bei den beiden letztgenannten Varianten den Schülerinnen und Schülern das Kompetenzmodell (anhand von Beispielaufgaben) vorzustellen, so dass diese ihre Fähigkeiten selbst einschätzen können.

[104] Nach Auffassung des Autors stellt sich vor allem die Frage, wie in einer Aufgabe je nach Kompetenzstufe einfache bzw. komplexe Algorithmen bereitgestellt werden können.

- *Der Kompetenztest wird von der Lehrperson kontrolliert und bewertet.*
- *Es fließen bei jeder der Komponenten A, B, C und D nur die Aufgaben einer Stufe in die Bewertung ein. (Die Auswahl regelt die Lehrperson.)*
- *Wir schlagen vor, dass Sie die Einschätzung nach folgenden Kriterien vornehmen:*
  - *Um Stufe I zu erlangen, müssen mindestens 18 Punkte erreicht werden.*
  - *Um Stufe II zu erlangen, müssen mindestens 36 Punkte erreicht werden.*
  - *Um Stufe III zu erlangen, müssen mindestens 54 Punkte erreicht werden.*
- *Die Lehrerin bzw. der Lehrer legt fest, ob und ggf. wie der Test benotet werden soll.*

Fothe (2008, S. 112) vertritt in Bezug auf Aufgaben in der Abiturprüfung Informatik die Position, dass *bewusst knappe Hinweise sinnvolle Spielräume beim Korrigieren und Bewerten der Schülerantworten auf der Grundlage des wirklich stattgefundenen Unterrichts ermöglichen.* Da es in der hier vorgestellten Hauptuntersuchung keine Vorgaben zum durchzuführenden Unterricht in Form eines Lehrplans oder in anderer Form gab, sondern nur ein Kompetenzmodell, zugehörige Aufgaben und ein passendes Programmiersystem zur Verfügung gestellt wurden, sollte die Bewertung des Tests durch die Lehrpersonen auf Grundlage des wirklich stattgefundenen Unterrichts erfolgen. Dementsprechend musste von den Lehrerinnen und Lehrern selbstständig ein Erwartungshorizont zu den Aufgaben erstellt werden.

**Einschätzung des Kompetenztests durch die Lehrpersonen** Einige Lehrkräfte gaben Feedback zum Kompetenztest, das im Folgenden dargestellt und analysiert werden soll:

- Drei Gymnasiallehrer (L3, L34, L35) empfanden die Aufgaben zur Komponente A auf Kompetenzstufe I als zu leicht.
  Die drei Lehrer hatten festgelegt, dass alle Lernenden Stufe I bearbeiten sollten. Durch die eigenständige Wahl der zu bearbeitenden Stufe hätten die Schülerinnen und Schüler die Komplexität der zu bearbeitenden Aufgabe selbst bestimmen können. Weiterhin wäre es möglich, den Lehrkräften, die der ganzen Klasse die gleichen Aufgaben zur Bearbeitung geben, freizustellen, bei den verschiedenen Komponenten unterschiedliche Stufen für die Schülerinnen und Schüler festzulegen. So könnte z. B. an einer Schule ein Kompetenztest geschrieben werden, in dem Aufgaben der Komponente A

auf Stufe III, der Komponente B auf Stufe I, der Komponente C auf Stufe I und der Komponente D auf Stufe II bearbeitet werden.

- Eine Lehrerin (L29) gab an, dass die Ansprüche für Haupt- und Realschüler zum Teil zu komplex waren. Sie stellte dar, dass ihre Schülerinnen und Schüler einige Kompetenzen auf Stufe III erworben hatten, anderen aber nicht. Als Beispiel gab sie an, dass die Lernenden Probleme hatten, ein Beispiel für einen Algorithmus zu nennen, der beim Benutzen eines Mobiltelefons auftritt.
  Die Lehrerin hatte das Kompetenzmodell nicht im Unterricht vorgestellt. Die Schülerinnen und Schüler wählten die im Test zu bearbeitende Stufe selbst aus. Die von der Lehrerin als Beispiel angegebene Aufgabe stammt aus dem Kompetenztest Stufe III Komponente A und stellt somit „umfassendere“ Kompetenzen dar. Evtl. haben sich die Schülerinnen und Schüler bei der eigenständigen Wahl der Kompetenzstufe überschätzt. Es kann darauf hingewiesen werden, dass bereits die Kompetenzen der Stufe I die Mindeststandards der GI-Empfehlungen umsetzen.
- Zwei Gymnasiallehrer (L34, L35) gaben an, dass die Testaufgabe zur Komponente D auf Stufe I zum Kopfrechnen „verführt“.[105]
  Dies war bei der Aufgabenstellung durchaus erwünscht, denn auf Stufe I sollten die richtigen Ausgaben des Programms einfach zu bestimmen sein, damit die Schülerinnen und Schüler ihre Programme leicht auf Funktionalität überprüfen können.
- Drei Lehrpersonen (L16, L32, L36) gaben ausdrücklich positives Feedback zum Kompetenztest.

Keine der Lehrpersonen berichtete über Probleme der Lernenden bei der Bearbeitung des Kompetenztests. Dies wird als Indiz dafür gedeutet, dass die Testaufgaben, der im Abschnitt 3.3 dargestellten Bedingung „prinzipielle Verstehbarkeit ohne

[105] Ausschnitt aus der Aufgabe:
*Marcos jüngere Schwester Anne lernt im Mathematikunterricht, Flächen von Rechtecken und Quadraten zu berechnen. In einer Hausaufgabe hat sie Folgendes berechnet:*
*Quadrat 1: Seitenlänge: 4 cm Flächeninhalt: 16 cm²*
*Rechteck 1: Seitenlänge a: 5 cm Seitenlänge b: 3 cm Flächeninhalt: 20 cm²*
*Anne wünscht sich ein Programm, mit dem sie überprüfen kann, ob sie bei ihren Hausaufgaben richtig gerechnet hat.*
*...*
*Testen Sie das Programm mit den Werten von Annes Hausaufgabe. Welche Flächeninhalte gibt Ihr Programm aus?*
*Quadrat 1:*
*Rechteck 1:*

externe Unterstützung“ genügen (vgl. Blum, 2006, S. 18). Erwähnenswert erscheint außerdem, dass von keiner einzigen Lehrkraft beanstandet wurde, dass kein einheitlicher Erwartungshorizont bereitgestellt wurde. Das Bewertungskriterium, nach dem mindestens 75 % der auf der jeweiligen Stufe möglichen Punkte erreicht werden mussten, wurde größtenteils als „genau richtig“ eingeschätzt. Es gab keine Anmerkungen zu Problemen bei der Korrektur des Tests. Diese Erkenntnisse werden als Indiz dafür gedeutet, dass die Testaufgaben auch der dritten im Abschnitt 3.3 dargestellten Bedingung „verlässliche Korrigierbarkeit“ genügen (vgl. Blum, 2006, S. 18).

**Ergebnisse der Lernenden beim Kompetenztest** Zwei Lehrer, die den Kompetenztest einsetzten, machten keine Angaben zu den Ergebnissen ihrer Schülerinnen und Schüler und werden deshalb im Folgenden nicht betrachtet. Nach den freiwilligen Angaben der restlichen 21 Lehrpersonen haben am Kompetenztest insgesamt 356 Schülerinnen und Schüler teilgenommen. Durchschnittlich wurden 17,0 Schülerinnen und Schüler einer Klasse getestet (SD=7,9). 57,6 % der Schülerinnen und Schüler haben aufgrund der Festlegung der Lehrperson lediglich Stufe I des Tests bearbeitet. Dadurch konnten diese maximal Kompetenzen der Stufe I nachweisen. Weiterhin schränkt die Bewertung des Kompetenztests durch unterschiedliche Lehrpersonen anhand individueller Erwartungshorizonte die Vergleichbarkeit der Testergebnisse ein. Trotzdem werden die Ergebnisse der Lernenden beim Kompetenztest nachfolgend vorgestellt und analysiert.
Wie in Tabelle 11 dargestellt ist, haben insgesamt 12,1 % der Schülerinnen und Schüler Stufe III, 14,6 % Stufe II und 54,5 % Stufe I erlangt. Rechnerisch ergibt sich, dass 18,8 % der Schülerinnen und Schüler keine Kompetenzstufe erreichten. Eine Analyse der Angaben der Lehrpersonen, die den Kompetenztest benoteten, ergab, dass bei dieser Gruppe lediglich 7,7 % der Schülerinnen und Schüler keine Kompetenzstufe erreichten.[106] Bei unbenoteten Tests gab es mit 30,2 % eine sehr viel größere Anzahl an Schülerinnen und Schülern, die keine Stufe erlangten (vgl. Tabelle 11). Auch wenn sich Klieme u. a. (2003, S. 83 f.) deutlich gegen eine Verwendung von standardbezogenen Testverfahren für Zwecke der Benotung und Zertifizierung aussprechen, kann doch festgestellt werden, dass bei einem benoteten Test zumindest in dieser Untersuchung von einem größeren Anteil der Lernenden Kompetenzen nachgewiesen werden konnten. Warum dies so ist, kann aufgrund der erhobenen Daten nicht festgestellt werden.[107]
Zwei Lehrpersonen (L13, L26) merkten an, dass die geringe Erfolgsquote der Schülerinnen und Schüler im Kompetenztest auf die geringe Zeitspanne, in der nach

[106] Ein Lehrer (L17) machte keine Angaben zur Benotung des Tests.
[107] Die Vermutung, dass die Benotung des Tests motivierend wirkte, liegt nahe.

dem Kompetenzmodell unterrichtet wurde, zurückzuführen ist. Die Ergebnisse des Tests wurden aufgrund dieser Aussagen hinsichtlich der Dauer des Einsatzes der Materialien untersucht. Wie in Tabelle 11 dargestellt ist, erreichten jeweils 20 - 30 % der Schülerinnen und Schüler, bei denen die Materialien mindestens 20 Stunden im Unterricht eingesetzt wurden, die Stufe I, II und III (vgl. Tabelle 11). Dementsprechend ist es mit Hilfe der festgelegten Stufen unter der Voraussetzung eines entsprechenden Unterrichts möglich, ein differenziertes Abbild des Leistungsspektrums der Zielpopulation zu geben. Bei Betrachtung der Gymnasiasten ergab die statistische Analyse einen starken Zusammenhang zwischen Dauer des Einsatzes der Materialien und der durchschnittlich von den Schülerinnen und Schülern im Test erreichten Kompetenzstufe (r=0,605; p=0,022).[108] Dies deutet darauf hin, dass die in Stufe II und III genannten Kompetenzen wirklich vertiefte bzw. umfassendere Kompetenzen darstellen, die erst nach einer längeren Beschäftigung mit dem Thema erworben werden. Die große Anzahl der mindestens 20 Stunden nach den gegebenen Materialien unterrichteten Schülerinnen und Schüler, die Stufe III erreichten, wird als Beleg dafür interpretiert, dass die dort angegeben Kompetenzen bei entsprechendem Unterricht nicht zu komplex für Lernende dieser Altersklasse sind.
Wenn lediglich die Schülerinnen und Schüler betrachtet werden, die mindestens 20 Stunden nach den gegebenen Materialien unterrichtet wurden und deren Test benotet wurde, so ergibt sich, dass all diese Lernenden zumindest Stufe I im Test erreicht haben (vgl. Tabelle 11). Dies ist ein Indiz dafür, dass die in Kompetenzstufe I angegebenen Anforderungen unter den genannten Voraussetzungen wirklich von allen Schülerinnen und Schülern erwartet werden können. Da die Kompetenzen am Inhaltsbereich „Algorithmen“ der GI-Empfehlungen (Jahrgangsstufen 8 bis 10) orientiert sind, deuten diese Ergebnisse auch darauf hin, dass die dort genannten Kompetenzen unter der Voraussetzung eines entsprechenden Unterrichts wirklich Minimalstandards sein können (vgl. Abschnitt 2.5 und 5.2).
Es konnten keine signifikanten Zusammenhänge zwischen der Benotung des Tests bzw. der Dauer des Unterrichts und Alter, Geschlecht, Unterrichtserfahrung oder Schulform der Lehrpersonen festgestellt werden. In der Gruppe „mindestens 20 Stunden Unterricht und benoteter Test“ waren neben fünf Gymnasialklassen auch zwei Realschulklassen und eine Hauptschulklasse vertreten. Dies wird als Indiz dafür interpretiert, dass das in Anlehnung an die GI-Empfehlungen entwickelte Kompetenzmodell „Algorithmen“ an allen Schulformen eingesetzt werden kann und dass die GI-Empfehlungen – zumindest für den untersuchten Inhaltsbereich

[108] Um eine hinsichtlich ihres Leistungsniveaus möglichst homogene Stichprobe zu untersuchen, wurden die Ergebnisse der Schülerinnen und Schüler am Gymnasium betrachtet. Da lediglich die Testergebnisse von insgesamt fünf Real- und Hauptschulklassen vorlagen, war eine seperate Betrachtung dieser Gruppen aus statistischen Gründen nicht sinnvoll.

und unter den genannten Bedingungen – die Merkmale „Verbindlichkeit für alle“ und „Realisierbarkeit“ aufweisen (vgl. Abschnitt 2.5).
Ein starker Zusammenhang zwischen der unterrichteten Jahrgangsstufe und der Dauer des Einsatzes der Materialien im Unterricht wurde festgestellt (r=0,471; p=0,002). Bei Schülerinnen und Schülern höherer Jahrgangsstufen wurden die bereitgestellten Materialien durchschnittlich länger im Unterricht eingesetzt. Insgesamt erzielten die Lernenden höherer Jahrgangsstufen durchschnittlich auch bessere Ergebnisse im Kompetenztest (vgl. Tabelle 11).[109] Es wird vermutet, dass dies neben dem durchschnittlich längeren Einsatz der Materialien auch dem Umstand geschuldet ist, dass die Schülerinnen und Schüler der höheren Klassenstufen auf ein breiteres Wissensspektrum zurückgreifen können.

Auch wenn sich diese Untersuchung auf das Lehrerhandeln konzentrierte, konnten durch die freiwilligen Angaben zu den Ergebnissen der Schülerinnen und Schüler beim Kompetenztest Erfahrungen zu diesem gesammelt werden. Aufgrund des Umstandes, dass durch die Lehrperson häufig die zu bearbeitende Stufe festgelegt wurde und dadurch ein Großteil der Lernenden Stufe I bearbeitete und aufgrund der nicht standardisierten Bewertung der Tests durch die Lehrpersonen ist es allerdings fraglich, inwieweit die dargestellten Ergebnisse vergleichbar sind. Für ausführlichere Analysen wäre es sinnvoll, die Ergebnisse einer größeren Anzahl von Schülerinnen und Schüler anhand einheitlicher Kriterien auszuwerten, um so weitere Rückschlüsse auf Kompetenzmodell und Aufgaben ziehen und Konsequenzen ableiten zu können.

**Allgemeine Einschätzung von Kompetenztests** 87,5 % der Lehrkräfte gaben an, dass Kompetenztests hilfreich sind, um die Kompetenzen der Schülerinnen und Schüler einzuschätzen. 40,0 % der Befragten schätzen Kompetenztests als hilfreich ein, um die Wirksamkeit ihres pädagogischen Handelns zu beurteilen. Insgesamt beurteilten 95,0 % der Lehrpersonen Kompetenztests als hilfreich. Diese positive Bewertung muss natürlich vor dem Hintergrund des Untersuchungsszenarios interpretiert werden, denn der Test wurde von den Lehrpersonen freiwillig dem erteilten Unterricht entsprechend durchgeführt, von ihnen selbstständig ausgewertet und zog keine Konsequenzen nach sich. Kompetenztests mit der Zielstellung des Bildungsmonitorings bzw. der Schulevaluation werden vermutlich in anderen Szenarien eingesetzt. Mit ihnen ist nach den Aussagen von Klieme u. a. (2003, S. 10) Individualdiagnostik aus methodischen Gründen meist nicht möglich. Dementsprechend bleibt es offen, ob solche Tests ähnlich positiv bewertet werden.

[109] Die beiden Lehrkräfte, die die Materialien in den Jahrgangsstufen 7 und 11 einsetzten, nutzten den Kompetenztest nicht im Unterricht.

| | Teilnehmer N (%) | Stufe III N (%) | Stufe II N (%) | Stufe I N (%) | keine Stufe (rechnerisch ermittelt) N (%) |
|---|---|---|---|---|---|
| Komptenztest insgesamt | 356 (100 %) | 43 (12,1 %) | 52 (14,6 %) | 194 (54,5 %) | 67 (18,8 %) |
| benoteter Test | 208 (100 %) | 37 (17,8 %) | 24 (11,5 %) | 131 (63,0 %) | 16 (7,7 %) |
| unbenoteter Test | 129 (100 %) | 6 (4,7 %) | 28 (21,7 %) | 56 (43,4 %) | 39 (30,2 %) |
| mindestens 20 Stunden Unterricht | 127 (100 %) | 37 (29,1 %) | 28 (22,1 %) | 36 (28,3 %) | 26 (20,5 %) |
| weniger als 20 Stunden Unterricht | 229 (100 %) | 6 (2,6 %) | 24 (10,5 %) | 158 (69,0 %) | 41 (17,9 %) |
| mindestens 20 Stunden Unterricht und benoteter Test | 78 (100 %) | 37 (47,4 %) | 24 (30,8 %) | 17 (21,8 %) | 0 (0,0 %) |
| Jahrgangsstufe 8 | 114 (100 %) | 2 (1,8 %) | 14 (12,3 %) | 56 (49,1 %) | 42 (36,8 %) |
| Jahrgangsstufe 9 | 173 (100 %) | 21 (12,1 %) | 17 (9,8 %) | 126 (72,8 %) | 9 (5,2 %) |
| Jahrgangsstufe 10 | 69 (100 %) | 20 (29,0 %) | 21 (30,4 %) | 12 (17,4 %) | 16 (23,2 %) |

Tabelle 11: Ergebnisse der Schülerinnen und Schüler verschiedener Gruppen beim Kompetenztest

## 8.6 Ergebnisse zur visuellen Programmiersprache Puck

**Verringern von Syntaxfehlern durch Puck** Alle befragten Lehrpersonen gaben an, dass das Verringern der Syntaxfehler durch Puck günstig für den Anfangsunterricht war. Die Hälfte der Lehrkräfte hatte durch die Arbeit mit der visuellen Programmiersprache sogar mehr Zeit für die individuelle Förderung einzelner Schülerinnen und Schüler. Wie Abbildung 37 zeigt, stimmten 90,0 % der Lehrpersonen der Aussage zu, dass auch leistungsschwächere Schülerinnen und Schüler durch die Arbeit mit dem Puck-System Erfolge erzielen konnten. Insgesamt werden die Angaben der Lehrpersonen so interpretiert, dass durch das Verhindern von Syntaxfehlern in Puck und die damit einhergehende bessere Betreuung durch die Lehrkraft auch leistungsschwächere Lernende bessere Ergebnisse im Informatikunterricht erzielen können. Dies ist vor allem dann wichtig, wenn Mindeststandards umgesetzt und somit alle Schülerinnen und Schüler in Deutschland Kompetenzen informatischer Bildung erwerben sollen*, gleich, welcher Herkunft sie sind, welchen sozialen Hintergrund sie haben und welche möglichen Behinderungen sie aufweisen [...]* (vgl. GI, 2008, S. 3).

**Vorbereitung auf textuelles Programmieren durch Puck** Lediglich fünf Prozent der Lehrpersonen gaben an, dass den Schülerinnen und Schülern durch die Arbeit mit dem Puck-System ein falsches Bild von Programmierung vermittelt wird. Sie begründeten diese Einschätzung nicht. 70,0 % der Lehrkräfte waren gegenteiliger Meinung. Es stellt sich die Frage, wie die Lernenden nach der Arbeit mit Puck mit textuellen Programmiersprachen umgehen. 60,0 % der Lehrkräfte waren der Auffassung, dass die Schülerinnen und Schüler durch den Unterricht mit dem Puck-System auf die Arbeit mit einer textuellen Programmiersprache gut vorbereitet worden sind. Die Lehrkräfte gaben hier nur eine persönliche Einschätzung ab, konnten sich aber nicht sicher sein, ob diese Einschätzung auch wirklich zutrifft, da sie nicht auf entsprechende Erfahrung mit einem Übergang zu einer textuellen Programmiersprache zurückgreifen können. Trotzdem weisen die Einschätzungen der Lehrpersonen zu den beiden zuletzt analysierten Aussagen darauf hin, dass es durch die Arbeit mit Kontrollstrukturen, Anweisungen und Prozeduren in Puck möglich ist, wesentliche strukturelle Grundlagen für das textuelle Programmieren zu legen.

**Motivation durch die Arbeit mit Puck** Lediglich 15,0 % der Lehrpersonen, stimmten der Aussage zu, dass sich für viele Schülerinnen und Schüler die Arbeit mit dem Puck-System als schwierig erwies. 80,0 % der Lehrkräfte waren gegenteiliger Ansicht. Wie Abbildung 38 zeigt, waren 77,5 % der Lehrerinnen und Lehrer

sogar der Meinung, dass das Puck-System Schülerinnen und Schüler motiviert, sich in den Informatikunterricht aktiv einzubringen. Ein einfaches motivierendes Programmiersystem im Unterricht ist nicht nur für Lehrende und Lernende attraktiv, es kann auch dazu beitragen, dem Fachkräftemangel in der Informatik entgegenzuwirken, indem es das Interesse der Schülerinnen und Schüler für diese Disziplin bereits in der Schule weckt. Die Antworten der Lehrkräfte geben Anlass zur Annahme, dass Puck ein solches motivierendes Werkzeug ist.
Da die Lehrkräfte diejenigen sind, die über einen Einsatz im Unterricht bestimmen, sind deren Einschätzungen wichtig für die Verbreitung eines Werkzeuges. Der Einsatz der visuellen Programmiersprache Puck motivierte 90,0 % der befragten Lehrkräfte für die Unterrichtsvorbereitung und -durchführung. Der Aussage, dass der Einarbeitungsaufwand in das Puck-System für Informatiklehrerinnen und -lehrer sehr hoch ist, stimmte lediglich ein Lehrer (2,5 %) zu. 87,5 % der Lehrkräfte wollten Puck im nächsten Schuljahr wieder einsetzten. Die generell positiven Einschätzungen der Befragten gaben keinen Anlass grundlegende Veränderungen an Puck vorzunehmen.

**Unterrichtsmaterialien zu Puck** Der Aussage, dass es nicht genügend geeignete Unterrichtsmaterialien für den Einsatz des Puck-Systems gibt, stimmten 40,0 % der Lehrpersonen zu, 52,5 % der Lehrkräfte hielten die vorhanden Materialien für ausreichend, 7,5 % waren unentschlossen. Die Forderung nach weiteren Unterrichtsmaterialien kann also bei einer größeren Anzahl an Lehrkräften festgestellt werden. Fraglich ist, ob alle diese Lehrenden auch alle vorhandenen Materialien kannten. Um diese noch besser zu verbreiten wurden sie auf der Internetseite `www.ipuck.de` bereitgestellt. Weiterhin wurde es auf dieser Webseite ermöglicht, Materialien und Programme zur visuellen Programmiersprache auszutauschen. Zusätzlich könnten weitere Unterrichtsmaterialien entwickelt werden.

**Frühstmöglicher Einsatz von Puck** Die Frage, ab welcher Jahrgangsstufe Puck frühestens eingesetzt werden kann, wurde differenziert beantwortet. 60,0 % der Lehrpersonen halten einen Einsatz ab Jahrgangsstufe 8 bzw. 9 für sinnvoll. 35,0 % könnten sich vorstellen, Puck bereits in der 7. Jahrgangsstufe oder sogar noch früher einzusetzen. Der Lehrer (L14), der die bereitgestellten Materialien in der 7. Klasse eingesetzt hatte, gab an, dass *das Prinzip und das System Puck sehr gut ankamen*. Eine Erprobung des Systems in den Jahrgangsstufen 5 bis 7 wäre somit ein Ansatz für weitere Forschungsarbeiten.

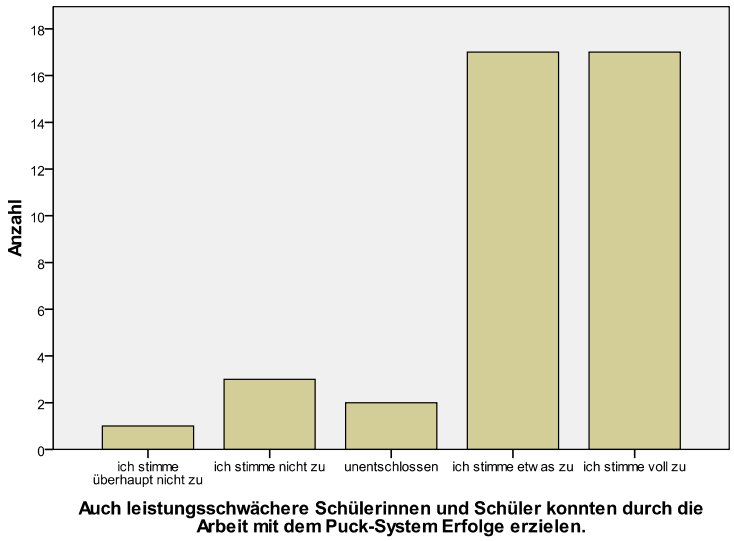



Abbildung 37: Einschätzung der Lehrpersonen zu der Aussage „Auch leistungsschwächere Schülerinnen und Schüler konnten durch die Arbeit mit dem Puck-System erfolge erzielen.“

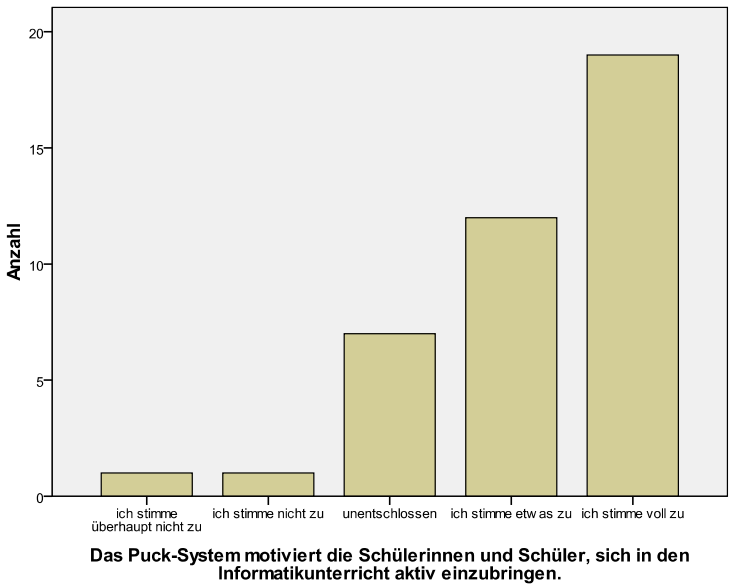



Abbildung 38: Einschätzung der Lehrpersonen zu der Aussage „Das Puck-System motiviert die Schülerinnen und Schüler, sich in den Informatikunterricht aktiv einzubringen.“

**Anmerkungen zu Puck** Die Anmerkungen der Lehrerinnen und Lehrer zum Puck-System bestanden zum größten Teil aus Funktionswünschen und Verbesserungsvorschlägen (vgl. Anhang F). Fünf Lehrpersonen (L22, L24, L28, L29, L35) gaben außerdem positives Feedback zu Puck. Die meisten der genannten Vorschläge wie z. B. das Kopieren von Bausteinen, das Generieren von Struktogrammen, das Umsetzen von Funktionen, Real- und String-Variablen, waren schon bei der Weiterentwicklung von Puck im Rahmen dieser Arbeit diskutiert und unter Beachtung der Forderung nach Einfachheit und dem Verzicht auf unnötige Funktionen nicht umgesetzt worden (vgl. Abschnitt 4.4 bzw. Arnold und Hartmann, 2007b, S. 180). Einige andere Verbesserungsvorschläge, wie in der Größe anpassbare Bildschirmbereiche, eine Linux- bzw. Macintoshversion und die Möglichkeit, Programme zu drucken erscheinen sinnvoll und würden die Einfachheit des Systems wohl kaum beeinflussen da sich dadurch an dessen Bedienung nichts Grundlegendes ändern würde. Sie sollten deshalb bei der zukünftigen Weiterentwicklung von Puck genauer analysiert werden. Trotz der verschiedenen Anregungen waren 72,5 % der Lehrkräfte der Ansicht, dass die Funktionalität von Puck für eine Einführung in die Programmierung ausreicht. Dies spricht dafür, das Grundkonzept der visuellen Programmiersprache beizubehalten.

Insgesamt war das Feedback der Lehrpersonen zu Puck nach der Erprobung im Unterricht größtenteils positiv. Es wird geschlussfolgert, dass sich die visuelle Programmiersprache zur Einführung in die Programmierung in den Jahrgangsstufen 8 bis 10 eignet.

## 8.7 Einordnung der Ergebnisse

Mit Bezug auf die zweite Forschungsfrage wird aus dem häufigen Einsatz und der generell positiven Beurteilung der bereitgestellten Materialien geschlussfolgert, dass Kompetenzmodell und Aufgaben als Grundlage kompetenzorientierten Unterrichts in der Sekundarstufe I geeignet sind (vgl. die Abschnitte 8.3, 8.4 und 8.5). Diese Ergebnisse sprechen auch für die in den Abschnitten 5.6 und 6.5 entwickelten Anforderungen zur Entwicklung von Kompetenzmodellen und zugehörigen Aufgaben.
Weiterhin wurde zur dritten Forschungsfrage dargelegt, dass sich die visuelle Programmiersprache Puck prinzipiell eignet, Schülerinnen und Schülern der Jahrgangsstufen 8 bis 10 eine Einführung in die Programmierung zu geben. Nachfolgend sollen zur Relativierung einige Punkte genannt werden, die die Verallgemeinerbarkeit der Ergebnisse auf den Informatikunterricht in deutschsprachigen Ländern einschränken.

- Weil die Teilnahme an der Untersuchung das selbstständige Einarbeiten in ein Programmiersystem und das Arbeiten mit einem Kompetenzmodell erforderte, ist anzunehmen, dass sich besonders engagierte und an der Fortentwicklung ihres Unterrichts interessierte Lehrpersonen angemeldet haben. Fraglich ist, ob die Materialien im Unterricht weniger engagierter Lehrkräfte ähnlich eingesetzt bzw. eingeschätzt worden wären.
- Unter den befragten Lehrerinnen und Lehrern gab es einen relativ großen Anteil an Gymnasiallehrkräften. Dies schränkt die Verallgemeinerbarkeit der Ergebnisse auf allgemeinbildende Schulen ein.
- Verpflichtender Informatikunterricht in den Jahrgangsstufen 8 bis 10 ist derzeit in den an der Untersuchung beteiligten Ländern nur gering verbreitet (vgl. hierzu Weeger, 2007). Es ist wahrscheinlich, dass der Unterricht zumindest in einem Teil der Schulen von den Schülerinnen und Schülern freiwillig belegt bzw. zumindest das Fach Informatik aktiv gewählt werden konnte. Damit war der Anteil der motivierten und am Fach interessierten Schülerinnen und Schülern bei der Durchführung des Kompetenztests vermutlich recht hoch. Der Unterricht bzw. der durchgeführte Kompetenztest wurde an einigen Schulen nicht benotet. Es ist daher möglich, dass die Schülerinnen und Schüler dem Fach bzw. dem Test nicht die gleiche Aufmerksamkeit schenkten wie anderen benoteten Fächern bzw. Tests. Die genannten Tatsachen schränken die Verallgemeinerbarkeit auf benoteten Informatikunterricht, der als Pflichtfach erteilt wird, ein.
- Das Kompetenzmodell ist programmiersprachenunabhängig formuliert. Durch die Arbeit mit Puck wurden Syntaxfehler verhindert, somit blieb bei einem Großteil der Lehrkräfte mehr Zeit für die individuelle Förderung einzelner Schülerinnen und Schüler (vgl. Abschnitt 8.6). Es bleibt offen, ob der Unterricht auch beim Verwenden eines anderen Programmiersystems zu ähnlichen Ergebnissen und Einschätzungen geführt hätte
- Obwohl der Fragebogen anonymisiert war, kann nicht ausgeschlossen werden, dass bei der Beantwortung der Fragen auch Effekte sozialer Erwünschtheit aufgetreten sind.

## 8.8 Zusammenfassung

In diesem Kapitel wurde die Hauptuntersuchung dargestellt, bei der sich 84 Lehrkräfte bereit erklärt hatten, das in dieser Arbeit entwickelte Kompetenzmodell und die zugehörige Aufgabensammlung unter Verwendung der visuellen Programmiersprache Puck im Unterricht zu erproben und anschließend einen Kompetenztest

durchzuführen. Am Ende des Halbjahres füllten 40 Lehrerinnen und Lehrer den in Anhang E dargestellten Fragebogen vollständig aus.
Trotz einzelner kritischer Anmerkungen wurden das Kompetenzmodell „Algorithmen“ und die zugehörigen Aufgaben durch die Lehrpersonen generell positiv beurteilt und häufig eingesetzt. Fast alle befragten Lehrpersonen setzten die Materialien ein, um Unterricht zu planen und einzuschätzen. Das Einordnen der Aufgaben in die Komponenten und Stufen des Kompetenzmodells wurde von den meisten Lehrkräften als wichtige Hilfe wahrgenommen. Der bereitgestellte Kompetenztest wurde von mehr als der Hälfte der Lehrpersonen eingesetzt.
Das Verwenden der visuellen Programmiersprache Puck im Unterricht gestaltete sich weitgehend als problemlos. Die Angaben der Lehrpersonen legten den Schluss nahe, dass das Verringern der Syntaxfehler durch Puck die Einstiegshürden in die Programmierung vermindert und Lehrpersonen so weniger korrigierend behilflich sein müssen. Die Ergebnisse der Befragung ergaben weiterhin, dass mit Puck ein motivierendes Werkzeug existiert, mit dem nach Einschätzung der Lehrkräfte auch leistungsschwächere Lernende erfolgreich arbeiten können, ohne ein falsches Bild vom Programmieren zu erhalten. Dies wird insbesondere dann wichtig, wenn Informatik als Unterrichtsfach für alle Schülerinnen und Schüler verpflichtend werden soll. Insgesamt sprechen die Ergebnisse für den gewählten Ansatz visueller Programmierung mit Bausteinen an allgemeinbildenden Schulen.

# 9 Ergebnisse und Ausblick

> *In der Wissenschaft gleichen wir alle nur den Kindern,*
> *die am Rande des Wissens hier und da einen Kiesel aufheben,*
> *während sich der weite Ozean des Unbekannten vor unseren Augen*
> *erstreckt.*
> (Sir Isaac Newton)[110]

Nachfolgend werden im Abschnitt 9.1 die Ergebnisse zu den im Kapitel 1 dargestellten wissenschaftlichen Fragestellungen dieser Arbeit präsentiert. Anschließend erfolgt im Abschnitt 9.2 ein Ausblick auf weitere Arbeiten.

## 9.1 Ergebnisse zu den wissenschaftlichen Fragestellungen dieser Arbeit

### Ergebnisse zur ersten Forschungsfrage

**Zur Entwicklung von Kompetenzmodellen für einen Bereich des Informatikunterrichts** In der vergleichsweise jungen Fachdidaktik Informatik liegen kaum gesicherte empirische Befunde zur Kompetenzentwicklung von Schülerinnen und Schülern vor. Auch auf größere Aufgabenpools zum Erfassen von Kompetenzen kann derzeit nicht zurückgegriffen werden. Die kürzlich vorgestellten GI-Empfehlungen zu Bildungsstandards Informatik (2008) wurden unter der Mitwirkung von vielen Fachdidaktikerinnen und Fachdidaktikern sowie Lehrerinnen und Lehrern erarbeitet. Sie legen Kompetenzen fest, über die Schülerinnen und Schüler am Ende der 7. bzw. 10. Jahrgangsstufe mindestens verfügen sollen und werden voraussichtlich Einfluss auf die weitere Entwicklung des Informatikunterrichts im deutschsprachigen Raum nehmen.
Um auch über die in den GI-Empfehlungen festgelegten Mindeststandards hinaus kompetenzorientierten Unterricht zu ermöglichen, wurde in dieser Arbeit auf Grundlage der dort für die Jahrgangsstufen 8 bis 10 im Inhaltsbereich „Algorithmen“ angegebenen minimalen Kompetenzen ein Kompetenzmodell entwickelt. Zur besseren Übersichtlichkeit und begründeten Vorwegnahme der Struktur eines

[110] zitiert nach Puntsch (1993, S. 997)

Kompetenztests wurden die Kompetenzen in vier Komponenten aufgeteilt. Um differenzierte Leistungsbeurteilung und individuelle Förderung von Schülerinnen und Schülern besser zu ermöglichen, wurden zu jeder Komponente jeweils Kompetenzen auf drei Stufen formuliert. Aufgrund einer Einordnung der Kompetenzen in die SOLO-Taxonomie wurde festgestellt, dass die Anzahl der Kompetenzen höherer Komplexität von Stufe I über Stufe II zu Stufe III ansteigt. Damit im Unterricht Aufgaben in beziehungsreichen Kontexten bereitgestellt werden können, wurden Beziehungen des Kompetenzmodells zu den Kompetenzen anderer Inhalts- und Prozessbereiche der GI-Empfehlungen aufgezeigt. In einem Reflexionsprozess wurden anschließend folgende acht Anforderungen an die Entwicklung eines Kompetenzmodells für einen Bereich der Informatik zusammengetragen:

1. Bei der Entwicklung des Kompetenzmodells relevante Forschungsergebnisse und die aktuelle Situation in Schulen beachten
2. Das Kompetenzmodell in verschiedene Komponenten unterteilen
3. Kompetenzen verschiedener Stufen angeben
4. Die Kompetenzen verständlich und konkret formulieren, ohne sich im Detail zu verlieren
5. Das Kompetenzmodell einheitlich, übersichtlich, kurz und handhabbar bereitstellen
6. Im Kompetenzmodell nicht auf spezielle Software, Programmiersprachen oder Entwicklungsumgebungen eingehen
7. Parallel zum Kompetenzmodell zugehörige Aufgaben bereitstellen
8. Für Verbreitung des Kompetenzmodells sorgen

Weiterhin ergab die Hauptuntersuchung, dass bei der Entwicklung von Kompetenzmodellen die Möglichkeit in Betracht gezogen werden sollte, dass das Modell den Lernenden im Unterricht präsentiert wird.

**Zur Entwicklung von Aufgaben für einen Bereich des Informatikunterrichts**
Im Inhaltsbereich „Algorithmen“ ist es aufgrund der Möglichkeit, im Unterricht verschiedene Anweisungen, Datentypen und Darstellungsformen zu verwenden, schwierig, konkrete Testaufgaben zu konstruieren, die unabhängig vom vorhergehenden Unterricht eingesetzt werden können. Um dem zu begegnen, wurden zunächst dem Untersuchungsdesign entsprechende Festlegungen zu den im Modell offen formulierten Kompetenzen getroffen, die es ermöglichten, einen Kompetenztest und weitere Aufgaben zu erstellen, die in den Untersuchungen dieser Arbeit von allen Lehrpersonen eingesetzt werden konnten. Zwölf Beispielaufgaben

verdeutlichen und konkretisieren die Kompetenzen und dienen somit der Verständigung zwischen Lehrenden und Lernenden über Komponenten und Stufen des Modells. Die 70 bereitgestellten Unterrichtsaufgaben dienen dem Erwerb von Kompetenzen. Anhand von zwölf Testaufgaben kann überprüft werden, inwieweit die im Modell angegebenen Kompetenzen von den Lernenden erreicht wurden. Beispiel- und Testaufgaben wurden zu jeder Komponente auf jeder Stufe des oben genannten Modells konstruiert, um die dort genannten Kompetenzen möglichst genau verdeutlichen bzw. überprüfen zu können. Aufgrund der Erfahrungen der Voruntersuchung wurden in der Hauptuntersuchung verschiedene Möglichkeiten zur Wahl der im Test zu bearbeitenden Kompetenzstufe präsentiert. Außerdem wurde ein Bewertungskriterium empfohlen, das es ermöglicht, Schülerinnen und Schülern das Erreichen einer niedrigeren Stufe zu bescheinigen, auch wenn der Test auf einer höheren Stufe bearbeitet wurde. Die Unterrichtsaufgaben wurden aufgrund der Erfahrungen der Voruntersuchung in die Komponenten und Stufen des Kompetenzmodells eingeordnet. Somit war es den Lehrpersonen in der Hauptuntersuchung möglich, für ihren Unterricht Aufgaben anhand der zu erwerbenden Kompetenzen auszuwählen. Um die Beziehungen zwischen Inhalts- und Prozesskompetenzen anzugeben, wurde eine Aufgabe exemplarisch mithilfe einer Bewertungsmatrix analysiert. In einem Reflexionsprozess wurden folgende sechs Anforderungen an die Entwicklung von Aufgaben zu Kompetenzmodellen in der Informatik zusammengetragen:

1. Zwischen Beispiel-, Unterrichts- und Testaufgaben unterscheiden
2. Beispiel- und Testaufgaben anhand der im Kompetenzmodell geforderten Kompetenzen konstruieren und in klare, überschaubare Teilaufgaben untergliedern
3. Vielfältige, abwechslungsreiche Unterrichtsaufgaben zusammenstellen
4. Die Unterrichtsaufgaben in einem digitalen, einfach modifizierbaren Format bereitstellen
5. In den Aufgaben nicht auf spezielle Software, Programmiersprachen oder Entwicklungsumgebungen eingehen
6. Aufgaben vor einem größeren Einsatz erproben

Die positiven Ergebnisse der Hauptuntersuchung sprechen für die dargestellten Anforderungen zur Entwicklung von Kompetenzmodellen und zugehörigen Aufgaben in der Informatik.

### Ergebnisse zur zweiten Forschungsfrage

**Zur Umsetzung kompetenzorientierten Informatikunterrichts** Um einen Unterricht zu ermöglichen, in dessen Zentrum der Aufbau von Kompetenzen steht, wurden im Rahmen dieser Arbeit verschiedene Materialien zur Verfügung gestellt. Lehrpersonen strukturieren und planen ihren Unterricht anhand eines Kompetenzmodells, d. h. sie legen vor größeren Unterrichtssequenzen bzw. einzelnen Unterrichtstunden fest, welche Kompetenzen die Schülerinnen und Schüler erwerben sollen und analysieren im nachhinein, inwieweit Kompetenzen erworben wurden. Weiterhin ist es möglich, das Kompetenzmodell im Unterricht vorzustellen, damit Schülerinnen und Schüler ihre eigenen Leistungen beurteilen, Zielvorstellungen entwickeln und Konsequenzen für ihr Handeln ableiten können. Lehrpersonen wählen Beispiel- und Unterrichtsaufgaben entsprechend der zu erwerbenden Kompetenzen bei der Planung ihres Unterrichts aus und setzen diese im Unterricht ein. Außerdem überprüfen sie anhand von Testaufgaben, inwieweit die Lernenden Kompetenzen erworben haben und schätzen anhand der Testergebnisse den durchgeführten Unterricht und die individuellen Leistungen ein. Im Allgemeinen wird ein (Software-)Werkzeug eingesetzt, welches auch Auswirkungen auf die Strukturierung, Planung, Durchführung und Auswertung des Unterrichts hat.

**Zum Einsatz von Kompetenzmodell und Aufgaben in der Voruntersuchung** In der Voruntersuchung wurden erste Versionen des entwickelten Kompetenzmodells und der zugehörigen Aufgaben von sechs Thüringer Lehrkräften erprobt. Das Kompetenzmodell wurde von drei Lehrkräften zur Planung von Unterrichtssequenzen bzw. -stunden eingesetzt. In den drei Schulen, in denen das Kompetenzmodell zur Vorbereitung von Unterrichtssequenzen und -stunden genutzt wurde, wurde häufig mit Hilfe von verschiedenen Aufgaben differenziert und die Schülerinnen und Schüler bekamen die Möglichkeit, im Kompetenztest selbstständig die zu bearbeitende Stufe auszuwählen. Die Sammlung von Unterrichtsaufgaben wurde von allen Lehrpersonen in nahezu jeder Unterrichtsstunde verwendet und anschließend nahezu vollständig positiv beurteilt. Vereinzelte negative Einschätzungen gab es lediglich zu mathematischen Aufgaben. Die Beispielaufgaben und der Kompetenztest wurden in allen sechs Klassen eingesetzt. Die Analyse der Schülerleistungen bei den bearbeiteten Testaufgaben ergab, dass die Stufen des Kompetenzmodells bereits so gewählt waren, dass mit ihnen Leistungsunterschiede zwischen Lernenden abgebildet werden konnten. Alle an der Voruntersuchung beteiligten Lehrkräfte schätzten den von außen bereitgestellten Test als hilfreich ein, um über den eigenen Unterricht zu reflektieren.
Insgesamt wurden die Materialien positiv beurteilt, konnten aber aufgrund der ge-

wonnenen Erfahrungen noch verbessert werden. Als Konsequenz auf die Ergebnisse der Voruntersuchung wurden die Unterrichtsaufgaben in die Komponenten und Stufen des Kompetenzmodells eingeordnet. Dadurch konnten Modell und zugehörige Aufgaben besser miteinander verzahnt werden. Weiterhin wurden Aufgaben zu den Komponenten A und B entwickelt, um den Lehrpersonen die Möglichkeit zu geben, die zugehörigen Kompetenzen im Unterricht besser thematisieren zu können.

**Zum Einsatz von Kompetenzmodell und Aufgaben in der Hauptuntersuchung**
In der Hauptuntersuchung wurden 40 Lehrkräfte zu ihren Erfahrungen mit dem Einsatz von Kompetenzmodell und Aufgaben befragt. Die meisten Lehrerinnen und Lehrer richteten ihren Unterricht am Kompetenzmodell „Algorithmen“ aus und nutzten die Aufgaben zur Unterrichtsvorbereitung. Das Einordnen der Unterrichtsaufgaben in die Komponenten und Stufen des Kompetenzmodells wurde größtenteils positiv bewertet. Vereinzeltes negatives Feedback zu Kompetenzmodell und Aufgaben relativierte sich meist nach genauerer Analyse. Mehr als die Hälfte der befragten Lehrpersonen setzte den Kompetenztest am Ende der Untersuchung ein, um die von den Lernenden erworbenen Kompetenzen zu überprüfen. Es zeigte sich, dass die Testaufgaben geeignet sind, um die Kompetenzen des Modells zu überprüfen. Die Testergebnisse ergaben weiterhin, dass bei längerem Einsatz der Materialien im Unterricht von den Lernenden durchschnittlich eine höhere Stufen erreicht wurde. Dies deutet darauf hin, dass auf den Stufen II und III wirklich vertiefte bzw. umfassendere Kompetenzen angegeben sind, die erst nach einer längeren Beschäftigung mit dem Thema erworben werden. Fehlende kritische Anmerkungen der Lehrpersonen zu Bearbeitungsdauer, Problemen bei der Bearbeitung und Korrektur sind Indizien dafür, dass der Kompetenztest ohne externe Unterstützung in überschaubarer Zeit bearbeitet und verlässlich korrigiert werden kann (vgl. hierzu Blum, 2006, S. 18).
Die Hauptuntersuchung zeigte außerdem, dass das in Anlehnung an die GI-Empfehlungen zu Bildungsstandards (2008) entwickelte Kompetenzmodell „Algorithmen“ an allen Schulformen eingesetzt werden kann und ermöglichte positive Rückschlüsse auf den untersuchten Inhaltsbereich der GI-Empfehlungen. Insgesamt konnte festgestellt werden, dass die Materialien von einem großen Teil der im Rahmen der Hauptuntersuchung befragten Lehrpersonen bei der Umsetzung kompetenzorientierten Informatikunterrichts genutzt wurden und dass das vorgestellte Kompetenzmodell und die zugehörigen Aufgaben dementsprechend als Grundlage für einen solchen Unterricht geeignet sind.

### Ergebnisse zur dritten Forschungsfrage

**Zum Einsatz von Puck in der Voruntersuchung** In der Voruntersuchung wurde die visuelle Programmiersprache Puck von sechs Thüringer Informatiklehrkräften im Unterricht der 8. bzw. 10. Jahrgangsstufe ein Schulhalbjahr lang eingesetzt. Die Lehrerinnen und Lehrer schätzten Puck positiv ein. Sie hoben insbesondere die verminderte Fehlerproblematik als günstig für den Anfangsunterricht hervor. Dadurch wurde es fünf der sechs Lehrpersonen ermöglicht, einzelne Schülerinnen und Schüler besser individuell zu fördern. Trotz einzelner Verbesserungsvorschläge wurden aufgrund der insgesamt positiven Ergebnisse in Vorbereitung auf die Hauptuntersuchung keine Veränderungen am System vorgenommen.

**Zum Einsatz von Puck in der Hauptuntersuchung** Im Rahmen der Hauptuntersuchung haben 40 Lehrkräfte Puck größtenteils in den Jahrgangsstufen 8, 9 bzw. 10 eingesetzt. Es wurde häufig angegeben, dass das Verringern der Syntaxfehler durch Puck die Einstiegshürden in die Programmierung verminderte. Der größte Teil der Lehrpersonen wurde durch die Arbeit mit der visuellen Programmiersprache für die Unterrichtsvorbereitung motiviert und konnte auch eine erhöhte Motivation bei den Schülerinnen und Schülern feststellen. Diese motivierenden Eigenschaften von Puck sind nicht nur für die Verbreitung des Systems wichtig, sondern können auch dem Fachkräftemangel in der Informatik entgegenwirken, indem sie das Interesse der Schülerinnen und Schüler für das Fach bereits in der Schule wecken. Trotz verschiedener Anregungen für Weiterentwicklungen waren ca. drei Viertel der Lehrkräfte der Auffassung, dass die Funktionalität von Puck für eine Einführung in die Programmierung ausreicht. Dies spricht dafür, das Grundkonzept der visuellen Programmiersprache beizubehalten und keine Erweiterungen vorzunehmen, die die Einfachheit des Systems beeinträchtigen könnten. Um Lehrenden und Lernenden eine bessere Möglichkeit zum Austausch von Programmen und Materialien zu bieten, wurde die Homepage `www.ipuck.de` eingerichtet.

## 9.2 Ausblick

**Zur Entwicklung kompetenzorientierten Informatikunterrichts** Eine Untersuchung, die die Ergebnisse kompetenzorientierten Unterrichts mit denen herkömmlichen Unterrichts vergleicht und somit überprüft, ob der Einsatz der in dieser Arbeit vorgestellten oder ähnlicher Materialien zur qualitativen Weiterentwicklung von Informatikunterricht führt, könnte ein Ansatz für weitere Forschungsarbeiten sein.

Ca. die Hälfte der Lehrpersonen stellte das Kompetenzmodell im Unterricht der Hauptuntersuchung vor. Für weitere Forschungsarbeiten wäre es sinnvoll zu untersuchen, inwieweit Schülerinnen und Schüler unterschiedlicher Jahrgangsstufen anhand von Kompetenzmodellen und Beispielaufgaben ihre eigenen Fähigkeiten in Vorbereitung auf einen Kompetenztest wirklich richtig einschätzen und Konsequenzen für ihr Handeln ableiten können.

Bei der Auswertung der Hauptuntersuchung konnte bei der Mehrzahl der Lehrkräfte ein Interesse an kompetenzorientiertem Unterricht festgestellt werden. Ein Ansatz für weitere Forschungsarbeiten wäre es, zu untersuchen, wie Lehrerinnen und Lehrer bei ihrem pädagogischem Handeln in einem solchen kompetenzorientierten Informatikunterricht weiter unterstützt werden können.

**Kompetenzmodelle im Informatikunterricht** In dieser Untersuchung wurde ein Kompetenzmodell für den Inhaltsbereich „Algorithmen“ entwickelt und erprobt. Es zeigte sich, dass anhand dieses Kompetenzmodells Informatikunterricht strukturiert, vorbereitet und ausgewertet werden kann. Kompetenzmodelle und zugehörige Tests können mit unterschiedlichen Zielstellungen eingesetzt werden. Es ist offen, inwieweit das Kompetenzmodell und die zugehörigen Aufgaben für Bildungsmonitoring und/oder Schulevaluation nutzbar sind. Eine empirische Überprüfung der Materialien, bei der die Ergebnisse einer größeren Anzahl von Lernenden in einem Kompetenztest systematisch ausgewertet werden, könnte zur Beantwortung dieser Frage beitragen. Die Erkenntnisse einer solchen Untersuchung könnten zusammen mit denen zu anderen Inhalts- und Prozessbereichen wie im Abschnitt 3.2 am Beispiel des europäischen Referenzrahmens für Sprachen beschrieben wurde, zu einem globalen Kompetenzmodell informatischer Bildung zusammengefasst werden.

Die Mehrzahl der Lehrpersonen sprach sich für Fortbildungen zum Einsatz von Kompetenzmodellen im Informatikunterricht und für den Einsatz von Kompetenzmodellen in anderen Bereichen des Informatikunterrichts aus. Hier wird Fortbildungs- und Forschungsbedarf in der Fachdidaktik Informatik deutlich.

**Kompetenzorientierte Aufgaben im Informatikunterricht** Ein Ansatz für weitere Forschungsarbeiten wäre es, weitere Beispiel-, Unterrichts- und Testaufgaben zusammenzutragen, die die Kompetenzen des entwickelten Modells auf andere Weise konkretisieren, also z. B. andere Datentypen, Anweisungen und Darstellungsformen nutzen, um so die Vielfalt im Informatikunterricht zu unterstützen und evtl. Vergleiche mit den hier vorgestellten Konkretisierungen zu ermöglichen.

Die Befragung der Lehrkräfte ergab, dass es aus organisatorischen Gründen sinnvoll wäre, einen Kompetenztest mit zwei Teilen gleicher Bearbeitungsdauer zu entwickeln, von denen ein Teil mit und ein anderer ohne Computer gelöst werden kann.

Weiterhin könnten weitere kompetenzorientierte Aufgaben entwickelte werden, bei denen die Kompetenzen verschiedener Inhalts- und Prozessbereiche noch stärker miteinander verzahnt sind, um so auch die Aufgabenkultur in der Informatik weiterzuentwickeln. Die im Abschnitt 6.4 eingesetzte Bewertungsmatrix kann hierbei hilfreich sein. Wenn Schülerinnen und Schüler bei der Bearbeitung von Aufgaben aufgefordert werden, „laut zu denken“, oder wenn sie nach der Aufgabenbearbeitung zur Lösung interviewt werden, so sind genauere Aussagen zu Lösungsansätzen und jeweils benötigten Kompetenzen möglich.

Einige Lehrpersonen gaben ihren Schülerinnen und Schülern beim bereitgestellten Kompetenztest die Möglichkeit, die zu bearbeitende Stufe selbstständig zu wählen. Warum in der Hauptuntersuchung jedoch größtenteils einheitlich von den Lehrpersonen festgelegt wurde, welche Stufe bearbeitet werden sollte, blieb offen. Ziel weiterer Arbeiten könnte es sein, weitere Methoden zu erforschen, mit denen die Kompetenzen unterschiedlicher Stufen in einem einheitlichen Test erfasst werden können. Die von Schlüter und Brinda (2007, S. 291 f.) vorgestellte Möglichkeit, unterschiedliche Hilfestellungen zur Verfügung zu stellen, ist ein erster Ansatz in dieser Richtung.

Matthias Kohl (2008b, S. 5) stellt am Beispiel der IT-Ausbildung und des IT-Weiterbildungssystems dar, dass kompetenzorientierte Prüfungen, die unter anderem die Durchführung, Dokumentation und Präsentation einer betrieblichen Projektarbeit beinhalten, Aussagen über die Fachkompetenz, aber auch über prozessbezogene Fähigkeiten oder soziale Kompetenzen erlauben. Nach Auffassung des Autors sollte auch in der Schulinformatik geprüft werden, inwieweit Projektarbeiten und andere für Testsituationen untypische Aufgabenformate zum Erwerben, Verdeutlichen und Überprüfen von Kompetenzen genutzt werden können.

**Die visuelle Programmiersprache Puck** Im Rahmen dieser Arbeit wurde gezeigt, dass die von Arnold und Hartmann (2007b) gegebenen Empfehlungen zur Entwicklung von interaktiven Lernumgebungen auch im Entwicklungsprozess von Puck nachvollzogen werden konnten. Die Ergebnisse der Erprobung von Puck waren weitgehend positiv. Dies wird als Beleg für die Sinnhaftigkeit der Empfehlungen interpretiert, der dazu führen sollte, dass diese auch bei der Entwicklung weiterer interaktiver Lernumgebung beachtet werden. Anschließende Reflexionen können zur Weiterentwicklung und Fundierung der Empfehlungen führen.

In der Hauptuntersuchung konnte sich ca. ein Drittel der befragten Lehrpersonen vorstellen, Puck auch bei jüngeren Lernenden, also vor der 8. Jahrgangsstufe einzusetzen. Ob das wirklich möglich ist, welche Aufgaben dafür geeignet sind und welche Kompetenzen dabei erworben werden sollen, könnten Fragestellungen für weitere Forschungsarbeiten sein. Weiterhin ist noch nicht belegt, dass die beim Einsatz von Puck erworbenen Kompetenzen direkt auf die Arbeit mit textuellen Programmiersprachen übertragbar sind.

Die visuelle Programmiersprache Puck hat sich im Unterricht der Vor- und der Hauptuntersuchung bewährt. Konzeptionelle Veränderungen, die die Einfachheit des Systems beeinträchtigen könnten, erscheinen deshalb nicht angebracht. Zur besseren Verbreitung des Systems wäre es aber sinnvoll, eine plattformunabhängige Puck-Version zu entwickeln, die auch in verschiedenen Sprachen verfügbar sein sollte. Weitere im Rahmen dieser Arbeit zusammengetragene Vorschläge sollten bei der Weiterentwicklung des Systems diskutiert werden. Zwei Fünftel der in der Hauptuntersuchung befragten Lehrkräfte waren der Ansicht, dass nicht genügend Unterrichtsmaterialien zu Puck zur Verfügung standen. Die Verbreitung der vorhandenen sowie die Entwicklung weiterer Materialien wären dementsprechend sinnvoll.

Ob das Konzept visueller Programmierung auch auf andere Programmierparadigmen übertragen werden kann und ob es möglich ist, ein visuelles System zu entwickeln, mit dem weiterführende Kompetenzen zur Programmierung erworben werden können, sind weitere spannende Forschungsfragen.

## Fazit

Die vorgestellten Entwicklungen verschiedener visueller Werkzeuge für den Informatikunterricht bergen ein großes Potenzial zur Fortentwicklung desselben. Die Entwicklung allein reicht aber nicht aus, es müssen auch Aufgaben, Fortbildungs- und Unterrichtsmaterialien bereitgestellt werden, die Werkzeuge müssen einem breiten Publikum zugänglich gemacht werden und schließlich muss der Einsatz der Werkzeuge im Unterricht wissenschaftlich untersucht werden, um so positive wie negative Effekte darstellen zu können. In dieser Arbeit wurde gezeigt, dass mit der visuellen Programmiersprache Puck eine Einführung in die Programmierung in den Jahrgangsstufen 8 bis 10 möglich ist, dass sich das Vermindern der Syntaxfehler positiv auf den Unterricht auswirkt und dass das System zur Motivation von Lehrenden und Lernenden beiträgt.
Weiterhin wurden in dieser Arbeit erste Erkenntnisse zur Kompetenzorientierung im Informatikunterricht zusammengetragen, ein Kompetenzmodell mit zugehörigen Aufgaben wurde entwickelt und im Unterricht erprobt. Insgesamt stehen die Entwicklungen dieser Richtung in der Fachdidaktik Informatik aber noch am Anfang und viele Fragen insbesondere zur Gestaltung kompetenzorientierten Unterrichts sind noch offen.

# Literaturverzeichnis


[ACM 2003] ACM: *A Model Curriculum for K-12 Computer Science: Final Report of the ACM K-12 Task Force Curriculum Committee*. Internetdokument. 2003. – URL `http://www.acm.org/education/curric_vols/k12final1022.pdf`. – Zugriffsdatum: 01.10.2008

[Aebli 1994] Aebli, Hans: *Denkprozesse*. 2. Auflage. Klett-Cotta, 1994

[Ala-Mutka 2003] Ala-Mutka, Kirsti: *Problems in Learning and Teaching Programming*. Internetdokument. 2003. – URL `http://www.cs.tut.fi/~codewitz/literature_study.pdf`. – Zugriffsdatum: 01.10.2008

[Arnold 2007] Arnold, Ruedi: *Interactive Learning Environments for Mathematical Topics*, ETH Zürich, Dissertation, 2007. – URL `http://e-collection.ethbib.ethz.ch/eserv/eth:30045/eth-30045-01.pdf`. – Zugriffsdatum: 01.10.2008

[Arnold und Hartmann 2007a] Arnold, Ruedi ; Hartmann, Werner: LogicTraffic – Logik in der Allgemeinbildung. In: *Informatik-Spektrum* 30 (2007), Nr. 1, S. 19–26

[Arnold und Hartmann 2007b] Arnold, Ruedi ; Hartmann, Werner: Pragmatische Empfehlungen zur Entwicklung von interaktiven Lernumgebungen. In: (Schubert, 2007), S. S. 171–182

[Artelt u. a. 2007] Artelt, Cordula ; McElvany, Nele ; Christmann, Ursula ; Richter, Tobias ; Groeben, Norbert ; Köster, Juliane ; Schneider, Wolfgang ; Stanat, Petra ; Ostermeier, Christian ; Schiefele, Ulrich ; Valtin, Renate ; Ring, Klaus ; Saalbach, Henrik: *Bildungsforschung*. Bd. 17: *Förderung von Lesekompetenz – Expertise*. Bonn : Bundesministerium für Bildung und Forschung (BMBF), 2007. – URL `http://www.bmbf.de/pub/bildungsreform_band_siebzehn.pdf`. – Zugriffsdatum: 02.10.2008

[Baldwin und Kujis 2004] Baldwin, Lynne P. ; Kujis, Jasna: Visualisation Techniques for Learning and Teaching Programming. In: *Journal of Computing and Information Technology* (2004), Nr. 4, S. 285–291. – URL `http://cit.zesoi.fer.hr/downloadPaper.php?paper=305`. – Zugriffsdatum: 01.10.2008

[Barnes und Kölling 2000] Barnes, David J. ; Kölling, Michael: *Objektorientierte Programmierung mit Java – Eine praxisnahe Einführung mit BlueJ*. Pearson Studium, 2000

[Baumann 1996] Baumann, Rüdeger: *Didaktik der Informatik*. Klett, 1996

[Bayrhuber 2007] Bayrhuber, Horst: Beitrag der Fachdidaktik zur Qualitätsverbesserung im Bildungssystem. In: Bayrhuber, Horst (Hrsg.) ; Elster, Doris (Hrsg.) ; Krüger, Dirk (Hrsg.) ; Vollmer, Helmut J. (Hrsg.): *Kompetenzentwicklung und Assessment*. Innsbruck : Studien Verlag, 2007, S. 9–15

[Bell u. a. 2006] Bell, Tim ; Witten, Ian H. ; Fellows, Mike: *Computer Science Unplugged*. Internetdokument. 2006. – URL `www.csunplugged.org`. – Zugriffsdatum: 01.10.2008

[Ben-Ari u. a. 2002] Ben-Ari, Mordechai ; Myller, Niko ; Sutinen, Erkki ; Tarhio, Jorma: Perspectives on Program Animation with Jeliot. In: Diehl, Stephan (Hrsg.): *Software Visualization - International Seminar Dagstuhl Castle, Deutschland, Mai 2001*. Springer Verlag, 2002, S. 31–45

[Bergin und Reilly 2005] Bergin, S. ; Reilly, R.: The Influence of motivation and comfort-level on learning to program. In: *Proceedings of the 17th Annual Workshop of the Psychology of Programming Interest Group*, URL `http://www.ppig.org/papers/17th-bergin.pdf`. – Zugriffsdatum: 01.10.2008, 2005, S. 293–304

[Bühl und Zöfel 2000] Bühl, Achim ; Zöfel, Peter: *SPSS Version 10 – Einführung in die moderne Datenanalyse unter Windows*. 7. Auflage. Addison Wesley Verlag, 2000

[Biggs und Collins 1982] Biggs, John B. ; Collins, Kevin F.: *Evaluating the Quality of Learning*. Academic Press Inc., 1982

[Bloom 1972] Bloom, Benjamin S.: *Taxonomie von Lernzielen im kognitiven Bereich*. Beltz, 1972

[Blum 2006] Blum, Werner: Die Bildungsstandards Mathematik – Einführung. In: Blum, Werner (Hrsg.) ; Drüke-Noe, Christina (Hrsg.) ; Hartung, Ralph (Hrsg.) ; Köller, Olaf (Hrsg.): *Bildungsstandards Mathematik: konkret*. Berlin : Cornelsen-Scriptor-Verlag, 2006, S. 14–29

[Boles 2002] Boles, Dietrich: *Programmieren spielend gelernt mit dem Java-Hamster-Modell*. Teubner, 2002

[Boles und Boles 2004] Boles, Dietrich ; Boles, Cornelia: *Objektorientierte Programmierung spielend gelernt mit dem Java-Hamster-Modell*. Teubner, 2004

[Bortz und Döring 2006] Bortz, Jürgen ; Döring, Nicola: *Forschungsmethoden und Evaluation*. 4. Auflage. Springer Medizin Verlag, 2006

[du Bouley 1989] Bouley, B. du: Some difficulties of learning to program. In: Soloway, E. (Hrsg.) ; Sphorer, J. C. (Hrsg.): *Studying the novice programmer*. Lawrence Erlbaum, 1989, S. 283–299

[Braband 2007] Braband, Claus: Constructive Alignment for Teaching Model-Based Design for Concurrency. In: Gomes, Luis (Hrsg.) ; Christensen, Soren (Hrsg.): *Proceedings of TeaConc 2007 - The 2nd Workshop on Teaching Concurrency*. Siedlce, Polen : Publishing House of University of Podlasie, 2007. – URL `http://www.brics.dk/~brabrand/teaconc.pdf`. – Zugriffsdatum: 01.10.2008

[Braband und Dahl 2007] Braband, Claus ; Dahl, Bettina: Constructive Alignment and the SOLO Taxonomy: A Comparative Study of University Competences in Computer Science vs. Mathematics. In: Lister, Raymond (Hrsg.) ; Simon (Hrsg.): *Koli Calling 2007, Proceedings of the Seventh Baltic Sea Conference on Computing Education Research* Bd. 88, Australian Computer Society, 2007

[Brabrand und Andersen 2006] Brabrand, Claus ; Andersen, Jacob: *Teaching Teaching & Understanding Understanding*. 19 minute award-winning short-film (DVD) about Constructive Alignment, Aarhus University Press, University of Aarhus, Denmark. 2006. – URL `http://www.daimi.au.dk/~brabrand/short-film/`. – Zugriffsdatum: 15.10.2008

[Breier und Hilger 2008] Breier, Norbert ; Hilger, Sabrina: Mein Computer spricht mit mir! – Sprachdialogsysteme in einem kontextbezogenen Informatikunterricht. In: (Brinda u. a., 2008), S. 119–128

[Bresciani und Eppler 2008] Bresciani, Sabrina ; Eppler, Martin J.: *The Perils of Visualization A Classification of Risks associated with Graphic Representations of Information*. Internetdokument. 2008. – URL `http://www.knowledge-communication.org/Bresciani_Eppler_Risks_of_Visualization_working%20paper_2008.pdf`. – Zugriffsdatum: 01.10.2008

[Brinda 2004] Brinda, Torsten: *Didaktisches System für objektorientiertes Modellieren im Informatikunterricht der Sekundarstufe II*. Siegen, Dissertation, 2004. – URL `http://deposit.ddb.de/cgi-bin/dokserv?idn=971190747`. – Zugriffsdatum: 01.10.2008

[Brinda u. a. 2008] Brinda, Torsten (Hrsg.) ; Fothe, Michael (Hrsg.) ; Hubwieser, Peter (Hrsg.) ; Schlüter, Kirsten (Hrsg.): *Didaktik der Informatik – Aktuelle Forschungsergebnisse 5. Workshop der GI-Fachgruppe "Didaktik der*

*Informatik“ 24.-25. September 2008 in Erlangen.* Bonn : Gesellschaft für Informatik e.V., 2008

[Brinda und Mägdefrau 2008] Brinda, Torsten ; Mägdefrau, Jutta: Zur Gestaltung der informatischen Bildung an Realschulen – Eine kriterienorientierte Bertrachtung. In: *LOG IN* (2008), Nr. 150/151, S. 12–18

[Brinda und Schulte 2005] Brinda, Torsten ; Schulte, Carsten: Beiträge der Objektorientierung zu einem Kompetenzmodell des informatischen Modellierens. In: (Friedrich, 2005), S. 137–148

[Börstler 2007] Börstler, Jürgen: Objektorientiertes Programmieren – Machen wir irgendwas falsch? In: (Schubert, 2007), S. 9–20

[Bruner 1988] Bruner, Jerome: Über kognitive Entwicklung. In: Bruner, Jerome (Hrsg.) ; Olver, Rose R. (Hrsg.) ; Greenfield, Patricia M. (Hrsg.): *Studien zur kognitiven Entwicklung.* 2. Auflage. Klett-Cotta, 1988, S. 21–54

[Bruner 1974] Bruner, Jerome S.: *Entwurf einer Unterrichtstheorie.* Düsseldorf : Pädagogischer Verlag Schwann, 1974

[Brusilovsky u. a. 1997] Brusilovsky, Peter ; Calabrese, Eduardo ; Hvorecky, Jozef ; Kouchnirenko, Anatoly ; Miller, Philip: Mini-languages: A Way to Learn Programming Principles. In: *Education and Information Technologies* 2 (1997), Nr. 1, S. 65–83. – URL `http://www.schule.bayern.de/karol/minilang/minilang.htm`. – Zugriffsdatum: 01.10.2008

[Calloni u. a. 1997] Calloni, Ben A. ; Bagert, Donald J. ; Haiduk, H. P.: Iconic programming proves effective for teaching the first year programming sequence. In: *SIGCSE Bulletin* 29 (1997), Nr. 1, S. 262–266. – URL `http://doi.acm.org/10.1145/268085.268189`

[Cilliers u. a. 2005] Cilliers, Charmain ; Calitz, André ; Greyling, Jéan: The effect of integrating an Iconic programming notation into CS1. In: *ITiCSE ’05: Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education.* New York, USA : ACM, 2005, S. 108–112. – URL `http://doi.acm.org/10.1145/1067445.1067478`

[Cunniff u. a. 1986] Cunniff, N. ; Taylor, R. P. ; Black, J. B.: Does programming language affect the type of conceptual bugs in beginners’ programs? A comparison of FPL and Pascal. In: *SIGCHI Bulletin* 17 (1986), Nr. 4, S. 175–182. – URL `http://doi.acm.org/10.1145/22339.22368`

[Demmer 2003] Demmer, Marianne: Bildungsstandards: Selektion perfektionieren oder überwinden? In: (GEW, 2003). – URL `http://www.gew.de/Binaries/Binary35637/bildungsstandards.pdf`. – Zugriffsdatum: 01.10.2008

[Dorninger 2007] Dorninger, Christian: IT-Bildungsstandards – Das Kompetenzmodell angewandte Informatik für die Sekundarstufe II. In: *PCNews* (2007), Nr. 106, S. 8–11. – URL `http://pcnews.at/_pdf/n1060008.pdf`. – Zugriffsdatum: 01.10.2008

[Durda 2007] Durda, Tommy: Evaluierungsmöglichkeiten von Bildungsstandards. In: *LOG IN* (2007), Nr. 146/147, S. 23–26

[ECDL Switzerland 2008] ECDL Switzerland: *FAQs zum ECDL Advanced*. Internetdokument. 2008. – URL `http://www.ecdl.ch/index.php?id=619`. – Zugriffsdatum: 02.10.2008

[Feldgen und Clua 2003] Feldgen, M. ; Clua, O.: New motivations are required for freshman introductory programming. In: *Proceedings of the 33rd ASSE/IEEE Frontiers in Education Conference*. Boulder, USA, 2003, S. T3C – T24. – URL `http://fie.engrng.pitt.edu/fie2003/papers/1248.pdf`. – Zugriffsdatum: 01.10.2008

[Feldgen und Clua 2004] Feldgen, M. ; Clua, O.: Games As A Motivation For Freshman Students To Learn Programming. In: *Proceedings of the 34rd ASSE/IEEE Frontiers in Education Conference*. Savannah, USA, 2004, S. SH1–11 – SH1–16. – URL `http://fie.engrng.pitt.edu/fie2004/papers/1222.pdf`

[Fothe 1989] Fothe, Michael: *80 Programme in Turbo-Pascal*. 2. Auflage. Akademie der Wissenschaften der DDR Berlin, 1989

[Fothe 2002] Fothe, Michael: *Problemlösen mit Python*. Thüringer Institut für Lehrerfortbildung, Lehrplanentwicklung und Medien, 2002. – URL `http://www.uni-jena.de/data/unijena_/faculties/minet/casio/Publikationen/python.pdf`. – Zugriffsdatum: 01.10.2008

[Fothe 2004] Fothe, Michael: Neue Einheitliche Prüfungsanforderungen in der Abiturprüfung Informatik - ein Schritt von 1989 nach 2004. In: *CD Austria* 5 (2004), S. 18–20

[Fothe 2005] Fothe, Michael: EPA Informatik. In: *LOG IN* (2005), Nr. 135, S. 46–49

[Fothe 2007] Fothe, Michael: Algorithmen in spielerischer Form. In: (Schubert, 2007), S. 31–43

[Fothe 2008] Fothe, Michael: Bildungsstandards Informatik für die Sekundarstufe II – Vorüberlegungen zur Entwicklung. In: (Brinda u. a., 2008), S. 107–116

[Fothe und Kerner 1988] Fothe, Michael ; Kerner, Immo O.: Problem des

Lucas. In: *Wurzel – Zeitschrift für Mathematik an Ober- und Spezialschulen* (1988), Nr. 2, S. 18–23

[Fothe und Ludwig 2008] Fothe, Michael ; Ludwig, Heidrun: Zu Möglichkeiten der externen Unterstützung von Informatiklehrerinnen und -lehrern in der gymnasialen Oberstufe. In: *informatica didactica* (2008), Nr. 8. – URL `http://www.informatica-didactica.de/InformaticaDidactica/FotheLudwig2008.pdf`. – Zugriffsdatum: 01.10.2008

[Fothe u. a. 2006] Fothe, Michael ; Ludwig, Heidrun ; Küspert, Klaus ; Wenzel, Marcus: Unterrichtsreflexion mit ungewöhnlichen Mitteln. In: *LOG IN* (2006), Nr. 141/142, S. 52–63

[Free Software Foundation 2007] Free Software Foundation: *GNU GENERAL PUBLIC LICENSE*. Internetdokument. 2007. – URL `http://www.gnu.org/licenses/old-licenses/gpl-2.0.txt`. – Zugriffsdatum: 01.10.2008

[Freiberger 2002] Freiberger, Ulli: *Karel – Eine Übersicht über verschiedene Entwicklungen, die auf der Idee von "Karel, the Robot" basieren.* Internetdokument. 2002. – URL `http://www.schule.bayern.de/karol/data/uebersicht.pdf`. – Zugriffsdatum: 01.10.2008

[Freiberger und Krsko 2003] Freiberger, Ulli ; Krsko, Ondrej: *Robot Karol – Eine Programmiersprache für Schülerinnen und Schüler Version 2.2.* Internetdokument. 2003. – URL `http://www.schule.bayern.de/karol/data/handbuch.pdf`. – Zugriffsdatum: 01.10.2008

[Friedrich 2003] Friedrich, Steffen: Informatik und PISA – vom Wehe zum Wohl der Schulinformatik. In: (Hubwieser, 2003), S. 133–144

[Friedrich 2005] Friedrich, Steffen (Hrsg.): *INFOS 2005 Unterrichtskonzepte für informatische Bildung 11. GI-Fachtagung Informatik und Schule 28.-30. September 2005 in Dresden.* Gesellschaft für Informatik e.V., 2005

[Friedrich und Puhlmann 2007] Friedrich, Steffen ; Puhlmann, Hermann: Bildungsstandards Informatik – von Wünschen zu Maßstäben für eine informatische Bildung. In: (Schubert, 2007), S. 21–32

[Garner u. a. 2005] Garner, Sandy ; Haden, Patricia ; Robins, Anthony: My program is correct but it doesn't run: a preliminary investigation of novice programmers' problems. In: *ACE '05: Proceedings of the 7th Australasian conference on Computing education.* Darlinghurst, Australia : Australian Computer Society, 2005, S. 173–180. – URL `http://www.cs.otago.ac.nz/staffpriv/anthony/publications/pdfs/GarnerHadenRobins.pdf`. – Zugriffsdatum: 01.10.2008

[GEW 2003] GEW (Hrsg.): *Nationale Bildungsstandards – Wundermittel oder Teufelszeug? Funktionen, Hintergründe und Positionen der GEW.* URL

http://www.gew.de/Binaries/Binary35637/bildungsstandards.pdf. – Zugriffsdatum: 01.10.2008, 2003

[GI 2000] GI: *Gesamtkonzept zur informatischen Bildung an allgemein bildenden Schulen.* Internetdokument. 2000. – URL http://www.gi-ev.de/fileadmin/redaktion/empfehlungen/gesamtkonzept_26_9_2000.pdf. – Zugriffsdatum: 01.10.2008

[GI 2004] GI: *Digitale Spaltung verhindern – Schulinformatik stärken!* Memorandum der GI. 2004. – URL http://www.nw.schule.de/gi/material/memorandum_schulinformatik.pdf. – Zugriffsdatum: 01.10.2008

[GI 2008] GI: Grundsätze und Standards für die Informatik in der Schule – Bildungsstandards Informatik für die Sekundarstufe I. In: *LOG IN* (2008), Nr. 150/151, S. Beilage zur Zeitschrift. – URL http://www.informatikstandards.de/. – Zugriffsdatum: 01.10.2008

[Gieseke 2007] Gieseke, Werner: Das Zentralabitur Informatik in Niedersachsen. In: *LOG IN* (2007), Nr. 148/149, S. 32–35

[Girmes 2003] Girmes, Renate: Die Welt als Aufgabe?! In: Ball, Helga (Hrsg.) ; Becker, Gerold (Hrsg.) ; Bruder, Regina (Hrsg.) ; Girmes, Renate (Hrsg.) ; Stäudel, Lutz (Hrsg.) ; Winter, Felix (Hrsg.): *Aufgaben.* Seelze : Friedrich Verlag, 2003, S. 6–11

[Goethe 1947] Goethe, Johann W.: *Chinesisch-deutsche Jahres- und Tageszeiten.* Krefeld : Scherpe-Verlag, 1947

[Greyling und Brown 2002] Greyling, J.H. ; Brown, D.: Issues highlighted by the implementation of a visual programming tool for an introductory programming course. In: Wells, George (Hrsg.) ; McNeill, John (Hrsg.): *Proceedings of the 32nd Annual SACLA Conference*, URL http://www.sacla.org.za/SACLA2002/Proceedings/Papers/Greyling_Brown.pdf. – Zugriffsdatum: 01.10.2008, 2002

[Gutknecht und Hromkovic 2008] Gutknecht, Jürg ; Hromkovic, Juraj: *Plädoyer für den Programmierunterricht.* Internetdokument. 2008. – URL http://www.abz.inf.ethz.ch/media/archive1/aktuelles/Plaedoyer_fuer_den_Programmierunterricht.pdf. – Zugriffsdatum: 01.10.2008

[Hammann 2004] Hammann, Marcus: Kompetenzentwicklungsmodelle: Merkmale und ihre Bedeutung. In: *MNU* 4 (2004), S. 196–203

[Hartmann 2005] Hartmann, Werner: Informatik - EIN/AUS - Bildung. In: (Friedrich, 2005)

[Hartmann u. a. 2006] Hartmann, Werner ; Näf, Michael ; Reichert, Raimond: *Informatikunterricht planen und durchführen*. Berlin : Springer Verlag, 2006

[Hartmann und Nievergelt 2002] Hartmann, Werner ; Nievergelt, Jürg: Informatik zwischen Wandel und Beständigkeit. In: *Informatik Spektrum* (2002), Nr. 6, S. 465–476

[Hartmann u. a. 2004] Hartmann, Werner ; Nievergelt, Jürg ; Reichert, Raimond: *Programmieren mit Kara*. Berlin : Springer Verlag, 2004

[Hartwig und Klieme 2006] Hartwig, Johannes ; Klieme, Eckard: Kompetenz und Kompetenzdiagnostik. In: Schweizer, Karl (Hrsg.): *Leistung und Leistungsdiagnostik*. Berlin : Springer Verlag, 2006, S. 127–143

[Heine 1978] Heine, Heinrich ; Matt, Peter von (Hrsg.): *Die Bäder von Lucca; Die Stadt Lucca*. Reclam, 1978

[Helmke und Hosenfeld 2004] Helmke, Andreas ; Hosenfeld, Ingmar: Vergleichsarbeiten – Standards – Kompetenzstufen: Begriffliche Klärung und Perspektiven. In: Wosnitza, M. (Hrsg.) ; Frey, A. (Hrsg.) ; Jäger, R. S. (Hrsg.): *Lernprozess, Lernumgebung und Lerndiagnostik: wissenschafliche Beiträge zum Lernen im 21. Jahrhundert*. Verlag Empirische Pädagogik, 2004, S. 56–75

[Henriksen und Kölling 2004] Henriksen, Poul ; Kölling, Michael: greenfoot: Combining Object Visualisation with Interaction. In: *Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications (OOPSLA)*, URL `http://www.cs.kent.ac.uk/pubs/2004/2192/content.pdf`. – Zugriffsdatum: 01.10.2008, 2004, S. 73–82

[Herper 2005] Herper, Henry: Algorithmen – Überlegungen zur Konstruktion von Aufgaben für den Informatikunterricht der Sekundarstufe I. In: *LOG IN* (2005), Nr. 135, S. 74–76

[Herzog 2006] Herzog, Walter: *Bildungsstandards – Aussichten einer Selbstverständlichkeit*. Internetdokument. 2006. – URL `https://www.phbern.ch/fileadmin/Bilder_und_Dokumente/01_PHBern/01_Studientage/2006/Studientage2006_Hauptreferat_Herzog.pdf`. – Zugriffsdatum: 01.10.2008

[Hubwieser 2003] Hubwieser, Peter (Hrsg.): *INFOS 2003 Informatische Fachkonzepte im Unterricht 10. GI-Fachtagung Informatik und Schule 17.-19. September in Garching bei München*. Gesellschaft für Informatik e.V., 2003

[Hubwieser 2004] Hubwieser, Peter: *Didaktik der Informatik Grundlagen, Konzepte, Beispiele*. 2. Auflage. Springer Verlag, 2004

[Humbert 2005] Humbert, Ludger: *Didaktik der Informatik*. 1. Auflage. Wiesbaden : Teubner, 2005

[Humbert und Pasternak 2005] Humbert, Ludger ; Pasternak, Arno: Informatikkompetenzen – Zur Entwicklung von Standards für die allgemeine Bildung im Schulfach Informatik. In: *LOG IN* (2005), Nr. 135, S. 24–28

[IQB 2007] IQB: *Institut zur Qualitätsentwicklung im Bildungswesen – Flyer*. Internetdokument. 2007. – URL `http://www.iqb.hu-berlin.de/institut/dateien/flyer_iqb_dt.pdf`. – Zugriffsdatum: 01.10.2008

[ISTE 1998] ISTE: *National Educational Technology Standards for Teachers*. Internetdokument. 1998. – URL `http://www.iste.org/Content/NavigationMenu/NETS/ForStudents/1998Standards/NETS_for_Students_1998_Profiles.pdf`. – Zugriffsdatum: 06.10.2008

[Jank und Meyer 1994] Jank, Werner ; Meyer, Hilbert: *Didaktische Modelle*. 3. Auflage. Frankfurt am Main : Cornelsen Scriptor, 1994

[Kelleher 2006] Kelleher, Caitlin: *Motivating Programming: using storytelling to make computer programming attractive to middle school girls*. Pittsburgh, Carnegie Mellon University, Dissertation, 2006. – URL `http://www.cse.wustl.edu/~ckelleher/kelleherThesis_CSD.pdf`. – Zugriffsdatum: 01.10.2008

[Kelleher und Pausch 2005] Kelleher, Caitlin ; Pausch, Randy: Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. In: *ACM Computing Surveys* 37 (2005), Nr. 2, S. 83–137. – URL `http://www.cs.cmu.edu/~caitlin/papers/NoviceProgSurvey.pdf`. – Zugriffsdatum: 01.10.2008

[Keller und Moser 2005] Keller, Florian ; Moser, Urs: Test your ICT-Knowledge. In: *CD Austria* 12 (2005), S. 11–13

[Khalil 2006] Khalil, Fida: *Gestaltung von PISA-Items für die Informatik zum Thema Informationssicherheit im Internet*, Universität Siegen, Diplomarbeit, 2006. – URL `http://www.die.informatik.uni-siegen.de/DIE_BIB/Arbeiten/Diplom/Diplomarbeit_Khalil.pdf`. – Zugriffsdatum: 01.10.2008

[Klieme 2003] Klieme, Eckhard: Bildungsgerechtigkeit zu fördern, ist ein wesentliches Ziel. In: (GEW, 2003), S. 14–19. – URL `http://www.gew.de/Binaries/Binary35637/bildungsstandards.pdf`. – Zugriffsdatum: 01.10.2008

[Klieme 2004a] Klieme, Eckhard: Begründung, Implementation und Wirkung von Bildungsstandards: Aktuelle Diskussionslinien und Befunde. In: *Zeitschrift für Pädagogik* 5 (2004), Nr. 50, S. 625–634

[Klieme 2004b] Klieme, Eckhard: Was sind Kompetenzen und wie lassen sie sich messen? In: *Pädagogik* (2004), Nr. 6, S. 10–13

[Klieme 2007] Klieme, Eckhard: Bildungsstandards, Leistungsmessung und Unterrichtsqualität. In: Labudde, Peter (Hrsg.): *Bildungsstandards am Gymnasium*. 1. Auflage. Bern : h.e.p.-Vertrag, 2007

[Klieme u. a. 2003] Klieme, Eckhard ; Avenarius, Hermann ; Blum, Werner ; Döbrich, Peter ; Gruber, Hans ; Prenzel, Manfred ; Reiss, Kristina ; Riquarts, Kurt ; Rost, Jürgen ; Tenorth, Heinz-Elmar ; Vollmer, Helmut J. ; BMBF (Hrsg.): *Bildungsreform*. Bd. 1: *Zur Entwicklung nationaler Bildungsstandards – Eine Expertise*. 4. unveränderte Auflage. Bonn, 2003. – URL `http://www.bmbf.de/pub/zur_entwicklung_nationaler_bildungsstandards.pdf`. – Zugriffsdatum: 01.10.2008

[Kölling u. a. 2003] Kölling, M. ; Quig, B. ; Patterson, A. ; Rosenberg, J.: The BlueJ system and its pedagogy. In: *Journal of Computer Science Education* 13 (2003), Dezember, Nr. 4, S. 249–268. – URL `http://www.bluej.org/papers/2003-12-CSEd-bluej.pdf`. – Zugriffsdatum: 01.10.2008

[KMK 2002] KMK: *Vereinbarung über Einheitliche Prüfungsanforderungen in der Abiturprüfung*. Internetdokument. 2002. – URL `http://db2.nibis.de/1db/cuvo/datei/eparahme.pdf`. – Zugriffsdatum: 01.10.2008

[KMK 2003a] KMK: *Bildungsstandards für die erste Fremdsprache (Englisch/Französisch) für den Mittleren Schulabschluss Beschluss vom 4.12.2003*. Internetdokument. 2003. – URL `http://www.kmk.org/schul/Bildungsstandards/1.Fremdsprache_MSA_BS_04-12-2003.pdf`. – Zugriffsdatum: 01.10.2008

[KMK 2003b] KMK: *Bildungsstandards im Fach Deutsch für den Mittleren Schulabschluss Beschluss vom 4.12.2003*. Internetdokument. 2003. – URL `http://www.kmk.org/schul/Bildungsstandards/Deutsch_MSA_BS_04-12-03.pdf`. – Zugriffsdatum: 01.10.2008

[KMK 2003c] KMK: *Bildungsstandards im Fach Mathematik für den Mittleren Schulabschluss Beschluss vom 4.12.2003*. Internetdokument. 2003. – URL `http://www.kmk.org/schul/Bildungsstandards/Mathematik_MSA_BS_04-12-2003.pdf`. – Zugriffsdatum: 01.10.2008

[KMK 2004a] KMK: *Bildungsstandards der Kultusministerkonferenz – Erläuterungen zur Konzeption und Entwicklung*. Veröffentlichung der Kultusministerkonferenz vom 16.12.2004, Internetdokument. 2004. – URL `http://www.kmk.org/schul/Bildungsstandards/Argumentationspapier308KMK.pdf`. – Zugriffsdatum: 01.10.2008

[KMK 2004b] KMK: *Einheitliche Prüfungsanforderungen in der Abiturprüfung Informatik*. Beschluss der Kultusministerkonferenz vom 01.12.1989 i. d. F. vom 05.02.2004, Internetdokument. 2004. – URL `http://www.kmk.org/doc/beschl/EPA-Informatik.pdf`. – Zugriffsdatum: 01.10.2008

[KMK 2007] KMK: *Ergebnisse der 319. Plenarsitzung der Kultusministerkonferenz*. Pressemitteilung, Internetdokument. 2007. – URL `http://www.kmk.org/aktuell/pm071018.htm`. – Zugriffsdatum: 01.10.2008

[Koerber und Peters 1998] Koerber, Bernhard ; Peters, Ingo-Rüdiger: *Informatische Bildung in Deutschland – Die Wurzeln der Zukunft*. S. 19–36. In: Koerber, Bernhard (Hrsg.) ; Peters, Ingo-Rüdiger (Hrsg.): *Informatische Bildung in Deutschland*, LOG IN Verlag, 1998

[Kohl 2004a] Kohl, Lutz: *Entwurf und Implementierung einer visuellen Programmiersprache für den Einsatz in Schulen*, Friedrich-Schiller-Universität Jena, Diplomarbeit, 2004. – URL `http://www.uni-jena.de/data/unijena_/faculties/minet/casio/LutzKohl/DiplomarbeitLutzKohl.pdf`. – Zugriffsdatum: 01.10.2008

[Kohl 2004b] Kohl, Lutz: *Konzepte der visuellen Programmierung und ihrer Einsatzmöglichkeiten an Schulen*. Internetdokument. 2004. – URL `http://www.uni-jena.de/data/unijena_/faculties/minet/casio/LutzKohl/StudienarbeitLutzKohl.pdf`. – Zugriffsdatum: 01.10.2008

[Kohl 2005] Kohl, Lutz: Puck – eine visuelle Programmiersprache für die Schule. In: (Friedrich, 2005), S. 309–318

[Kohl 2006a] Kohl, Lutz: Mit Puck einfach Programmieren lernen. In: *Jenaer Schriften zur Mathematik und Informatik* (2006), Nr. 1. – URL `http://www.uni-jena.de/data/unijena_/faculties/minet/casio/Puck/Lutz+Kohl+Mit+Puck+einfach+Programmieren+lernen.pdf`. – Zugriffsdatum: 01.10.2008

[Kohl 2006b] Kohl, Lutz: Puck - eine visuelle Programmiersprache für die Schule. In: *informatica didactica* 7 (2006). – URL `http://www.informatica-didactica.de/InformaticaDidactica/Kohl2006.pdf`. – Zugriffsdatum: 01.10.2008

[Kohl 2007a] Kohl, Lutz: Informatik mit der visuellen Programmiersprache Puck. In: Stechert, Peer (Hrsg.): *Informatische Bildung in der Wissensgesellschaft. Praxisband der 12. Fachtagung Ïnformatik und Schule - INFOS 2007"der Gesellschaft für Informatik e. V.* Siegen : universi, 2007, S. 75–76

[Kohl 2007b] Kohl, Lutz: Puck - a visual programming system for schools. In: Lister, Raymond (Hrsg.) ; Simon (Hrsg.): *Koli Calling 2007, Proceedings of*

*the Seventh Baltic Sea Conference on Computing Education Research* Bd. 88, Australian Computer Society, 2007. – URL `http://crpit.com/confpapers/CRPITV88Kohl.pdf`. – Zugriffsdatum: 01.10.2008

[Kohl 2008a] Kohl, Lutz: Arbeitsmaterial zur Untersuchung "Kompetenzorientierter Informatikunterricht mit der visuellen Programmiersprache Puck". In: *Jenaer Schriften zur Mathematik und Informatik* 4 (2008). – URL `http://www.minet.uni-jena.de/preprints/kohl_08/KohlArbeitsmaterialUntersuchung.pdf`. – Zugriffsdatum: 01.10.2008

[Kohl und Fothe 2007] Kohl, Lutz ; Fothe, Michael: Algorithmen aus einer anderen Perspektive. In: *LOG IN* (2007), Nr. 146/147, S. 20–22

[Kohl u. a. 2007] Kohl, Lutz ; Meissner, Gabor ; Schmidt, Harald: Puck – ein Sommernachtstraum. In: *LOG IN* (2007), Nr. 144, S. 39–47. – URL `http://www.log-in-verlag.de/wwwredlogin/Archiv/2007/144/LOG_IN_144_39-47_Kohl.PDF`. – Zugriffsdatum: 01.10.2008

[Kohl und Romeike 2006] Kohl, Lutz ; Romeike, Ralf: Aktueller Stand der Objektorientierung bei Informatiklehrerinnen und -lehrern. In: Forbig, P. (Hrsg.) ; Siegel, G. (Hrsg.) ; Schneider, M. (Hrsg.): *Hochschuldidaktik der Informatik, HDI 2006*, Gesellschaft für Informatik e. V., 2006, S. 63–75. – URL `http://www.uni-jena.de/img/unijena_/faculties/minet/casio/veroeffentlichungenKohl/KohlRomeikeHDI2006.pdf`. – Zugriffsdatum: 01.10.2008

[Kohl 2008b] Kohl, Matthias: Experten für das "Lernen lernen"?! – Lernprozessbegleitung in der IT-Aus- und Weiterbildung als Praxisbeispiel für den Umgang mit veränderten Anforderungen an das Bildungspersonal. In: *bwp@* (2008), Nr. 8. – URL `http://www.bwpat.de/ht2008/ws25/kohl_ws25-ht2008_spezial4.pdf`. – Zugriffsdatum: 02.10.2008

[Koubek 2005] Koubek, Jochen: E-Mail-Kompetenzen – Ein Beispiel zu Standards für die informatische Bildung. In: *LOG IN* (2005), Nr. 135, S. 61–65

[Kujath 2006] Kujath, Bertold: Ein Test- und Analyseverfahren zur Kontrastierung von Problemlöse-Prozessen informatischer Hoch- und Niedrigleister – erste Ergebnisse einer Pilotstudie. In: Schwill, Andreas (Hrsg.) ; Schulte, Carsten (Hrsg.) ; Thomas, Marco (Hrsg.): *3. Workshop der GI-Fachgruppe "Didaktik der Informatik" 19.-20. Juni 2006 an der Universität Potsdam*. Bonn : GI, 2006, S. 49–70

[Kurz 2006] Kurz, Sabine: Outputorientierung in der Qualitätsentwickung. In: Rauner, Felix (Hrsg.): *Handbuch Berufsbildungsforschung*. 2. Auflage. 2006, S. 427–435

[Landerer 2006] Landerer, Claudio: *Die Nintendo-Generation lernt Programmieren*, Universität Salzburg, Diplomarbeit, 2006. – URL `http://www.uni-salzburg.at/pls/portal/docs/1/288296.PDF`. – Zugriffsdatum: 01.10.2008

[Lange und Axelsson 2006] Lange, Martin ; Axelsson, Roland: *Teaching Teaching & Understanding Understanding*. Deutsche Untertitel des Videos, Internetdokument. 2006. – URL `http://www.daimi.au.dk/~brabrand/short-film/part-3-DE.html`. – Zugriffsdatum: 02.10.2008

[Lankes u. a. 2006] Lankes, Eva-Maria ; Lorenzen, Hinrich ; Petersen, Christiane ; Urban, Sabine von ; Zelewski, Hans-Dieter von ; Zielinski, Dieter ; Institut für Qualitätsentwicklung an Schulen Schleswig-Holstein (Hrsg.): *Kompetenzorientierter Mathematikunterricht, Anregungen für die Arbeit mit den Bildungsstandards zum Hauptschulabschluss und mittleren Abschluss (Sekundarstufe I)*. Internetdokument. 2006. – URL `http://faecher.lernnetz.de/links/materials/1152252908.pdf`. – Zugriffsdatum: 01.10.2008

[Leuders 2006] Leuders, Timo: Kompetenzorientierte Aufgaben im Unterricht. In: Blum, Werner (Hrsg.) ; Drüke-Noe, Christina (Hrsg.) ; Hartung, Ralph (Hrsg.) ; Köller, Olaf (Hrsg.): *Bildungsstandards Mathematik: konkret*. Cornelsen Scriptor, 2006, S. 81–95

[Lister u. a. 2006] Lister, Raymond ; Simon, Beth ; Thompson, Errol ; Whalley, Jacqueline L. ; Prasad, Christine: Not seeing the forest for the trees: novice programmers and the SOLO taxonomy. In: *Proceedings of the 11th annual SIGCSE conference on Innovation and technology in computer science education (ITICSE '06)*. Bolonga, Italy. : ACM Press, 2006, S. 118–122. – URL `http://www.cs.ucsd.edu/~bsimon/pubs/papers/ListerEtAlITiCSE2006.pdf`. – Zugriffsdatum: 01.10.2008

[Magenheim 2005] Magenheim, Johannes: Towards a Competence Model for Educational Standards of Informatics. In: Samways, Brian (Hrsg.): *8th IFIP World Conference on Computers in Education (WCCE 2005); 4-7 July 2005*. University of Stellenbosch. Cape Town, South Africa, 2005. – URL `http://ddi.uni-paderborn.de/fileadmin/Informatik/AG-DDI/Veroeffentlichungen/Paper/2005/towards_a_competence_model_for_educational_standards_of_informatics.pdf`. – Zugriffsdatum: 01.10.2008

[Maloney u. a. 2004] Maloney, John ; Burd, Leo ; Kafai, Yasmin ; Rusk, Natalie ; Silverman, Brian ; Resnick, Mitchel: Scratch: A Sneak Preview. In: *Second International Conference on Creating, Connecting and Collaborating*

*through Computing* (2004), S. 104–109. – URL `http://llk.media.mit.edu/papers/ScratchSneakPreview.pdf`. – Zugriffsdatum: 01.10.2008

[Mannila 2006] Mannila, Linda: Progress reports and novices' understanding of program code. In: *Proceedings of the 6th Baltic Sea conference on Computing education research*, ACM, 2006, S. 27–31. – URL `http://doi.acm.org/10.1145/1315803.1315810`. – Zugriffsdatum: 01.10.2008

[Matthäus 2006] Matthäus, Wolf-Gert: *Grundkurs Programmieren mit Delphi*. 2. Auflage. Vieweg+Teubner, 2006

[Meyerhöfer 2004] Meyerhöfer, Wolfram: Zum Kompetenzstufenmodell von PISA. In: *Journal für Mathematik Didaktik* (2004), Nr. 3/4, S. 294–305

[Möller 1995] Möller, Christine: Die curriculare Didaktik. In: Gudjons, Herbert (Hrsg.) ; Teske, Rita (Hrsg.) ; Rainer, Winkel (Hrsg.): *Didaktische Theorien*. 8. Auflage. Hamburg : Bermann + Helbig Verlag, 1995, S. 62–77

[Modrow 2005] Modrow, Eckart: *Standards für den Informatikunterricht in der Sekundarstufe I des Gymnasiums*. Internetdokument. 2005. – URL `http://www.vlin.de/sek1/standards.pdf`. – Zugriffsdatum: 01.10.2008

[Moser 2006] Moser, Heinz: Standards für die Medienbildung. In: *Computer und Unterricht* 63 (2006), S. 49–55

[Naps u. a. 2003] Naps, Thomas L. ; Rössling, Guido ; Almstrum, Vicki ; Dann, Wanda ; Fleischer, Rudolf ; Hundhausen, Chris ; Korhonen, Ari ; Malmi, Lauri ; McNally, Myles ; Rodger, Susan ; Velazquez-Iturbide, J. A.: Exploring the Role of Visualization and Engagement in Computer Science Education. In: *SIGCSE Bulletin* 35 (2003), Nr. 2, S. 131–152. – URL `http://www.cs.hut.fi/Research/SVG/publications/exploring_role_of_vis_and_engagement_in_cse.pdf`. – Zugriffsdatum: 02.10.2008

[National Research Council 2001] National Research Council (Hrsg.): *Knowing what students know – The science and design of educational assessment*. National Research Council, 2001

[NCTM 2000] NCTM (Hrsg.): *Principles and Standards for School Mathematics*. Reston : NCTM, 2000. – URL `http://standards.nctm.org/`. – Zugriffsdatum: 02.10.2008

[Neuweg 2004] Neuweg, Georg H.: Bildungsstandards in Österreich – Über die gute Absicht, die Vereinbarkeit von Einsicht und Aufsicht und die gebotene Vorsicht. In: *Pädaktuell* (2004), Nr. 2, S. 4–13. – URL `http://www.wipaed.jku.at/images/stories/forschung/BildungsstandardsOesterreich.pdf`. – Zugriffsdatum: 01.10.2008

[Neuweg 2005] Neuweg, Georg H.: Vorsichtsstandards für den Umgang mit Bildungsstandards. In: *bwp@* (2005), Nr. 8. – URL `http://www.bwpat.de/ausgabe8/neuweg_bwpat8.pdf`. – Zugriffsdatum: 02.10.2008

[OECD 2004] OECD (Hrsg.): *Lernen für die Welt von morgen – Erste Ergebnisse von PISA 2003*. URL `http://www.pisa.oecd.org/dataoecd/18/10/34022484.pdf`. – Zugriffsdatum: 02.10.2008, 2004

[OECD 2005] OECD (Hrsg.): *PISA 2003 Technical Report*. URL `http://www.pisa.oecd.org/dataoecd/49/60/35188570.pdf`. – Zugriffsdatum: 01.10.2008, 2005

[Ohrnberger u. a. 2007] Ohrnberger, Elfriede ; Klein, Werner ; Kaufmann, Helmut: Gesamtstrategie der Kultusministerkonferenz zum Bildungsmonitoring. In: Labudde, Peter (Hrsg.): *Bildungsstandards am Gymnasium*. hep-Verlag, 2007, S. 87–96

[Orlamünder u. a. 2002] Orlamünder, Dieter ; Liskowsky, Rüdiger ; Hussmann, Heinrich: *Softwareentwicklung mit Delphi – Eine systematische Einführung*. Hanser Fachbuchverlag, 2002

[Papert 1985] Papert, Seymour ; Moos, Ludwig (Hrsg.) ; Waffender, Manfred (Hrsg.): *Gedankenblitze – Kinder, Computer und neues Lernen*. Rowohlt, 1985

[Pattis 1995] Pattis, Richard E. ; Roberts, Jim (Hrsg.) ; Stehlik, Mark (Hrsg.): *Karel the Robot, A Gentle Introduction to the Art of Programming*. 2. Auflage. John Wiley & Sons, 1995

[Paulsen und von Saldern 2005] Paulsen, Arne ; Saldern, Matthias von: *Die nationalen Bildungsstandards für den Mittleren Schulabschluss der Kultusministerkonferenz im Vergleich zu den Vorschlägen des Gutachtens 'Zur Entwicklung nationaler Bildungsstandards' ('Klieme-Gutachten') und den Erkenntnissen nach PISA*. Internetdokument. 2005. – URL `http://www.leuphana.de/fileadmin/user_upload/PERSONALPAGES/Fakultaet_1/von_Saldern_Matthias/ASS_Material/Arbeitsberichte/2005_01_Bildungsstandards.pdf`. – Zugriffsdatum: 02.10.2008

[Perkins u. a. 1989] Perkins, D. N. ; Hancock, C. ; Hobbs, R. ; Martin, F. ; Simmons, R.: Conditions of Learning in Novice Programmers. In: Soloway, E. (Hrsg.) ; Sphorer, J. C. (Hrsg.): *Studying the novice programmer*. Lawrence Erlbaum, 1989, S. 261–279

[Pomberger 1999] Pomberger, Gustav: Prozedurorientierte Programmierung. In: Pomberger, Gustav (Hrsg.) ; Rechenberg, Peter (Hrsg.): *Informatik-Handbuch*. 2. Auflage. Hanser Verlag, 1999, S. 517–528

[Puhlmann 2003] Puhlmann, Hermann: Informatische Literalität nach dem PISA-Muster. In: (Hubwieser, 2003), S. 145–154

[Puhlmann 2005] Puhlmann, Hermann: Standards-orientierte Aufgaben. In: *LOG IN* (2005), Nr. 135, S. 29–31

[Puhlmann und Friedrich 2007] Puhlmann, Hermann ; Friedrich, Steffen: Die Standards – und wie weiter? Zur Beurteilung von Aufgaben für Unterricht und Test. In: *LOG IN* (2007), Nr. 146/147, S. 16–19

[Puntsch 1993] Puntsch, Eberhard: *Zitate Handbuch Für Wissenschaftler, Journalisten, Politiker Künstler, Manager, Redner, Erzieher, Korrespondenten.* Weltbild Verlag, 1993

[Reiss 2007] Reiss, Kristina: Bildungsstandards – eine Zwischenbilanz am Beispiel der Mathematik. In: Bayrhuber, Horst (Hrsg.) ; Elster, Doris (Hrsg.) ; Krüger, Dirk (Hrsg.) ; Vollmer, Helmut J. (Hrsg.): *Kompetenzentwicklung und Assessment.* Innsbruck : Studien Verlag, 2007, S. 19–34

[Röhner 2005] Röhner, Gerhard: *JAVA - Kursmaterial – Algorithmen und Datenstrukturen.* Internetdokument. 2005. – URL `http://informatik.bildung-rp.de/fileadmin/user_upload/informatik.bildung-rp.de/Fortbildung/pdf/J1-Kurs.pdf`

[Röhner 2007] Röhner, Gerhard: Das Landesabitur Informatik in Hessen. In: *LOG IN* (2007), Nr. 148/149, S. 36–41

[Robins u. a. 2003] Robins, A. ; Rountree, J. ; Rountree, N.: Learning and Teaching Programming: A Review and Discussion. In: *Computer Science Education* 13 (2003), Nr. 2, S. 137–172. – URL `http://www.cs.otago.ac.nz/staffpriv/anthony/publications/pdfs/RobinsRountreeRountree.pdf`. – Zugriffsdatum: 02.10.2008

[Robins u. a. 2006] Robins, Anthony ; Haden, Patricia ; Garner, Sandy: Problem distributions in a CS1 course. In: *ACE '06: Proceedings of the 8th Austalian conference on Computing education.* Darlinghurst, Australia : Australian Computer Society, 2006, S. 165–173. – URL `http://portal.acm.org/citation.cfm?id=1151891&dl=GUIDE&coll=GUIDE`. – Zugriffsdatum: 02.10.2008

[Romeike 2007a] Romeike, Ralf: Animationen und Spiele gestalten – Ein kreativer Einstieg in die Programmierung. In: *LOG IN* (2007), Nr. 146/147, S. 36–44

[Romeike 2007b] Romeike, Ralf: Kriterien kreativen Informatikunterrichts. In: (Schubert, 2007)

[Rust 1985] Rust, Hans-Peter: Informatikunterricht in der Sekundarstufe I. In: *Die Deutsche Schule* 77 (1985), S. 367–370

[Schiffer 1998] Schiffer, Stephan: *Visuelle Programmierung: Grundlagen und Einsatzmöglichkeiten*. Addison-Wesley-Longman, 1998

[Schlüter und Brinda 2007] Schlüter, Kirsten ; Brinda, Torsten: Auf dem Weg zu Bildungsstandards für Konzepte der Theoretischen Informatik in der Sekundarstufe. In: (Schubert, 2007)

[Schott und Ghanbari 2008] Schott, Franz ; Ghanbari, Shahram A.: *Kompetenzdiagnostik, Kompetenzmodelle, kompetenzorientierter Unterricht*. Münster : Waxmann, 2008

[Schubert 2007] Schubert, Sigrid (Hrsg.): *INFOS 2007 Didaktik der Informatik in Theorie und Praxis 12. GI-Fachtagung Informatik und Schule 19.-21. September 2007 in Siegen*. Gesellschaft für Informatik e.V., 2007

[Schubert und Schwill 2004] Schubert, Sigrid ; Schwill, Andreas: *Didaktik der Informatik*. Berlin : Spektrum Akademischer Verlag, 2004

[Schubert u. a. 2005] Schubert, Sigrid ; Stechert, Peer ; Freischlad, Stefan: Die Phisher im Internet - Ein Beitrag zu Standards der informatischen Bildung. In: *LOG IN* (2005), Nr. 135, S. 66–68

[Schwarz u. a. 2007] Schwarz, Günther ; Leonhartsberger, Christa ; Kroupa, Thomas ; Hutterer, Florian ; Eizinger, Johann ; Pühler, Josef ; Kloimböck, Christian ; Herbst, Marisa: *In der Unterstufe spielerisch Programmieren lernen mit JavaKara und Robot Karol*. Projektbericht, Internetdokument. 2007. – URL `http://imst.uni-klu.ac.at/materialien/2007/2230_556_Langfassung_Schwarz.pdf`. – Zugriffsdatum: 02.10.2008

[Schwill 1993] Schwill, Andreas: Fundamentale Ideen der Informatik. In: *Zentralblatt für Didaktik der Mathematik* (1993), Nr. 1, S. 20–31. – URL `http://www.informatikdidaktik.de/Forschung/Schriften/ZDM.pdf`

[Sedlmeier und Renkewitz 2008] Sedlmeier, Peter ; Renkewitz, Frank: *Forschungsmethoden und Statistik in der Psychologie*. 1. Auflage. Pearson Studium, 2008

[Shakespeare 1989] Shakespeare, William: *A Midsummer Night's Dream*. Reclam, 1989

[Taylor u. a. 1986] Taylor, Robert P. ; Cunniff, Nancy ; Uchiyama, Minh: Learning, research, and the graphical representation of programming. In: *ACM '86: Proceedings of 1986 ACM Fall joint computer conference*. Los

Alamitos, USA : IEEE Computer Society Press, 1986, S. 56–63. – URL http://portal.acm.org/citation.cfm?id=324543&dl=ACM&coll=portal. – Zugriffsdatum: 02.10.2008

[The College Board 2007] The College Board: *AP Course Description: Computer Science*. 2007. – URL http://apcentral.collegeboard.com/apc/public/repository/ap08_compsci_coursedesc.pdf. – Zugriffsdatum: 02.10.2008

[Thüringer Kultusministerium 1999] Thüringer Kultusministerium: *Lehrplan für das Gymnasium Informatik*. Internetdokument. 1999. – URL http://www.thillm.de/thillm/pdf/lehrplan/gy/gy_lp_if.pdf. – Zugriffsdatum: 01.10.2008

[Thüringer Kultusministerium 2008] Thüringer Kultusministerium: *Thüringer Medienschulen*. Internetdokument. 2008. – URL http://www.thueringen.de/de/tkm/schule/schulwesen/schulentwicklung/medienschulen/. – Zugriffsdatum: 17.10.2008

[Trim u. a. 2001] Trim, John ; North, Brian ; Coste, Daniel ; Sheils, Joseph ; Goethe-Institut Inter Nations (Hrsg.) ; KMK (Hrsg.) ; EDK (Hrsg.) ; BMBWK (Hrsg.): *Gemeinsamer europäischer Referenzrahmen für Sprachen: lernen, lehren, beurteilen*. Langenscheidt-Verlag, 2001. – URL http://www.goethe.de/z/50/commeuro/deindex.htm

[Vollmer 2005] Vollmer, Helmut J.: *Kompetenzen und Bildungsstandards – Stand der Entwicklung in verschiedenen Fächern*. Internetdokument. November 2005. – URL http://gfd.physik.hu-berlin.de/texte/Vollmer-Workshop-Einleitung.doc. – Zugriffsdatum: 02.10.2008

[Vollmost 2005] Vollmost, Manfred: Ein Kerncurriculum Informatik. In: *LOG IN* (2005), Nr. 135, S. 54–60

[Weeger 2007] Weeger, Moritz: *Synopse zum Informatikunterricht in Deutschland*. Bakkalaureatsarbeit, TU-Dresden. 2007. – URL http://output.inf.tu-dresden.de/homepages/uploads/media/synopse_weeger.pdf. – Zugriffsdatum: 02.10.2008

[Weinert 1999] Weinert, Franz E. ; OECD (Hrsg.): *Konzepte der Kompetenz*. 1999

[Weinert 2002] Weinert, Franz E.: Vergleichende Leistungsmessung in Schulen – eine umstrittene Selbstverständlichkeit. In: Weinert, Franz E. (Hrsg.): *Leistungsmessungen in Schulen*. Weinheim, 2002, S. 17–31

[Wittmann 1976] Wittmann, Erich: *Grundfragen des Mathematikunterrichts*. 4. Auflage. Vieweg, 1976

[Zentrale für Unterrichtsmedien im Internet e.V. 2008] Zentrale für Unterrichtsmedien im Internet e.V.: *ZUM-Wiki*. Internetdokument. 2008. – URL `http://wiki.zum.de/Hauptseite`. – Zugriffsdatum: 02.10.2008

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Abkürzungsverzeichnis

BMBF Bundesministerium für Bildung und Forschung

BMBWK Österreichisches Bundesministerium für Bildung, Wissenschaft und Kultur

ACM Association for Computing Machinery

ECDL European Computer Driving Licence

EDK Schweizerische Konferenz der Kantonalen Erziehungsdirektoren

EPA einheitlichen Prüfungsanforderungen für die Abiturprüfung

GNU GNU is not Unix

GUI Graphical User Interface

ICT Information and Communications Technology

IQB Institut zur Qualitätsentwicklung im Bildungswesen

IS Informatics System

JDK Java Development Kit

K-12 amerikanische Bezeichnung für den Unterricht vom Kindergarten über die Primarstufe bis zur Sekundarstufe, die 12 steht für die Anzahl der Schuljahre

n. s. nicht signifikant

NCTM National Council of Teachers of Mathematics

OECD Organisation for Economic Co-operation and Development

SD (engl.: standard deviation) Standardabweichung

UML Unified Modeling Language

z. B. zum Beispiel

GI Gesellschaft für Informatik e. V.

ISTE International Society for Technology in Education

KMK Die ständige Konferenz der Kultusminister der Länder in der Bundesrepublik Deutschland

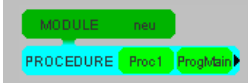
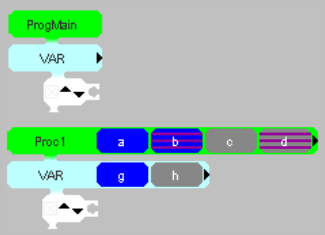
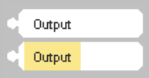
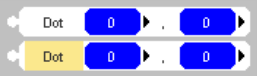
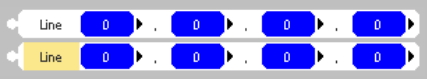
SOLO Structure of the Observed Learning Outcome

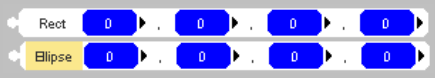
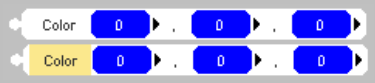
u. a. und andere

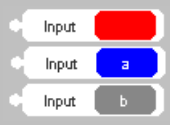
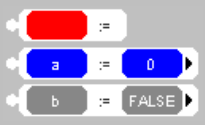
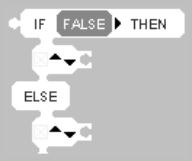
# Anhang

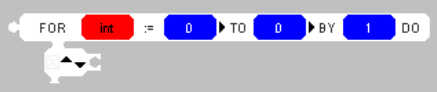
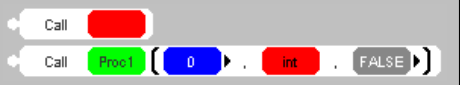
# A Funktionalität, Einstellmöglichkeiten und Darstellung der Puck Bausteine

In der folgenden Tabelle sind in der linken Spalte die im Puck-System verwendbaren Bausteine abgebildet, in der rechten Spalte werden dazu jeweils Funktionalität, Einstellmöglichkeiten und Darstellung erläutert. Die visuelle Programmiersprache Puck wurde im Kapitel 4 vorgestellt.

| **Baustein** | **Funktionalität, Einstellmöglichkeiten und Darstellung** |
|---|---|
| Modul   | Im MODULE-Baustein kann der Name des Programms sowie eine kurze Beschreibung festgelegt werden. Prozeduren können deklariert und benannt werden. Die Startprozedur ProgMain ist bereits vordefiniert. |
| Prozedur   | Prozeduren dienen als Unterprogramme und können mit Hilfe von Parametern gesteuert werden. Paramter können entweder als Wertparameter (keine Schraffierung) oder als Referenzparameter (violette Schraffierung) übergeben werden. Variablen und Parameter können vom Datentyp Integer (blau) oder Boolean (grau) sein. Mit den Anschlusstellen einer Prozedur können die verschiedenen Anweisungs-Bausteine verbunden werden. |
| Textausgabe   | Dem Anwender wird ein Text ausgegeben. Je nach Wahl erfolgt die Textausgabe mit Zeilenumbruch (weißer Baustein) oder ohne Zeilenumbruch (gelber Baustein). |
| Dot   | Mit dem Dot-Baustein ist es möglich, einen Punkt an der mit der x- und y-Koordinate angegebenen Stelle in dem Ausgabefenster zu erzeugen. Je nach Wahl wird der Punkt mit der aktuellen Vordergrundfarbe gezeichnet (weißer Baustein) oder gelöscht (gelber Baustein). |
| Line   | Durch den Line-Baustein können Linien im Ausgabefenster erzeugt werden. Dafür müssen die Koordinaten von zwei Punkten angegeben werden. Je nach Wahl wird die Linie mit der aktuellen Vordergrundfarbe gezeichnet (weißer Baustein) oder gelöscht (gelber Baustein). |

| Baustein | Funktionalität, Einstellmöglichkeiten und Darstellung |
|---|---|
| Rectangle/Ellipse   | Durch den Rectangle/Ellipse-Baustein können Rechtecke und Ellipsen im Ausgabefenster erzeugt werden. Dafür müssen die Koordinaten von zwei Punkten (z. B. links unten und rechts oben) angegeben werden. Je nach Wahl in der Attributtabelle ist der Baustein mit Rect bzw. Ellipse bezeichnet und wird ausgefüllt (weißer Baustein) oder ungefüllt (gelber Baustein) gezeichnet. |
| Sleep   | Der Sleep-Baustein stoppt die Ausführung des Programms für eine bestimmte Zeit. Die Dauer der Programmunterbrechung wird als Integer-Ausdruck in hundertstel Sekunden angegeben. |
| Color   | Durch den Color-Baustein kann die aktuelle Zeichenfarbe geändert werden (weißer Baustein) bzw. der Hintergrund neu eingefärbt werden (gelber Baustein). Die Auswahl erfolgt in der Attributtabelle. In den drei Integerausdrücken werden Werte von 0 bis 255 für die Farben Rot, Grün und Blau angegeben. |
| Sound   | Durch den Sound-Baustein kann ein Ton gespielt werden. Tonhöhe (0=c,1=cis,2=d,3=dis, ...) und Tondauer (z. B. 1=$\frac{1}{16}$, 2=$\frac{1}{8}$, 4=$\frac{1}{4}$, 8=$\frac{1}{2}$, 16=$\frac{1}{1}$) werden als Integer-Ausdrücke angegeben. Ein Instrument (Piano, Xylophon, Guitar, Violin, Pause) kann mit der rechten Maustaste ausgewählt werden. Außerdem werden die Töne je nach Wahl gepuffert (gelber Baustein) oder gespielt (weißer Baustein). Gepufferte Töne werden in einer Liste gesammelt und vor dem Abarbeiten des nächsten gespielten Tons mit ausgegeben. |
| Random   | Durch den Random-Baustein wird eine Variable mit einem Zufallswert initialisiert. Wenn noch keine Variable ausgewählt wurde, ist der Baustein rot gekennzeichnet. Bei Integer-Variablen (blau) wird hinter *range* die Obergrenze des Bereichs angegeben, aus dem die Zufallszahl gewählt wird (1..range). Bei Boolean-Variablen (grau) wird hinter *probability* eine Wahrscheinlichkeit angegeben, mit der *true* gewählt wird (80 = Mit einer Wahrscheinlichkeit von 80 % wird *true* gewählt). |

| Baustein | Funktionalität, Einstellmöglichkeiten und Darstellung |
|---|---|
| Input   | Der Benutzer wird zur Eingabe aufgefordert. Der eingegebene Wert wird in die ausgewählte Variable gespeichert. Wenn noch keine Variable ausgewählt wurde, ist der Baustein rot gekennzeichnet. Bei Integer-Variablen (blau) kann eine Eingabeaufforderung bzw. eine Frage angegeben werden. Bei Boolean-Variablen (grau) kann zusätzlich noch ein Text für *true* bzw. *false* angegeben werden. Die eingegebenen Texte werden dann als Antwortmöglichkeiten auf die in der Eingabeaufforderung gestellte Frage aufgeführt. |
| Output   | Dem Anwender wird der Wert eines Integer-Ausdrucks im Ausgabefenster ausgegeben. In der Attributtabelle kann ein Text angegeben werden, der vor dem Wert des Integer-Ausdrucks ausgegeben wird. |
| Zuweisung   | Bei einer Zuweisung wird in der Variablen auf der linken Seite der Wert des Ausdrucks auf der rechten Seite gespeichert. Wenn noch keine Variable ausgewählt wurde, ist der Baustein rot gekennzeichnet. Je nach Wahl der Variablen entsteht auf der rechten Seite der Zuweisung ein Integer- (blau) bzw. ein Boolean-Ausdruck (grau). |
| IF- THEN   | Beim IF-THEN-Baustein wird eine Bedingung überprüft. Ist die Bedingung erfüllt, so werden die Anweisungen des THEN-Zweiges ausgeführt. Ist die Bedingung nicht erfüllt, so werden die Anweisungen des ELSE-Zweiges ausgeführt. |
| WHILE-DO   | Beim WHILE-DO-Baustein wird eine Bedingung jeweils am Anfang überprüft. Die mit den Anschlussstellen verbundene Anweisungsfolge wird so oft wiederholt abgearbeitet, wie das Auswerten der Bedingung *true* ergibt. |
| REPEAT-UNTIL   | Beim REPEAT-UNTIL-Baustein wird die mit den Anschlussstellen verbundene Anweisungsfolge abgearbeitet und anschließend wird eine Bedingung überprüft. Solange das Auswerten der Bedingung *false* ergibt, wird erneut mit der Abarbeitung begonnen. Erst wenn das Ergebnis der Überprüfung *true* ist, wird die Abarbeitung der Schleife abgebrochen. |

| Baustein | Funktionalität, Einstellmöglichkeiten und Darstellung |
|---|---|
| FOR   | Bei einem FOR-Baustein wird der Wert einer Integer-Variablen von einem Startwert bis zu einem Endwert um eine bestimmte Schrittweite erhöht. Die mit den Anschlussstellen verbundenen Bausteine werden in jedem Schritt abgearbeitet. |
| Prozeduraufruf   | Eine Prozedur ist ein Unterprogramm. Bei jedem Prozeduraufruf durch den mit „Call“ beschrifteten Baustein wird die gewählte Prozedur (grün) abgearbeitet. Je nach Prozedur müssen für die aktuellen Paramter Ausdrücke (Wertparameter) bzw. Variablen (Referenzparameter) der Datentypen Integer (blau) bzw. Boolean (grau) übergeben werden. Ausdrücke sind dem Datentyp entsprechend initialisiert. Wenn Variablen übergeben werden müssen, so ist der aktuelle Parameter zu Beginn rot gekennzeichnet und enthält eine Kurzform des Datentyps (int bzw. bool). |

# B Einordnung der Kompetenzen des entwickelten Modells in die SOLO-Taxonomie

Nachfolgend werden für die vier Komponenten die einzelnen Kompetenzen der Stufen I, II und III in die SOLO-Taxonomie eingeordnet. Das Kompetenzmodell und die SOLO-Taxonomie wurden im Kapitel 5 vorgestellt.

## Komponente A – Eigenschaften von Algorithmen

**Den Algorithmusbegriff und die wesentlichen Eigenschaften von Algorithmen erklären** Auf Stufe I sollen die Schülerinnen und Schüler den im Unterricht behandelten Algorithmusbegriff und die Eigenschaften von Algorithmen erklären können. Insgesamt kann die Kompetenz SOLO 2 zugeordnet werden, da auf dieser Stufe lediglich eine Wiedergabeleistung gefordert ist. Auf Stufe II sollen sich die Erklärungen auf bekannte Beispiele beziehen. Auch das kann als einfache Wiedergabe von Inhalten deklariert werden (SOLO 2). Das Erklären der Eigenschaften von Algorithmen an selbst konstruierten Beispielen, z. B. zu einem vorgegebenen Themenbereich erfordert von den Lernenden ein tieferes Verständnis des Algorithmusbegriffs und das in Betracht ziehen der verschiedenen Eigenschaften. Dies entspricht SOLO 3.

**Die wesentlichen Eigenschaften von Algorithmen überprüfen** Auf Stufe I sollen die wesentlichen Eigenschaften von Algorithmen in einfachen Fällen überprüft werden. Hierfür ist es nötig die verschiedenen Eigenschaften zu kennen und diese in Beispielsituationen korrekt anwenden zu können. Da die Eigenschaften auf dieser Stufe noch einzeln und nicht im Zusammenhang betrachtet werden sollen, kann die Kompetenz SOLO 3 zugeordnet werden. Wenn in Stufe II und III aufgrund der Eigenschaften begründet werden soll, ob ein Handlungsablauf ein Algorithmus ist, so kann dies nur Erfolgen wenn die Eigenschaften von Algorithmen im Zusammenhang und in Verbindung mit dem Algorithmusbegriff verstanden wurden. Das Begründen kann also SOLO 4 zugeordnet werden.

**Probleme nennen, die mit Hilfe von Algorithmen lösbar, bzw. nicht lösbar sind** Die Kompetenz, Probleme zu nennen, die mit Hilfe von Algorithmen lösbar bzw. nicht lösbar sind, kann auf allen drei Stufen SOLO 2 zugeordnet werden. Denn hier soll nur bekanntes Wissen wiedergegeben werden.

## Komponente B – Algorithmische Grundbausteine und Datentypen

**Algorithmische Grundbausteine erklären und anwenden** Die Erklärung der algorithmischen Grundbausteine und ihre Anwendung kann auf allen drei Stufen SOLO 3 zugeordnet werden. Beide Tätigkeiten können unabhängig voneinander ausgeführt werden, ein Herstellen von Zusammenhängen ist an dieser Stelle nicht gefordert. Die Erweiterung der Kompetenz um Prozeduren und Parameter auf Stufe III ist lediglich inhaltlicher Art.

**Algorithmische Grundbausteine darstellen** Während auf Stufe I lediglich Pseudocode als Darstellungsform für algorithmischen Grundbausteine gefordert ist (SOLO 2), werden in Stufe II verschiedene Darstellungsformen gefordert (SOLO 3) und um Stufe III zu erreichen, müssen die Lernenden zwischen unterschiedlichen Darstellungsformen wechseln können. Hierfür ist ein tieferes Verständnis der Darstellungsformen und deren Verhältnis zueinander nötig (SOLO 4).

**Datentypen verwenden** Auf Stufe I des Kompetenzmodells muss lediglich ein numerischer Datentyp verwendet werden (SOLO 2). Auf Stufe II und III wird die Anwendung von verschiedenen Datentypen gefordert (SOLO 3).

## Komponente C – Arbeit mit Algorithmen

**Gegebene Algorithmen analysieren** Um gegebene Algorithmen zu analysieren und nachzuvollziehen müssen im Allgemeinen mehrere algorithmische Grundbausteine bekannt sein. Auf Stufe I sollen die Lernenden in der Lage sein, einfache in Pseudocode gegebene Algorithmen zu lesen. Dafür reicht es meist aus, die Funktionsweise der einzelnen Grundbausteine zu betrachten (SOLO 3). Für die Analyse des Leistungsumfangs von Algorithmen auf Stufe II und III müssen verschiedener Grundbausteine miteinander in Beziehung gesetzt und zu einem Gesamtverständnis des Algorithmus zusammengefügt werden. Die Kompetenz kann auf Stufe II und III somit SOLO 4 zugeordnet werden.

**Algorithmen prüfen** Das Prüfen von Algorithmen, auch unter Verwendung von Durchlauftabellen (Stufe II) kann als Abarbeiten eines bekannten Schemas interpretiert werden. Da hierfür die verschiedenen algorithmischen Grundbausteine bekannt sein müssen, die allerdings isoliert voneinander betrachtet werden können, wird dieser Kompetenz auf den Stufen I und II SOLO 3 zugeordnet. Während bei Stufe I und II die Beispiele gegeben werden, müssen die Lernenden auf Stufe III typische und untypische Beispiele selbst finden. Dafür muss der gesamten Algorithmus sowie dessen Funktionsweise verstanden worden sein. Deshalb wird dieser Kompetenz auf Stufe III SOLO 4 zugeordnet.

**Algorithmen in Programme umsetzen** Um gegebene Algorithmen in Programme umzusetzen, müssen von den Lernenden verschiedene Fertigkeiten, beherrscht werden (vgl. Abschnitt 4.3). Hierzu zählen: der Umgang mit einer Programmierumgebung sowie der Test des entwickelten Programms und evtl. Korrekturen. Die Fertigkeiten müssen auch in Beziehung zueinander gesetzt werden, so müssen z. B. Fehlermeldungen interpretiert und durch Veränderungen im Programm beseitigt werden. Das Umsetzen von Algorithmen wird deshalb auf allen Stufen SOLO 4 zugeordnet.

**Algorithmen bzw. Programme modifizieren und ergänzen** Wenn ein gegebener Algorithmus verändert werden soll, so muss er analysiert werden, die richtige Stelle für Modifikationen muss gefunden werden und schließlich müssen algorithmische Grundbausteine entfernt, hinzugefügt, bzw. verändert werden. Das Modifizieren und Ergänzen von Algorithmen enthält also verschiedene Aspekte, die zu einem kohärenten Ganzen zusammengefügt werden müssen und wird somit auf den Stufen I und II SOLO 4 zugeordnet. Wenn auf Stufe III Programme nicht nur nach Vorgaben, sondern auch nach selbst gesetzten Zielen modifiziert und erweitert werden sollen, so entspricht das SOLO 5, denn hier müssen eigene Ideen umgesetzt werden.

**Algorithmen bzw. Programme korrigieren** Um fehlerhafte Algorithmen bzw. Programme korrigieren zu können, muss ein Fehler in einem Programm erst einmal gefunden werden. Hierfür ist eine Vorstellung von der korrekten Arbeitsweise des Programms notwendig. Dies setzt Kenntnisse über die Funktionsweise der algorithmischen Grundbausteine im Zusammenhang voraus. Insgesamt kann dem komplexen Prozess der Korrektur von Algorithmen und Programmen SOLO 4 zugeordnet werden.

## Komponente D – Programmentwicklung

**Programme entwerfen** Beim Entwerfen von Programmen müssen verschiedene algorithmische Grundbausteine zu einem kohärenten Programmentwurf zusammengefügt werden. Dies entspricht SOLO 4. Auf Stufe II und III soll der Entwurf schriftlich erfolgen, das hat keinen Einfluss auf die Einordnung in die SOLO-Taxonomie.

**Programme mit einem Programmiersystem implementieren** Um Programme mit einem Programmiersystem zu implementieren, müssen von den Lernenden ähnliche Fertigkeiten wie beim Umsetzen eines Algorithmus in ein Programm beherrscht werden. Hierzu zählen: der Umgang mit einer Programmierumgebung, das Übersetzen des Entwurfs in einzelne algorithmische Grundbausteine, der Test des entwickelten Programms, evtl. Korrekturen sowie die benutzungsfreundliche Gestaltung (Stufe II und III). Da auch hier die genannten Fertigkeiten in Beziehung zueinander gesetzt werden müssen wird das Implementieren von Programmen mit einem Programmiersystem auf allen Stufen SOLO 4 zugeordnet.

**Programme auf Funktionalität testen** Das Testen von Programmen anhand gegebener Beispiele auf Stufe I und II erfordert lediglich Kenntnisse darüber wie ein Programm gestartet, wie Daten eingegeben und Ergebnisse abgelesen werden. Dies entspricht SOLO 3, da Zusammenhänge im Allgemeinen nicht hergestellt werden müssen. Wenn allerdings auf Stufe III die Eingaben zur Überprüfung der Funktionalität eines Programms selbstständig gewählt werden müssen, so setzt dies zusätzlich Kenntnisse über die Funktionsweise des Programms, bzw. der einzelnen algorithmischen Grundbausteine voraus. Somit müssen verschiedene Aspekte im Zusammenhang beachtet werden. Dies entspricht SOLO 4.

**Über den Lösungsweg reflektieren** Wenn auf Stufe II über den Lösungsweg und auf Stufe III zusätzlich über Vor- und Nachteile der Lösung reflektiert werden soll, so setzt dies ein Gesamtverständnis der gegebenen Aufgabe, des entwickelten Lösungswegs und möglicher Alternativen voraus. Somit wird die Kompetenz auf den Stufen II und III SOLO 4 zugeordnet.

**Programme eigenständig verbessern** Das eigenständige Verbessern von Programmen setzt das Verständnis der Aufgabenstellung, einen sinnvollen Entwurf und ein implementiertes, getestetes Programm voraus. Selbstständig entwickelte Ideen,

die über die gegebenen Informationen hinaus gehen, werden als eigenständige Verbesserungen umgesetzt. Dies entspricht SOLO 5.

# C Beziehungen des Kompetenzmodells zu den GI-Empfehlungen

Im Folgenden werden Beziehungen zwischen den Kompetenzen des entwickelten Kompetenzmodells „Algorithmen“ und den Kompetenzen der Inhalts- und Prozessbereiche der GI-Empfehlungen dargestellt. Zu jedem Inhalts- und jedem Prozessbereich gibt es eine Tabelle. In der linken Spalte sind jeweils Kompetenzen angegeben, die von Schülerinnen und Schülern der Jahrgangsstufe 8 bis 10 in dem entsprechenden Inhalts- bzw. Prozessbereich erwartet werden. Die mittlere Spalte zeigt jeweils die im Kompetenzmodell „Algorithmen“ geforderten Kompetenzen, die mit den Kompetenzen des Inhalts- bzw. Prozessbereichs in Verbindung stehen. In der rechten Spalte der Tabellen werden die Beziehungen zwischen den in der linken und den in der mittleren Spalte angegebenen Kompetenzen erläutert.

## Informationen und Daten

| GI-Empfehlungen<br><br>Schülerinnen und Schüler | Kompetenzmodell „Algorithmen“<br>Schülerinnen und Schüler | Erläuterungen |
|---|---|---|
| • stellen Informationen in unterschiedlicher Form dar | • stellen algorithmische Grundbausteine in verschiedenen Darstellungsformen dar (B II) | Die Darstellung eines Algorithmus beinhaltet die Darstellung von Informationen. |
| • interpretieren Daten im Kontext der repräsentierten Information | • testen Programme anhand gegebener Eingaben auf ihre Funktionalität (D II) | Die Ausgaben von Programmen müssen als richtig oder falsch interpretiert werden. |

| GI-Empfehlungen<br><br>Schülerinnen und Schüler | Kompetenzmodell „Algorithmen“<br>Schülerinnen und Schüler | Erläuterungen |
|---|---|---|
| • kennen und verwenden die Datentypen Text, Zahl und Wahrheitswert | • verwenden verschiedene Datentypen (B II) | In den Aufgaben zum Kompetenzmodell wird auf Stufe I ein numerischer Datentyp gefordert. In Stufe II und III müssen mehrere Datentypen verwendet werden. |
| • kennen und verwenden arithmetische und logische Operationen<br>• stellen Datentypen und Operationen formal dar und nutzen sie sachgerecht | • verwenden verschiedene Datentypen (B II)<br>• analysieren die Funktionsweise und den Leistungsumfang gegebener Algorithmen (C II)<br>• prüfen Algorithmen mit gegebenen Beispielen mithilfe von Durchlauftabellen (Schreibtischtest) (C II)<br>• setzen gegebene Algorithmen in Programme um (C II)<br>• modifizieren und ergänzen Algorithmen bzw. Programme nach Vorgaben (C II)<br>• fertigen einen schriftlichen Entwurf für Programme an (D II)<br>• implementieren Programme mit einem Programmiersystem benutzungsfreundlich (D II) | Um Datentypen sinnvoll verwenden zu können, müssen auch die zugehörigen Operationen bekannt sein und angewendet werden können. Beim Analysieren, Prüfen, Umsetzen und Modifizieren von Algorithmen ist es nötig, Datentypen sachgerecht zu nutzen und die in den gegebenen Algorithmen verwendeten Operationen zu interpretieren, umzusetzen bzw. weitere hinzuzufügen. Auch beim Entwerfen und Implementieren von Programmen müssen Datentypen und zugehörige Operationen verwendet bzw. sachgerecht genutzt werden. |

## Sprachen und Automaten

| GI-Empfehlungen<br><br>Schülerinnen und Schüler | Kompetenzmodell „Algorithmen“<br>Schülerinnen und Schüler | Erläuterungen |
|---|---|---|
| • geben Problemlösungen in einer Dokumentenbeschreibungssprache, Abfragesprache oder Programmiersprache an | • setzen gegebene Algorithmen in Programme um (C II)<br>• implementieren Programme mit einem Programmiersystem benutzungsfreundlich (D II) | Das Implementieren von Programmen erfolgt in einer Programmiersprache. |
| • interpretieren Fehlermeldungen bei der Arbeit mit Informatiksystemen und nutzen sie produktiv | • setzen gegebene Algorithmen in Programme um (C II)<br>• implementieren Programme mit einem Programmiersystem benutzungsfreundlich (D II)<br>• testen Programme anhand gegebener Eingaben auf ihre Funktionalität (D II) | Bei der Arbeit mit Programmiersystemen, insbesondere beim Implementieren und Testen von Programmen, können verschiedenartige Fehlermeldungen auftreten, die vom Benutzer interpretiert werden müssen, um produktiv weiterarbeiten zu können. |

## Informatik, Mensch und Gesellschaft

| GI-Empfehlungen<br><br>Schülerinnen und Schüler | Kompetenzmodell „Algorithmen“<br>Schülerinnen und Schüler | Erläuterungen |
|---|---|---|
| • kommentieren automatisierte Vorgänge und beurteilen deren Umsetzung | • reflektieren über den Lösungsweg, sowie über Vor- und Nachteile der Lösung (D III) | Bei der Reflexion über einen Lösungsweg bzw. über die Vor- und Nachteile einer Lösung werden programmiersprachlich umgesetzte Vorgänge beurteilt und kommentiert. |

## Modellieren und Implementieren

| GI-Empfehlungen<br><br>Schülerinnen und Schüler | Kompetenzmodell „Algorithmen“<br>Schülerinnen und Schüler | Erläuterungen |
|---|---|---|
| • analysieren Sachverhalte und erarbeiten angemessene Modelle | • analysieren die Funktionsweise und den Leistungsumfang gegebener Algorithmen (C II)<br>• prüfen Algorithmen mit gegebenen Beispielen mithilfe von Durchlauftabellen (Schreibtischtest) (C II)<br>• fertigen einen schriftlichen Entwurf für Programme an (D II)<br>• testen Programme anhand gegebener Eingaben auf ihre Funktionalität (D II) | Das Analysieren, Prüfen und Testen von Algorithmen bzw. Programmen entspricht einer Analyse eines Sachverhaltes. Beim Entwurf eines Programms wird ein angemessenes Modell zur Lösung eines informatischen Problems entwickelt. |

| GI-Empfehlungen<br><br>Schülerinnen und Schüler | Kompetenzmodell „Algorithmen“<br>Schülerinnen und Schüler | Erläuterungen |
|---|---|---|
| • verwenden bei der Implementierung die algorithmischen Grundbausteine | • setzen gegebene Algorithmen in Programme um (C II)<br>• implementieren Programme mit einem Programmiersystem benutzungsfreundlich (D II) | Beim Umsetzen von Algorithmen und beim Implementieren von Programmen werden stets algorithmische Grundbausteine verwendet. |
| • beeinflussen das Modellverhalten durch zielgerichtete Änderungen | • modifizieren und ergänzen Algorithmen bzw. Programme nach Vorgaben (C II)<br>• korrigieren gegebene fehlerhafte Algorithmen bzw. Programme (C III)<br>• verbessern Programme eigenständig (D III) | Das Modifizieren, Ergänzen, Korrigieren und Verbessern von Algorithmen bzw. Programmen impliziert zielgerichtete Änderungen am Modell der Problemlösung. |
| • beurteilen das Modell, die Implementierung und die verwendeten Werkzeuge kritisch | • reflektieren über den Lösungsweg (D II) | Eine Reflexion über den Lösungsweg beinhaltet eine Reflexion über die modellierte Problemlösung, deren Implementierung und evtl. auch über das verwendete Werkzeug. |

## Begründen und Bewerten

| GI-Empfehlungen<br><br>Schülerinnen und Schüler | Kompetenzmodell „Algorithmen“<br>Schülerinnen und Schüler | Erläuterungen |
|---|---|---|
| • stützen ihre Argumente auf erworbenes Fachwissen | • begründen anhand der Eigenschaften von Algorithmen, ob gegebene Handlungsabläufe Algorithmen sind (A II)<br>• analysieren die Funktionsweise und den Leistungsumfang gegebener Algorithmen (C II)<br>• reflektieren über den Lösungsweg sowie über Vor- und Nachteile der Lösung (D III) | Wenn im Rahmen der dargestellen Kompetenzen begründet, analysiert und reflektiert werden soll, so wird dies im Allgemeinen im Zusammenhang mit Fachwissen zum Thema „Algorithmen“ geschehen. |

## Strukturieren und Vernetzen

| GI-Empfehlungen<br><br>Schülerinnen und Schüler | Kompetenzmodell „Algorithmen“<br>Schülerinnen und Schüler | Erläuterungen |
|---|---|---|
| • planen Arbeitsabläufe und Handlungsfolgen | • fertigen einen schriftlichen Entwurf für Programme an (D II) | Mit einem Entwurf eines imperativen Algorithmus wird eine Handlungsfolge für einen Computer geplant. |
| • nutzen Analogien zwischen informatischen Inhalten oder Vorgehensweisen, um Neues mit Bekanntem zu verknüpfen | • modifizieren und ergänzen Algorithmen bzw. Programme nach Vorgaben (C II)<br>• fertigen einen schriftlichen Entwurf für Programme an (D II)<br>• implementieren Programme mit einem Programmiersystem benutzungsfreundlich (D II)<br>• testen Programme anhand gegebener Eingaben auf ihre Funktionalität (D II)<br>• reflektieren über den Lösungsweg (D II)<br>• verbessern Programme eigenständig (D III) | Beim Entwerfen, Implementieren, Testen, Modifizieren, Verbessern von und beim Reflektieren über Programme werden meist Probleme bearbeitet, die für die Schülerinnen und Schüler neu sind. Bei der Problemlösung wird im Allgemeinen auf bekannte Pläne und Schemas zurückgegriffen. |

## Kommunizieren und Kooperieren

| GI-Empfehlungen<br><br>Schülerinnen und Schüler | Kompetenzmodell „Algorithmen“<br>Schülerinnen und Schüler | Erläuterungen |
|---|---|---|
| • kommunizieren mündlich strukturiert über informatische Sachverhalte<br>• stellen informatische Sachverhalte unter Benutzung der Fachsprache schriftlich sachgerecht dar | • erklären die algorithmischen Grundbausteine wie Variablen, Wertzuweisungen, Verzweigungen und Wiederholungen und wenden diese Erklärungen an (B II)<br>• stellen die algorithmischen Grundbausteine in verschiedenen Darstellungsformen dar (B II)<br>• analysieren die Funktionsweise und den Leistungsumfang gegebener Algorithmen (C II)<br>• fertigen einen schriftlichen Entwurf für Programme an (D II)<br>• reflektieren über den Lösungsweg sowie über Vor- und Nachteile der Lösung (D III) | Lösungen zu den Aufgaben der angegebenen Kompetenzen können sowohl mündlich als auch schriftlich dargestellt werden, dabei sollte stets auf ein richtiges Verwenden der Fachsprache geachtet werden. |

## Darstellen und Interpretieren

| GI-Empfehlungen<br><br>Schülerinnen und Schüler | Kompetenzmodell „Algorithmen“<br>Schülerinnen und Schüler | Erläuterungen |
|---|---|---|
| • gestalten Diagramme und Grafiken, um informatische Sachverhalte zu beschreiben und mit anderen darüber zu kommunizieren | • stellen die algorithmischen Grundbausteine in verschiedenen Darstellungsformen dar (B II)<br>• fertigen einen schriftlichen Entwurf für Programme an (D II) | Bei einigen Darstellungsformen wie z. B. Struktogrammen und Programmablaufplänen werden grafische Elemente verwendet. Beim Anfertigen eines Entwurfs können Diagramme und Grafiken, insbesondere zur Veranschaulichung der Ausgabe eines Programms, hilfreich sein. |
| • interpretieren Diagramme, Grafiken sowie Ergebnisdaten | • stellen die algorithmischen Grundbausteine in verschiedenen Darstellungsformen dar und wechseln zwischen Darstellungsformen (B III)<br>• testen Programme anhand gegebener Eingaben auf ihre Funktionalität (D II) | Wenn zwischen Darstellungsformen gewechselt wird, so müssen evtl. auch grafisch dargestellte Algorithmen interpretiert werden. Wenn entschieden werden soll, ob ein Programm funktionert, müssen die Ergebnisdaten interpretiert werden. |

# D Musterlösungen zu den Aufgaben

Nachfolgend werden Musterlösungen zu den im Kapitel 6 angegebenen Aufgaben vorgestellt.

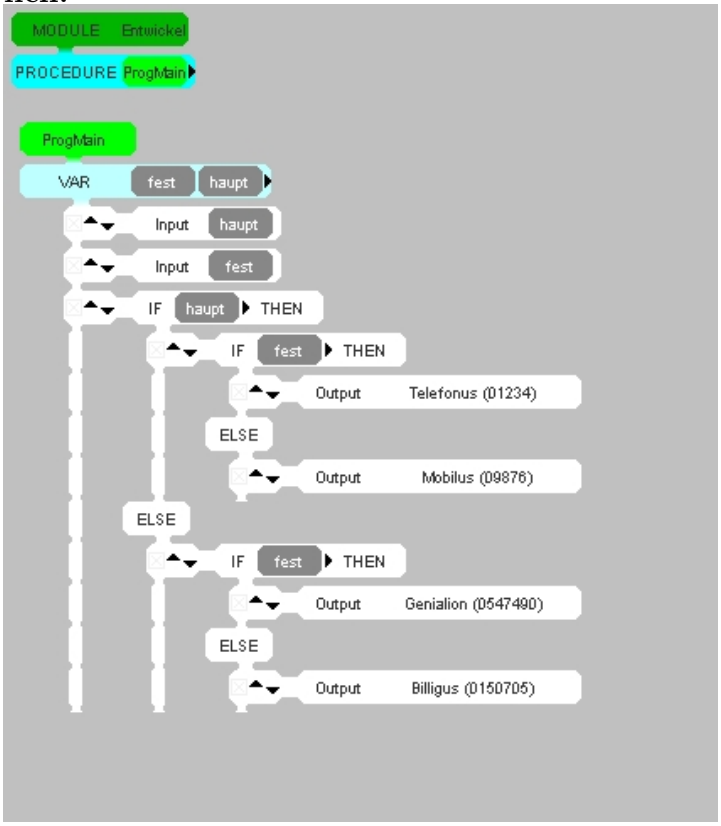
## Musterlösung zu der in Abbildung 21 dargestellten Testaufgabe zu Komponente A auf Stufe I

- **Eigenschaften der Handlungsvorschrift:**
  Ja, die Handlungsvorschrift erfüllt die Eigenschaft Allgemeinheit.
  Nein, die Handlungsvorschrift erfüllt nicht die Eigenschaft Eindeutigkeit.
- **Erklären Sie den Begriff Algorithmus.**
  Ein Algorithmus ist eine Verarbeitungsvorschrift, die aus einer endlichen Folge von eindeutig ausführbaren Anweisungen besteht, mit der man eine Vielzahl gleichartiger Aufgaben lösen kann.
- **Nennen Sie ein Beispiel für ein Problem, das mithilfe von Algorithmen lösbar ist.**
  Das Sortieren einer größeren Menge von Adressen nach der Postleitzahl ist ein Beispiel für ein Problem, das mithilfe von Algorithmen lösbar ist.

## Musterlösung zu der in Abbildung 22 dargestellten Testaufgabe zu Komponente D auf Stufe II

- **Fertigen Sie einen schriftlichen Entwurf für ein Programm an, bei dem der Benutzer zwischen Haupt- und Nebenzeit sowie zwischen Festnetz- und Mobilnummer auswählen kann. Anschließend soll der günstigste der vier gegebenen Anbieter mit Nummer ausgegeben werden.**
  Der Benutzer wird aufgefordert einzugeben, ob er zur Hauptzeit oder zur Nebenzeit telefonieren möchte. Das Ergebnis wird in die Booleanvariable hauptzeit gespeichert.
  Der Benutzer wird aufgefordert einzugeben, ob er mit einer Festnetz- oder eine Mobilfunknummer telefonieren möchte. Das Ergebnis wird in die Booleanvariable festnetz gespeichert.
  Wenn hautpzeit=true, dann wird folgendes getan:
  Wenn festnetz=true, dann wird „Telefonus (01234)“ ausgegeben.
  Wenn festnetz=false, dann wird „Mobilus (09876)“ ausgegeben.
  Wenn hautpzeit=false, dann wird folgendes getan:
  Wenn festnetz=true, dann wird „Genialion (0547490)“ ausgegeben.
  Wenn festnetz=false, dann wird „Billigus (0150705)“ ausgegeben.

- **Implementieren sie das entworfene Programm benutzungsfreundlich.**




- **Testen Sie Ihr Programm mit folgenden Eingaben.**
  **A) ein Telefonat zu einer Festnetznummer in der Nebenzeit**
  **B) ein Telefonat zu einer Mobilfunknummer in der Hauptzeit**
  **Notieren Sie alle Eingaben und Ausgaben Ihres Programms bei beiden Testfällen.**
  zu A) Genialion (0547490)
  zu B) Mobilus (09876)

- **Reflektieren Sie über Ihren Lösungsweg. Welche Schwierigkeiten traten beim Entwurf und bei der Implementierung auf? Wie haben Sie diese Probleme gelöst?**
  Für die Eingabe des Benutzers nutzte ich zuerst Integervariablen, bei denen der Benutzer eine Zahl eingeben musste (z. B. 1=Hauptzeit; 2=Nebenzeit). Den eingegebenen Werten entsprechend wurde dann der günstigste Anbieter mit Nummer ausgegeben. Später kam ich auf die Idee, die Eingabe direkt in Variablen vom Typ Boolean zu speichern. Dadurch wurde das Programm noch benutzungsfreundlicher.

## Musterlösung zu der in Abbildung 23 dargestellten Beispielaufgabe zu Komponente B auf Stufe III

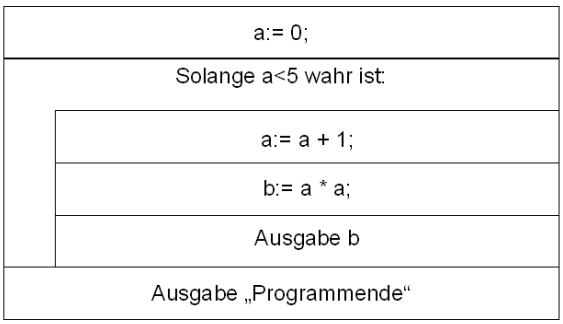
- **Nennen Sie zwei Ihnen bekannte Parameterarten und erklären Sie die Funktionsweise der Parameterübergabe.**

  **Wertparametern** kann ein Ausdruck des entsprechenden Datentyps als Startwert übergeben werden. Wertparameter verhalten sich innerhalb einer Prozedur wie lokale Variablen. Sie haben einen bestimmten Datentyp und einen Wert, der innerhalb der Prozedur verändert werden kann. Der Wert eines Wertparameters wird nach der Abarbeitung der Prozedur nicht an die aufrufende Stelle zurückgegeben.

  **Referenzparametern** wird eine Variable übergeben. Der Wert des Referenzparameters kann dann in der Prozedur verändert werden. Nach Abarbeitung der Prozedur wird der Variablen der letzte Wert des Referenzparameters zugewiesen. Somit können Ergebnisse an die aufrufende Stelle zurückgegeben werden.

- **Überführen Sie die gegebenen Beispiele in eine andere Darstellungsform.**

  Struktogramm:




  Pseudocode:

  - Fordere den Benutzer mit „Gib dein Alter ein!“ auf, eine Zahl einzugeben. Speichere die eingegebene Zahl in die Variable alter.
  - Wenn die Bedingung (alter >= 18) wahr ist, gib den Text „Willkommen“ aus, ansonsten gib den Text „Zugriff verweigert“ aus.

- **Geben Sie für zwei Datentypen jeweils zwei Werte und zwei Operationen an.**

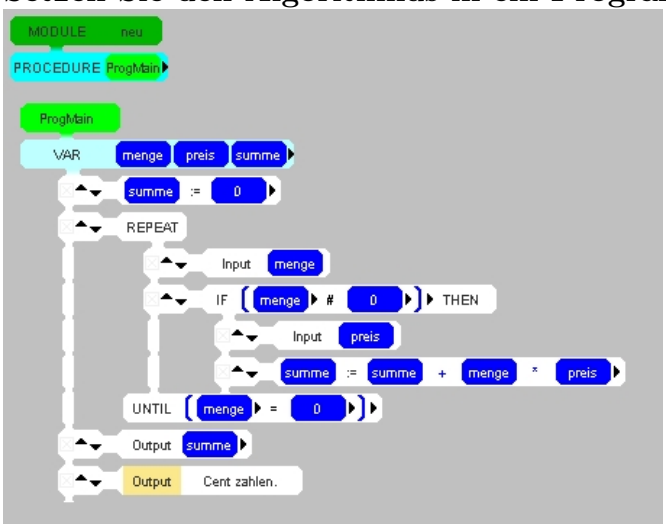
| Datentyp | Werte | Operationen |
|---|---|---|
| Integer | 2,7 | +, - |
| Boolean | true, false | and, or |

## Musterlösung zu der in Abbildung 24 dargestellten Beispielaufgabe zu Komponente C auf Stufe II

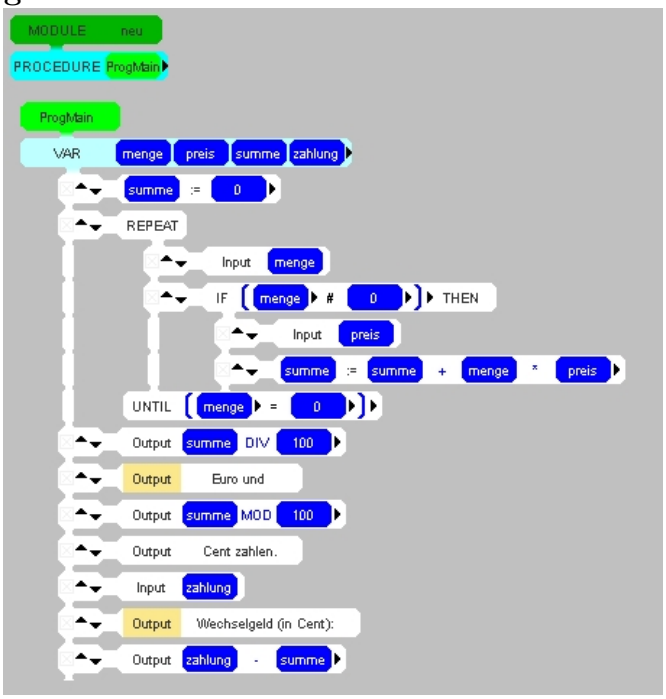
- **Lesen Sie den Algorithmus und geben Sie seine Funktionsweise mit eigenen Worten wieder.**
  Der Algorithmus liest vom Benutzer so lange die Anzahl der Artikel ein, bis dieser die Anzahl null eingibt. Wenn die Artikelanzahl größer als null ist, wird auch der Preis in Cent abgefragt. Anschließend wird jeweils das Produkt aus Anzahl und Preis zur aktuellen Gesamtsumme addiert. Nach der Eingabe der Zahl null bei der Artikelanzahl wird erst der Text „Der Kunde muss “, dann die Gesamtsumme und am Ende der Text „ Cent zahlen.“ ausgegeben.
- **Frau Meier kauft: zwei Becher Joghurt für je 50 Cent, vier Kiwis für je 25 Cent und eine Mango für 99 Cent.**
  **Überprüfen Sie den gegebenen Algorithmus für den Einkauf von Frau Meier mithilfe einer Durchlauftabelle.**

| menge | preis | summe |
|---|---|---|
| 2 | 50 | 100 |
| 4 | 25 | 200 |
| 1 | 99 | 299 |
| 0 | 99 | 299 |

- **Setzen Sie den Algorithmus in ein Programm um.**

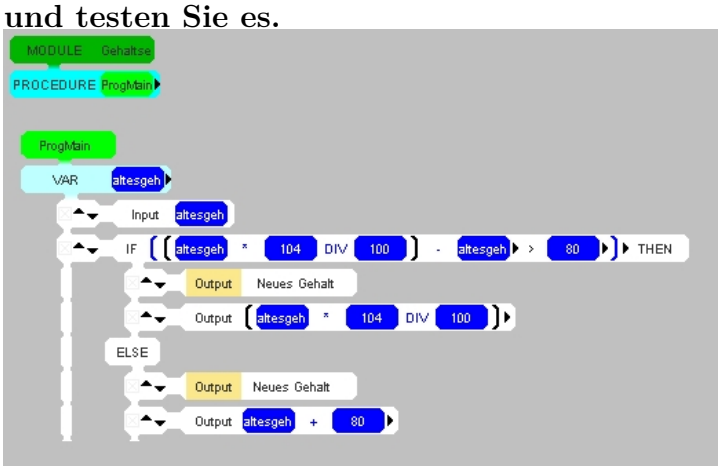



- **Verändern Sie das Programm so, dass die Summe in Euro und Cent ausgegeben wird.**
  **Außerdem soll nach der Berechnung der Summe noch eingegeben werden, mit welchem Betrag der Kunde zahlt. Anschließend soll ausgegeben werden, wie viel Wechselgeld der Kassierer zurückzugeben hat.**

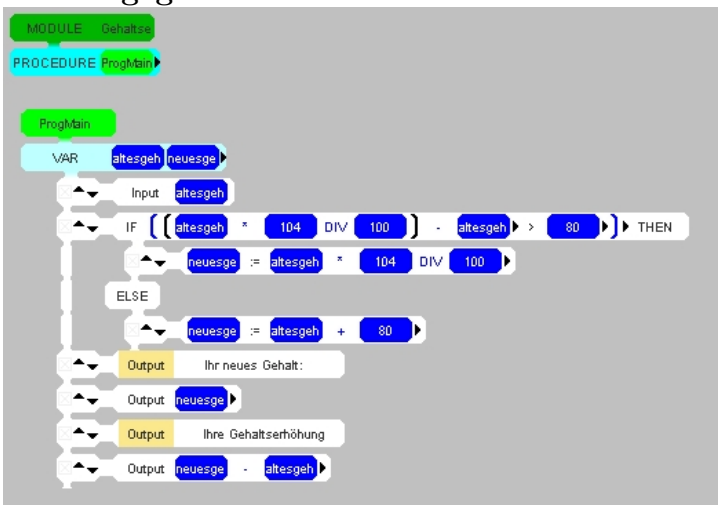


## Musterlösung zu der in Abbildung 25 dargestellten Unterrichtsaufgabe zu Komponente C auf Stufe I

- **Welche Ausgaben liefert der Algorithmus für das Gehalt 1000 € und für das Gehalt 10000 €?**
  Gehalt 1000 → Neues Gehalt: 1080
  Gehalt 10000 → Neues Gehalt:10400
- **Lesen Sie den Algorithmus, übertragen Sie ihn in ein Programm und testen Sie es.**




- **Finden Sie durch mehrfaches Testen heraus, ab welcher Gehaltsgrenze mehr als 80 € Gehaltserhöhung fällig ist.**
  Ab der Eingabe 2025 sind mehr als 80€ Gehaltserhöhung fällig.
- **Ergänzen Sie das Programm so, dass jeweils die Gehaltserhöhung mit ausgegeben wird.**



## Musterlösung zu der in Abbildung 27 dargestellten Unterrichtsaufgabe zu Komponente C auf Stufe III

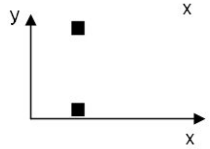
- **Informieren Sie sich im Internet über die Funktionsweise einer Binäruhr.**
  Eine binäre Uhr stellt die aktuelle Uhrzeit binär dar. Dabei wird jede Dezimalziffer der auszugebenden Zeit einzeln in das Dualsystem umgerechnet und angezeigt. Diese Anzeige der Dualzahlen ist häufig durch Leuchtdioden (LEDs) realisiert. (Quelle http://de.wikipedia.org/wiki/Binäre_Uhr – Zugriffsdatum 20.10.2008)
- **Analysieren Sie den Algorithmus mithilfe der angegebenen Durchlauftabelle.**

| s1 | s2 | Prozeduraufruf |
|---|---|---|
| 0 | 0 | ausg(0,200); ausg (0,300) |
| 0 | 1 | ausg(0,200); ausg (1,300) |
| 0 | 2 | ausg(0,200); ausg (2,300) |
| 0 | 3 | ausg(0,200); ausg (3,300) |
| 0 | 4 | ausg(0,200); ausg (4,300) |
| 0 | 5 | ausg(0,200); ausg (5,300) |
| 0 | 6 | ausg(0,200); ausg (6,300) |
| 0 | 7 | ausg(0,200); ausg (7,300) |
| 0 | 8 | ausg(0,200); ausg (8,300) |
| 0 | 9 | ausg(0,200); ausg (9,300) |
| 1 | 0 | ausg(1,200); ausg (0,300) |
| 1 | 1 | ausg(1,200); ausg (1,300) |
| 1 | 2 | ausg(1,200); ausg (2,300) |
| 1 | 3 | ausg(1,200); ausg (3,300) |

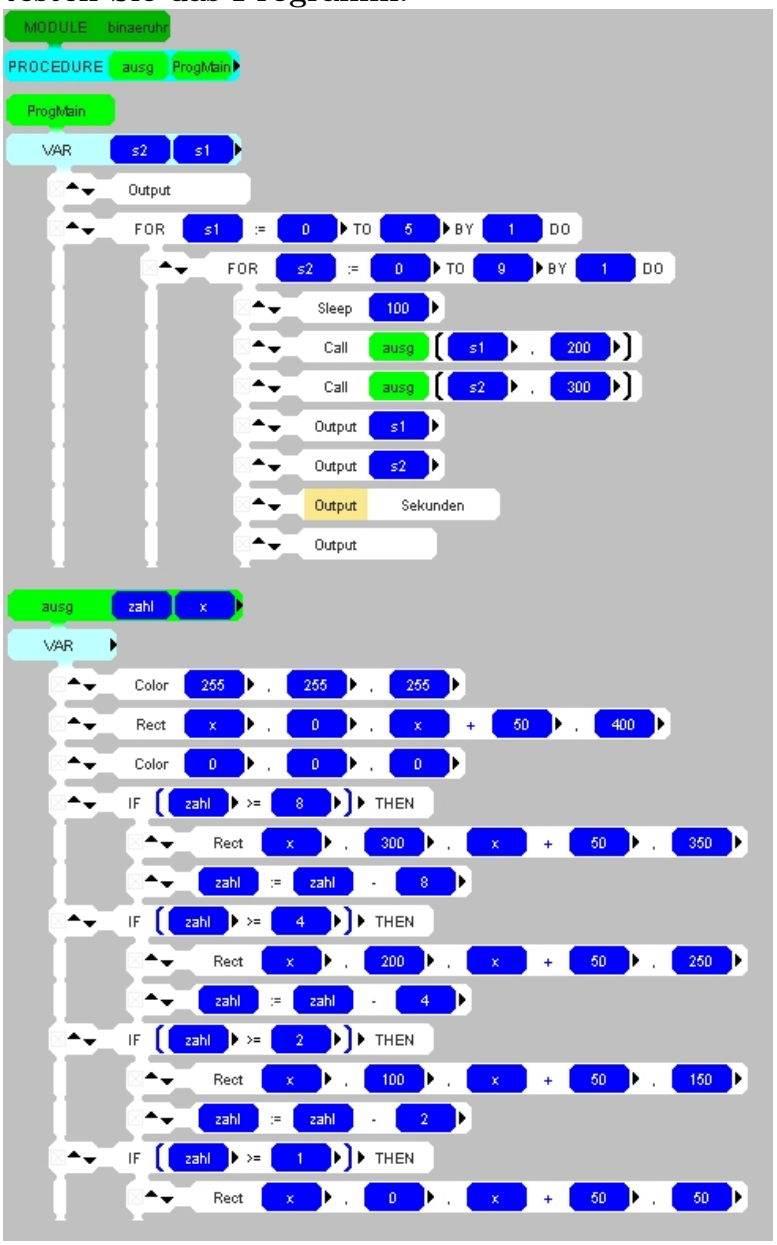
- **Analysieren Sie an selbst gewählten Beispielen zwei unterschiedliche Aufrufe der Prozedur ausg. Zeichnen Sie auch die Ausgabe rechts neben die Durchlauftabellen.**

| zahl | x |
|---|---|
| 7 | 300 |
| 3 | |
| 1 | |
| | |

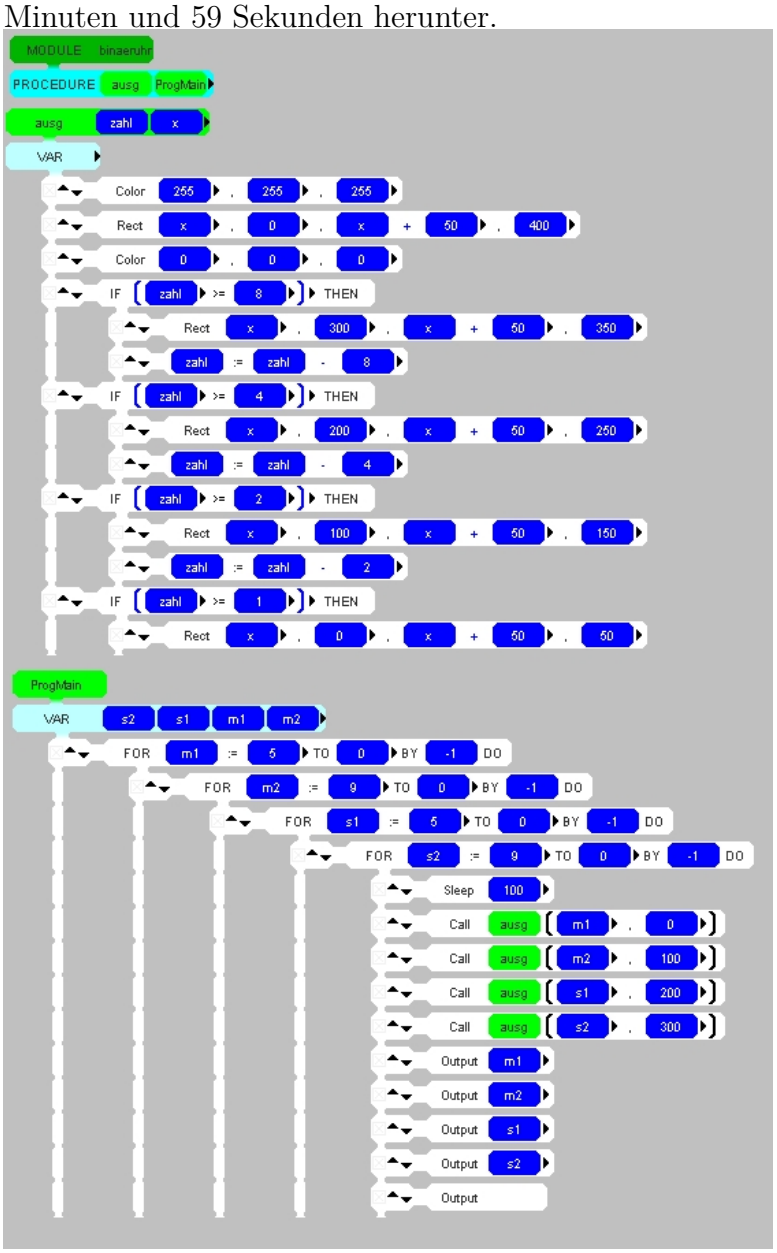

| zahl | x |
|---|---|
| 9 | 200 |
| 1 | |
| | |
| | |



- **Übertragen Sie den gegebenen Algorithmus in ein Programm und testen Sie das Programm.**



- **Verändern Sie das Programm so, dass es zusätzlich zu den Sekunden auch Minuten ausgeben kann.**
  **Verbessern oder erweitern Sie das Programm in irgendeiner Weise.**
  Das folgende Programm gibt Minuten und Sekunden als Zahlenwerte und in der binären Darstellung aus. Es zählt dabei einen Countdown von 59 Minuten und 59 Sekunden herunter.



# E Fragebogen zur Hauptuntersuchung

Der nachfolgend abgebildete Fragebogen wurde am Ende der im Kapitel 8 dargestellten Hauptuntersuchung eingesetzt, um die Erfahrungen der Lehrpersonen zu den bereitgestellten Materialien und der visuellen Programmiersprache Puck sowie allgemeine Einschätzungen zu kompetenzorientiertem Informatikunterricht zu erheben.

Dipl.-Inf. Lutz Kohl
Friedrich-Schiller-Universität Jena
Casio-Stiftungsprofessur für Didaktik der Informatik / Mathematik
Ernst-Abbe-Platz 2
D-07743 Jena
Tel. +49 (0) 3641 / 946497
Fax +49 (0) 3641 / 946252
E-Mail lutz.kohl@uni-jena.de
http://www.uni-jena.de/casio.html

Sehr geehrte Informatiklehrerin, sehr geehrter Informatiklehrer,

es freut uns sehr, dass Sie das Kompetenzmodell, die Aufgaben für den Unterricht, den Kompetenztest und das visuelle Programmiersystem Puck in Ihrem Unterricht eingesetzt haben und nun bereit sind, uns einige Fragen zu Ihren Erfahrungen damit zu beantworten. Durch Ihr Engagement wird eine Einschätzung und Verbesserung von kompetenzorientiertem Informatikunterricht sowie eine Evaluation der genannten Materialien möglich. Herzlichen Dank!

Die Beantwortung der Fragen erfolgt selbstverständlich freiwillig und anonym. Bei Problemen wenden Sie sich bitte an die oben angegebene Adresse.

**Inhalt des Fragebogens:**

Im ersten Teil erfragen wir einige Daten zu Ihrer Person und zu Ihrem Unterricht.

Der zweite Teil des Fragebogens gliedert sich inhaltlich in die Themen: Kompetenzmodell; Aufgaben für den Unterricht; die visuelle Programmiersprache Puck; der Kompetenztest; allgemeine Fragen zu kompetenzorientiertem Informatikunterricht

Am Ende eines jeden Themenbereichs besteht jeweils die Möglichkeit, Kritik, Lob, Anregungen und Verbesserungsvorschläge anzugeben.

Bitte beantworten Sie alle Fragen wahrheitsgemäß. Lassen Sie möglichst keine Frage aus. Falls die vorgegebenen Antwortmöglichkeiten Ihre Meinung nicht genau treffen, wählen Sie bitte diejenige Antwort aus, die Ihrer Auffassung am nächsten kommt. Die Beantwortung des Fragebogens dauert ca. 20 Minuten.

Nochmals herzlichen Dank für Ihre Unterstützung.

Prof. Dr. Michael Fothe,

Dipl.-Inf. Lutz Kohl

Bitte klicken Sie auf "weiter", um die Befragung zu starten!

---

| Alter | |
|---|---|
| ☐ | bis 30 |
| ☐ | 31-40 |
| ☐ | 41-50 |
| ☐ | 51-60 |
| ☐ | ab 61 |

| Geschlecht |
|---|
| ☐ männlich |
| ☐ weiblich |

| Besitzen Sie eine Lehrbefähigung im Fach Informatik? |
|---|
| ☐ ja |
| ☐ nein |

| Seit wie vielen Jahre sind Sie im Schuldienst? |
|---|
| ☐ bis 5 |
| ☐ 6-15 |
| ☐ 16-25 |
| ☐ 26-35 |
| ☐ ab 36 |

| Seit wie vielen Jahren unterrichten Sie das Fach Informatik? |
|---|
| ☐ bis 5 |
| ☐ 6-10 |
| ☐ 11-15 |
| ☐ ab 16 |

| In welchem Land / Staat unterrichten Sie? | | |
|---|---|---|
| ☐ Baden-Württemberg | ☐ Bayern | ☐ Berlin |
| ☐ Brandenburg | ☐ Hessen | ☐ Mecklenburg-Vorpommern |
| ☐ Niedersachsen | ☐ Nordrhein-Westfalen | ☐ Rheinland-Pfalz |
| ☐ Sachsen-Anhalt | ☐ Schleswig-Holstein | ☐ Thüringen |
| ☐ Österreich | ☐ Schweiz | |
| ☐ Sonstige | | |

| In welcher Schulform unterrichten Sie? |
|---|
| ☐ Gesamtschule |
| ☐ Gymnasium |
| ☐ Hauptschule |
| ☐ Oberschule |
| ☐ Realschule |
| ☐ Regelschule |
| ☐ Andere |

**Sind Sie derzeit im Referendariat?**

- ☐ ja
- ☐ nein

**In welcher Klassenstufe haben Sie die Materialien eingesetzt?**
**Falls Sie in mehreren Klassenstufen mit den Materialien unterrichtet haben, so wählen Sie bitte für die Beantwortung dieser und aller folgenden Fragen eine der Klassenstufen aus.**

- ☐ 8. Klasse
- ☐ 9. Klasse
- ☐ 10. Klasse
- ☐ Sonstige

**Welchen Schulzweig besuchten die Schülerinnen und Schüler, bei denen Sie die Materialien eingesetzt haben?**

- ☐ Gymnasium
- ☐ Hauptschule
- ☐ Realschule
- ☐ Sonstiges

**Wurde der nach den bereitgestellten Materialien erteilte Unterricht mit einer Note bewertet?**

- ☐ ja
- ☐ nein

**Wie viele Unterrichtsstunden (je 45 Minuten) haben Sie mit den bereitgestellten Materialien unterrichtet?**

**Haben Sie den Kompetenztest im Unterricht eingesetzt?**

- ☐ ja
- ☐ nein

---

**Wie oft haben Sie Ihren Unterricht am gegebenen Kompetenzmodell ausgerichtet?**

- ☐ nie
- ☐ selten
- ☐ gelegentlich
- ☐ oft
- ☐ immer

**Haben Sie den Schülerinnen und Schülern das Kompetenzmodell vorgestellt?**

- ○ Nein, das Kompetenzmodell wurde nicht vorgestellt.
- ○ Ja, das Kompetenzmodell wurde kurz vorgestellt.
- ○ Ja, das Kompetenzmodell wurde ausführlich erläutert.

**Ich habe das Kompetenzmodell eingesetzt zur Planung …**

- ☐ ... der Stoffverteilung im gesamten Halbjahr.
- ☐ ... von Unterrichtssequenzen (z. B. Monatsplanung).
- ☐ ... von einzelnen Unterrichtsstunden.
- ☐ ... einer jeden Unterrichtsstunde.
- ☐ ... von Leistungsüberprüfungen.

**Ich habe das Kompetenzmodell eingesetzt zur Einschätzung des Unterrichts am Ende …**

- ☐ ... vom Halbjahr.
- ☐ ... von Unterrichtssequenzen.
- ☐ ... von einzelnen Unterrichtsstunden.
- ☐ ... einer jeden Unterrichtsstunde.

**Wie schätzen Sie die Einteilung des gegebenen Kompetenzmodells ein?**

- ☐ Die Einteilung in verschiedene Stufen ist sinnvoll.
- ☐ Die Einteilung in verschiedene Komponenten ist sinnvoll.
- ☐ Die Einteilung ist ungünstig, weil ...

---

**Warum haben Sie den Unterricht nie bzw. selten am Kompetenzmodell ausgerichtet?**

**Das Kompetenzmodell ist ...**

- ○ sehr übersichtlich.
- ○ übersichtlich.
- ○ unübersichtlich.
- ○ sehr unübersichtlich.

**Bitte geben Sie Kritik, Lob, Anmerkungen und Verbesserungsvorschläge für das Kompetenzmodell an.**

| **Aufgaben für den Unterricht:** | **Eine Beispielaufgabe (Billard)** |
|---|---|
| **Die bereitgestellten Aufgaben für den Unterricht bestehen aus den Beispielaufgaben zur Veranschaulichung der geforderten Kompetenzen (Arbeitsmaterial Seiten 8 - 17) und der Aufgabensammlung (Arbeitsmaterial Seiten 18 - 69). Jede Aufgabe ist jeweils einer Komponente und einer Stufe des Kompetenzmodells zugeordnet.** | Entwerfen Sie schriftlich ein Programm, bei dem der Verlauf einer Kugel auf einem Billardtisch Punkt für Punkt aufgezeichnet wird …<br>Setzen Sie den Entwurf in ein Programm um.<br>Testen Sie das Programm.<br>Verbessern oder erweitern Sie das Programm in irgendeiner Weise. |

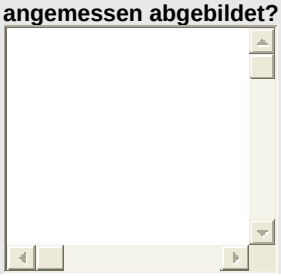
**Einsatz der Aufgaben für den Unterricht**

| | nie | selten | gelegentlich | oft | immer |
|---|---|---|---|---|---|
| Wie oft haben Sie die bereitgestellten Aufgaben bei der Unterrichtsvorbereitung eingesetzt? | ☐ | ☐ | ☐ | ☐ | ☐ |
| Wie oft wurden in Ihrem Unterricht Aufgaben unterschiedlicher Kompetenzstufen zur Differenzierung eingesetzt? | ☐ | ☐ | ☐ | ☐ | ☐ |

**Einschätzung der Aufgaben für den Unterricht**

| | ich stimme überhaupt nicht zu | ich stimme nicht zu | unentschlossen | ich stimme etwas zu | ich stimme voll zu |
|---|---|---|---|---|---|
| Die Einordnung der Aufgaben für den Unterricht in die Komponenten und Stufen des Kompetenzmodells war für mich eine wichtige Hilfe. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Die geforderten Kompetenzen wurden durch die Aufgaben für den Unterricht angemessen abgebildet. | ☐ | ☐ | ☐ | ☐ | ☐ |

**Welche Kompetenzen wurden durch die Aufgaben für den Unterricht nicht angemessen abgebildet?**



**Bitte geben Sie Kritik, Lob, Anmerkungen und Verbesserungsvorschläge für die bereitgestellten Aufgaben an.**
(z. B. konkrete Verbesserungsvorschläge für einzelne Aufgaben)

---

**Das Puck-System**

Puck ist eine visuelle Programmiersprache für den Einsatz in der Schule.
Das Puck-System kann unter www.ipuck.de heruntergeladen werden.

**Bitte geben Sie an, inwieweit Sie den Aussagen zur visuellen Programmiersprache Puck zustimmen.**

| | ich stimme überhaupt nicht zu | ich stimme nicht zu | unentschlossen | ich stimme etwas zu | ich stimme voll zu |
|---|---|---|---|---|---|
| Das Verringern der Syntaxfehler durch das Puck-System war günstig für den Anfangsunterricht. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Für viele Schülerinnen und Schüler erwies sich die Arbeit mit dem Puck-System als schwierig. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Das Puck-System motiviert die Schülerinnen und Schüler, sich in den Informatikunterricht aktiv einzubringen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Es gibt zu wenig geeignete Unterrichtsmaterialien für den Einsatz des Puck-Systems. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Der Einsatz des Puck-Systems motivierte mich für meine Unterrichtsvorbereitung und Unterrichtsdurchführung. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Der Einarbeitungsaufwand in das Puck-System ist für Informatiklehrerinnen und -lehrer sehr hoch. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Die Schülerinnen und Schüler sind durch den Unterricht mit dem Puck-System auf die Arbeit mit einer textuellen Programmiersprache gut vorbereitet worden. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Durch die Arbeit mit dem Puck-System hatte ich mehr Zeit für die individuelle Förderung einzelner Schülerinnen und Schüler. | ☐ | ☐ | ☐ | ☐ | ☐ |

| | | | | | |
|---|---|---|---|---|---|
| Durch die Arbeit mit dem Puck-System wird den Schülerinnen und Schülern ein falsches Bild von der Programmierung vermittelt. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Auch leistungsschwächere Schülerinnen und Schüler konnten durch die Arbeit mit dem Puck-System Erfolge erzielen. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Die Funktionalität von Puck reicht für eine Einführung in die Programmierung aus. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich möchte das Puck-System in den nächsten Schuljahren wieder einsetzen. | ☐ | ☐ | ☐ | ☐ | ☐ |

**Ich halte den Einsatz des Puck-Systems für sinnvoll ab Klassenstufe ...**

- ☐ 5
- ☐ 6
- ☐ 7
- ☐ 8
- ☐ 9
- ☐ 10
- ☐ Andere

**Bitte geben Sie Kritik, Lob, Anmerkungen und Verbesserungsvorschläge für die visuelle Programmiersprache Puck an.**

---

**Wer hat festgelegt, welche Stufen die Schülerinnen und Schüler im Kompetenztest bearbeiten?**

- ☐ Ich habe eine Stufe festgelegt, die von der ganzen Klasse einheitlich zu bearbeiten war.
- ☐ Ich habe individuell für jede Schülerin und jeden Schüler (oder für Schülergruppen) festgelegt, welche Stufe zu bearbeiten war.
- ☐ Jede Schülerin und jeder Schüler hat für sich festgelegt, welche Stufe zu bearbeiten war.
- ☐ Jede Schülerin und jeder Schüler hat für sich für jede der Komponenten die Stufe festgelegt, die zu bearbeiten war.
- ☐ Sonstiges

**Wie viele Schülerinnen und Schüler haben am Kompetenztest teilgenommen?**

**Wie viele Schülerinnnen und Schüler haben Stufe III erreicht?**

**Wie viele Schülerinnen und Schüler haben Stufe II, aber nicht Stufe III erreicht?**

**Wie viele Schülerinnen und Schüler haben Stufe I, aber nicht Stufe II oder III erreicht?**

**Wurde der Kompetenztest benotet?**

- ☐ ja
- ☐ nein

---

**Welche Stufe des Kompetenztests bearbeiteten die Schülerinnen und Schüler?**

- ☐ Stufe I
- ☐ Stufe II
- ☐ Stufe III
- ☐ Sonstiges

**Nach welchen Kriterien wurden die Noten beim Kompetenztest vergeben?**

**Wie schätzen Sie das vorgegebene Kriterium zum Erreichen der Stufen im Kompetenztest ein?**
**75 % der in einer Stufe möglichen Punkte mussten erreicht werden**

- ☐ 75 % ist zu gering.
- ☐ 75 % ist zu hoch.
- ☐ Es sollten in jeder der Komponenten eine minimale Punktzahl erreicht werden.
- ☐ Das Kriterium ist genau richtig.
- ☐ Sonstiges

**Wie viel Zeit hatten die die Schülerinnen und Schüler zur Bearbeitung des Kompetenztests?**

- ☐ 90 Minuten
- ☐ Bitte Zeit in Minuten angeben

**Abbildung der Kompetenzen durch den Kompetenztest**

| | ich stimme überhaupt nicht zu | ich stimme nicht zu | unentschlossen | ich stimme etwas zu | ich stimme voll zu |
|---|---|---|---|---|---|
| Die im Kompetenzmodell geforderten Kompetenzen wurden durch den bereit gestellten Kompetenztest angemessen abgebildet. | ☐ | ☐ | ☐ | ☐ | ☐ |

**Warum haben Sie den Kompetenztest nicht im Unterricht eingesetzt?**

**Bitte geben Sie Kritik, Lob, Anmerkungen und Verbesserungsvorschläge für den Kompetenztest an.**

Am Ende des Fragebogens sollen Sie noch einige allgemeine Fragen zu kompetenzorientiertem Unterricht beantworten.

**Die Einteilung von Kompetenzmodellen in einzelne Komponenten ist ...**

- ☐ ... sinnvoll für die Strukturierung von Kompetenzmodellen.
- ☐ ... sinnvoll für die Vorbereitung auf Kompetenztests.
- ☐ ... sinnvoll für die differenzierte Beurteilung von Leistungen.
- ☐ ... nicht nützlich
- ☐ sonstiges

| Die Einteilung von Kompetenzmodellen in einzelne Stufen ist ... |
|---|
| ☐ ... sinnvoll für die Strukturierung von Kompetenzmodellen. |
| ☐ ... sinnvoll für die Vorbereitung auf Kompetenztests. |
| ☐ ... sinnvoll für Differenzierung im Unterricht. |
| ☐ ... nicht nützlich. |
| ☐ sonstiges |

| Kompetenztests sind hilfreich bei ... |
|---|
| ☐ ... der Einschätzung der entwickelten Kompetenzen der Schülerinnen und Schüler. |
| ☐ ... der Einschätzung der Wirksamkeit des pädagogischen Handelns einer Lehrperson |
| ☐ sonstiges |

| Kompetenzorientierter Informatikunterricht | | | | | |
|---|---|---|---|---|---|
| | ich stimme überhaupt nicht zu | ich stimme nicht zu | unentschlossen | ich stimme etwas zu | ich stimme voll zu |
| Ich wünsche mir eine Fortbildung zum Einsatz von Kompetenzmodellen im Informatikunterricht. | ☐ | ☐ | ☐ | ☐ | ☐ |
| Ich würde Kompetenzmodelle auch in anderen Gebieten der Informatik einsetzen. | ☐ | ☐ | ☐ | ☐ | ☐ |

Vielen Dank für Ihre Teilnahme!

Um zum Fragebogen zurückzukehren, klicken Sie bitte auf "*zurück*".

Um die Befragung endgültig abzuschließen, klicken Sie bitte auf "*Abschließen, Antworten speichern*". Ihre Antworten werden dann in der Datenbank gespeichert.

# F Ergebnisse des Fragebogens der Hauptuntersuchung

Nachfolgend werden die Antworten der Lehrkräfte zum im Anhang E dargestellten Fragebogen der Hauptuntersuchung vorgestellt. Bei geschlossenen Fragen wird jeweils angegeben, wie häufig die vorgegebenen Antwortmöglichkeiten gewählt wurden. Bei offenen Fragen werden die Antworten der Lehrpersonen vollständig wiedergegeben (Bezeichnung der Lehrpersonen mit L1 bis L40). Die Ergebnisse der Befragung wurden im Kapitel 8 vorgestellt.

## Fragen zur Lehrkraft

| Alter | bis 30 | 31-40 | 41-50 | 51-60 | ab 61 |
|---|---|---|---|---|---|
| Häufigkeit (N=40) | 1 | 6 | 16 | 16 | 1 |

| Geschlecht | männlich | weiblich |
|---|---|---|
| Häufigkeit (N=40) | 31 | 9 |

| Besitzen Sie eine Lehrbefähigung im Fach Informatik? | ja | nein |
|---|---|---|
| Häufigkeit (N=40) | 27 | 13 |

| Seit wie vielen Jahre sind Sie im Schuldienst? | bis 5 | 6-15 | 16-25 | 26-35 | ab 36 |
|---|---|---|---|---|---|
| Häufigkeit (N=40) | 2 | 12 | 8 | 17 | 1 |

| Seit wie vielen Jahren unterrichten Sie das Fach Informatik? | bis 5 | 6-10 | 11-15 | ab 16 |
|---|---|---|---|---|
| Häufigkeit (N=40) | 3 | 14 | 8 | 15 |

| In welchem Land/Staat unterrichten Sie? | Häufigkeit (N=40) |
|---|---|
| Baden-Würtemberg | 1 |
| Bayern | 2 |
| Berlin | 5 |
| Brandenburg | 1 |
| Hessen | 11 |
| Niedersachsen | 4 |
| Nordrhein-Westfalen | 2 |
| Rheinland-Pfalz | 1 |
| Sachsen-Anhalt | 3 |
| Schleswig-Holstein | 1 |
| Thüringen | 8 |
| Schweiz | 1 |

| In welcher Schulform unterrichten Sie? | Gymnasium | Gesamtschule | Real- oder Hauptschule |
|---|---|---|---|
| Häufigkeit (N=40) | 30 | 5 | 5 |

| Sind Sie derzeit im Referendariat? | Ja | Nein | keine Angabe |
|---|---|---|---|
| Häufigkeit (N=40) | 1 | 1 | 38 |

## Fragen zum Unterricht

| In welcher Klassenstufe haben Sie die Materialien eingesetzt? | 8. Klasse | 9. Klasse | 10. Klasse | andere Jahrgangsstufe |
|---|---|---|---|---|
| Häufigkeit (N=39) | 10 | 18 | 9 | 2 |

| Welchen Schulzweig besuchten die Schülerinnen und Schüler, bei denen Sie die Materialien eingesetzt haben? | Gymnasium | Realschulzweig | Hauptschulzweig |
|---|---|---|---|
| Häufigkeit (N=39) | 28 | 8 | 3 |

| Wurde der nach den bereitgestellten Materialien erteilte Unterricht mit einer Note bewertet? | ja | nein |
|---|---|---|
| Häufigkeit (N=40) | 28 | 12 |

| Wie viele Unterrichtsstunden (je 45 Minuten) haben Sie mit den bereitgestellten Materialien unterrichtet? | 2 - 9 Stunden | 10 - 19 Stunden | 20 - 29 Stunden | mehr als 29 Stunden |
|---|---|---|---|---|
| Häufigkeit (N=40) | 4 | 20 | 9 | 7 |

| Haben Sie den Kompetenztest im Unterricht eingesetzt? | Ja | Nein |
|---|---|---|
| Häufigkeit (N=40) | 23 | 17 |

## Fragen zum Kompetenzmodell

| Wie oft haben Sie Ihren Unterricht am gegebenen Kompetenzmodell ausgerichtet? | nie | selten | gelegentlich | oft | immer |
|---|---|---|---|---|---|
| Häufigkeit (N=40) | 0 | 4 | 7 | 24 | 5 |

| Haben Sie den Schülerinnen und Schülern das Kompetenzmodell vorgestellt? | Häufigkeit (N=39) |
|---|---|
| Nein, das Kompetenzmodell wurde nicht vorgestellt. | 18 |
| Ja, das Kompetenzmodell wurde kurz vorgestellt. | 19 |
| Ja, das Kompetenzmodell wurde ausführlich erläutert. | 2 |

| Ich habe das Kompetenzmodell eingesetzt zur Planung ... | jeweilige Anzahl der Lehrerinnnen und Lehrer, die diese Option wählten (jeweils N=40) |
|---|---|
| der Stoffverteilung im gesamten Halbjahr. | 14 |
| von Unterrichtssequenzen (z. B. Monatsplanung). | 20 |
| von einzelnen Unterrichtsstunden | 20 |
| einer jeden Unterrichtsstunde | 3 |
| von Leistungsüberprüfungen. | 4 |

| Ich habe das Kompetenzmodell eingesetzt zur Einschätzung des Unterrichts am Ende ... | jeweilige Anzahl der Lehrerinnnen und Lehrer, die diese Option wählten (jeweils N=40) |
|---|---|
| vom Halbjahr. | 11 |
| von Unterrichtssequenzen. | 17 |
| von einzelnen Unterrichtsstunden | 10 |
| einer jeden Unterrichtsstunde. | 0 |

| Wie schätzen Sie die Einteilung des gegebenen Kompetenzmodells ein? | jeweilige Anzahl der Lehrerinnnen und Lehrer, die diese Option wählten (jeweils N=40) |
|---|---|
| Die Einteilung in verschiedene Stufen ist sinnvoll. | 35 |
| Die Einteilung in verschiedene Komponenten ist sinnvoll. | 25 |
| Die Einteilung ist ungünstig, weil...<br>L10: ...sie zu kompliziert und komplex ist.<br>L27: Die Schwierigkeitsstufen werden von den Schülern teilweise anders empfunden | 2 |

Warum haben Sie den Unterricht nie bzw. selten am Kompetenzmodell ausgerichtet?

L3: Zeitgründe. Ich habe eigene Aufgabenstellungen entwickelt.
L10: Mir ist das Modell zu kompliziert und komplex.
L28: Aus schulorganisatorischen Gründen konnte ich die Unterrichtseinheit erst im Februar beginnen.
L37: Der Unterricht wurde an den Wünschen der Schüler ausgerichtet. Die Vereinbarkeit mit

dem Lehrplan wurde dabei vom Lehrer beachtet. Die Ausrichtung zum Kompetenzmodell war nicht vordergründig.

| Das Kompetenzmodell ist ... | sehr übersichtlich. | übersichtlich. | unübersichtlich. | sehr unübersichtlich. |
|---|---|---|---|---|
| Häufigkeit (N= 40) | 7 | 27 | 5 | 1 |

Bitte geben Sie Kritik, Lob, Anmerkungen und Verbesserungsvorschläge für das Kompetenzmodell an.

L2: Bei mir war alles gut umsetzbar.
L3: Es müsste eine bessere Übersicht über die Unterrichtsstunden/Unterrichtseinheiten geben.
L5: Es fiel mir leicht, mit den angebotenen Materialien zu arbeiten und daraus eine sinnvolle Unterrichtssequenz zu planen, so dass die SchülerInnen mit großer Selbständigkeit arbeiten konnten.
L10: Prinzipiell ist es ja gut, die Aufgaben in Schwierigkeitsgrade und Kompetenzen aufzuteilen. Aber den Unterricht kann ich danach nicht ausrichten, da muss ich mich doch mehr an den Inhalten orientieren. Beides zusammen war mir zuviel. Deshalb habe ich den Kompetenztest auch nicht geschrieben (zumal dieses Wort wegen des Ma-KT vorbelegt ist und zwar nicht gerade positiv).
L11: Unterricht nach dem Kompetenzmodell war gut möglich. Gliederung und Aufgabenstellung waren für eine 9. Klasse Gymnasium angemessen.
L12: Das Kompetenzmodell an sich ist gut.
L13: Ich habe das Kompetenzmodell nur kurzfristig einsetzen können, bin mit den erreichten Ergebnissen aber weitestgehend zufrieden. Das Modell war eine gute Möglichkeit auch Schüler der Sekundarschule an das Programmieren heranzuführen.
L14: - übersichtlichere Darstellung der gedruckten Ausführungen; - mehr und leichtere Übungsaufgaben
L27: Manche theoretische Konstrukte (z. B. Bedingungen für einen Algorithmus) müssen den Schülern beispielhaft nähergebracht werden.
L29: Teilweise waren die Ansprüche für Haupt- und Realschüler zu komplex. Es war auch so gelagert, dass die Schüler nur einzelne Teile je Kompetenz nicht verstanden, jedoch andere Teile auch in einer höher liegenden Kompetenz gut bearbeiten konnten.
L33: - praktisch sehr gut handhabbar zur individuellen Lernerfolgskontrolle
L35: Die Behandlung des Algorithmusbegriffs ist mit den Schülern aus Stufe 8 nicht einfach. Ihnen fällt die Abstraktion schwer. Auch die Formulierung von Algorithmen im Pseudocode oder als Struktogramm bzw. Programmablaufplan ist für sie schwierig. Die Arbeit mit den Algorithmen und die Programmentwicklung geht vielen Schülern deutlich leichter von der Hand als das Arbeiten mit den Komponenten A und B. Es hat ein bisschen gedauert bis ich das Kompetenzmodell verstanden hatte. Wenn es klar ist, kann man aber insgesamt gut damit arbeiten.
L36: Lob für die gute Struktur!

## Fragen zu den Unterrichtsaufgaben

| Einsatz der Aufgaben für den Unterricht (jeweils N=40) | | | | | |
|---|---|---|---|---|---|
| | nie | selten | gele-gentlich | oft | immer |
| Wie oft haben Sie die bereitgestellten Aufgaben bei der Unterrichtsvorbereitung eingesetzt? | 1 | 1 | 8 | 24 | 6 |
| Wie oft wurden in Ihrem Unterricht Aufgaben unterschiedlicher Kompetenzstufen zur Differenzierung eingesetzt? | 7 | 14 | 12 | 6 | 1 |

| Einschätzung der Aufgaben für den Unterricht (jeweils N=40) | | | | | |
|---|---|---|---|---|---|
| | ich stimme über-haupt nicht zu | ich stimme nicht zu | unent-schlossen | ich stimme etwas zu | ich stimme voll zu |
| Die Einordnung der Aufgaben für den Unterricht in die Komponenten und Stufen des Kompetenzmodells war für mich eine wichtige Hilfe. | 1 | 2 | 4 | 19 | 14 |
| Die geforderten Kompetenzen wurden durch die Aufgaben für den Unterricht angemessen abgebildet. | 0 | 3 | 9 | 14 | 14 |

Welche Kompetenzen wurden durch die Aufgaben für den Unterricht nicht angemessen abgebildet?

L10: Ich weiß die Einordnung gerade nicht, aber für die Programmierung von Prozeduren (Aufteilung eines Problems in Prozeduren etc.) fehlten mir Aufgaben.

L29: - Schüler hatten große Probleme mit den Handlungsvorschriften; - hatten Probleme bei Aufgaben mit mathematischem Hintergrund die Mathematik zu durchschauen; - große Probleme ergaben sich auch mit dem Koordinatensystem
L32: - die grundlegenden Kompetenzen zum Algorithmusbegriff

Bitte geben Sie Kritik, Lob, Anmerkungen und Verbesserungsvorschläge für die bereitgestellten Aufgaben an.

L2: - keine Kritikpunkte
L5: Ich bin sehr zufrieden mit den bereitgestellten Aufgaben. Die SchülerInnen waren mit Begeisterung bei der Sache.
L10: Insgesamt ist das eine schöne Aufgabensammlung! Danke!
L11: Probleme der Schüler beim Lösen von Aufgaben traten vor allem dann auf, wenn sie die Angaben nicht genau gelesen hatten. Puck ist super – die Schüler haben sehr schnell gelernt damit umzugehen und mussten sich nicht mit lästiger Syntax-Fehlersuche herumschlagen.
L12: Die Aufgaben waren von Anfang an zu anspruchsvoll. So wurde bereits bei der ersten Aufgabe vorausgesetzt, dass die Schüler Schleifen verstehen können...
L14: - mehr und leichtere Aufgaben zum Eintrainieren (bezieht sich auf die Anforderungen in einer 7. Klasse, in der das Prinzip und das System Puck sehr gut ankamen)
L19: Es wäre hilfreich, an einer Aufgabe die Komplexität durch weitere Zusatzaufgaben in mehreren Stufen zu erarbeiten.
L22: positiv: Aufgaben mit grafischen Bausteinen, sehr motivierend; - Fehler bei Aufgabe 38 (Summieren)
L24: Aufgabe Römische Zahlen: Programm wurde von mir (bzw. natürlich auch von den Schülern) bis zur Zahl 5000 erweitert und zusätzlich wurden Ausgaben wie „IIII“ in „IV“ umgewandelt (für jeden Zahlenbereich). Damit konnte ein für die Schüler sehr realistisches Programm geschrieben werden und die Schüler waren sehr engagiert.
L26: Meinen Schülern half oft der Pseudocode, das Programm umzusetzen. Ich erarbeite oft die Struktogramm an der Tafel und gebe den Code als Unterstützung an die schwächeren Schüler aus. Die graphischen Aufgaben (Haus und Segelboot) haben ihnen sehr gut gefallen.
L27: Die Aufgaben wurden von den Schülern und mir als sinnvoll empfunden. Die Aufgabe zu den Prozeduren (Informatikanien) war für den Einstieg etwas komplex.
L29: Haupt- und die meisten Realschüler brauchen klein schrittigere Anweisungen, Hinführungen. Ich habe z. B. ein Koordinatensystem als Kopiervorlage erstellt und die Schüler konnten durch das Vorzeichnen der Figuren leichter die passenden Koordinaten finden (was jedoch dann auch nicht immer leicht war). Einfache mathematische Probleme lösen lassen.
L32: Der Begriff Algorithmus sollte besser über ein Rollenspiel veranschaulicht werden, d. h. eine Gruppe von Schülern überlegt sich, wie eine Maschine zu programmieren ist, damit sie bestimmte Aufgaben z. B. das Durchlaufen eines Labyrinths selbständig erledigt. Dies sollte simuliert und protokolliert werden. Zudem fehlt die Eigenschaft Terminiertheit.
L35: Die bereit gestellten Beispiel- und auch Übungsaufgaben sind für den Unterricht sehr hilfreich und lassen sich auch sehr zielgerichtet einsetzen. Auch hier schien mir die Gliederung zunächst etwas unübersichtlich. Nach dem Erfassen des Aufbaus habe ich mit den teilweise differenziert eingesetzten Aufgaben den wesentlichen Teil meines Unterrichts bestritten.
L36: Die Angleichung der Kompetenzstufen mit Programmierbausteinen ist gut gelungen.
L40: Viele Aufgaben, dadurch gute Wahlmöglichkeiten und Differenzierungsmöglichkeiten...

## Fragen zum Puck-System

| Zustimmung zu Aussagen bezüglich Puck (jeweils N=40) | | | | | |
|---|---|---|---|---|---|
| | ich stimme überhaupt nicht zu | ich stimme nicht zu | unentschlossen | ich stimme etwas zu | ich stimme voll zu |
| Das Verringern der Syntaxfehler durch das Puck-System war günstig für den Anfangsunterricht. | 0 | 0 | 0 | 5 | 35 |
| Für viele Schülerinnen und Schüler erwies sich die Arbeit mit dem Puck-System als schwierig. | 10 | 22 | 2 | 6 | 0 |
| Das Puck-System motiviert die Schülerinnen und Schüler, sich in den Informatikunterricht aktiv einzubringen. | 1 | 1 | 7 | 12 | 19 |
| Es gibt zu wenig geeignete Unterrichtsmaterialien für den Einsatz des Puck-Systems. | 7 | 14 | 3 | 12 | 4 |
| Der Einsatz des Puck-Systems motivierte mich für meine Unterrichtsvorbereitung und Unterrichtsdurchführung. | 0 | 1 | 3 | 17 | 19 |
| Der Einarbeitungsaufwand in das Puck-System ist für Informatiklehrerinnen und -lehrer sehr hoch. | 17 | 19 | 3 | 1 | 0 |

| Zustimmung zu Aussagen bezüglich Puck (jeweils N=40) | | | | | |
|---|---|---|---|---|---|
| | ich stimme überhaupt nicht zu | ich stimme nicht zu | unentschlossen | ich stimme etwas zu | ich stimme voll zu |
| Die Schülerinnen und Schüler sind durch den Unterricht mit dem Puck-System auf die Arbeit mit einer textuellen Programmiersprache gut vorbereitet worden. | 0 | 4 | 12 | 16 | 8 |
| Durch die Arbeit mit dem Puck-System hatte ich mehr Zeit für die individuelle Förderung einzelner Schülerinnen und Schüler. | 2 | 10 | 8 | 15 | 5 |
| Durch die Arbeit mit dem Puck-System wird den Schülerinnen und Schülern ein falsches Bild von der Programmierung vermittelt. | 15 | 13 | 10 | 2 | 0 |
| Auch leistungsschwächere Schülerinnen und Schüler konnten durch die Arbeit mit dem Puck-System Erfolge erzielen. | 1 | 3 | 2 | 17 | 17 |
| Die Funktionalität von Puck reicht für eine Einführung in die Programmierung aus. | 0 | 7 | 4 | 17 | 12 |
| Ich möchte das Puck-System in den nächsten Schuljahren wieder einsetzen. | 0 | 3 | 2 | 8 | 27 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Ich halte den Einsatz des Puck-Systems für sinnvoll ab Klassenstufe ... | 5 | 6 | 7 | 8 | 9 | 10 |
| Häufigkeit (N=39) | 2 | 2 | 9 | 12 | 12 | 2 |

Bitte geben Sie Kritik, Lob, Anmerkungen und Verbesserungsvorschläge für die visuelle Programmiersprache Puck an.

L2: - alles ok
L3: Es sollten mehr Beispiele für eine gesamte Unterrichtsreihe mit Varianten gegeben werden.
L7: Ich wünsche mir für die nächste Version von Puck die Möglichkeit, Bausteinquelle, Arbeitsbereich und Attributtabelle in ihrer Breite auf dem Bildschirm individuell einstellen zu können.
L10: Ich habe schon viele Verbesserungsvorschläge (z.B. Copy&Paste) gemailt. Wäre schön, wenn die umgesetzt würden.
L11: Habe leider nur mit der 9. Klasse Erfahrungen; Kollege hat Puck in der 7. Klasse gemacht - war wohl recht schwierig für die Schüler. Neuntklässler kamen dagegen sehr gut zurecht und haben schnell kapiert. Schüler haben auch viel selbstständig ausprobiert.
L14: - endlich so programmieren, dass man nicht auf Windows angewiesen ist; - Bildschirmausgabe optimieren (auf grafikschwachen Terminals ein einziges Geruckel und Gezuckel) also nur technische Kritikpunkte die Idee und die Verwendung (bis auf oben erwähnte Punkte) kritikfrei; - tolle Idee
L16: Als Mangel sehe ich an, dass Puck über keine Funktionen verfügt, obwohl gerade Java diese favorisiert. Der Baustein Output könnte verbessert werden, an vielen Stellen ist er „unhandlich“, z.B. wenn mehrere Daten hintereinander mit Text ausgegeben werden sollen.
L18: Bitte unbedingt Real-Typen einbauen. Es ist anstrengend, die Beispiele immer so hinzubiegen, dass mit Mod und Div sinnvolle Werte rauskommen.
L22: - keinen Frust durch Syntaxfehler; - ich habe viele Schülergespräche über algorithmische Fragestellungen erlebt; - toll wäre, wenn man mit Puck auch Struktogramme gestalten könnte
L23: Ich halte es für sinnvoll und unbedingt notwendig, das Entwickeln von Algorithmen vor der Arbeit mit dem PUCK-System durch das Erstellen von Struktogrammen zu unterstützen. Dadurch wird die Struktur der eingeschlagenen Lösungswege deutlicher. Im PUCK-System ist das alleinige Betrachten des Programms, insbesondere beim Einsatz verschachtelter Schleifen und Bedingungen, gewöhnungsbedürftig.
L24: Puck motiviert die Schüler sehr. Weitere Bausteine, wie zum Beispiel für die String-Verarbeitung (z. B. Umrechnung dezimal-binär ist mit Puck und ohne String-Verarbeitung für die Schüler doch sehr komplex und ähnelt der Umrechnung aus der Realität nicht besonders)
L25: Die Ausgabe der Ergebnisse (außer Grafiken) lässt sehr zu wünschen übrig. Ebenso ist die Beschränkung der Datentypen auf Boolean und Integer ein großer Mangel.
L26: Ich habe die Kopierfunktion vermisst und das Öffnen von Programmen von externen Datenträgern ist etwas zeitraubend.
L28: Es fehlen die Funktionen „kopieren/einfügen“ und „drucken“ Im übrigen ist Puck gut einsetzbar.
L29: Für die Hauptschule gab es in der 10. Klasse gute Arbeitsergebnisse, in der 9. Jahrgangsstufe liegt es ein wenig an der Motivation der gesamten Klasse; In den Realschulklassen gab es sowohl in der 9. als auch in der 10. Jahrgangsstufe sehr gute Arbeitsleitungen. Das Programm hakt in unserer Umgebung (Linux-Server, Windows 2000 Betriebssystem);
L32: Bei den Schleifen ist die Angabe von true und false anstatt eines vorgegebenen Vergleiches

gewöhnungsbedürftig und für Schüler nicht einfach.
L35: Das Prinzip der Bausteine, das das Erlernen einer Sprachsyntax ausklammert, ist m. E. ein voller Erfolg. Man kann mit den Schülern deutlich schneller Algorithmen umsetzen und Programme entwickeln. Für die meisten Schüler ist der schnelle Erfolg ihrer Bemühungen sehr motivierend. Puck ist ein sehr gelungenes Instrument zur Einführung in die Informatik. Verbesserungsvorschlag: Die Ausgabemöglichkeiten sind nicht sehr üppig. Vielleicht könnte man da etwas nachbessern (mir ist schon bewusst, dass dabei das Prinzip des einfachen, auf die wesentlichen Elemente beschränkten Systems nicht aufgegeben werden sollten).
L36: Die Beschränkung auf Integer-Variablen ist etwas hinderlich.
L38: - eine Erweiterung der Datentypen um real wäre wünschenswert; - die Gestaltung des Ausgabefensters könnte verbessert werden
L40: Mir fehlten die Kommazahlen und das Wurzelziehen...

## Fragen zum Kompetenztest

| Wer hat festgelegt, welche Stufen die Schülerinnen und Schüler im Kompetenztest bearbeiten? | Häufigkeit (N=23) |
|---|---|
| Ich habe eine Stufe festgelegt, die von der ganzen Klasse einheitlich zu bearbeiten war. | 19 |
| Ich habe individuell für jede Schülerin und jeden Schüler (oder für Schülergruppen) festgelegt, welche Stufe zu bearbeiten war. | 0 |
| Jede Schülerin und jeder Schüler hat für sich festgelegt, welche Stufe zu bearbeiten war. | 1 |
| Jede Schülerin und jeder Schüler hat für sich für jede der Komponenten die Stufe festgelegt, die zu bearbeiten war. | 2 |
| Sonstiges | 1<br>L16:Ich konnte den Test nicht komplett übernehmen, da ich einige Aufgaben im Unterricht behandelt hatte, keine PC's verwenden konnte und nur 60 Minuten zur Verfügung hatte. |

| Wie viele Schülerinnen und Schüler haben am Kompetenztest teilgenommen? | 6-10 | 11-15 | 16-20 | 21-25 | 26-35 |
|---|---|---|---|---|---|
| Häufigkeit (N=21) | 4 | 5 | 7 | 2 | 3 |

| Wie viele Schülerinnen und Schüler haben Stufe III erreicht? | 0 | 1-5 | 6-10 | 11-15 | 16-20 |
|---|---|---|---|---|---|
| Häufigkeit (N=21) | 15 | 3 | 2 | 0 | 1 |

| Wie viele Schülerinnen und Schüler haben Stufe II, aber nicht Stufe III erreicht? | 0 | 1-5 | 6-10 | 11-15 | 16-20 |
|---|---|---|---|---|---|
| Häufigkeit (N=21) | 9 | 9 | 2 | 1 | 0 |

| Wie viele Schülerinnen und Schüler haben Stufe I, aber nicht Stufe II erreicht? | 0 | 1-5 | 6-10 | 11-15 | 16-20 | 21-35 |
|---|---|---|---|---|---|---|
| Häufigkeit (N=21) | 3 | 7 | 4 | 2 | 3 | 2 |

| Wurde der Kompetenztest benotet? | Ja | Nein |
|---|---|---|
| Häufigkeit (N=22) | 11 | 11 |

| Welche Stufe des Kompetenztests bearbeiteten die Schülerinnen und Schüler? | Stufe I | Stufe II | Stufe III |
|---|---|---|---|
| Häufigkeit (N=19) | 12 | 4 | 3 |

Nach welchen Kriterien wurden die Noten beim Kompetenztest vergeben?

L2: - Vollständigkeit; - stringente Anwendung
L3:

| Note | Prozente | Punkte von bis |
|---|---|---|
| 1 | 90 - 100 | 22 - 24 |
| 2 | 76 - 89 | 18 - 21 |
| 3 | 61 - 75 | 15 - 17 |
| 4 | 46 - 60 | 11 - 14 |
| 5 | 21 - 45 | 5 - 10 |
| 6 | 0 - 20 | 0 - 4 |

L16: Notenspiegel nach KMK-Prozenten
L19: - Komplexität; - Vollständigkeit; - Ansatz
L31: Nach der üblichen Prozentverteilung entsprechend der erreichten Punkte (50 % = Note 4).
L34: Gemäß der Zuordnung Punkte → Note :
24-22→ 1; 21-19→ 2; 18-15→ 3; 14-11→ 4; 10-5→ 5; 4-0→ 6.
L35: Nach folgendem Bewertungsschlüssel: Punkte → Note
24-22→ 1; 21-19→ 2; 18-15→ 3; 14-11→ 4; 10-5→ 5; 4-0→ 6.
L36: - Punktgrenzen

| Wie schätzen Sie das vorgegebene Kriterium zum Erreichen der Stufen im Kompetenztest ein? (75 % der in einer Stufe möglichen Punkte mussten erreicht werden) | Häufigkeit (N=22) |
|---|---|
| 75 % ist zu gering. | 0 |
| 75 % ist zu hoch. | 6 |
| Es sollten in jeder der Komponenten eine minimale Punktzahl erreicht werden. | 3 |
| Das Kriterium ist genau richtig. | 13 |

| Wie viel Zeit hatten die die Schülerinnen und Schüler zur Bearbeitung des Kompetenztests? | 45 Minuten | 60 Minuten | 75 Minuten | 80 Minuten | 90 Minuten |
|---|---|---|---|---|---|
| Häufigkeit (N=19) | 3 | 2 | 2 | 1 | 11 |

| Die im Kompetenzmodell geforderten Kompetenzen wurden durch den bereit gestellten Kompetenztest angemessen abgebildet. | ich stimme überhaupt nicht zu | ich stimme nicht zu | unentschlossen | ich stimme etwas zu | ich stimme voll zu |
|---|---|---|---|---|---|
| Häufigkeit (N=22) | 0 | 1 | 2 | 11 | 8 |

Warum haben Sie den Kompetenztest nicht im Unterricht eingesetzt?

L4: Das hing mit der Lerngruppe und den miserablen Ergebnissen (kaum Lernfortschritte, schlechte Arbeitshaltung) zusammen.
L5: Meine Kurse haben viel mehr Schüler als PC's, so dass die Unterrichtszeit für die Durchführung des Kompetenztests mit zwei Gruppen nicht möglich war, ohne dass die Gefahr bestand, dass Informationen über die Aufgaben weitergegeben werden.
L9: Er passte nicht hinein.
L10: Weil ich Probleme mit der Einteilung in die Kompetenzen hatte. Außerdem war mir der Test zu lang.
L12: Es war unmöglich meine Schülerinnen und Schüler in einem Halbjahr soweit zu bringen, dass sie die Aufgaben in dem Kompetenztest auch nur ansatzweise verstanden hätten.
L15: Es war leider keine Zeit - aus schulinternen Gründen - bislang vorhanden. Kommt noch.
L18: Zeitgründe. Der Unterricht war fakultativ und fiel öfter aus.
L20: Entgegen der Schuljahresplanung war kein wöchentlicher Unterricht im Halbjahr möglich und damit konnte der Lehr-/Lernstoff nicht komplett vermittelt werden.
L21: Da die Zeit des Einsatzes des Materials zu kurz dazu war.
L22: Lerngruppe ist zu groß, so dass nicht jeder alleine an einem Computer arbeiten kann.
L23: Wir bieten Informatik im Wahlpflichtunterricht an. Die Schülerinnen und Schüler haben dieses Fach mit unterschiedlichen Erwartungen gewählt. Der Einsatz in diesem Schuljahr (ab August 2007) war sehr kurz. Ich werde das Puck-System auf jeden Fall weiter verwenden, um schrittweise die Komplexität von Programmen zu erhöhen. Dabei werde ich dann auch den Kompetenztest nach und nach einsetzen.
L24: Leider hat die Zeit dazu nicht ausgereicht.
L28: Organisatorische Gründe: die Unterrichtseinheit ist noch nicht beendet!
L39: Zeitfrage. Der Unterricht ist wegen Nachmittagsstunden oft ausgefallen.

### Bitte geben Sie Kritik, Lob, Anmerkungen und Verbesserungsvorschläge für den Kompetenztest an.

L2: keinerlei Kritikpunkte
L3: Die Aufgaben der Kompetenzstufe I zum Algorithmus waren zu leicht.
L5: Der Kompetenztest hätte je zur Hälfte aus einem theoretischen und einem praktischen Teil zu je 45 Minuten bestehen sollen. Dann hätte ich die Gruppe halbiert und der Kompetenztest wäre ohne Probleme durchführbar gewesen, wenn die Schüler nach der Hälfte der Zeit ihre Plätze getauscht hätten.
L10: Der Test sollte viel kürzer und übersichtlicher sein.
L13: Anmerkung: Die geringe Erfolgsquote in dem Kompetenztest liegt lediglich an der geringen Zeitspanne in der ich das Modell einsetzen konnte. Alle Schüler, die Interesse für das Programmieren zeigten hatten gute Ansätze.
L14: Er konnte noch nicht durchgeführt werden.
L16: Gut so. Der „Support“ über die Puck-Homepage könnte noch besser werden ... ist aber wohl in Arbeit.
L19: Im Augenblick keine Verbesserungen.
L22: - Verteilung zwischen theoretischen (Papier) und praktischen (Computer) Aufgaben 50 : 50, so dass im Wechsel gearbeitet werden kann; - Aufgaben so gestalten, dass am Ende die erreichte Punktzahl angibt, welche Kompetenzstufe erreicht wurde
L26: Aus stundenplantechnischen Gründen unterrichte ich nur 14tägig und habe diesen Test trotzdem durchgeführt. Dadurch erklärt sich auch das Ergebnis. Damit haben die Schüler einen Soll/Ist-Vergleich. Generell wäre es ein Zwischentest.

L29: Meine Schüler beherrschen zwar sogar Teile aus Stufe III, jedoch kamen sie mit allgemeinen Fragen, z. B. Algorithmus beim Benutzen eines Handys nicht klar. Solche Transferaufgaben überforderten sie sehr. Die Aufgaben sind insgesamt zu sehr an den gymnasialen Schülern ausgerichtet. Jedoch eignet sich Puck aufgrund der Struktur besonders gut als Programmiersprache für Haupt- und Realschüler. Eventuell wäre nach einem statt einem halben Jahr die Differenz auch weg.
L32: Ein großes Lob, der Kompetenztest war völlig in Ordnung.
L34: Die Aufgabenstellung in Komponente A war teilweise zu einfach (Nennen eines Beispiels); beim Ankreuzen hätte ich mir Begründungen gewünscht. Bei Komponente D verführte die Angabe der Programmausgabe zum Kopfrechnen.
L35: Reine Ankreuzfragen (Stufe I, Komp. A) sind zu einfach bzw. zu „zufallsbehaftet“. Auch die Frage nach den Flächeninhalten bei Stufe I, Komp. D, die man im Kopf ausrechnen kann scheint mir nicht so sinnvoll. Ansonsten ist der Kompetenztest in Ordnung.
L36: Gut durchdacht.

## Allgemeine Fragen zu kompetenzorientiertem Unterricht

| Die Einteilung von Kompetenzmodellen in einzelne Komponenten ist ... | jeweilige Anzahl der Lehrerinnnen und Lehrer, die diese Option wählten (jeweils N=40) |
|---|---|
| ... sinnvoll für die Strukturierung von Kompetenzmodellen. | 25 |
| ... sinnvoll für die Vorbereitung auf Kompetenztests. | 20 |
| ... sinnvoll für die differenzierte Beurteilung von Leistungen | 25 |
| ... nicht nützlich | 2 |
| sonstiges | 0 |

| Die Einteilung von Kompetenzmodellen in einzelne Stufen ist ... | jeweilige Anzahl der Lehrerinnnen und Lehrer, die diese Option wählten (jeweils N=40) |
|---|---|
| ... sinnvoll für die Strukturierung von Kompetenzmodellen. | 19 |
| ... sinnvoll für die Vorbereitung auf Kompetenztests. | 19 |
| ... sinnvoll für Differnzierung im Unterricht. | 26 |
| ... nicht nützlich | 0 |
| sonstiges | 0 |

| Kompetenztests sind hilfreich bei ... | jeweilige Anzahl der Lehrerinnnen und Lehrer, die diese Option wählten (jeweils N=40) |
|---|---|
| ... der Einschätzung der entwickelten Kompetenzen der Schülerinnen und Schüler. | 35 |
| ... der Einschätzung der Wirksamkeit des pädagogischen Handelns einer Lehrperson | 16 |
| sonstiges | 1 (L10: eher nicht nützlich) |

| Kompetenzorientierter Unterricht | | | | | |
|---|---|---|---|---|---|
| | ich stimme überhaupt nicht zu | ich stimme nicht zu | unentschlossen | ich stimme etwas zu | ich stimme voll zu |
| Ich wünsche mir eine Fortbildung zum Einsatz von Kompetenzmodellen im Informatikunterricht. (N=40) | 1 | 3 | 12 | 14 | 10 |
| Ich würde Kompetenzmodelle auch in anderen Gebieten der Informatik einsetzen. (N=39) | 0 | 4 | 9 | 13 | 13 |

Dipl.-Inf. Lutz Kohl
Dornburger Straße 113 · 07743 Jena
Tel. 03641/296402
E-Mail: lutz.kohl@googlemail.com

# Lebenslauf Dipl.-Inf. Lutz Kohl

| | |
|---|---|
| **Geboren am** | 08.11.1980 |
| **Geburtsort** | Leinefelde |
| **Familienstand** | verheiratet mit Susann Kohl seit 2005 |
| **Kinder** | Charlotte Kohl |

Schulbildung

| | |
|---|---|
| **08/1987 – 06/1991** | POS Beuren |
| **08/1991 – 06/1999** | Leibniz-Gymnasium Leinefelde |
| **06/1999** | Abitur |

Sonstige Tätigkeiten

| | |
|---|---|
| **06/1999 – 04/2000** | Grundwehrdienst in Hessisch Lichtenau |
| **05/2000 – 09/2000** | Kundenbetreuer in der VW-Autostadt Wolfsburg |

Studium

| | |
|---|---|
| **10/2000 – 09/2001** | Studium Informatik an der TU Ilmenau |
| **10/2001 – 12/2004** | Studium Informatik an der Friedrich-Schiller-Universität Jena |
| **02/2004 – 03/2004** | Praktikum in der Softwareentwicklung der Firma IT-Double GbR in Hamburg |
| **12/2004** | Abschluss: Diplom-Informatiker |

Berufstätigkeit

| | |
|---|---|
| **Seit 02/2005** | wissenschaftlicher Mitarbeiter in der Abteilung Didaktik der Mathematik und Informatik an der Friedrich-Schiller-Universität Jena |

30. Oktober 2008

Lutz Kohl