Sigrid Schubert (Hrsg.)

Didaktik der Informatik in Theorie und Praxis

12. GI-Fachtagung „Informatik und Schule – INFOS 2007“
19.-21. September 2007 an der Universität Siegen

Gesellschaft für Informatik e.V. (GI)

Lecture Notes in Informatics (LNI) - Proceedings
Series of the Gesellschaft für Informatik (GI)

Volume P-112

ISBN 978-3-88579-206-2
ISSN 1617-5468

Volume Editor
Prof. Dr. Sigrid Schubert
Universität Siegen
Fachbereich Elektrotechnik und Informatik
Hölderlinstr. 3
57068 Siegen, Germany
E-Mail: schubert@die.informatik.uni-siegen.de

printed by Köllen Druck+Verlag GmbH, Bonn

Vorwort

Obwohl sich die Didaktik der Informatik als Fachgebiet der Informatik in Form neuer Arbeitsgruppen an Hochschulen etablierte, bleibt die Stärkung der informatischen Bildung in der Schule eine noch zu lösende Aufgabe. Dazu leistet die 12. Fachtagung der Gesellschaft für Informatik (GI) „Informatik und Schule – INFOS 2007“ einen wichtigen Beitrag. Sie steht unter dem Motto „Didaktik der Informatik in Theorie und Praxis“ und wird vom GI-Fachausschuss Informatische Bildung in Schulen zusammen mit der Universität Siegen veranstaltet. Traditionsgemäß fördert sie den Erfahrungsaustausch zwischen Lehrenden und Forschenden und bietet Anregungen dafür, wie erfolgreicher Informatikunterricht gestalten werden kann und welche bildungspolitischen Entscheidungen dafür zu treffen sind. Drei Besonderheiten zeichnen die INFOS 2007 aus: die Kooperation mit „Der 5. e-Learning Fachtagung Informatik − DeLFI 2007“ der GI-Fachgruppe E-Learning, die Kooperation mit der GI-Fachgruppe Didaktik der Informatik, die ihren 7. Workshop „Didaktik der Informatik − aktuelle Forschungsergebnisse“ integrierte, und die Kooperation mit dem „6. Informatiktag Nordrhein-Westfalen“ der GI-Fachgruppe „Informatische Bildung in Nordrhein-Westfalen“. Den Aktiven im Fachausschuss und in den drei Fachgruppen danke ich für die ausgezeichnete Zusammenarbeit und Unterstützung.

Dieser Tagungsband enthält vier eingeladene Vorträge und 23 Beiträge, die vom Programmkomitee aus 46 eingereichten Beiträgen nach einem wissenschaftlichen Begutachtungsprozess ausgewählt wurden. Diese Beiträge beleuchten die folgenden Schwerpunkte der Tagung aus unterschiedlichen Perspektiven: Studium und Fortbildung, Informatiksysteme als Unterrichtsmittel, Forschungsansätze, Fundamentale Ideen in der Unterrichtspraxis, Anwendungsorientierter Informatikunterricht, Qualitätsentwicklung und Qualitätssicherung im Informatikunterricht. Eine Abrundung erfährt der Tagungsband durch acht zweiseitige Kurzbeiträge und die Vorträge des 7. Workshops „Didaktik der Informatik − aktuelle Forschungsergebnisse“. Im Rahmen der Konferenz finden außerdem 26 weitere Workshops und drei thematische Fortbildungen statt.

Die Organisation und Durchführung der gesamten Tagung und die Erstellung dieses Tagungsbandes waren nur durch das Engagement vieler Personen und Institutionen und durch die finanzielle Unterstützung der Sponsoren und Aussteller möglich, denen ich hiermit danke. Stellvertretend werden „LOG IN – informatische Bildung und Computer in der Schule“ und „Informatica Didactica – Zeitschrift für fachdidaktische Grundlagen der Informatik“ genannt. Außerdem möchte ich allen Autoren für ihre qualitativ hochwertigen Beiträge zu diesem Band und damit zum Gelingen der Tagung danken. Ein besonderer Dank gebührt den Mitgliedern des Programmkomitees der INFOS 2007, dem Vorbereitungsteam des Informatiktags Nordrhein-Westfalen und der Jury des Unterrichtswettbewerbs für ihre sorgfältige Begutachtung der Beiträge und für die konstruktive Diskussion bei der Erstellung des Tagungsprogramms, sowie dem Organisationskomitee unter Leitung von Peer Stechert, dem Karin Ofterdinger, Kirstin Schwidrowski, Gerd Müller und Christian Eibl angehören.

Siegen, im September 2007 Sigrid Schubert

Programmkomitee

Sigrid Schubert, Universität Siegen (Vorsitz)
Norbert Breier, Universität Hamburg
Torsten Brinda, Universität Erlangen
Katrin Büttner, Mittelschule Heidenau
Michael Fothe, Universität Jena
Steffen Friedrich, TU Dresden
Werner Hartmann, PH Bern
Peter Hubwieser, TU München
Udo Kelter, Universität Siegen
Bernhard Koerber, FU Berlin
Barbara Leipholz-Schumacher, Euregio-Kolleg
Roland Mittermeir, Universität Klagenfurt
Hermann Puhlmann, Leibniz-Gymnasium Altdorf
Gerhard Röhner, Studienseminar Darmstadt
Andreas Schwill, Universität Potsdam
Monika Seiffert, Amt für Bildung, Hamburg
Peer Stechert, Universität Siegen
Hiltrud Westram, Gymnasium Lechenich, Erftstadt

Organisation

Prof. Dr. Sigrid Schubert
Dipl.-Inf. Peer Stechert
Didaktik der Informatik und E-Learning
Universität Siegen
Fachbereich Elektrotechnik und Informatik
Hölderlinstr. 3,
57068 Siegen
http://www.die.informatik.uni-siegen.de
schubert@die.informatik.uni-siegen.de
stechert@die.informatik.uni-siegen.de

Inhaltsverzeichnis

Eingeladene Vorträge

Studium und Fortbildung

Informatiksysteme als Unterrichtsmittel

Forschungsansätze

Fundamentale Ideen in der Unterrichtspraxis

Anwendungsorientierter Informatikunterricht

Qualitätsentwicklung und -sicherung im Informatikunterricht

Kurzbeiträge

Workshop: „Didaktik der Informatik – aktuelle Forschungsergebnisse“

Objektorientiertes Programmieren – Machen wir irgendwas falsch?

Jürgen Börstler
Umeå University, Schweden
jubo@cs.umu.se

Abstract: Obwohl die Studierenden immer bessere Vorkenntnisse mitbringen, werden die Ergebnisse in der Einführung in die Programmierung immer schlechter. So, oder ähnlich, klagen die Hochschulen mehr oder weniger überall, auch bei uns in Schweden. Gleichzeitig gibt es eine Reihe von Publikationen, die aufzeigen, dass es möglich ist, in Einzelfällen, Verbesserungen zu erzielen (z. B. mit Hilfe spezieller Programmierumgebungen, Bibliotheken oder Visualisierungen). Des Weiteren werden auch ganzheitliche „Konzepte“ angeboten, um den Einstieg in die objektorientierte Programmierung zu erleichtern (wie z. B. im Rahmen des „TeachScheme!“ Projektes).
Also, um die Titelfrage zu beantworten; Offensichtlich machen wir etwas falsch. Aber was genau? Wie wird (objektorientiertes) Programmieren richtig unterrichtet? Leider lassen sich nur sehr wenige Studien verallgemeinern und viele Erklärungsmodelle sind nicht theoretisch verankert.
In diesem Beitrag werden wir einige Studien und Theorien diskutieren, die für die Entwicklung einer Einführung in die Programmierung relevant sind, aber aus den verschiedensten Gründen oft nicht beachtet werden.

1 Einleitung

Die Leistungen in den Einführungsveranstaltungen der Programmierung werden immer schlechter, lautet eine weit verbreitete Klage. Als eine Ursache für dieses Phänomen wird oft die Objektorientierung angeführt. Aber gibt es wirklich Belege dafür? Gibt es überhaupt Belege für eine generelle Verschlechterung der Leistungen in den Einführungsveranstaltungen der Programmierung? Meines Wissens gibt es keine zuverlässigen Trendstudien, die entweder das Eine oder das Andere belegen. Da sich die Rahmenbedingungen in der Informatik sehr schnell verändern, sind solche Studien jedoch auch sehr schwierig. Auf der anderen Seite gibt es eine große Menge von Publikationen, die zeigen, dass es, in Einzelfällen, sehr wohl möglich ist objektorientiertes Programmieren als erstes Paradigma zu lehren und zu lernen.

Wir sollten uns daher eine Reihe grundlegender Fragen stellen. Gibt es überhaupt ein Problem? Falls ja, was machen wir als Lehrkräfte und/oder EntwicklerInnen von Unterrichtsmaterial falsch? Gibt es Faktoren, die die Studienresultate in einem Programmierkurs maßgeblich beeinflussen, aber nicht beachtet werden?

In diesem Beitrag wollen wir einige Aspekte diskutieren, die unseres Erachtens in der Diskussion um die Didaktik des Programmierens bisher zu wenig Beachtung finden.

2 Gibt es überhaupt ein Problem?

Es gibt zwar keine Trendstudien über Lerninhalte und Ergebnisse von Programmierkursen, aber es gibt eine Reihe von internationalen Studien, die die theoretischen Kenntnisse und praktischen Fertigkeiten in der Programmierung näher untersucht haben. Zusammen genommen ergibt sich dabei ein eher düsteres Bild. Allen Studien, die im Folgenden vorgestellt werden, liegen objektorientierte Programmiersprachen zu Grunde. Doch geht es in allen Studien um das Programmieren im Allgemeinen, der spezifische Effekt der Objektorientierung wurde nicht untersucht.

In einer Studie von McCracken et al. wurden 216 Studierende im 2. Semester von 4 Universitäten in unterschiedlichen Ländern, auf ihre praktischen Fertigkeiten in der Programmierung getestet [Mc01]. Die Aufgabe bestand darin, einen einfachen Text-basierten Rechner zu implementieren. Von 110 erreichbaren Punkten, wurden im Schnitt nur 22,89 Punkte erreicht! In ihrer Analyse führen die Verfasser der Studie das schlechte Ergebnis auf allgemeine Defizite im Problemlösen zurück.

Lister et al. hypothetisierten, dass diese Probleme eher prinzipieller Natur sind [Li04]. Die Studierenden hätten möglicherweise Schwierigkeiten, grundlegende Tätigkeiten systematisch auszuführen, wie z. B. die Ausführung von Kod zu verfolgen. Dazu wurden 556 Studierenden von 12 unterschiedlichen Universitäten, 12 Multiple-Choice Fragen gestellt. Alle TeilnehmerInnen hatten mindestens einen Programmierungskurs abgeschlossen. In jeder Aufgabe wurde ein einfaches Kodfragment mit 4–5 möglichen Antworten präsentiert. Das Ergebnis dieser Studie fiel wesentlich besser aus (siehe Tabelle 1), als das von McCracken et al. Allerdings waren die Aufgaben wesentlich einfacher. Grundlegende Probleme mit der Verfolgung von (objektorientiertem) Kod werden z. B. auch von Ragonis und Ben Ari beschrieben [RBA05b].

Richtige Antworten	Anzahl Studierende
10–12	27%
8–9	24%
5–7	25%
0–4	23%

Tabelle 1: Ergebnisse der Studie von Lister et al.

In weiteren Studien haben Dehnadi und Bornat und Ma et al. untersucht, welches Verständnis ProgrammieranfängerInnen für das grundlegende Konzept der Zuweisung haben [DB06, Ma07]. Diese Studien zeigen, dass AnfängerInnen vielen unterschiedlichen Erklärungsmodellen folgen. Es zeigt sich auch, dass das Verständnis für das Konzept der Zuweisung signifikant schwieriger wird, wenn Referenzen ins Spiel kommen (siehe Tabelle 2). Für die Auswertung der Resultate wurden die TeilnehmerInnen in drei Gruppen unterteilt: (1) Studierende, deren Antworten konsistent angemessenen Erklärungsmodellen

entsprechen, (2) Studierende, deren Antworten konsistent unangemessenen Erklärungsmodellen entsprechen und (3) Studierende mit inkonsistenten Erklärungsmodellen. Für Zuweisungen ohne Referenzen ergibt sich ein erwartetes Bild. Die Studierenden mit konsistent angemessenen Modellen, erzielen im Schnitt die besten Resultate in der Abschlussprüfung. Die Studierenden mit konsistent unangemessenen Modellen schneiden im Schnitt am schlechtesten ab, und diejenigen mit inkonsistenten Modellen liegen dazwischen. Für Zuweisungen mit Referenzen ergibt sich jedoch ein anderes Bild. Die Studierenden mit konsistent angemessenen Modellen schneiden zwar immer noch im Schnitt am besten ab, aber die inkonsistente Gruppe schneidet am schlechtesten ab. Des Weiteren gibt es aus der Gruppe mit konsistent unangemessenen Modellen viele, die in der Abschlussprüfung sehr gut abschneiden.

Erklärungsmodelle	ohne Referenzen	mit Referenzen
konsistent angemessen	63	17
inkonsistent	23	41
konsistent unangemessen	14	42

Tabelle 2: Erklärungsmodelle für Zuweisungen: Kategorien in % der ProgrammieranfängerInnen.

Die bisher diskutierten Studien beziehen sich alle auf Studierende im Grundstudium, wie auch die überwiegende Mehrzahl aller veröffentlichten Studien. Wie aber steht es um die Leistungen der AbsolventInnen? Dieser Frage sind Eckerdal et al. nachgegangen [Ec06]. In dieser Studie werden Designfähigkeiten untersucht. Studierende in ihrem letzten Semester sollen einen Softwareentwurf für einen „Superwecker“ erstellen. Für diese Studie wurden Daten von über 100 Entwürfen von einer Vielzahl unterschiedlicher Universitäten ausgewertet. Die Autoren kommen zu dem beunruhigenden Ergebnis, dass nur etwa 9% aller Entwürfe zumindest partiell korrekt sind. Weitere 29% lassen Ansätze erkennen und die restlichen 62% kommen kaum über eine Umformulierung des Problemes hinaus.

Zusammenfassend ergibt sich das Bild, dass die Studierenden tatsächlich, zum Teil grundlegende, theoretische und praktische Defizite haben, die aber scheinbar nicht von den gängigen Leistungsnachweisen aufgefangen werden. Also, um auf die Ausgangsfrage zurückzukommen, machen wir zumindest in den Prüfungen etwas falsch, da die Studierenden weniger können und wissen, als wir glauben.

3 Ist die Objektorientierung der richtige Start?

Die Objektorientierung ist das zur Zeit vorherrschende Programmierparadigma. Es scheint daher auf den ersten Blick angebracht, grundlegenden Konzepte, wie z. B. Objekte und Klassen, so früh wie möglich zu behandeln, damit sie ausreichend geübt und vertieft werden können. Die FürsprecherInnen einer frühen Einführung der Objektorientierung behaupten, dass die Probleme mit objects-first oder -early Ansätzen nicht an der Objektorientierung an sich liegen, sondern an einem Mangel an geeigneten Werkzeugen und pädagogischer Erfahrung [Kö03, Gr06]. Diese Auffassung ist aber umstritten [Br04, Fe04]. Unumstritten ist hingegen die Auffassung, dass eine Umstellung zu einem objekt-

orientierten Ansatz mehr bedeutet, als nur die Programmiersprache zu wechseln. Das objektorientierte Programmieren verlangt nicht nur eine andere Art Probleme zu lösen, sondern auch eine andere Art des Unterrichts. Konnten im prozeduralen Ansatz die Konzepte noch Schritt für Schritt aufeinander aufbauend eingeführt werden, ist das im objektorientierten Ansatz nicht mehr möglich, da die grundlegenden Konzepte so eng miteinander verwoben sind (siehe Abbildung 3). Wir könnten also schließen, dass die Objektorientierung, wenn auch nicht prinzipiell schwieriger zu lernen, sicher schwieriger zu lehren ist.

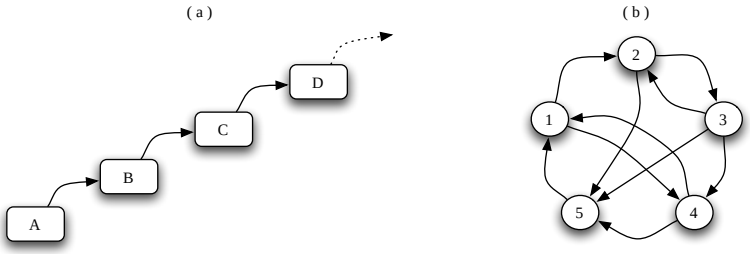


Abbildung 1: Schematische Abhängigkeiten grundlegender Konzepte in der prozeduralen (a) und objektorientierten (b) Programmierung.

Die Wahl des Paradigma beeinflusst welche Aspekte wie schnell gelernt oder verstanden werden. Wiedenbeck et al. [Wi99] haben z. B. untersucht inwiefern sich das Kodverständnis von AnfängerInnen im prozeduralen Paradigma von dem von AnfängerInnen im objektorientierten Paradigma unterscheidet. Für kleine Programme (um 20 Zeilen) ergaben sich nur kleine Unterschiede zwischen beiden Gruppen. Die objektorientierte Gruppe hatte sogar leichte Vorteile in Fragen, die sich auf die Funktionalität der Programme bezogen. Für größere Programme (um 150 Zeilen) hingegen, war die prozedurale Gruppe in allen Fragetypen überlegen. Die Verfasser der Studien führen das auf die steilere Lernkurve[1] für die Objektorientierung zurück.

4 Die Rahmenbedingungen haben sich geändert

Die Rahmenbedingungen für das Informatikstudium sind nicht mehr dieselben wie noch vor 7-8 Jahren. Mit dem Rückgang des Interesses für ein Informatikstudium, haben sich scheinbar auch die Qualifikationen und Einstellungen der StudienanfängerInnen geändert[2]. Immer mehr StudienanfängerInnen sind mit der Anwendung eines Computers vertraut und haben Programmiererfahrung. Trotzdem werden keine Informatik-relevanten Kenntnisse vorausgesetzt. Die Schere zwischen den nominellen Anforderungen an die Studierenden und deren tatsächlichen Vorwissen wird dadurch immer größer (siehe Abbildung 2). Etwa zeitgleich ist auf breiter Front der Übergang zum objektorientierten Paradigma vollzogen

[1] Im Sinne von aufwändiger, schwieriger oder mühsamer zu Lernen.
[2] Dies ist ein wahrgenommener zeitlicher Zusammenhang, der in Wirklichkeit möglicherweise gar nicht besteht. Die Vermutung einer Kausalität ist Spekulation.

worden. Dies führte zu einer größeren Menge an Stoff für die Einführungsveranstaltungen in die Programmierung, welche, gemessen an den nominellen Anforderungen, also schwieriger geworden sind. Trotz dieser Veränderungen sehen unsere Programmierveranstaltungen im wesentlichen noch genau aus wie vor 10 oder mehr Jahren.

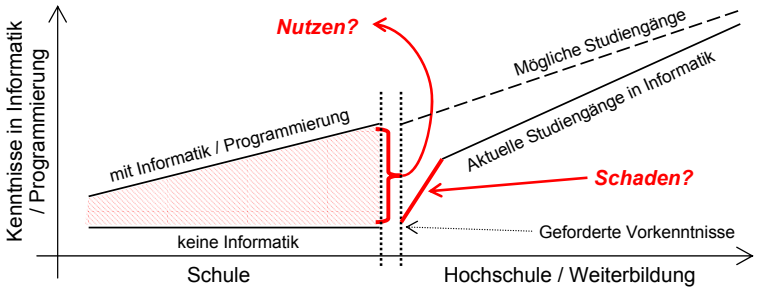


Abbildung 2: Anbindung der Schulinformatik an die Hochschulinformatik. Die zu bewältigende Stoffmenge am Anfang des Studiums ist (unnötig?) hoch.

Als nächste Änderung wird sich der Bologna-Prozess bemerkbar machen. Ein wichtiges Ziel diese Prozesses ist Definition von operationalisierbaren Lernzielen (Lernergebnisse), die sich an den Studierenden orientieren. Dieses an sich lobenswerte Ziel ist ein zweischneidiges Schwert. Die Instrumentalisierung von Lerninhalten kann leicht zu einer Fragmentisierung und Marginalisierung von Wissen und Fertigkeiten führen, insbesondere wenn die Lernergebnisse, wie vorgeschlagen, studienbegleitend geprüft werden. Werden Lernergebnisse isoliert betrachtet und geprüft, könnte das Wissen um das „Grosse Ganze“ auf der Strecke bleiben, und alles was nicht explizit als Lernergebnis definiert ist, irrelevant werden.

5 Eine Reihe von Überlegungen

Das Programmieren ist eine komplexe Fertigkeit. Nach Winslow [Wi96] ist der Weg zum Expertentum lang (etwa 10 Jahre). Des Weiteren korreliert die Fähigkeit Programme zu verstehen nur schwach mit der, Programme auch tatsächlich zu schreiben. Beides muss also explizit gelehrt und gelernt werden.

Mir scheint es, dass wir es unseren Studierenden unnötig schwer machen. Trotz der in den Abschnitten 2–4 geschilderten Umstände erwarten wir, dass sie schon nach dem ersten Semester „vernünftig“ programmieren können. Ist diese Erwartungshaltung realistisch oder gar angemessen? Ich würde sagen, eher nein.

In den folgenden Abschnitten wollen wir einige Faktoren diskutieren, die in der Diskussion um eine Didaktik des Programmierens mehr Beachtung finden sollten.

5.1 Kognitive Belastung

Im Gegensatz zum menschlichen Langzeitgedächtnis, ist das menschliche Kurzzeit- oder Arbeitsgedächtnis in seiner Kapazität sehr begrenzt [Co01]. In unserem Arbeitsgedächtnis werden Informationseinheiten[3] ständig manipuliert und umorganisiert um ihnen einen „Sinn“ zu geben. Die sich ergebenden Strukturen werden dabei fortlaufend (bewusst und unbewusst) mit dem verglichen, was wir bereits wissen, d. h. im Langzeitgedächtnis gespeichert haben. Dieses fortwährende „Testen“ kann zur Konstruktion neuen Wissens führen, welches dann in sogenannten Schemata im Langzeitgedächtnis gespeichert wird. Wir lernen. Ist das Arbeitsgedächtnis aber zu sehr belastet, reicht dessen Kapazität nicht mehr aus um neue Schemata zu bilden, d. h. das Lernen wird erschwert. Die Theorie der kognitiven Belastung (Cognitive Load Theory – CLT) beschreibt Faktoren, die die kognitive Belastung beeinflussen und welche Effekte dies auf das Lernen hat [SMP98, MS05].

Das Arbeitsgedächtnis ist zwar sehr begrenzt, doch können die Informationseinheiten, die bearbeitet werden im Prinzip beliebig komplex sein. Sie müssen jedoch für den Betrachter als Einheiten, sogenannte „chunks“, angesehen werden [Go01]. Was für erfahrene ProgrammiererInnen eine Einheit bildet (z. B. die Abstraktion „Rekursion“), kann aber für AnfängerInnen bereits zur kognitiven Überbelastung führen.

Die Programmierung ist eine sehr komplexe Fähigkeit. Des Weiteren ist die Programmierung ein vollständig abstraktes Gebiet, in dem alle Regeln künstlicher Natur sind (wie etwa in höherer Mathematik). AnfängerInnen haben daher kein Vorwissen oder schlüssiges Erklärungsmodell auf das sie „aufbauen“ können. Gleichzeitig verzeihen Compiler auch nicht die geringsten Fehler. Aber anstatt den Studierenden das Lernen leichter zu machen, sind die Lerninhalte eher umfangreicher und komplexer geworden. Die CLT kann uns vielleicht dabei helfen, die sich daraus ergebenden Probleme besser zu verstehen [SDT03].

5.2 AnfängerInnen arbeiten nicht wie ExpertInnen

Erfahrene ProgrammiererInnen kennen eine große Anzahl von sogenannten „Programmierplänen“ und wissen wie diese instantiiert und zu komplexen Programmen zusammengesetzt werden. AnfängerInnen kennen weniger Pläne, die, im Gegensatz zu den Plänen von ExpertInnen, auch sprachspezifisch sind. D. h. AnfängerInnen können ihre Pläne nicht so leicht generalisieren wie ExpertInnen. Die Pläne der AnfängerInnen sind aber dennoch ausreichend, um einfache Probleme zu lösen. Das Hauptproblem von AnfängerInnen liegt jedoch nicht in der Anzahl der bekannten Pläne, sondern in ihren begrenzten Fähigkeiten diese zu komplexen Programmen zusammenzusetzen [SS86].

Die Objektorientierung scheint dieses Problem noch zu verstärken, da Pläne und objektorientierte Strukturen im wesentlichen orthogonal sind. In einem Plan können viele Objekte beteiligt sein und ein einzelnes Objekt kann wiederum in vielen unterschiedlichen Plänen vorkommen [Ri95]. Eine gute Übersicht über diese und weitere kognitive und psychologische Aspekte des Programmierens findet sich in [RRR03] und [Dé02].

[3] Diese können externer (mit den Sinnen wahrgenommen) oder interner (unsere eigenen „Gedanken“) Natur sein.

Für einfache und vertraute Probleme kennen ExpertInnen alle notwendigen Pläne und wissen wie diese instantiiert werden müssen um es, mit Hilfe einer top-down forward expansion Stategie, zu lösen. Falls diese Strategie nicht zum Erfolg führt, wird zu einer bottom-up backward solution Strategie zurückgegriffen [Ri91]. Da AnfängerInnen weniger (generelle) Pläne kennen und, darüber hinaus, Schwierigkeiten haben Pläne zu komplexen Programmen zusammenzusetzen, greifen sie im wesentlichen immer auf die letztere Strategie zurück.

Auf eine Unterrichtssituation übertragen, ergibt sich aus der obigen Diskussion, dass sich Lehrkräfte und Studierende auf unterschiedlichen „Ebenen“ befinden (siehe Abbildung 3) und deshalb möglicherweise aneinander vorbei reden. Dieses Problem reflektiert sich auch im Inhalt von Lehrbücher der Programmierung, welche oft zu wenig Konzept- und Prozessorientiert sind [LW07].

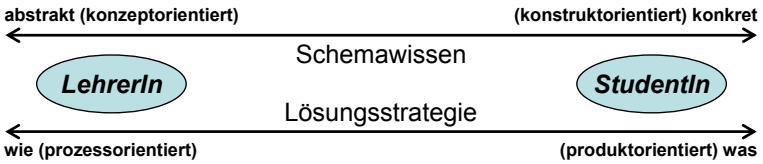


Abbildung 3: Gedankliche „Ebenen“ beim Lösen von Programmierproblemen.

5.3 Einfachere Sprachen und Werkzeuge

Programmiersprachen und -umgebungen werden häufig aufgrund ihrer Relevanz für das spätere Arbeitsleben gewählt. Didaktische Gesichtspunkte müssen oft zurück stehen. Unter Berücksichtigung der kognitiven Belastung (siehe Abschnitt 5.1) erweisen wir den Studierenden damit möglicherweise einen Bärendienst. Die Syntax und Semantik gebräuchlicher objektorientierter Programmiersprachen ist sehr komplex, erheblich umfangreicher und komplexer als die von z. B. Pascal. Hinzu kommen noch umfangreiche Bibliotheken. Trotz dieser Faktoren sind Java und C++ die dominierenden Programmiersprachen für die Einführungsveranstaltungen, obwohl grundlegende objektorientierte Konzepte in vielen Fällen gar nicht behandelt werden [Da05]. Erfahrungen und Studien zeigen, dass „einfachere“ Sprachen nicht nur das Lernen erleichtern, sondern auch eine gute Grundlage für „komplexe“ Sprachen sind [Fe04, MPS06].

Ähnliches gilt für Programmierumgebungen. Obwohl es Programmierumgebungen gibt, die speziell für AnfängerInnen entwickelt wurden, wie z. B. BlueJ[4], kommen häufig Umgebungen zum Einsatz, die für AnfängerInnen viel zu komplex und anspruchsvoll sind, wie z. B. Eclipse[5]. Dies führt unweigerlich zu einer höheren kognitiven Belastung.

[4] `http://www.bluej.org/`
[5] `http://www.eclipse.org/`

5.4 Vorsicht mit Notation und Terminologie

Konzepte müssen sehr genau erklärt werden. Viele hartnäckige Fehlauffassungen von Studierenden könnten von ungenauen Erklärungen herrühren, wie z. B. der verbreiteten Definition „a class is a set of objects“. Diese aus mengentheoretischen Gesichtspunkten korrekte Definition, kann Studierende dazu verleiten Objektinstanzen als Klassen anzusehen, wie z. B. eine Mannschaft in einem Programm welches Spielergebnisse verwalten soll [TH04].

Des Weiteren ist es verwirrend, dass es für dasselbe Konzept unterschiedliche Begriffe gibt, die je nach Kontext variieren, siehe Tabelle 3. Um die Sache noch verwirrender zu machen haben viele Programmiersprachen auch noch von diesen Begriffen abweichende Syntax, wie z. B. in Java, wo die Vererbung mit „extends“ angezeigt wird. Es ist daher wichtig Konzepte, Notationen und Terminologi sorgfältig zu trennen und zu definieren und dann auch konsistent zu verwenden.

	UML	Java	C++	Weitere
Attribut	attribute	(field) variable	data member / (instance/reference) variable	–
Metode	operation	method	member function	–
Eigenschaft	feature	member	member	property
Metodenimplementierung	method	method definition (body)	(member) function definition	method implementation
Subklasse	child	subclass	derived or child class	–
Superklasse	parent	superclass	base class	–
Vererbung	generalization	inheritance	inheritance	specialization

Tabelle 3: Beispiele für die Fülle von Begriffen für objektorientierte Konzepte.

5.5 Vorsicht mit Beispielen

Ein weiteres Problem ist die Bereitstellung von „exemplarischen[6]“ Beispielen. Gängige Beispiele einfach in eine objektorientierte Programmiersprache zu „übersetzen“ kann keine Lösung sein, da solche Beispiele dann oft nicht mehr im Einklang mit den objektorientierten Prinzipien stehen, die wir eigentlich lehren wollen [Bö07, CA02, We01]. Dieses Problem wird jedoch oft unterschätzt, oder erst gar nicht als solches identifiziert.

Beispiele nehmen eine zentrale Rolle in den frühen Phasen des Wissenserwerbs ein [Va96] und werden von den Lernenden als Vorlagen für ihre eigene Arbeit übernommen. Beispiele sollten daher generalisierbar sein und mit den Lernzielen übereinstimmende Eigenschaften haben. Ansonsten wird es für die Lernenden schwierig Muster zu erkennen und zufällige oberflächliche Eigenschaften eines Beispiels von denen zu unterscheiden, die prinzipielle Bedeutung haben. Werden kontinuierlich exemplarische Beispiele verwendet, wird es den Lernenden leichter gemacht, mit der Zeit erwünschte von unerwünschten Eigenschaften zu unterscheiden.

[6] Im Sinne von beispielhaft.

Mit sorgsam abgestimmten Beispielen kann das Risiko von Fehlinterpretationen und Fehlschlüssen verringert werden (siehe auch Abschnitt 5.6.2). In vielen Java-Lehrbüchern werden z. B. die Klassen „Math“ oder „String“ als Beispiele genutzt um das Klassenkonzept zu erläutern. Beide Beispiele widersprechen jedoch wichtigen grundlegenden Eigenschaften von „normalen“ Objekten und Klassen. Die Klasse „Math“ ist statisch und ihre Anwendung unterscheidet sich dadurch markant vom „Standardfall“. „String“-Objekte verhalten sich zumindest syntaktisch wie der „Standardfall“. Allerdings sind sie unveränderlich, d. h. sie widersprechen der Vorstellung, dass sich der (interne) Zustand eines Objektes, in Abhängigkeit der empfangenen Nachrichten, ändert.

Wenn wir wollen, dass die Lernenden mit der Zeit Gemeinsamkeiten und strukturelle Muster erkennen, sollten wir unsere Erklärungen und Beispiele nicht mit Hilfe „Spezialfällen“ beginnen.

5.6 Lerne deine Studierenden kennen

Für einen erfolgreichen Unterricht sollten wir die Studierenden dort abholen, wo sie mit ihrem Vorwissen stehen. Mit den in Abschnitt 4 beschriebenen Rahmenbedingungen, kann es allerdings schwierig werden einen gemeinsamen Ansatzpunkt für diese „Abholung“ zu definieren. Aber nur wenn wir uns informieren, können wir die Zahl der Studierenden minimieren, die entweder unterfordert oder überfordert werden. Darüber hinaus sollten wir uns auch über andere Faktoren Gedanken machen, die für den Erfolg im Programmieren Lernen von Bedeutung sind.

5.6.1 Erfolgsfaktoren

Die Forschung ist sich im Großen und Ganzen einig, dass gute Kenntnisse in Mathematik und Programmierung, ein allgemein hoher Notendurchschnitt, sowie Selbsteinschätzung (self-efficacy[7]) positiv mit guten Leistungen in der Einführung in die Programmierung korrelieren, siehe z. B. [BR06, Fi05, Wi02]. Intensives Computer-spielen scheint den Erfolg jedoch negativ zu beeinflussen [Wi02]. Die Auswirkungen dieser Faktoren sind aber nicht immer eindeutig, da sich viele Faktoren gegenseitig beeinflussen. So kommt z. B. Wiedenbeck [Wi05] zum Schluss, dass (übertriebene) Selbsteinschätzung auch negative Auswirkungen auf das Kursresultat haben kann. Des Weiteren scheinen Vorkenntnisse vorwiegend über den „Umweg“ der Selbsteinschätzung zu wirken, da die positiven Effekte von Vorkenntnissen schnell nachlassen [HW06, Wi05].

5.6.2 Schwierigkeiten

Es gibt eine Reihe von Forschungsergebnissen über spezifische Schwierigkeiten oder sogenannte „misconceptions“ von ProgrammieranfängerInnen, siehe z. B. [Cl04, Fl00, HGW97, Ja06, Le05, MR02, RBA05a, RHG06, SS86] für einen Überblick. Obwohl dieses „Wissen“ sehr wichtig für die Entwicklung von Konzepten und Materialien für den Unterricht

[7] „Perceived Self-Efficacy: People’s beliefs about their capabilities to produce effects.“ [Ba94]

ist, werden diese Ergebnisse im Einzelnen jedoch nur selten in spezifische konkrete Vorschläge für Verbesserungen umgesetzt. Es werden jedoch eine Menge von „Paketlösungen“ angeboten, die gleich ein Reihe von Schwierigkeiten angehen, wie z. B. Programmbibliotheken, Programmierumgebungen, Mikrowelten etc., auf die ich hier im Einzelnen nicht eingehen kann.

Viele Schwierigkeiten werden damit erklärt, dass den Studierenden ein konzeptuelles Modell fehlt, welches ihnen erlaubt das Verhalten eines Programmes bei der Ausführung vorherzusagen [BAY06, Bö05, BOM99, MR02, RBA05b]. Das gilt auch für einen Teil der in Abschnitt 2 beschriebenen Studien. Viele Verfasser, inklusive des Autors, vertreten daher die Auffassung, dass solche Modelle explizit gelehrt werden sollten. Intuitive Modelle können sicher dabei helfen die kognitive Belastung zu senken (siehe Abschnitt 5.1) und den Schemaerwerb zu erleichtern (siehe Abschnitt 5.2).

6 Zusammenfassung und Ausblick

Zusammenfassend ergibt sich ein recht düsteres Bild für die allgemeine Praxis im Programmierunterricht. Wir müssen uns sogar etwas provokativ fragen, wie wir die Lehrkräfte für die Programmierung im Schulunterricht angemessen vorbereiten wollen, wenn uns die Zeit schon für Studierende mit Hauptfach Informatik nicht ausreicht?

Dieser Beitrag zeigt einige Aspekte auf, die uns helfen können die Situation besser zu verstehen und langfristig auch zu verbessern.

Literaturverzeichnis

[Ba94] A. Bandura. Self-efficacy. In V.S. Ramachaudran, Hrsg., Encyclopedia of human behavior, Vol. 4, Seiten 71–81. 1994.

[BAY06] M. Ben-Ari und T. Yeshno. Conceptual models of software artifacts. Interacting with Computers, 18(6):1336–1350, 2006.

[Bö07] J. Börstler, M. Nordström, L. Kallin Westin, J.E. Moström und J. Eliasson. Transitioning to OOP—A Never Ending Story. In M. Kölling, J. Bennedsen und M.E. Caspersen, Hrsg., *Scandinavian Pedagogy of Programming*. 2007. to appear.

[Bö05] J. Börstler. Improving CRC-card role-play with role-play diagrams. Conference Companion – 20th Annual Conference on Object Oriented Programming Systems Languages and Applications, Seiten 356–364, 2005.

[BR06] S. Bergin und R. Reilly. Predicting introductory programming performance: A multi-institutional multivariate study. Computer Science Education, 16(4):303–323, 2006.

[Br04] Kim Bruce. Controversy on How to Teach CS1: A Discussion on the SIGCSE-members Mailing List. ACM SIGCSE Bulletin, 36(4):29–35, 2004.

[CA02] CACM Forum. ‘Hello, World’ Gets Mixed Greetings. Communications of the ACM, 45(2):11–15, 2002.

[Cl04] Michael Clancey. Misconceptions and Attitudes that Infere with Learning to Program. In Sally Fincher und Marian Petre, Hrsg., Computer Science Education Research, Seiten 85–100. Taylor & Francis, Lisse, The Netherlands, 2004.

[Co01] Nelson Cowan. The magical number 4 in short-term memory: A reconsideration of mental storage capacity. Behavioral and Brain Sciences, 24(1):87–114, 2001.

[Da05] Nell Dale. Content and emphasis in CS1. ACM SIGCSE Bulletin, 37(4):69–73, 2005.

[DB06] S. Dehnadi und R. Bornat. The camel has two humps, Working Paper. http://www.cs.mdx.ac.uk/research/PhDArea/saeed/, 2006.

[BOM99] B. du Boulay, T. O'Shea und J. Monk. The black box inside the glass box: presenting computing concepts to novices. International Journal of Human-Computer Studies, 51(2):265–277, 1999.

[Dé02] F. Détienne. Software Design – Cognitive Aspects. Springer, London, UK, 2002.

[Ec06] A. Eckerdal, R. McCartney, J.E. Moström, M. Ratcliffe und C. Zander. Categorizing student software designs: Methods, results, and implications. Computer Science Education, 16(3):197–209, 2006.

[Fi05] S. Fincher, B. Baker, I. Box, Q. Cutts, M. de Raadt, P. Haden, J. Hamer, M. Hamilton, R. Lister, M. Petre, A. Simon, Robins, K. Sutton, D. Tolhurst und J. Tutty. Programmed to succeed?: A multi-national, multi-institutional study of introductory programming courses. Technical Report 01-05, Computing Laboratory, University of Kent, Canterbury, UK, 2005.

[Fe04] M. Felleisen, R.B. Findler, M. Flatt und S. Krishnamurthi. The TeachScheme! Project: Computing and Programming for Every Student. Computer Science Education, 14(1):55–77, 2004.

[Fl00] Ann E. Fleury. Programming in Java: Student-Constructed Rules. In Proc. of the 31th Technical Symposium on Computer Science Education, Seiten 197–201, 2000.

[Gr06] R. Granerud, J. Kaasbøll, R. Borge, C. Holmboe und O. Smørdal. Childrens understanding of object-orientation. In Annita Fjuk, Amela Karahasanovic und Jens Kaasbøll, Hrsg., *Comprehensive object-oriented learning: The learner's perspective*, Seiten 27–47. Informing Science Press, Santa Rosa, CA, USA, 2006.

[Go01] F. Gobet, PC Lane, S. Croker, PC Cheng, G. Jones, I. Oliver und JM Pine. Chunking mechanisms in human learning. Trends in Cognitive Sciences, 5(6):236–243, 2001.

[HGW97] Simon Holland, Robert Griffiths und Mark Woodman. Avoiding Object Misconceptions. In Proceedings of the 28th Technical Symposium on Computer Science Education, Seiten 131–134, 1997.

[HW06] E. Holden und E. Weeden. What Makes Valuable Pre-experience for Students Entering Programming Courses? Issues in Informing Science and Information Technology, 3:279–293, 2006.

[Ja06] Matthew C. Jadud. An Exploration of Novice Compilation Behaviour in BlueJ. Dissertation, University of Kent, Canterbury, UK, 2006.

[Kö03] M. Kölling, B. Quig, A. Patterson und J. Rosenberg. The BlueJ System and its Pedagogy. Computer Science Education, 13(4):249–268, 2003.

[Le05] G. Lewandowski, A. Gutschow, R. McCartney, K. Sanders und D. Shinners-Kennedy. What novice programmers don't know. In Proceedings of the First International Computing Education Research Workshop, Seiten 1–12, 2005.

[Li04] R. Lister, O. Seppälä, B. Simon, L. Thomas, E.S. Adams, S. Fitzgerald, W. Fone, J. Hamer, M. Lindholm, R. McCartney et al. A multi-national study of reading and tracing skills in novice programmers. ACM SIGCSE Bulletin, 36(4):119–150, 2004.

[LW07] J.M.C. Lin und C.C. Wu. Suggestions for content selection and presentation in high school computer textbooks. Computers & Education, 48(3):508–521, 2007.

[Ma07] L. Ma, J. Ferguson, M. Roper und M. Wood. Investigating the Viability of Mental Models Held by Novice Programmers. In Proceedings of the 38th Technical Symposium on Computer Science Education, Seiten 499–503, 2007.

[MPS06] L. Mannila, M. Peltomäki und T. Salakoski. What about a simple language? Analyzing the difficulties in learning to program. Computer Science Education, 16(3):211–227, 2006.

[MR02] I. Milne und G. Rowe. Difficulties in Learning and Teaching Programming—Views of Students and Tutors. Education and Information Technologies, 7(1):55–66, 2002.

[Mc01] M. McCracken, T. Wilusz, V. Almstrum, D. Diaz, M. Guzdial, D. Hagan, Y.B.D. Kolikant, C. Laxer, L. Thomas und I. Utting. A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. ACM SIGCSE Bulletin, 33(4):125–180, 2001.

[RBA05a] N. Ragonis und M. Ben-Ari. A long-term investigation of the comprehension of OOP concepts by novices. Computer Science Education, 15(3):203–221, 2005.

[RBA05b] N. Ragonis und M. Ben-Ari. On understanding the statics and dynamics of object-oriented programs. In Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education, Seiten 226–230, 2005.

[RHG06] A. Robins, P. Haden und S. Garner. Problem distributions in a CS1 course. In Proc. of the 8th Austalian Conference on Computing Education, Seiten 165–173, 2006.

[Ri91] R.S. Rist. Knowledge Creation and Retrieval in Program Design: A Comparison of Novice and intermediate Student Programmers. Human-Computer Interaction, 6(1):1–46, 1991.

[Ri95] R.S. Rist. Program structure and design. Cognitive Science, 19(4):507–561, 1995.

[RRR03] A. Robins, J. Rountree und N. Rountree. Learning and Teaching Programming: A Review and Discussion. Computer Science Education, 13(2):137–172, 2003.

[SDT03] D. Shaffer, W. Doube und J. Tuovinen. Applying Cognitive load theory to computer science education. In Proc. Joint Conference EASE & PPiG, Seiten 333–346, 2003.

[SS86] James C. Spohrer und Elliot Soloway. Novice mistakes: are the folk wisdoms correct? Communications of the ACM, 29(7):624–632, 1986.

[SMP98] J. Sweller, J.J.G. van Merrienboer und F.G.W.C. Paas. Cognitive Architecture and Instructional Design. Educational Psychology Review, 10(3):251–296, 1998.

[TH04] Maryana Teif und Orit Hazzan. Junior High School Students' Perceptions of Object Oriented Concepts. In 8th Workshop on Pedagogies and Tools for the Teaching and Learning of Object Oriented Concepts, 2004. `http://www.cs.umu.se/~jubo/Meetings/ECOOP04/Submissions/TeifHazzan.pdf`.

[Va96] K. VanLehn. Cognitive Skill Acquisition. Annual Review of Psychology, 47:513–539, 1996.

[MS05] J.J.G. van Merriënboer und J. Sweller. Cognitive Load Theory and Complex Learning: Recent Developments and Future Directions. Educational Psychology Review, 17(2):147–177, 2005.

[We01] R. Westfall. 'Hello, World' Considered Harmful. Communications of the ACM, 44(10):129–130, 2001.

[Wi05] Susan Wiedenbeck. Factors affecting the success of non-majors in learning to program. In Proceedings of the First International Computing Education Research Workshop, Seiten 13–24, 2005.

[Wi02] Brenda Cantwell Wilson. A Study of Factors Promoting Success in Computer Science Including Gender Differences. Computer Science Education, 12(1):141–164, 2002.

[Wi96] Leon E. Winslow. Programming pedagogy—a psychological overview. ACM SIGCSE Bulletin, 28(3):17–22, 1996.

[Wi99] S. Wiedenbeck, V. Ramalingam, S. Sarasamma und C.L. Corritore. A comparison of the comprehension of object-oriented and procedural programs by novice programmers. Interacting with Computers, 11(3):255–282, 1999.

Bildungsstandards Informatik – von Wünschen zu Maßstäben für eine informatische Bildung

Steffen Friedrich
TU Dresden, Fakultät Informatik
AG Didaktik der Informatik
01062 Dresden
steffen.friedrich@tu-dresden.de

Hermann Puhlmann
Leibniz-Gymnasium Altdorf
Fischbacher Straße 23
90518 Altdorf
hermann@puhlmann.name

Abstract: Die Kompetenzen einer informatischen Bildung der Schülerinnen und Schüler sind hinsichtlich der notwendigen Bestandteile noch nicht ausreichend genug charakterisiert. In diesem Kontext haben sich mit Unterstützung der Gesellschaft für Informatik e.V. Fachleute aus Schulen und Hochschulen zusammengefunden, um Standards für eine informatische Bildung auszuarbeiten.
Der Beitrag stellt die Genese und den Stand der Arbeiten dar. In ihm werden Mindeststandards für einen mittleren Bildungsabschluss vorgeschlagen. Durch die Differenzierung in die Klassenstufen 5-7 und 8-10 wird eine sofortige und unmittelbare Nutzung im Unterricht unterstützt.

1 „Informatische Bildung“ – oder was man darunter versteht

Die Bildungsmisere in Deutschland ist inzwischen ein Thema geworden, mit dem es sich ganz gut in der Öffentlichkeit punkten lässt. Es scheint fast akzeptiert, dass deutsche Schülerinnen und Schüler in internationalen Vergleichen nur im Mittelfeld landen, dass die Wirtschaft insbesondere im Bereich der Naturwissenschaften und der Technik bereits jetzt Nachwuchs intensiv sucht. Was kann man aktuell beobachten?

PISA-Aufgabe: [PI03]

„Du hast Änderungen an einem Textdokument vorgenommen und möchtest sowohl die geänderte Datei speichern als auch die ursprüngliche Version des Textes behalten. Was tust du?
a) Ich wähle in der Textverarbeitung den Menüpunkt "Änderungen in einer neuen Datei speichern“.
b) Ich rufe in der Textverarbeitung den Menüpunkt "Versionsvergleich“ auf.
c) Ich speichere die geänderte Datei unter einem neuen Namen.
d) Ich verschiebe die Datei vor dem Speichern in ein anderes Verzeichnis.“

Feststellung IT-Wirtschaft: [BK05]

> „Die Leistungen deutscher Schüler im … mathematischen und naturwissenschaftlichen Bereich sind im internationalen Vergleich drittklassig, das deutsche Bildungssystem ist Mittelmaß. Seit Anfang der 90er Jahre verharrt der Anteil junger, hochqualifizierter Menschen in Deutschland auf einem vergleichbaren Niveau …“

Diese Liste der Beispiele wäre beliebig fortzusetzen. Letztlich wird deutlich, dass nach wie vor eine gesellschaftliche Entwicklung hin zu einer wie auch immer definierten Informations- und Wissensgesellschaft proklamiert und deren ständiger Veränderung auch ein hohes Tempo bescheinigt wird. Die Anpassung der Bildungsangebote in allen Ebenen bleibt insbesondere für Schulen in strukturellen Debatten oder regionalen Unterschieden stecken. Für einen nachhaltigen Fortschritt im Leistungsprofil der Absolventen von Schulen (auch Hochschulen) ist die Struktur der Schularten oder auch die Studienstruktur zweitrangig, wenn die inhaltlichen Anforderungen das notwendige Niveau erreichen. Deshalb geht es primär darum, mit welchem Inhalt unabhängig von Schulstrukturen erreicht wird, dass die Lernenden dann in der Lage sind, die Anforderungen in Ausbildung, Studium und vor allem in der künftigen Arbeitstätigkeit zu bewältigen.
Für die informatische Bildung ist eine solche Betrachtung aktueller denn je, weil dieser Bereich nach wie vor um eine bildungspolitische Anerkennung ringt. So wird informatische Bildung immer noch auf Bedienkompetenzen im Umgang mit Informatiksystemen reduziert. Damit werden diese Fertigkeiten zu wenig von einer informatischen Bildung unterschieden. Das schlägt sich dann beispielsweise in Modellen empirischer Untersuchungen nieder, die Bedienfertigkeiten als computerbezogene Kompetenzen deklarieren. [SE04] Es ist in diesem Zusammenhang anerkennenswert, dass diese Kompetenzen überhaupt in Studien und Untersuchungen einbezogen werden und dafür nach Basismodellen gesucht wird. Mathematische Kompetenzen werden jedoch auch nicht auf die Kompetenzen zur Nutzung moderner Taschenrechner reduziert.
Die Entwicklung von Bildungskonzepten und Schulfächern war auch in der Vergangenheit von gesellschaftlichen und wirtschaftlichen Entwicklungen beeinflusst. So mussten sich sowohl die klassischen Naturwissenschaften als auch die Geographie ihren Platz in der Schule sukzessive erstreiten. [GO99] Immer haben dabei die gesellschaftlichen und ökonomischen Anforderungen an die Bildung den entscheidenden Durchbruch bewirkt. Es ist an der Zeit, dass wir diesen Veränderungen Rechnung tragen und die Vielgestaltigkeit der informatischen Bildung in Deutschland [WE07] als entwicklungshemmend akzeptieren und geeignete Konzepte vorlegen. Hier hat sich die Gesellschaft für Informatik e.V. in den letzten Jahren Verdienste erworben, weil sie immer wieder mit konkreten Vorschlägen aufgewartet hat [GI00], um informatische Bildung zu befördern. Wenn in einigen Bundesländern Initiativen zum Ausbau informatischer Bildung in den Schulen inzwischen erste Fortschritte zeigen, ist das sicher auch ein Erfolg dieser Bemühungen. Perspektivisch ist es notwendig, die Ergebnisse und Kompetenzen zu präzisieren, die durch informatische Bildung bei Schülerinnen und Schülern erreicht werden müssen, also: Bildungsstandards zur informatischen Bildung vorzulegen.

2 Bildungsstandards – eine vielfältige Diskussion

Spätestens seit der Veröffentlichung der ersten Ergebnisse vergleichender empirischer Studien wie TIMSS oder PISA haben verschiedene Debatten und fast unzählige wissenschaftliche und journalistische Schriften dafür gesorgt, Interesse für Arbeiten zu entwickeln, die unter dem Stichwort "Bildungsstandards" erscheinen. Es ist in der Tat inzwischen ein reichlich diffuses Bild zu Absichten, Teilergebnissen und möglichen Perspektiven entstanden. Standardisierungen in der Schule sind nicht neu. Sie existieren bereits in Form von Lehrplänen, zentralen Prüfungen, äußeren Strukturen von Schulen oder auch in der Verwaltung. "Versteht man unter 'Standards' dauerhafte Lösungen für wiederkehrende Probleme, die auf einem bestimmten Niveau gehalten werden und für einen bestimmten Geltungsbereich bestimmt sind, dann wird damit der Alltag erfasst." [OE04] Bei Beachtung bisheriger Festlegungen für den Unterricht, die man im weiten Sinne einer Standardisierung zuordnen kann, ergeben sich vor allem Fragen nach Neuwert und Alltagstauglichkeit dieser aktuellen Überlegungen zu Bildungsstandards.
Die eigentliche Zielrichtung wird bereits im Klassiker, der "Expertise zur Entwicklung nationaler Bildungsstandards" [KL03], betont, indem auf die Notwendigkeit verwiesen wird, sich weniger an Lehrplänen u.ä. (am "Input"), sondern an den Leistungen der Schule und den Lernergebnissen der Schülerinnen und Schüler (am "Output") zu orientieren. Dies ist vom Ansatz her sofort einleuchtend und verschiebt natürlich bildungspolitische Diskussionen vom Streit um die Inhalte hin zur Betrachtung und Interpretation vorliegender Resultate, die bisher auf der Basis von Unterricht nach Lehrplänen oder Richtlinien und unter vorhandenen Rahmenbedingungen entstanden sind. Der Kern des Ansatzes lag vor allem darin, dass die selbst gesteckten und in Lehrplänen festgehaltenen Bildungsziele häufig nicht erreicht werden und sich die Leistungen von Lernenden stärker als in anderen untersuchten PISA-Staaten unterscheiden. Diese Ergebnisse wurden bereits zu diesem Zeitpunkt als Hinweis darauf interpretiert, dass es wenigstens für die untersuchten Bereiche an Mindeststandards fehlt, die von Schülerinnen und Schülern dieser Altersstufe erreicht werden müssen. [KL03] Gleichzeitig wurde damit auch die Erwartung verbunden, dass eine höhere Zielklarkeit und bessere Chancen zur Überprüfung des Erreichten eine rechtzeitige und zielgenaue Unterstützung ermöglichen. Das bedeutet schließlich, dass Standards einerseits der Orientierung aller Beteiligten (Lehrende, Eltern, Administration) über normativ gesetzte Anforderungen dienen, andererseits eine Basis für Leistungsüberprüfungen darstellen, deren Ebenen und Instrumente sorgsam zu unterscheiden sind. [BL06] Diese beiden Seiten sind es, die manche kontroverse Diskussion befördern. Man könnte zum einen annehmen, dass dies auch durch bereits existierende Festlegungen erreicht wird, und zum anderen, dass eine einseitige Testorientierung eintreten wird.
Unabhängig vom Blickwinkel auf Standardisierungen in der Bildung sollte das Bemühen überwiegen, für guten Unterricht, auf den sich Eltern und Schüler verlassen können, zu sorgen. Das führt zwangsläufig zu einer Orientierung auf Ergebnisse - im Sinne von Kompetenzen - die erst einmal unabhängig davon sind, in welchem Kontext von Schule und Ausbildung diese erreicht werden oder auch erreicht werden sollen. Die Bedeutungsvielfalt des Begriffes 'Kompetenz' macht es erforderlich, darauf zu verweisen, dass im Zusammenhang mit der Erarbeitung von Bildungsstandards (bezugnehmend auf WEINERT [WN99]) diese in der Regel im Sinne pädagogisch - psychologischer Erkenntnisse verstanden werden - als erlernte, anforderungsspezifische Leistungsdispo-

sitionen, die durch kontinuierlichen Aufbau von Wissen und Können in einem Inhalts- und Erfahrungsbereich entwickelt werden. In diesem Verständnis haben Kompetenzmodelle die Aufgabe, die Ziele, die Struktur und die Ergebnisse fachlicher Lernprozesse zu beschreiben und, umgesetzt in Aufgaben und Tests, den Leistungsstand von Schülerinnen und Schülern zu erfassen und darzustellen. Die so formulierte Auffassung zu Kompetenzen als kontextspezifische kognitive Leistungsdispositionen unter Ausschluss motivationaler und affektiver Faktoren bietet der weiteren Bildungsforschung eine gute Arbeitsgrundlage. Sie schafft gleichzeitig eine brauchbare Abgrenzung von in psychologischen Untersuchungen verwendeten generalisierten kognitiven Leistungsdispositionen, die eine nahe inhaltliche Verwandtschaft zu gängigen Definitionen der Intelligenz aufweisen. Die Differenzierung geschieht insbesondere durch den Kontextbezug (d.h. es werden Kompetenzen immer bereichsspezifisch, auf bestimmte Aufgaben und Situationen bezogen, betrachtet), durch die Lernbarkeit (d.h. es sind prinzipiell erlernbare bereichsspezifische Kenntnisse, Fertigkeiten und Strategien) und durch die Binnenstruktur (d.h. es ergeben sich Strukturen aus Anforderungen der Aufgaben). [SC06] Es bleibt ein Arbeitsfeld der Kompetenzdiagnostik, im Rahmen von Schulleistungsuntersuchungen in Form standardisierter Tests Kompetenzen empirisch zu erfassen. Die damit im Zusammenhang stehenden Fragen der theoretischen Modellierung führen zu unterschiedlichen Formen, von denen Kompetenzstrukturmodelle und Kompetenzniveaumodelle näher betrachtet werden sollen.

Kompetenzstrukturmodelle beschreiben die Dimensionen von Kompetenzen, die meist durch faktorenanalytische Untersuchungen gewonnen werden. Durch Zusammenfassung und Interpretation von Messungen zu geeigneten Dimensionen oder durch Ableitung von Erwartungen aus theoretischen Annahmen wird darstellbar, welche Kompetenzen in einem bestimmten Zusammenhang erfasst werden können oder sollen. Bei der Entscheidung für ein bestimmtes Strukturmodell wird schließlich auch die Frage nach dem Grad der Differenzierung spezifischer Kompetenzen zu beantworten sein. Gerade im Bereich von Untersuchungen zu Schulleistungen gestaltet sich dies eher schwierig und muss wenigstens zu einer inhaltlich differenzierten und konkreten Beschreibung der jeweiligen Kompetenzen führen. [SC06] Kompetenzniveaumodelle beschreiben die unterschiedlichen Ausprägungen erfasster Kompetenzen. Insbesondere geht es um die Frage, welche Anforderungen mit unterschiedlichem Ausprägungsgrad bewältigt werden können. Als Interpretationshilfe konkret vorliegender Messungen werden Kompetenzniveaus oder Kompetenzstufen benutzt, die dann eine kriteriumsorientierte Einschätzung der erreichten Kompetenzen ermöglichen. Dieses Herangehen kann zur Grundlage für eine Beschreibung und Kodierung von anforderungsrelevanten Aufgabenmerkmalen genutzt werden, beispielsweise hinsichtlich der auszuführenden kognitiven Operationen, der spezifischen fachlichen Besonderheiten oder der Aufgabenformate. [SC06]

In einer Reihe von Veröffentlichungen wurde bereits darauf verwiesen [KL04], dass diese Auffassung zur Kompetenz eher eine Einschränkung auf kognitive Leistungsbereiche vornimmt und selbst WEINERT eine Begriffserweiterung vorgenommen hat. Für die Erarbeitung von Bildungsstandards erscheint dies gerechtfertigt, wenn man sich dieser Einschränkung bewusst ist. In der Konsequenz verdeutlichen Kompetenzmodelle, wie eine Aufschlüsselung von Dimensionen der jeweiligen fachwissenschaftlichen Grundbildung in Kompetenzen und Kompetenzstufen vorgenommen werden kann. Daraus erwachsen grundlegende Aufgaben an die Fachdidaktiken, entsprechende Modelle zu entwickeln und auszudifferenzieren und durch geeignete Untersuchungen Kompe-

tenzstrukturen empirisch zu sichern. [HA04] Wenn dies durch Bereitstellung notwendiger Ressourcen und in enger Kooperation mit der Schulpraxis erfolgen kann, sind grundlegende Veränderungen in der Bildung denkbar, die über eine bildungspolitische Strukturdiskussion weit hinausgehen. Das reicht von länderübergreifenden Anforderungsstrukturen, die durch Kompetenzen begründet sind, bis zu konkreten Hilfestellungen für Lehrende für die Gestaltung von Lehr-Lern-Prozessen. Damit dies möglich wird, sind nach Klieme [KL03] bei der Ausarbeitung von Bildungsstandards eine Reihe von Merkmalen zu beachten, die allen Beteiligen die verbindlichen Ziele und Kompetenzanforderungen möglichst eindeutig vermitteln:

Fachlichkeit: Bildungsstandards sind jeweils auf einen bestimmten Lernbereich bezogen und arbeiten die Grundprinzipien der Disziplin bzw. des Unterrichtsfaches klar heraus.

Fokussierung: Die Standards decken nicht die gesamte Breite des Lernbereiches bzw. Faches in allen Verästelungen ab, sondern konzentrieren sich auf einen Kernbereich.

Kumulativität: Bildungsstandards beziehen sich auf die Kompetenzen, die bis zu einem bestimmten Zeitpunkt im Verlauf der Lerngeschichte aufgebaut worden sind.

Verbindlichkeit für alle: Sie drücken die Mindestvoraussetzungen aus, die von allen Lernern erwartet werden. Diese Mindeststandards müssen schulformübergreifend für alle Schülerinnen und Schüler gelten.

Differenzierung: Die Standards legen aber nicht nur eine „Messlatte“ an, sondern differenzieren zwischen Kompetenzstufen, die über und unter bzw. vor und nach dem Erreichen des Mindestniveaus liegen.

Verständlichkeit: Die Bildungsstandards sind klar, knapp und nachvollziehbar formuliert.

Realisierbarkeit: Die Anforderungen stellen eine Herausforderung für die Lernenden und die Lehrenden dar, sind aber mit realistischem Aufwand erreichbar.

Auch wenn häufig die Intentionen und Chancen von Bildungsstandards dargestellt werden, sollte der Blick auch auf Gefahren gerichtet bleiben, die diese Entwicklung in sich birgt. So besteht bei einer überzogenen Orientierung auf Ergebnisse die Gefahr, dass Unterricht zur Testvorbereitung mutiert und auf eine automatische Ausprägung von Kompetenzen hoffend, einzig und allein die Lösung von Standardaufgaben in den Mittelpunkt gestellt wird. Selbst für einen eher aufgabenorientierten Unterricht in der Mathematik wird eine Nutzung von Lern- statt Testaufgaben zur Ausprägung bestimmter Kompetenzen als durchaus wünschenswert betrachtet. [BL06] Andererseits kann eine Verbesserung von Qualität im Unterricht nicht allein durch die Einführung verbindlicher Standards erwartet werden. So hat beispielsweise die Lernfeldstrukturierung im Bereich der beruflichen Bildung keinesfalls automatisch besseren Unterricht hervorgebracht. Es wird ein systematischer Veränderungsprozess einzuleiten sein, in den die Lehrenden von Beginn an einbezogen sind und zu deren Realisierung sie zeitliche Ressourcen und fachliche und didaktische Unterstützung benötigen. Folglich muss auch dieser Arbeitsbereich

zum wissenschaftlich akzeptierten und auch geförderten Tätigkeitsgebiet fachdidaktisch Arbeitender gehören, deren gemeinsames Wirken dort unterstützt werden muss. Es bedarf gerade in den teilweise als "Erfahrungswissenschaften" titulierten Fachgebieten aufwendiger und bildungspolitisch unabhängiger Untersuchungen, um zu verallgemeinerbaren Aussagen zu gelangen. Auf ein weiteres Problem verweist Herzog [HE06], indem er deutlich macht, dass nicht einfach von Experten über inhaltliche Strukturen entschieden werden kann. Vielmehr bräuchte es Verfahren, die festlegen, wie inhaltliche Entscheidungen zustande kommen. Zur Realisierung bedarf es letztlich auch der Professionalität und des Vertrauens in die Lehrenden. Hier betont HERZOG, dass "Professionalität bedeutet, in der Lage [zu] sein, auf die Bedingungen der eigenen Berufsarbeit gestaltend Einfluss zu nehmen. .... Vertrauen – das wissen wir spätestens seit Lenin – ist die Alternative zu Kontrolle. Dieses Vertrauen ist in Berufen, die mit Menschen zu tun haben, unabdingbar. Bildungsstandards, die auf Kontrolle ausgerichtet sind, scheinen dies vergessen zu lassen." [HE06]

3 Informatische Bildung – auf dem Wege zu Standards

Die Diskussion um Inhalte von Schulfächern und deren quantitative und qualitative Bestimmung ist nahezu so alt wie die Schule selbst. Dabei sollte beachtet werden, dass jede Schulbildung genau genommen zwei Aufgaben hat: Den Einzelnen auf ein Leben als mündiger Bürger vorzubereiten und ihm für weiteres schulisches oder berufliches Lernen anschlussfähiges Wissen sowie die geeigneten Kompetenzen zu vermitteln. Allerdings hatten es neue Inhalte und Schulfächer schon immer schwer, sich gegen den Widerstand der tradierten Fächer durchzusetzen. Dies gilt für die Informatik heute genau so wie vor hundert Jahren für die Naturwissenschaften. [GO99] Weil die vorhandenen Strukturen der Schulen auf die Anforderungen der Industrialisierung damals nicht reagieren konnten und keine Rahmenbedingungen für neue Fachgebiete existierten, wurden "neue Schulen" in Form von Realgymnasien geschaffen. Vor dem Hintergrund der tiefgreifenden gesellschaftlichen Veränderungen im Informationszeitalter war die Einführung der Informatik in das neu entstandene Kurssystem der gymnasialen Oberstufe in den 70-er Jahren des letzten Jahrhunderts nur ein ganz kleiner Schritt, der keine Veränderungen der etablierten Fächerstrukturen erforderte.
Seither hat es verschiedene Ansätze und Initiativen gegeben, um eine zeitgemäße informatische Bildung in den Schulen fest zu installieren, insbesondere deshalb, weil tiefere fachliche Einsichten für ein kompetentes und vor allem effizientes Benutzen von Anwendungen der Informatik, für die Entscheidung zum adäquaten Einsatz von Informatiksystemen, für die Einschätzung von Möglichkeiten und Gefahren der Nutzung dieser Anwendungen sowie deren gesellschaftlicher Relevanz und schließlich für einen unkomplizierten Anschluss von Studium und Berufsausbildung notwendig sind. Die aktuellen Entwicklungen auf dem Arbeitsmarkt machen zusätzlich deutlich, welchen Stellenwert Bildung in diesem Bereich der Informations- und Kommunikationstechnologie hat und vor allem künftig haben wird. In diesem Zusammenhang ist möglicherweise das Erreichen von Bedienkompetenzen (Computerführerschein) etwas, das auch in anderen Fächern angesiedelt sein könnte. Erfahrungen mit den Empfehlungen der Bund-Länder-Kommission [GK87] machen allerdings deutlich, wie dieses Konzept recht bald an

Grenzen gelangte oder gar nichts passierte. Die tieferen Einsichten in Grundlagen und Zusammenhänge versprechen dagegen, dass letztlich auch die erworbenen Bedienkompetenzen transferfähig werden, also die Übertragung auf bisher unbekannte Anwendungen ermöglicht wird.
In der Vergangenheit haben trotz oder gerade wegen der immer noch umstrittenen Verortung informatischer Bildung in Schulen didaktische Diskussionen stattgefunden, die sowohl fachliche Schwerpunkte als auch unterrichtlich Machbares herausgearbeitet haben und heute der Erarbeitung von Bildungsstandards helfen. Wenn im Folgenden an wenigen Beispielen auf historische Wurzeln der jetzt vorliegenden Entwürfe von Bildungsstandards Informatik [BI07] (ein Gesamtüberblick dazu folgt im Abschnitt 4) eingegangen wird, dann um zu zeigen, dass diese nicht plötzlich entstehen, sondern auf Unterrichtserfahrungen und deren didaktischen Verallgemeinerungen beruhen.

Algorithmen in der Schule

Die Algorithmenorientierung war eines der ersten didaktischen Konzepte in der Schulinformatik, das stark angelehnt an universitäre Vorgehensweisen zu ähnlichen Unterrichtsstrukturen führte. Ausgehend vom Algorithmenbegriff wurden schrittweise Realisierungen in den jeweils vorhandenen Programmiersprachen vorgenommen.

Lehrplan Berlin: [LBE85]

> „In der Algorithmik steht die Einführung in grundlegende Elemente der Algorithmisierungsmethodik an Hand kleiner Programme im Vordergrund."

Lehrplan DDR: [LD89]

> „Die Schüler kennen den Begriff ‚Algorithmus' und wesentliche Eigenschaften von Algorithmen. Sie besitzen sichere Kenntnisse über Algorithmenstrukturen (..) und wesentliche Möglichkeiten ihrer Darstellung, .."

Da das Unterrichtsfach auf Kurse in der gymnasialen Oberstufe beschränkt war und so nur von Schülerinnen und Schülern besucht wurde, die an diesem Fach interessiert waren, fand das Vorgehen eine weite Verbreitung. Möglicherweise entstand auch Liebe zum Detail und daraus die heute noch zitierte Vorstellung, dass im Informatikunterricht lediglich Programmierfreaks unterrichtet werden, folglich diese Bildung nicht für alle notwendig ist. Auf die grundlegende Bedeutung der Algorithmik, die nicht in der Syntax einer Programmiersprache liegt, wurde inzwischen häufig genug hingewiesen, auch auf die Potenz, die diese Bildung im Rahmen einer informatischen Bildung besitzt. [RE04] Deshalb enthalten die Bildungsstandards Informatik auch einen gesonderten Inhaltsbereich "Algorithmen":

> "Unter einem Algorithmus versteht man eine genau definierte Handlungsvorschrift zur Lösung eines Problems oder einer Klasse von Problemen. Dieser Begriff wird auch umgangssprachlich im Sinne eines nach festgelegten Regeln ablaufenden Prozesses verwendet. Die grundlegenden Eigenschaften, die ein solcher Prozess haben muss, damit man von einem Algorithmus sprechen kann, sind mitunter weniger bekannt und werden kaum bewusst verwendet. Gerade wegen dieser Eigenschaften, die im Zusammenhang mit einer Realisierung von Algorithmen stehen, wird dieser Begriff inzwischen als eine fundamentale Idee der Informatik eingeordnet." [BI07]

Es ist nur folgerichtig, dass daraus Anforderungen abgeleitet werden, die alle Schülerinnen und Schüler mindestens erfüllen müssen. Zum Erreichen dieser Kompetenzen sollten nie die Werkzeuge im Vordergrund stehen, die in Lehr-Lern-Szenarien verwendet werden.

Modellierung als wichtiges Konzept

Bereits in Zeiten algorithmenorientierter didaktischer Konzepte wurden Grundprinzipien der informatischen Abstraktion und Modellierung durch schülergeeignete Darstellungen realisiert. In den im Rahmen der informationstechnischen Grundbildung meist verwendeten Anwendungen hatten, bedingt durch einfachere Benutzungsoberflächen, die Fertigkeiten zu deren Verwendung ein größeres Gewicht als das Hinterfragen der zu Grunde liegenden Struktur. Trotzdem entstanden erste Überlegungen zu einer Modellierung und Strukturierung dieser Anwendungen [FB91], die auch in Lehrplänen für den Sekundarbereich I aufgegriffen wurden [LSN92]. Dabei ging es aus didaktischer Sicht darum, in analoger Weise für Anwendungen wie Textverarbeitung, Tabellenkalkulation, o.ä. eine verständliche Modellierung zu schaffen, diese altersgemäß aufzubereiten und durch Ausbildung und über geeignete Materialien die Lehrenden zu befähigen, ein solches Vorgehen in einem Fachunterricht umzusetzen. Auch dabei zeigt sich ein fortschreitender Erkenntnisprozess, der nur durch einen engen Kontakt zu schulischen Erprobungen zielführend ist und auch vor theoretischen Überhöhungen bewahrt.
An Begriffsbildungen aus verschiedenen Materialien wie Lehrbüchern und Handreichungen kann ein solcher Prozess recht gut veranschaulicht werden. Wir wollen das exemplarisch an der Auffassung zur Beschreibung von Objekten einer Standardanwendung wie der Textverarbeitung zeigen:

Akademiebericht Dillingen [FB91]
: Überblick über Strukturelemente der Textverarbeitung: Zeichen, Wort, Satz, Zeile/Absatz, Seite/Gesamttext (Dokument)

Lehrplan Sachsen: [LSN92]
: Grundbegriffe der Textverarbeitung: Objekte (z.B. Zeichen, Absatz, Bereich); Attribute (z.B. fett, Blocksatz, zweizeilig); Operationen (z.B. formatieren, ersetzen, kopieren)

Lehrbuch „Informatik und Alltag“: [FR96]
: Objekte in der Textverarbeitung sind Bestandteile von Texten. Sie besitzen Eigenschaften. Die elementaren Objekte sind Zeichen, Absatz und Dokument.

Lehrbuch Didaktik: [HU00]
: Nun übertragen wir die soeben erworbene (objektorientierte) Begriffswelt auf ein Textverarbeitungssystem. Dabei identifizieren wir z.B. die Klassen „Textdokument“, „Seite“, „Absatz“, „Zeichen“ mit den … aufgezählten Attributen.

Lehrplan Sachsen: [LSN04]
: Allgemeines Ziel: Die Schüler erkennen, dass Abstraktion und Modellbildung grundlegend für das Verständnis einer automatischen Informationsverarbeitung sind.
Klasse 7: .... Die Schüler nutzen den Zusammenhang Objekt – Attribut – Methode als Modell zum Verständnis von Anwendungen.

Lehrbuch „Informatik“: [FH04]

> Wir behandeln daher die einzelnen Zeichen eines Textes als Objekte der Klasse Zeichen. Alle Buchstaben, Ziffern, Sonderzeichen und Leerzeichen sind Objekte dieser Klasse.

Handreichung ISB München: [ISB05]

> Hefteintrag: In Texten kommen Objekte der Klasse Zeichen vor. Diese können Buchstaben, Ziffern, Leerzeichen und Sonderzeichen (z.B. @, ?, §) sein.

Letztlich schaffen die in dieser Weise reflektierten Erfahrungen eine gewisse Sicherheit zu dem, was für Schülerinnen und Schüler im Themengebiet der informatischen Modellierung unter Nutzung von Standardanwendungen zu verstehen ist, welche Kompetenzen auch in diesem Bereich mindestens angestrebt werden müssen.

In den nun vorgelegten Bildungsstandards Informatik findet sich dieser Themenkomplex im Inhaltsbereich "Information und Daten" [BI07] wieder. Weitere Anforderungen sind sowohl in den anderen Inhaltsbereichen als auch in den Darstellungen zu Prozesskompetenzen, insbesondere "Modellieren und Implementieren", zu finden.

Schülerinnen und Schüler aller Jahrgangsstufen sollen	**In den Jahrgangsstufen 5 bis 7 müssen alle Schülerinnen und Schüler**
... den Zusammenhang von Information und Daten sowie verschiedene Darstellungsformen für Daten verstehen.	… Begriffe „Klasse“, „Objekt“, „Attribut“ kennen und in Anwendungssituationen verwenden.

Im erläuternden Text wird dann davon ausgegangen, dass ein Bekanntmachen mit dieser Begriffswelt an Grafik-Anwendungen erfolgen sollte. Hinsichtlich der Verarbeitung von Texten wird dann an gleicher Stelle u.a. ausgeführt:

> "... Textverarbeitungsprogramme (und Editoren) sind die zweite wichtige Anwendung, die Schülerinnen und Schüler möglichst früh kennen lernen müssen. Dabei ist es wichtig, sich nicht in den sehr umfangreichen Möglichkeiten eines Textsystems zu verlieren. Statt dessen soll der allen Textsystemen gemeinsame Kern herausgestellt werden. Das sind vor allem die Strukturierungseinheiten Zeichen, Absatz und Seite. Die Schülerinnen und Schüler sehen anhand der Menüstruktur, dass dies die Einheiten sind, auf die sich Formatierungen beziehen. Objektdiagramme für einfache Beispiele oder Klassendiagramme zum Aufzeigen des allgemeinen Prinzips können helfen, dies zu verdeutlichen..."

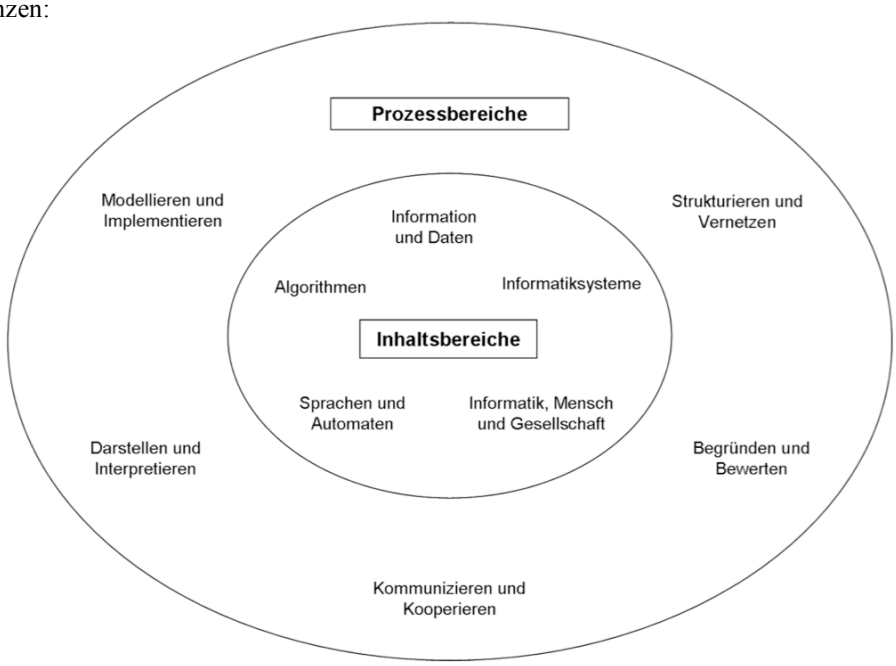
Gerade hier besteht allerdings die Schwierigkeit, dass Lehrende in der eignen Ausbildung eher selten mit entsprechenden Modellierungen konfrontiert wurden und es deshalb gerade dafür eines effektiven Fortbildungskonzeptes [FI07] bedarf.
Hier sollen Bildungsstandards ansetzen, eine Gesamtsicht auf eine informatische Bildung liefern und gleichzeitig die Komplexität in diesem Bereich deutlich machen. Die informatische Bildung hat längst den Ruf abgelegt, der Technologie nachzujagen, sondern beinhaltet genügend Grundlagen, die zur minimalen Bildung eines jeden Lernenden im heutigen gesellschaftlichen Umfeld gehören. Die Erfahrungen des Wirkens für einen

verpflichtenden Informatikunterricht zeigen, dass dazu ein Spagat zwischen Wünschbarem und Machbarem nötig ist. Bei aller Orientierung an Kompetenzen, die durch Aufgaben zu überprüfen wären, bedarf es Grundaussagen, Hinweisen und Materialien, um in den durch Lehrerinnen und Lehrer organisierten Lernprozessen die Kompetenzen bei Lernenden erst einmal zu entwickeln. Eine Einbindung von reinen Testszenarien, wie beispielsweise EDCL, liefert ein Arbeiten nur für den Test. So gibt es vergleichsweise auch keinen ernstzunehmenden Vorschlag, im Bereich der fremdsprachlichen Kompetenzen in der Allgemeinbildung bereits ausgearbeitete Testszenarien (wie z.B. TOEFL) zu verwenden.

4 Bildungsstandards Informatik – Entwurf und Konsequenzen

Das Ziel von Bildungsstandards kann nicht sein, eine zentralisierte innere Systematik von ausgewählten Schulfächern zu beschreiben. Dadurch wird für die Qualitätsentwicklung in der Bildung kaum etwas zu erreichen sein. Die Darstellung sollte genau das beschreiben, worauf sich aufbauende Bildungsgänge oder die berufliche Praxis verlassen können. Ein Fachunterricht wird wesentliche Beiträge leisten müssen, aber nicht allein ausreichen. So, wie Lesekompetenzen nicht ausschließlich im Unterrichtsfach Deutsch entstehen, wird auch für andere Kompetenzbereiche zu entscheiden sein, wo deren Herausbildung stattfindet. Bei der Erstellung von Bildungsstandards Informatik müssen schließlich auch solche Inhalte und Methoden sichtbar gemacht werden, die sich deutlich von denen anderer Fächer unterscheiden und den systematischen Fachunterricht im Fach Informatik von anderen Einsatzformen des Computers im Schulalltag signifikant abgrenzen. Hinsichtlich der informatischen Kompetenzen kommt man so zur Konsequenz, dass deren Entwicklung ohne einen Fachunterricht Informatik nicht möglich ist.
Angeregt durch Beiträge auf wissenschaftlichen Fachtagungen [PU03] ging es bei der Formulierung von Standards für die Schulinformatik zuerst um eine Standortbestimmung und Orientierung der notwendigen Arbeiten. Dabei wurde deutlich, dass eine Orientierung an den Standards für Mathematik des NCTM [NC00] gegenüber dem Herangehen der KMK gewisse Vorteile bietet. Während der Fokus bei den Vorschlägen des NCTM primär auf der Unterstützung der Arbeit der Mathematiklehrer lag, und zwar mit dem Ziel, Kindern eine qualitativ hochwertige Ausbildung angedeihen zu lassen, ging es seitens der KMK eher darum, festzulegen, was Lernende zu einem bestimmten Zeitpunkt können sollen. Damit wurde zugleich ein theoretischer Rahmen für das Testen von Kompetenzen geschaffen. Das begründet auch, warum in der Erarbeitung von Bildungsstandards der KMK Regelstandards zur Grundlage genommen wurden. Die Folgerung aus den bisherigen Darstellungen zum Stand der informatischen Bildung kann eigentlich nur darauf gerichtet sein, solche Kompetenzen zu beschreiben und für Lehrende nutzbar darzustellen, die das Minimum informatischer Bildung für alle Schüler erfassen, also Mindeststandards zu formulieren.
In verschiedenen Workshops [GI07] wurden die Ausarbeitungen begonnen und erste Entwürfe diskutiert. So konnten auf der GI-Tagung „Informatik & Schule – INFOS'05“ im September 2005 in Dresden [GI05] bisherige Überlegungen dargestellt und die weitere Zusammenarbeit im Sinne des Vorgehens des NCTM angeregt werden. In der Folge wurde ein erster Gesamtentwurf [CU06] vorgelegt und in den Fachgruppen der Informa-

tiklehrerinnen und -lehrer sowie unter Fachdidaktikern diskutiert und weiterentwickelt. Es wurden die Inhaltsbereiche „Information und Daten“, „Informatiksysteme“, „Algorithmen“, „Sprachen und Automaten“ sowie „Informatik, Mensch und Gesellschaft“ herausgearbeitet und Kompetenzen formuliert, die Schülerinnen und Schüler unter dieser inhaltlichen Sicht erwerben müssen. Diese Kompetenzen sind als Unterstützung für die Planung und Gestaltung des Unterrichts für die Klassenstufen 5-7 und 8-10 konkretisiert. Zusätzliche erläuternde Texte und Aufgabenbeispiele sollen verständlicher machen, was eigentlich mit der geforderten Kompetenz gemeint bzw. welches Niveau anzustreben ist. Die Prozesskompetenzen beschreiben dann „Arten des Umgangs mit informatischen Inhalten“ und sind in analoger Weise dargestellt.
Die folgende Übersicht zeigt diese Inhaltsbereiche und die prozessbezogenen Kompetenzen:



Es würde den Umfang des Beitrages sprengen, wenn an dieser Stelle die Bildungsstandards Informatik [BI07] vollständig vorgestellt würden.
Bei der Formulierung von Standards für die Schulinformatik ist es natürlich auch notwendig, darüber nachzudenken, auf der Basis welcher Inhalte welche Kompetenzen entwickelt werden können. Dabei sind immer Inhaltsbereiche und prozessbezogene Kompetenzen in Einklang zu bringen. Ein losgelöstes Betrachten oder vielleicht formales Abarbeiten stimmt mit den Grundüberlegungen zu Bildungsstandards nicht überein und wird informatische Bildung in der Allgemeinbildung keinesfalls befördern. Die vorliegenden Ausarbeitungen zu den Standards belegen auf andere Weise, dass informatische Bildung ins mittlere Schulalter gehört. Es erscheint realistisch, die vorgeschlagenen Kompetenzen zum Maßstab eines Unterrichtsfaches oder auch eines anders strukturierten Lehr-Lern-Prozesses zu machen. Keinesfalls sollte weiterhin vom

Anfangsunterricht Informatik gesprochen werden, wenn damit die Klassenstufe 11 in der gymnasialen Oberstufe gemeint ist.
Ohne es im Detail auszuführen: Die vorgelegten Bildungsstandards Informatik genügen den von KLIEME geforderten Merkmalen, weil sowohl Fachlichkeit als auch nötige Breite beachtet wurden, weil Mindeststandards formuliert sind, die für alle gelten sollen und eine Differenzierung über die Klassenstufen vorgenommen wurde. Die weitere Diskussion und vor allem ihre Nutzung im Schulalltag werden zeigen, ob die Formulierungen verständlich und nachvollziehbar und ob die Anforderungen auch mit dem gedachten Aufwand realistisch erreichbar sind. Dazu braucht es Lehrerinnen und Lehrer, die durch eignes Tun, durch Darstellen von Erfahrungen und kritisches Diskutieren diesen Prozess aktiv begleiten. Es braucht ferner Materialien, um Lehrende für ihre Tätigkeit anzuregen [HA06], ohne dabei Rezepte darzustellen, die die Lehrenden eigentlich gar nicht erwarten. Und es braucht schließlich auch solche Rahmenbedingungen für die Fachdidaktiken, die die konkrete Unterstützung der Schulpraxis höher bewerten und stärker fördern als praxisferne Publikationen, die lediglich den wissenschaftlichen Diskurs befördern. Wenn es der Bildungspolitik um die Standards ernst ist und diese die Qualität von Schule und Bildung wirklich verbessern sollen, ist die bisherige "Projekt-Manie" untauglich. Es sind völlig neue und vor allem langfristig angelegte Szenarien der Unterstützung von Forschung und Unterricht (sicher einschließlich der Ressourcen) notwendig, die vom gemeinsamen Erarbeiten von Unterrichtsmaterialien und der Überprüfung ihrer Brauchbarkeit bis zu vereinfachten Verfahren der Unterrichtsbeobachtung und deren Reflektion reichen und so der wissenschaftlichen Qualifikation von Lehrenden dienen.
In diesem Sinne hat die Arbeit an den Bildungsstandards Informatik erst begonnen, obwohl sicher mit der nun vorgelegten Fassung ein wichtiger Grundstein gelegt ist. Es geht nicht nur um "einen Wechsel der Perspektive, der die Praxis ja schon anders macht" [OE06], sondern um Lösungen für die Praxis. Insbesondere sollte es für die informatische Bildung darum gehen, diese Mindestforderungen unter den vorhandenen oder veränderten Rahmenbedingungen auch zu realisieren und Erfahrungen dabei offen auszutauschen. Ein besonderer Stellenwert wird dabei der Erarbeitung weiterer Aufgaben zukommen, 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.

Literaturverzeichnis:

[BI07] Bildungsstandards Informatik. In: LOG IN 146/147 (2007), (im Druck)

Eine umfangreiche Angabe der weiteren Quellen konnte aus Kapazitätsgründen hier nicht mehr angefügt werden.
Sie steht unter `http://dil.inf.tu-dresden.de/infos07/quellen.pdf` vollständig zur Verfügung.

Exploratory Learning

Ivan Kalas, Daniela Lehotska

Department of Informatics Education
Faculty of Mathematics, Physics and Informatics
Comenius University
842 48 Bratislava, Slovak Republic
kalas@fmph.uniba.sk
lehotska@fmph.uniba.sk

Abstract: In our department we have a long history of developing interfaces for learning, which are framed by constructivist and constructionist theories of learning. Thus we try to create effective opportunities for communication, exploratory and collaborative learning and working in common learning spaces. We build this specialization upon software platforms like SuperLogo and Imagine Logo (software environments being used in dozens of countries and thousands of schools for educational purposes), which have been creating since 90-ties with the aim to provide children, students, teachers and (professional) educational software developers with intuitive yet powerful means to discover, communicate and cooperate. In our software platforms we are making use of the combination of several modern and innovative features like parallel processes, object-oriented paradigm, events, tools for straightforward on-line communication through the Internet, and programmable shapes – the possibility to specify the actors' (i.e. turtles') shapes by simple Logo commands.

During recent years we have been exploring interesting opportunities emerged from these features, which allow us to create visual interactive objects in many diverse and inventive ways. Actors with shapes specified through programmable shapes may change in accordance with external settings; they may become pieces of interactive "jigsaw puzzle", blocks of a building set inviting children to explore. They may continuously visualize actual states of other objects or relations among them – in such role we call them gauges and use them for increasing the children's motivation and supporting them while children explore, modify or build behaviours in their learning processes.

Gradually we have discovered that visual interactive objects might have diverse shapes and could be used in many effective and attractive ways in working with different age groups – from simple building blocks for pre-schoolers, through flexible objects for different representations of fractions, to complex items of software laboratories, in which future teachers construct their own knowledge and understanding of key informatics concepts.

We will conclude our survey of challenging means for exploratory learning by presenting a few observations obtained by us or our partners from several recent European research projects.

1 Introduction

In our department we have a long history of developing interfaces for learning, such as SuperLogo, Thomash the Clown, Imagine Logo (see [KB03]) and others, often developed or evaluated within European research projects like Playground, Match, CoLabs, Kaleidoscope 6FP NoE with partners from IoE London, Logotron Cambridge, IoE Warwick, ELTE Budapest, Cnotinfor Portugal and others. Our interfaces for learning are framed by constructivist and constructionist theories of learning. We share Piaget's understanding that the knowledge can't be transferred from one person to another. It is constructed by an individual through his/her own experience. "To understand is to discover. A student who achieves certain knowledge through free investigation and spontaneous effort will be able to retain it" [Pi73].

In development of our environments for learning we try to create effective opportunities for communication, exploratory and collaborative learning and working in common learning spaces. That is why we are interested both in cognitive oriented and socially oriented constructivism which stresses the collaboratory efforts of learners as sources of learning.

Most of all, however, we are influenced and motivated by the work of Papert who explored the theory of constructivism and went beyond it. He calls his theory constructionism as the opposite of instructionism. Papert claims, that constructivist learning happens when people are engaged in constructing a product, something external to themselves such as a sand castle, a machine, a computer program or a book. In parallel with constructing the things children construct their knowledge. In Papert's view "better learning will not come from finding better ways for the teacher to instruct, but from giving the learner better opportunities to construct". He considers computer to be the medium for development of constructionistic learning – in [Pa93] he speaks about The Knowledge Machine which would allow children a rich exploration of the world.

Constructionism and exploratory learning have been our main visions also in several recently completed research and development projects. There we have been using Imagine Logo platform for developing and studying means which would allow children to discover, communicate and cooperate. We have been rather successful in reaching these goals mainly due to effective and productive combination of several modern and innovative features of Imagine Logo – we will briefly present them in chapter 2. After that we will show that interfaces for learning can be employed in many effective and attractive ways in working with different age groups. To prove this we will put forward outcomes of several projects: the gauges of the Space Travel Games Construction Kit project [Ka06a] (in chapter 3), the interfaces for collaborative learning of the CoLabs project (in chapter 4), interesting dynamic mathematics environment of Visual Fractions (in chapter 5), and a series of interactive Imagine Logo microworlds from experimental seminar for our future teachers. We are using these environments to provide our students with tangible experience through which they construct their own knowledge and understanding of many key informatics concepts (see chapter 6). In the closing chapter 7 we present some interesting observations from the research with the interfaces for exploratory learning, conducted by our European partners or by us.

2 Imagine Logo Platform for Our Research and Development

Imagine Logo represents new generation of educational and professional programming environment, which provides users with powerful and modern techniques like object-oriented paradigm and on-line cooperation through the Internet. This approach makes Imagine Logo an appropriate learning environment not only for children, but for teachers and professional educational software developers as well. It is becoming a widely-used tool in European educational systems for prototyping and building active learning environments, educational microworlds or stand-alone applications, and for educational research as well. Beside traditional elements of turtle geometry it offers several new attractive features and concepts for education. In [BK01] and [KB03] we have explored whether and how these new enhancements support understanding in modern exploratory learning. In our research projects we make use of Imagine Logo as a tool for quick and elegant development of interactive interfaces for learning. The most important features, which allow us to do so, are:

- Events and parallel processes – beyond all doubt they represent new approach in the development of interactive interfaces for learning. In rather natural and powerful way they support what we develop – either with children or for children: in many iterative design loops we build actors or objects, we build their shapes, behaviours, interactions with the environment or between each other. Parallel processes can be considered as forces, which independently and in parallel watch over relations and dependencies among actors, make them move, make objects follow certain paths or curves, run various experiments, simulations, gadgets, dynamic models etc.

- Multiple turtles and object-oriented structure – nearly unlimited numbers of actors (that can be organized into "families" of similar objects with related or identical behaviours) help us construct scenes with dozens or hundreds of actors of any kind. These actors (or any active pieces) may be dynamically created, may react to outer circumstances, alter their reactions, become models for new clones, may cease to exist etc.

- Programmable shapes – the possibility to specify actors' shapes (or any picture, still or animated) directly by simple Logo commands. In [KB00] and [KB03] we presented broad palette of possibilities offered by this innovative feature of Imagine Logo. Note that all shapes specified in this way are independent of the absolute frame. In harmony with the traditional Logo philosophy, all programmable pictures stick to the relative frame and as such Imagine itself rotates them according to actual headings of turtles. Being turtles, such visual objects may easily move along the screen, change their headings, colours, sizes, speeds etc.

In an extended process during recent years we have been deeply exploring the possibility to represent the actor's (i.e. turtle's) shape through a piece of simple Logo commands. We have considered the ways how this feature – combined with event-driven programming, object-oriented structure and parallel independent processes – qualifies Imagine as

a powerful development platform for computational systems aimed at representing, exploring and manipulating visual interactive objects in diverse innovative ways.
In [KB03] we presented several simple examples of how to turn common Logo turtles into squares, arrows, cogwheels, numbers, angles or fraction parts of different units. Inside the "setShape [...]" command brackets we simply insert usual Logo commands which would otherwise draw the same shape into the background.

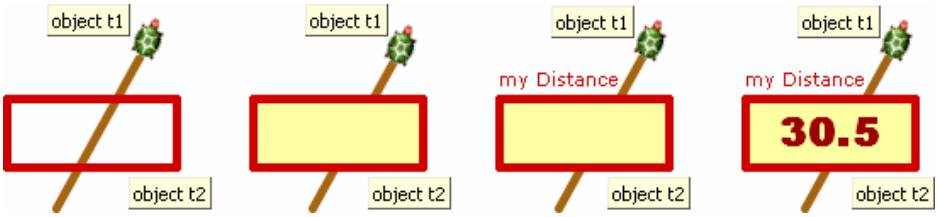


Fig 1: Turtle t1 draws a line in the background (to clarify the difference), while turtle t2 "draws" its own shape by "setShape" using the same language. Any turtle created in such way can move and rotate – its shape isn't drawn into the background

3 Imagine Logo Turtles as Gauges

Shapes of Logo actors (i.e. turtles) which are specified in such programmable way may change in accordance with some external settings. Figure 2 for example illustrates a situation, when slider 1 has an event attached to it, which will – whenever triggered by resetting the slider – address object t1 and make it accept the actual value of the slider as its actual diameter and update its shape correspondingly. Naturally, there may be several external factors which dynamically trigger the process of updating a shape.

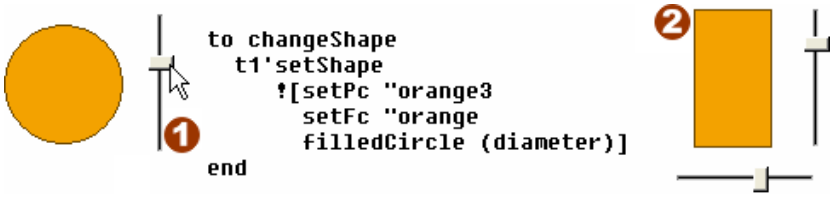


Fig 2: Rectangle 2 will dynamically change its width and height according to a pair of attached sliders

Objects which are parts of the environment may employ their programmable shapes to look like numbers, cards with words or sentences, fractions, ... simply any building items appropriate for the topic of the concrete environment for exploratory learning. Such object:

- is a common Logo turtle, which means it can easily be moved and rotated – without any need to think about how to specify its newly rotated shape,
- can contain its own internal information, its own private data, state and attributes,

- can have shape which dynamically changes depending on its actual state (for example, its actual position) or in accordance with some external factors (for example variables stored in the page, attributes of some other objects etc.). Using the mechanism of user-defined events, any change of any of those factors may trigger forward propagation to update all dependant shapes, see [To03],
- can employ the mechanism of parallel independent processes to – besides reacting to any external changes – continuously execute some other operations or movements. An object can thus monitor some other values (not affecting its own shape) or relations, it can move, change its size, colours etc.,
- can contain any number of sensitive spots which differently react to clicking or dragging by the mouse. For example, if we drag red point 1 of the angle, see Figure 3, its internal setting size will change and thus the shape of the angle will change as well. If we drag its blue point 2 the whole angle will rotate around its central point 3. If we drag the perimeter 4 of the angle, we can increase or decrease its radius. Moreover, any angle object can be either transparent or coloured. This example demonstrates how flexible and interactive these objects are.

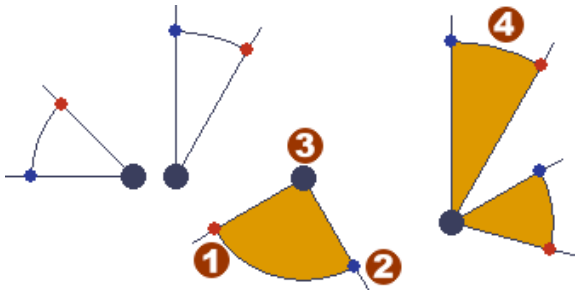


Fig 3: We extensively exploit programmable shapes for representing mathematical objects or building blocks in various exploratory microworlds

In the Space Travel Games Construction Kit project, see [Ka06a], we have started to engage such dedicated objects neither for building nor for modifying the behaviour of any actor or tool – their purpose has only been to continuously visualize (through their shapes) actual attributes of the things involved. We started to call such objects gauges. Observations reported by Kahn et al. prove that gauges can play important role in increasing the children's motivation and support them while exploring, modifying and building behaviours in their learning process.
One of the challenges in the Space Travel Games Construction Kit is to make the Lander safely land on the surface of a planet. To manage this, children explore, apply and modify behaviour gadgets (pieces of behaviour encoded in instructions) and visual gauges that allow them to monitor several key quantities of the Lander. In Figure 4 we see the gauges panel (A) with icons for three different models of gauges (B). If we decide to use one of them, we click it and drag its instance (C) into the working space.

Fig 4: While the gauge C isn't activated, it shows as a default value a question mark – any other initial value could be confusing

To activate the gauge we have to drag its anchor (D) towards an object we want to explore, the Lunar Lander (see E) in this case. When we release the anchor, the object displays the menu (F) of all different values it exports to the gauge, see Figure 5, and thus allows us to get into its internal state in the most readable way.

Fig 5: When the gauge is activated, the colour of its anchor alters (see G) and the gauge starts to continuously display the actual value of a chosen variable (see H). The gauge also displays the name of the object and the variable currently monitored

The gauges showed in Figure 6 have some additional control elements: circle 1 can be dragged vertically – in this way we can change the scale of all collected values, for example, between f(x)/1 and f(x)/10. Similar extensions and modifications of gauges are limitless, which results from the fact they are specified in ordinary Logo commands. Their look, colours, sizes and any additional control elements are designed by the author of the microworld, see Figure 6. Designing them is similar to drawing them in traditional Logo turtle way. The gauge in the right part of the figure, for example, contains four anchors to monitor four different quantities in parallel.

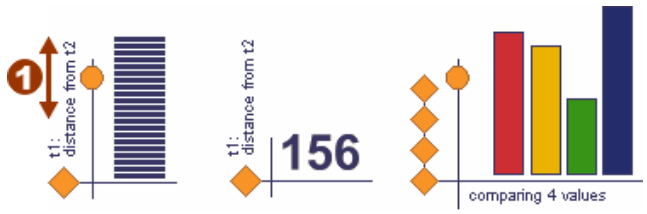


Fig 6: Gauges may have very different shapes

Based on the educational projects mentioned above and research conducted within them we may specify benefits and properties of engaging gauges in the interfaces for learning:

- Gauges are visual and highly interactive objects and thus are useful tools to observe dynamic processes, changing states, relations and quantities.
- They allow us to access areas of the environments (microworlds, interfaces etc.), which are due to different reasons hard to access or completely hidden. It is the designer of the environment who decides which attributes or quantities will be made available in which context.
- Their styles, sizes, colours etc. are relatively easy to change by the designer (or even by the user) as far as these are basic Logo objects with shapes specified by the Logo commands. Advanced user or designer may either modify them or build new gauges with different look.
- They are highly flexible tools which may contain several additional control elements for customizing the visualization, for example the size, scale, units of measurement, frequency etc.
- They can be employed in different layers of learning in the sense of [Ka06a], namely at the first layer for reading and observing, at the second and third layers for manipulating, predicting, testing and active exploring and at the fourth layer as a means for projecting modifications or new development within the environment.
- Gauges can be used in different architectures: naturally they would fit into component-oriented systems (see [KKH97]) or in less modular systems where they may provide the probe into private internal states of other objects.
- As reported by Kahn, gauges support collaborative learning (see below).
- Gauges support employment of multiple representations in parallel. We have managed to benefit considerably from this property in our Visual Fractions.

4 Visual Fractions

Exploratory learning has been our main vision also in recently completed CoLabs project. Our aim has been to develop educational microworlds for collaborative work and learning – collaboratories. As one such collaboratory we decided to create a complex

dynamic interactive computer environment, which would allow children to explore and discover fractions and fractional relations by themselves, or with peers and teachers. We wanted our collaboratory to offer a set of powerful jigsaw puzzle pieces for building one's own understanding of the topic.
Visual Fractions provides ten different visual representations: pie, box, decimal fraction, percentage, ratio, picture, family, number line, fraction and balloon. Fraction objects can be connected together to create dependences, which means that some objects represent values of some other objects. These dependencies are dynamic – we can change the value of an independent object and consecutively the value of a dependent object will change. These dynamic dependencies and interactions give us opportunity to observe how a fraction represented in one way looks like in another representation, what happens if we change the value of some object, investigate when and find the reasons why a fraction represented in one way cannot be expressed in another way etc. The relations between objects can also be more complex – more objects can depend upon the same object. Also the chains of dependencies can be created.

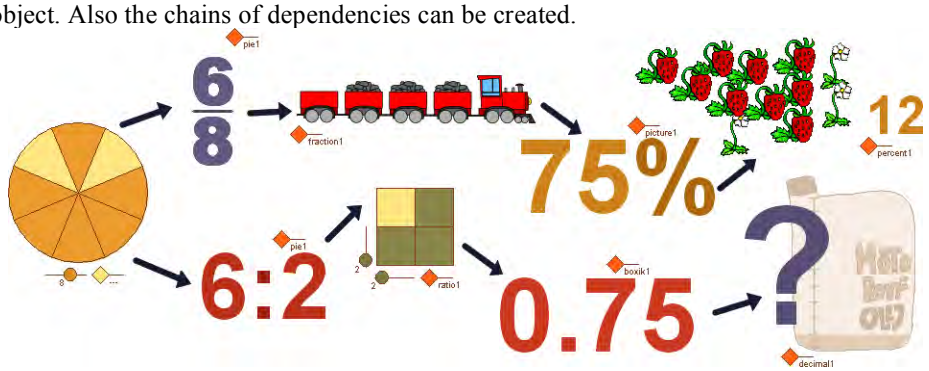


Fig 7: Two chains of dependencies of fraction objects staring with a pie with the value of 6/8. Fractions represented by a picture (a train, an oil canister...) have fixed denominator (for the train in this figure it is 4, for the oil canister it is 10). Therefore they sometimes cannot represent a value of another fraction, e.g. we cannot represent 6/8 by a canister with 10 segments

For each fraction object several properties can be set up. In some representations we can determine which control gadgets of the object will be presented to the user. Some other settings are specific mathematical properties. E.g. pie, box and fraction representations have a property that determines whether the object should always be reduced to the smallest denominator.
Beside fraction representations there are other useful tools in Visual Fractions including basic arithmetic operations (addition, subtraction etc.), operations for comparing (less, equal, greater than), regions, generators, textboxes and components for representing True and False values. These tools work on the principle of dependencies as well. For example, regions are used to find out whether groups of objects meet certain condition, like whether all objects within the region are less than (equal, greater than) the region's leading value, whether all objects in the region are identical, true, ordered, whether their sum equals the region's leading value. With generators we can create sets of values that are either randomly or in the specified order used as input values (parameters) to an exploratory task.

Since the beginning of the Visual Fractions development we have planned to support different forms of its usage in the learning process. We consider the situation when a pupil solves pre-prepared activities the least innovative style of learning, the least challenge to collaborative learning and constructivist explorations. At the opposite pole to such learning situation is a group of pupils working in a common learning space – either at their computers or in front of a large interactive smart board, where together with their teacher they solve problems and discover relations using additional blocks of the fractions building set and thus explore mathematics in a more efficient way. They can build dynamic fraction models with dependencies; they can modify the items' settings and observe the effects. They can construct experiments about fractional relations; they can discuss the ways how the models behave. The teacher may stimulate the explorations by questions like What would happen if... or How to make it evident that... or How could we explain it or demonstrate it to others? In such learning situations the distinctions between users of activities and authors of activities are disappearing. Solving problems in such constructionist way means explore relations between objects, modify their representations, add new blocks and insert them into the chains of dependencies and discuss the consequences.

5 Developing Interfaces for Collaborative Learning

In our research we explore what Imagine Logo offers to target groups of users who expect small, immersive, open and flexible microworlds supporting on-line collaboration – developed for everyday learning situations by the means that are accessible also to non-professional developers of educational software. If we examine the development of Imagine microworlds from the on-line collaboration point of view, we can distinguish four dimensions of this process:

- Technicalities – copes with the low level communication through the TCP/IP protocol to run the transfer of data between Imagine kernel and the network. It also provides complete or partial support for overcoming linguistic barriers etc.
- Connection interface – is responsible for creating and activating the connection between two or several computers.
- Sharing – covers higher network programming (communication) by exploiting already running connection between computers. It is responsible for sending and receiving all kinds of information and objects, and thus creating and managing common work space, characters and behaviours.
- Activity – implements the educational goal through designing and programming the microworld itself (covers all programming except network communication).

In [KW06] we have analyzed the development process along each of these dimensions; here we will concentrate on building connection interface only. The tasks being solved here include creating and activating an on-line connection between two or several computers with running Imagine (through local network or Internet). Based on our previous experience in building such interfaces we have developed a metaphor, which

hides all technical details and concepts of networking such as server, client etc. and makes it feasible to understand and carry out the following steps.
To establish and maintain such connection the Imagine "Net" class of objects is used. An object (instance) of that class represents a connection for all participants of the activity, i.e. two or several computers. One of the participants – let us call him an initiator of activity – creates a Net object and activates it (steps 1 and 2, see Figure 8). We may consider the initiator as the one who invites us to join.
Other user (or users) must know that there is an open invitation and must react by taking the following steps: he creates a "Net" object as well and accepts the invitation (steps 1 and 3; step 1 is identical for all, step 3 can be run only after step 2). This is similar to making a phone call: step 1 means "have a mobile phone", step 2 "switch it on". Step 3 means either "dial the number" of a friend (i.e. enter his IP address) or "look up his name in the address book" and press it (i.e. enter the other user computer's name).
As soon as the active connection is established, all differences among participants (who was inviting, who accepted) can be forgotten – all of them become equal partners (as if connected in the peer-to-peer private network, although the developer must understand the difference between the initiator-server and other users-clients). When creating the connection (step 1) each user may specify his own nickname, which will later be used by other users to address him. When the connection is activated and established, the users will not need any technicalities (like IP address) any more.

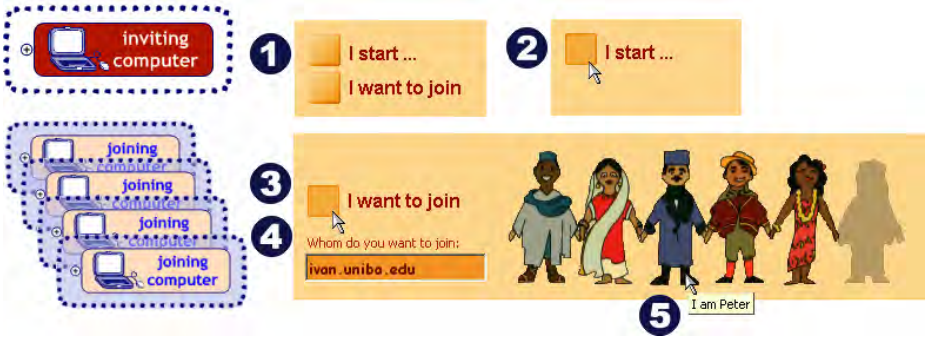


Fig 8: Sample interface for creating 1-team, identical-roles, up-to-6 users connection

The key motive for us why we continually study collaboration through Logo on-line microworlds is the ease with which we achieve high level of interactivity, visualization and openness. By openness of an educational application we understand its readiness to open the inside to its user and easily accommodate modifications of its internal resources. We distinguish openness (a) at the level of data – Can pictures, sounds, backgrounds or other data be accessed and modified or replaced?, (b) at the level of application didactics – Are there more ways how to use it?, and (c) at the level of tasks – Can other tasks or other type of tasks be inserted to fit into the same didactics? Net connections of Imagine Logo lead us even further than most of other Logo versions. It allows, for example, constructing our own version of forwarding, then publishing and sharing it with all other users – in the form of shared and open piece of code, shared behaviour or shared procedure.

6 Exploratory Learning in Our Future Teachers Training

The potential of straightforward development of visual interactive objects can easily find space for creative integration into surprisingly wide spectrum of situations and ages of learners – as illustrated by our innovative seminar on Discovering Basics of Informatics for future teachers of informatics. During 12 weeks we pass through series of interesting and complex topics, however, we treat them in a completely informal way encouraging modern constructionist learning. To achieve this we are developing small interactive software interfaces for exploratory learning, devoted to one or another key issue of informatics, for an example see Figure 9.

For our students the basics of informatics are hard and often unpopular domain. When we tried to analyze the reasons for their negative attitude, we came to the following conclusion: Because of the shortage of time the students are exposed to key issues and problems before they manage to discover those problems by themselves. We present to students formal analysis of the fundamental problems which aren't their fundamental problems yet. Moreover, we do so using language which is rather unfamiliar to them.

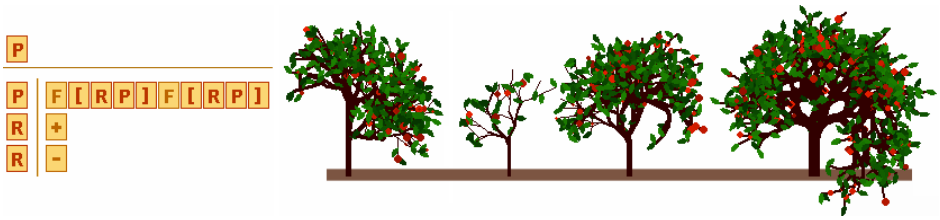


Fig 9: When developing software applications for our seminar we provide students with constructivist interfaces for exploratory learning. This figure shows an application, in which students construct L-systems, have them generate words and interpret them in a Logo turtle way

To improve this situation we want our students to obtain concrete experience within several key issues like constructing logic circuits, modifying them and exploring corresponding Boolean functions – before they start formal treatment of problems with minimization of DNF expressions etc. We want them to play with regular expressions before they start proving theorems about them, before they start talking about them using yet another formal language etc. Although it is too early to draw any serious conclusions about students' attitude to the basics of informatics, in [Ka06b] we could already report about how this innovation has accelerated students' interest in exploratory learning.

7 Some Interesting Observations

"Collaborative learning is a category of learning by cooperating, common exploring and discovering, creating solutions and solving problems, discussing and improving together", see [Wi00]. New digital technologies play unique role in such learning. Therefore, it is natural that one of the key challenges in the development of interfaces for learning at our department is the enhancement of collaborative exploratory learning.

By referring to several educational design projects we have illustrated a powerful idea of how to employ innovative features of Imagine Logo to support exploratory and collaborative learning. We (either alone or together with our partners in the projects mentioned above) have conducted research activities, which show interesting results. While testing the Space Travel Games Construction Kit, for example, gauges proved very successful in two player game – the ability to attach gauges to the opponent's Lander was a particularly successful feature, enabling one group to make a close comparison with the other and to adjust the strategy second by second, see [Ka06a].
Several experiments have been run during the development of the Visual Fractions at the universities of Warwick, Budapest and Bratislava, inspecting affordances of the interface, functional design of the tools, fraction representations and their control items – how and when were they used by children, how intuitive they are etc. In our own experiments with teachers and children we have come to rather interesting observation. Even if Visual Fractions is seemingly non-programming environment, using it in a constructionist way develops (and requires) certain level of algorithmic thinking. When teachers and children use it as authors – building models, exploring relations etc. – they develop their algorithmic skills in several aspects, see [LK05].

Bibliography

[BK01] Blaho, A.; Kalas. I.: Object Metaphor Helps Create Simple Logo Projects, Proc. of EuroLogo 2001, A Turtle Odyssey, Linz 2001, pp. 55 – 65

[Ka06a] Kahn, K.; Noss, R.; Hoyles, C.; Jones, D.: Designing for diversity through web-based layered learning: a Prototype Space Travel Games Construction Kit, CD Proc of 17th ICMI Study Conference Technology Revisited, HUT, Vietnam 2006

[Ka06b] Kalas, I.: Discovering informatics fundamentals trough interactive interfaces for learning. In: Informatics Education – The Bridge between Using and Understanding Computers. Berlin, Springer-Verlag LNCS 4226, 2006. pp. 13 – 24

[KB00] Kalas, I.; Blaho, A.: Imagine New Generation of Logo: Programmable Pictures, Proc. of WCC2000, Educational Uses of ICT, Beijing 2000, pp. 427 − 430

[KB03] Kalas, I.; Blaho, A.: Exploring visible mathematics with Imagine: Building new mathematical cultures with a powerful computational system. In: Marshall, G., Katz, Y.: Learning in School, Home and Community. IFIP Kluwer 2003, pp. 53 − 64

[KKH97] Kynigos, Ch.; Koutlis, M.; Hadzilacos, T.: Mathematics with component-oriented exploratory software, Int J of Computers for Mathematical Learning, Springer, Vol 2, No 3, 1997, pp. 229 – 250

[KW06] Kalas, I.; Winczer, M.: Building interfaces for on-line collaborative learning, Int. J Education and Information Technologies, 2006; 11 (3), pp. 371 – 384

[LK05] Lehotska, D.; Kalas, I.: LVF – Interface for Dynamic Mathematics, Proc of the 7th Int Conference on Technology in Mathematics Teaching. Bristol 2005, pp. 108 − 116

[Pa93] Papert, S.: The Children's Machine. Rethinking School in the Age of the Computer. Basic Books, 1993, New York.

[Pi73] Piaget, J.; Memory and intelligence: New York: BasicBooks.

[To03] Tomcsanyi, P.: Implementing object dependencies in Imagine Logo, Proc. of EuroLogo 2003, Porto, pp. 127 − 140

[Wi00] Wieserma, N.: How does Collaborative Learning actually work in a classroom and how do students react to it? Retrieved June 6, 2007, from www.city.londonmet.ac.uk/deliberations/collab.learning/wiersema.html

Strictly models and objects first – Unterrichtskonzept für objektorientierte Modellierung

Ira Diethelm

Gaußschule, Löwenwall 18a, 38100 Braunschweig
und
Universität Kassel, Wilhelmshöher Allee 73, 34121 Kassel
ira.diethelm@uni-kassel.de

Abstract: In diesem Artikel fasse ich die wesentlichen Überlegungen zur Entwicklung von Leitideen für ein Unterrichtskonzept zur OOM im Rahmen meiner Dissertation zusammen und stelle die Ableitung der dafür nötigen Eigenschaften von hilfreichen Unterrichtsmethoden vor. Die ausführliche Beschreibung der Unterrichtsmethoden finden Sie in [Di07].

1 Einleitung

Objektorientierte Modellierung (OOM) im Unterricht ist immer noch ein breit diskutiertes Thema - in der Didaktik akzeptiert und gewünscht, von der Praxis oft als unnötiger Overhead oder als schlicht zu komplex empfunden. Ich habe mich daher gefragt, wie man OOM in der Praxis vereinfachen kann. Ausgehend von den in der Literatur dokumentierten Konzepten zur OOM und ihren Kritikpunkten habe ich in [Di07] ein Unterrichtskonzept entwickelt, das aus Erkenntnissen der Lernpsychologie, allgemeiner Didaktik, Fachdidaktik und auch der Softwaretechnik Unterrichtsmethoden herleitet, um den berichteten Schwierigkeiten wie z.B. dem „Lernen auf Vorrat“ zu begegnen.

Mein Konzept folgt vier Leitideen: models first, strictly objects first, Nachvollziehbarkeit und Ausführbarkeit. Die strikte Umsetzung dieser Ideen führte zu einem Unterrichtskonzept, das einerseits von Beginn an das Ziel der Modellierung berücksichtigt und oft von der dynamischen Sicht des Problems ausgeht. Da es weitgehend auf der grafischen Modellierungsebene verbleibt, werden viele Probleme eines Programmierkurses vermieden und dennoch entstehen als Ergebnis der Modellierung ausführbare Programme.

In diesem Beitrag stelle ich zunächst die Leitideen des Konzepts vor, bevor ich nach Überlegungen zu Werkzeugen und Beispielen in der zweiten Hälfte dieses Beitrags die daraus entwickelten Unterrichtsmethoden für signifikante Abschnitte im Unterricht in OOM erläutere. Da die meisten Unterrichtsmethoden bereits ausführlich in [DGZ02, DGZ03, DGZ05a, DGZ05b, DGS06] beschrieben wurden, werde ich diese hier aus Platzgründen kurz fassen. Ich schließe mit einem Fazit und Ausblick.

2 Entwicklung der Leitideen

2.1 Ausgangslage

Auf den ersten Blick scheinen sich die Aussagen, dass einerseits objektorientierte Modellierung an den Anfang zu setzen sei und lerntheoretische Vorteile hat, vgl. z.B. [Sc95], mit denen zu widersprechen, die von den Schwierigkeiten in der Umsetzung berichten, vgl. z.B. [Sp01]. Wie kann etwas intuitiv sein, das so viele Lernhürden für die Schüler und Schwierigkeiten beim Unterrichten für die Lehrer mit sich bringt? Muss man aus Problemen beim Unterricht in OOM schließen, dass die erste Annahme falsch ist? Ist somit objektorientiertes Denken weder leicht noch wünschenswert?

Laut Holland et al in [HGW97] ist ein objektorientiertes Unterrichtskonzept nicht von Haus aus schwierig, es gebe nur viele Möglichkeiten für den Lehrer, etwas falsch zu machen und damit bei den Schülern Fehlvorstellungen hervorzurufen. Dies bedeutet also, dass die berichteten Probleme ihre Ursachen in den benutzen Werkzeugen und der Methodik haben, mit denen OOM gelehrt wird. Um dies zu verbessern, bin ich zwei Fragen nachgegangen:

1. Welche Kriterien muss eine Unterrichtskonzept und dessen Methodik zur objektorientierten Modellierung erfüllen, um die lerntheoretischen Vorteile der OOM zu unterstützen und die genannten Probleme von vornherein zu reduzieren?

2. Welche methodischen Hilfen können aus den genannten Erfahrungen, der Lerntheorie, der allgemeinen Didaktik und auch aus den Erkenntnissen der Softwaretechnik hergeleitet werden, um bestimmte Lernschwierigkeiten beim Unterricht in OOM gezielt zu vermindern?

2.2 Models first

Objektorientierte Modellierung ist nicht dasselbe ist wie objektorientierte Programmierung und diese sollte wiederum nicht als Erweiterung der strukturierten Programmierung verstanden werden. Der Umstieg von einer strukturierten auf eine objektorientierte Programmiersprache wurde in mehreren Studien als langwierig und fehlerträchtig beschrieben, vgl. z.B. [Be00]. Die Bedeutung des ersten Eindrucks, der ersten Programmiersprache, auf die Bildung der mentalen Modelle der Schüler sollte daher nicht unterschätzt werden und verdient somit große Beachtung bei der Entwicklung eines Unterrichtskonzepts zur objektorientierten Modellierung, vgl. [Sc95, BB04]. Somit sollte ein erfolgreiches Unterrichtskonzept, das seinen Schwerpunkt auf die objektorientierte „Modellierung“ legt auch mit objektorientierter „Modellierung“ beginnen.

Der Begriff „models first“ bedeutet hier, dass die objektorientierte Modellierung von Anfang an im Informatikunterricht unterrichtet wird, insbesondere, dass nicht nur die Modellierungssprache, sondern auch die Modellierungstechnik von Beginn an gelehrt werden soll. Damit soll das Ziel erreicht werden, dass die Modellierung die Denkweise bei der

Problemlösung bestimmt und durch den ersten Eindruck so gewissermaßen zur Muttersprache des informatischen Problemlösens wird.

2.3 Strictly objects first

Ebenso sollte ein Unterrichtskonzept zur objektorientierten Modellierung aus denselben Gründen bei den Objekten beginnen. Die Analyse vieler Berichte, die OOM als schwieriges Konzept bezeichnen, deuten darauf hin, dass die Schwierigkeiten mit objektorientierter Modellierung und Programmierung daher rühren, dass Objekte zwar die Akteure in den Programmen sind, es aber nicht die Objekte sind, sondern die (meist Java-)Klassen, die implementiert werden. Somit kann leicht Verwirrung bei Schülern bei der Unterscheidung zwischen Klassen und Objekten entstehen.

Hier ist „objects first“ nicht etwa als „classes first“ oder „OOP first“, sondern im wörtlichen Sinne gemeint. Die Modellierung sollte buchstäblich bei den Objekten beginnen, mit deren Eigenschaften, Beziehungen und Fähigkeiten. Die Klassen sollten zweitrangig behandelt werden und sich aus den Objekten ergeben. Letzteres gilt auch für die Modellierung des Verhaltens.

Um dies zu erreichen kann von Anwendungsfällen und Szenarien ausgegangen werden. In den darin beteiligten Objekten sind alle Informationen enthalten, die für die Modellierung wichtig sind. Aus dem Unterschied zwischen der Modellierung einer Anfangssituation mit Objekten und der Modellierung der Endsituation ergibt sich außerdem ein Beispielverhalten in dem gegebenen Szenario, das auch ohne Klassen zumindest verbal beschrieben werden kann. Die Konzentration auf eine klar definierte Situation reduziert zudem die Komplexität im Vergleich zu den vielen möglichen Ausführungen, die bei der Betrachtung der Klassen eine Rolle spielen. Durch diese Verlagerung des Focus wird vermieden, dass die Schüler überfordert werden.

2.4 Nachvollziehbarkeit

An etlichen Unterrichtskonzepten zur OOM wurde die Notwendigkeit des „Lernens auf Vorrat“ kritisiert, z.B. von [Fü99]. Bei Bottom-up-Ansätzen liegt es in der Natur des Konzepts, dass sich aus den gelernten Teilen erst nach einer gewissen Zeit für die Schüler ein Gesamtbild ergibt und sie somit erst spät erfahren, wozu sie bestimmte Konstrukte wie z.B. „public static void main“ bisher gelernt haben und warum diese Konstrukte so aufgebaut sind. Die mithilfe von OOM vermittelte Problemlösestrategie sollte aber von den Schülern subjektiv als konsequente und natürliche Vorgehensweise empfunden werden. Daher sollte beim Aufbau von Unterrichtseinheiten besonders darauf geachtet werden, die Systematik der Modellierung nachvollziehbar zu machen. Gleichzeitig sollen auch Alternativen für Entscheidungen aufgezeigt und verschiedene Wege zugelassen werden, um so die Modellierungsentscheidungen für die Schüler transparenter zu machen und ihnen Modellierung als Gestaltungsprozess mit mehreren Möglichkeiten zu vermitteln und so bei den Schülern Sicherheit im Umgang mit den gelernten Modellierungtechniken zu erzeugen.

Späteres Revidieren von Entscheidungen und die damit verbundene Änderung des Modells treten in der Realität sehr oft während eines Softwareprojektes auf. Somit würde eine Auslassung dieser Phase im Unterricht ein verzerrtes Bild von informatischem Problemlösen vermitteln. Modelle sollten somit im Laufe des Unterrichts zumindest einmal geändert und dem Problem angepasst werden.

2.5 Ausführbarkeit

Modellierung sollte den Modellbildungprozess als Problemlösestrategie unterstützen und im Unterricht nicht als Selbstzweck gelehrt werden. Hierzu muss sie einer Problemlösung entgegenstreben. Die Schüler können sich dann durch Anwendung ihres Modells von dessen Tauglichkeit für die Lösung des Problems überzeugen. Die Anwendbarkeit des Modells auf das gegebene Problem und damit die Güte des Modells wird im Informatikunterricht in der Regel durch die Ausführung des mithilfe des Modells implementierten Programms verdeutlicht. Die darin enthaltene Bestätigung ist zudem ein beträchtlicher Motivationsfaktor. Aber gerade die Implementierung des Modelles mithilfe einer textuellen Programmiersprache stellt oft eine sehr große Hürde dar, die allein durch die Motivation kaum überwunden werden kann. Etwas zu schaffen, das man evtl. sogar mit nach Hause nehmen kann, um es seinen Eltern oder Freunden oder evtl. über das Internet der ganzen Welt zu zeigen, was man selbst erstellt hat, ist eine der großen Chancen des Informatikunterrichts.

3 Auswahlkriterien für Werkzeuge und Beispiele

3.1 Die Wahl der Werkzeuge

Da der bisherige Weg zum ausführbaren Programm stets über eine textuelle Programmiersprache führte und dies offenbar viele Probleme mit sich bringt, habe ich versucht, den Programmtext zu vermeiden. CASE-Tools sind in der Lage aus Klassendiagrammen Quelltext zu erzeugen, jedoch müssen bei fast allen trotzdem anschließend die Methoden und damit die eigentliche Funktionalität noch als Text ausformuliert werden. Erstrebenswert wäre aber ein einheitlicher Weg, der ohne den Wechsel der Repräsentationsebene zum Programmtext auskommt. Dazu müsste auch die Ausführung selbst grafisch sichtbar gemacht werden und auch der Programmablauf und die Fehlersuche müssten in der Modellebene visualisiert werden.

Derzeit ist Fujaba (From UML to Java and back again) das einzige Tool dieser Art, das die objektorientierte Modellierung in dieser Weise konsequent erlaubt. Es gestattet die o.g. Programmierung mit Regeldiagrammen, die aus Aktivitätsdiagrammen mit eingebetteten Kollaborationsdiagrammen bestehen. Mithilfe dieser Diagrammart wird es möglich, die Modellierungsebene auch bei der Herstellung von ausführbaren Programmen beizubehalten und so einen Medienbruch zu vermeiden. Fujaba besitzt zudem diverse Mechanismen zur Ausführung und zum Debuggen der so erzeugten funktionsfähigen Modelle auf der

gleichen Ebene, auf der sie erzeugt wurden, ohne zu irgendeinem Zeitpunkt diese Ebene verlassen und auf die textuelle Ebene wechseln zu müssen. Leider besitzt es aber auch große Anforderungen an die Rechnerresourcen und lässt sich auch aufgrund von einer fehlenden Hilfe und unverständlichen Fehlermeldungen nicht intuitiv bedienen.

3.2 Die Wahl der Beispiele

Die Anforderungen an Beispiele, die die Leitideen unterstützen, lassen sich wie folgt zusammenfassen und genauer spezifizieren, vgl. [HGW97] und [DGS06]: Das Einstiegsbeispiel sollte aus einem den Schülerinnen und Schülern vertrauten Erfahrungsbereich stammen, so komplex sein, dass es das Durchspielen geeigneter Szenarien und Anwendungsfälle erlaubt und komplex genug sein, um Änderungen im Modell zu provozieren z.B. muss es eine Gruppenarbeit ermöglichen, in der voneinander gelernt und miteinander diskutiert wird. Konkret auf die objektorientierte Modellierung bezogen bedeutet dies, dass das Einstiegsbeispiel aus mindestens drei Klassen bestehen sollte, damit deutlich wird, dass Objektorientierung Probleme in Klassen zerlegt. Zwischen den Klassen muss es Assoziationen mit verschiedenen Kardinalitäten geben sollte um die Begrifflichkeit zu unterstützen. Darüber hinaus lassen sich noch weitere Kriterien für Beispiele finden, vgl. [DGS06, Di07].

Viele dieser Anforderungen erfüllt das Beispiel, eine Computerspielversion für das Spiel „Mensch ärgere dich nicht“ zu entwerfen. Man kann guten Gewissens annehmen, dass alle Schüler dieses Spiel kennen und so ist man sich (scheinbar) völlig im Klaren darüber, welche Teile, Regeln und Funktionalität ein Programm für dieses Spiel besitzen muss. Ich habe aber festgestellt, dass auch hier schon genügend Modellierungsentscheidungen getroffen werden müssen. Gestaltungsfragen sind beispielsweise, ob ein oder mehrere Benutzer gemeinsam spielen sollen, welche Aspekte des Spiels automatisiert werden sollen, etc. Es treten aber im Gegensatz zu realitätsnäheren Problemen wie z.B. bei einem elektronischen Kalender hier keine Entscheidungen auf, die die Schüler zu diesem Zeitpunkt mangels Modellierungserfahrung noch gar nicht begründet in die eine oder andere Richtung treffen können und somit dem Lehrer folgen müssten.

Die Motivation wird durch reales Anschauungsmaterial gesteigert aber noch größer, wenn die Schüler ihre Unterrichtsergebnisse zu Hause präsentieren können, wenn also nicht nur das Programm am Ende spielbar ist, sondern z.B. als jar der kleinen Schwester gegeben werden kann. Am größten ist meiner Erfahrung nach die Motivation allerdings, wenn sich ein Roboter bewegt. Lego Mindstorms Roboter z.B. lassen sich auch sehr gut mit OOM beschreiben und erfüllen fast alle der o.g. Kriterien.

4 Entwicklung der Unterrichtsmethoden

Damit die hier vorgestellten Leitideen im Unterricht umgesetzt werden können, muss die OOM näher in Teilbereiche des informatischen Problemlösens im Unterricht aufgeteilt werden. Es lassen sich zunächst die folgenden drei Teilbereiche identifizieren: 1. die

„Einführung der Objekte“ als Anfangspunkt der objektorientierte Modellierung, 2. das Modellieren des Verhaltens und somit dem „Entwurf von Methoden“. Hier muss ausgehend von einer oder mehreren Ausgangssituationen ein Weg zu einer gewünschten Endsituation gefunden werden. Und 3. müssen die Schüler den „Ablauf der Methoden“ verstehen, um Problemlöseprozess am Beispiel existierender Programme und Methoden zu rekonstruieren oder das eigene Modell überprüfen zu können. Alle drei Bereiche verweben sich zu einem Ganzen bei der „Durchführung eines Projektes“.

Und nicht zuletzt ist die „Einführung einer textuellen Programmiersprache“ nicht nur für die objektorientierte Problemlösung ein wichtiger Bereich. Da das Paradigma und die damit verbundene Modellierungstechnik meist sehr zeitnah mit der Syntax der Programmiersprache unterrichtet werden, besteht beim Lernen dieser Dinge die Gefahr der Überlagerung durch zu große inhaltliche Nähe (Ähnlichkeitsinterferenz).

4.1 Einführung der Objekte

Die Einführung des Objektbegriffs sollte entsprechend der zweiten Leitidee bei den Objekten beginnen. Also sollte dies bereits ab der ersten Unterrichtsstunde thematisiert werden. Mit Handlungen mit realen Objekten kann das mentale Modell der Schüler auf den Objektbegriff vorbereitet werden. Hier werden die später zu modellierenden Situationen mit den realen Objekten nachgespielt. Ist z.B. ein Brettspiel Thema der einführenden Unterrichtseinheit, so könnte die Schüler zunächst ein paar Züge dieses Brettspiels in der Gruppe mit einem echten Spielbrett spielen. Auf diese Weise erfolgt nicht nur eine textuelle/auditive, sondern auch eine visuelle und handlungsmäßige Einführung in den Kontext, der zu modellieren ist. Auf allen drei mentalen Repräsentationsebenen werden so die Grundlagen für die kommende neue Information geschaffen.

4.1.1 Vom Objekt zur Klasse

Nach dieser Vorbereitung kann die Modellierung mithilfe von Objektdiagrammen eingeführt werden, in dem die für die konkrete Situation wichtigen realen Objekte genannt und in einem ersten Objektdiagramm festgehalten werden. Bei diesem Vorgehen halte ich einen ständigen Abgleich des Modells mit der betrachteten Situation für sehr wichtig, um die Nachvollziehbarkeit der gelehrten Vorgehensweise zu gewährleisten. Ausgehend vom Sprachgebrauch, der bereits bestimmte Gegenstände ähnlicher Natur zu einem Oberbegriff zusammenfasst und mit Blick auf das entstandene Objektdiagramm können dann die Klassen, immer noch anhand des betrachteten Beispiels, eingeführt werden, vgl. [DGZ05b].

Die Ableitung des Klassendiagramms aus Objektdiagrammen sollte sehr systematisch und nachvollziehbar geschehen. Das Verfahren sollte es den Schülern ermöglichen, Schritt für Schritt zum Klassendiagramm zu kommen ohne auf Erfahrung zurückgreifen zu müssen. Daher ist auch hier die strenge Begrenzung des Vorgehens auf genau die eine betrachtete Situation sehr wichtig.

Wird dieses Vorgehen anschließend für viele verschiedene Situationen desselben Zusammenhangs jeweils in einer Gruppe pro Situation geübt, kann die Systematik bei den Schü-

lern gefestigt werden. In der Kleingruppe können die Schüler untereinander erste Erfahrungen in dieser Modellierungstechnik erwerben und bei Unklarheiten kurz bei ihren Mitschülern nachfragen. Bei einer anschließenden Vorstellung der Gruppenergebnisse und Diskussion in der Klasse müssen sich die Gruppen auf ein gemeinsames Klassendiagramm einigen, da die einzelnen Klassendiagramme der Gruppen zum Teil eine recht große Überlappung in der modellierten Information, nicht immer aber in der gewählten Darstellung bei der Modellierung aufweisen. Die Schüler müssen so ihre Entscheidungen vor ihren Mitschülern verantworten und begründen, vgl. [Di07].

4.1.2 Vererbung

Durch komplexere Konzepte der Objektorientierung wie Aggregation und Komposition, Vererbung und Design Pattern wird das Projekt und damit die Problemlösung übersichtlicher und wiederverwendbar. In einer frühen Phase des Unterrichts, in der sie noch nicht oder wenig das Verhalten der Objekte betrachtet haben, ist die Behandlung dieser Konzepte jedoch nicht nötig. Die Schüler würden sie als leeres Konstrukt oder nur zur Ersparnis von Quelltext verwenden, ohne den Nutzen dieser Strukturierungsmöglichkeit zu kennen.

Die Vererbung fasst Klassen mit ähnlichen Eigenschaften zusammen. Um diesen Abstraktionsschritt leisten zu können, benötigen die Schüler aber bereits eine gewisse Sicherheit im objektorientierten Modellieren. Daher sollte sich die Vererbung nicht direkt an die Einführung von Klassen anschließen, sondern zu einem späteren Zeitpunkt, nach der Erarbeitung von einigen Methoden, an einem prägnanten Beispiel eingeführt werden, bei dem die Ähnlichkeit und damit eine Verwandtschaft der Klassen durch das Verhalten deutlich wird.

4.2 Methoden entwerfen

Auch beim Entwurf von Methoden sollte meiner Einschätzung nach aus denselben Gründen wie bei der Einführung der Objekte der Unterricht immer von den Objekten ausgehen und die Modellierung in den Vordergrund stellen. Um dies und die Nachvollziehbarkeit auch bei unterschiedlich schwierigen Methoden zu unterstützen, sollte für die Modellierung der Methoden eine Schreibweise verwendet werden, die die Objekte und ihre Beziehungen zueinander und den Kontrollfluss leicht verständlich darstellt. Die Regeldiagramme, die Fujaba zur Implementierung von Methoden anbietet, entsprechen diesen Anforderungen. Da Fujaba aus diesen Diagrammen Java-Quelltext generiert, wird hier zusätzlich die vierte Leitidee erfüllt, dass die erstellten Modelle ausführbar sein sollten.

Das Objektspiel, vgl. auch [Sc04, BS05, DGZ05b], veranschaulicht das Zusammenspiel der Objekte. Jeder Schüler nimmt die Rolle eines Objektes ein. Dadurch gewinnen die Schüler Erkenntnisse über die Modellwelt und vervollständigen ihr mentales Modell, indem sie gezwungen sind, sich aktiv als Teil dieser Modellwelt zu verhalten., vgl. [DGS06].

Bei komplexeren Methoden kann das Objektspiel in der grob skizzierten Form nur Hinweise geben. Es wird erforderlich, dass sich die Schüler genau in die Rolle des Objektes versetzen: Üblicherweise besitzt der Schüler in der Entwurfsphase eine Übersicht über

die gesamte Ausgangssituation, in der die Methode aufgerufen werden soll, z.B. in Form eines Objektdiagramms. Sein Verhalten im Objektspiel kann durch diese Draufsicht beeinträchtigt werden, da das Objekt, dessen Rolle er einnimmt, diese Draufsicht nicht hat, vgl. [DGZ05a]. Es fällt den Schülern schwer, sich vorzustellen, welche Möglichkeiten diese haben und welchen Einschränkungen sie unterliegen.

Nicht nur bei der Konstruktion eines neuen Systems als Lösung für ein Problem, sondern auch bei der Analyse und Dekonstruktion eines bestehenden Systems muss sich der Schüler in einer Erkundungsphase in die Lage des Computers versetzen und sich dessen beschränkte Möglichkeiten bewusst machen: „Mit welchen Werten, anderen Objekten, Methoden kann ich hier Änderungen durchführen?“, „Wie finde ich benötigte Informationen / Partnerobjekte?“. Zum besseren Verständnis des Verhaltens der Objekte ist also ein Perspektivwechsel nötig.

Ein Objektspiel ist geeignet, um diesen Perspektivwechsel zu fördern, jedoch erfordert es zusätzliche Maßnahmen, um die Problematik anschaulicher zu machen. Erteilt man den Schülern den Auftrag, die Augen während des Objektspiels zu schließen oder verbindet man ihnen die Augen, können sie die Beschränktheit der Möglichkeiten eines Objekts am eigenen Leib erfahren. Aus den gesammelten Erfahrungen können die Schüler in der gemeinsamen Diskussion ein allgemeines Vorgehen für verschiedene Ausgangssituationen für den gleichen Methodenaufruf erarbeiten und somit die Methode nachvollziehbar entwerfen, vgl. Abb. 1 links und [DGZ05a].

Nicht alle Methoden lassen sich auf diese Weise von Schülern leicht erarbeiten. Bei dem Entwurf von Methoden, die einen rekursiven Aufruf beinhalten werden, ist die zugrundeliegende Systematik auch durch verbundene Augen nicht leicht aufzudecken. Hier können aber Vorgehensweisen aus der Softwaretechnik helfen, wie z.B. das Story-Driven-Modeling vgl. z.B. [DGZ02].

4.3 Den Ablauf von Methoden verstehen

Nach dem Entwurf eigener Methoden oder währenddessen sollten die Schüler in einer Testphase genau die Arbeitsweise des Programms nachvollziehen, um zu überprüfen, ob die Lösung der aktuellen Problemstellung entspricht und um ggf. Fehler zu finden. Auch bei der Dekonstruktion fremder, gegebener Programme, müssen die Schüler die Funktionsweise einzelner Programmteile genau nachvollziehen, um sich ihrerseits eine innere Repräsentation des untersuchten Gegenstands zu konstruieren.

Die Übertragung von imperativen Methoden zur Darstellung des Ablaufs eines Programms wie einer Tracetabelle auf objektorientierte Programmierung wirft jedoch immer dann Probleme auf, wenn nicht nur Attributwerte geändert werden, sondern auch die Objektstruktur verändert wird. Wenn z.B. ein neuer Link erzeugt wird, kann dies mit solchen Tabellen nicht ausreichend protokolliert werden. Auch Ausgaben unterstützen nur begrenzt die Anschaulichkeit dieser Abläufe. Möglicherweise sind gerade diese Unzulänglichkeiten der bewährten Methoden der imperativen Programmierung die Gründe, wieso Objektorientierung als schwer und als für Anfänger ungeeignet angesehen wird.

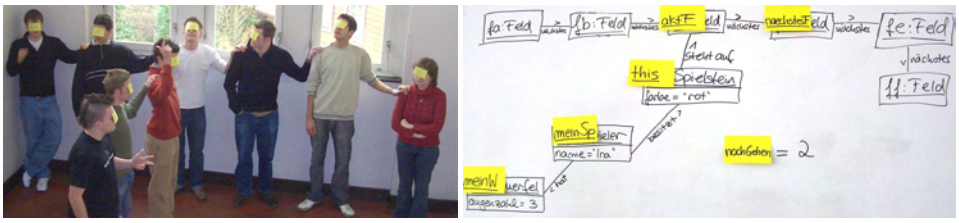


Abbildung 1: Objektspiel und Zetteltest

Um den Ablauf von Programmen objektorientiert darzustellen, kann man die Tracetabelle gegen Klebezettel (z.B. Post-its) und eine Kamera eintauschen, vgl. Abb. 1 rechts und [DGZ05a]. Dieser grafische Durchgang durch den Ablauf einer Methode bietet dem Schüler einerseits Hilfen an, sein mentales Modell des Ablaufs der Methode auszubilden. Andererseits kann er sein vorhandenes mentales Modell mit dieser Darstellung abgleichen. Ist es ähnlich, erhält der Schüler eine Bestätigung und somit eine positive Rückmeldung. Dies stärkt die Sicherheit im Entwurf von Methoden und erhöht die Motivation.

Carsten Schulte nutzte als Alternative zu Meldungen auf der Standardausgabe in [Sc04] bereits das Dynamic Object Browsing System (Dobs), das eine Erweiterung von Fujaba ist und Testen auf Modellierungsebene erlaubt. Dort lassen sich Objektstrukturen beliebig anlegen und Methoden direkt auf den Objekten ausführen.

4.4 Ein Projekt durchführen

Beim handlungsorientierten wie beim projektorientierten Unterricht wird mit den Schülern ein Handlungsprodukt vereinbart. Im Informatikunterricht der dem objektorientierten Paradigma folgt, ist dies zumeist ein Programm, das aus mehreren Methoden und Klassen besteht. Ein vom Handlungsprodukt geleiteter Unterricht ist in anderen Fächern ein sehr motivierender Unterricht, da das Handlungsprodukt sichtbar und fassbar ist, z.B. als Kollage oder als Gegenstand. Im Informatikunterricht wird die Motivation nicht automatisch vom sichtbaren Entstehungsprozess genährt, da leicht die Gefahr besteht, dass stundenlang keine Verbesserung sichtbar wird, obwohl die Schüler sehr viel Mühe in das Projekt stecken.

Ein Informatik-Unterrichtsprojekt sollte zusätzlich softwaretechnische Aspekte eines Projektes vermitteln, also auf ein Produkt zustreben, das solche Teilerfolge sichtbar macht und dadurch den Lernprozess steuert, an dem alle Schüler gleichberechtigt beteiligt sind. Dabei sollten gleichzeitig die typischen Phasen der Softwareentwicklung durchlaufen werden. Zudem sollte dies in Gruppen geschehen und der äußere Rahmen des Projekts so gestaltet sein, dass die Schüler ihren Lernprozess und damit auch den zeitlichen Rahmen selbst steuern können. In einem normalen Schulalltag sind all diese Anforderungen nicht zu erfüllen, daher ist es hier wünschenswert das Projekt über mindestens 2 Tage außerhalb der Schule in einer Umgebung durchzuführen, in der jede Gruppe ihren eigenen Raum besitzt und auch das weitere äußere Umfeld wenig den Lernprozess stört, sondern eher dem Alltag eines Softwareentwicklers entspricht.

Da im Sinne eines handlungsorientierten Unterrichts das Produkt des Projekts für die Schüler greifbar sein und auch „mit der Hand“ entstehen sollte, sollte es im Gegensatz zu einem normalen Programm teilweise Materie besitzen. Roboter, insbesondere Lego Mindstorms Roboter, erfüllen diese Anforderung. Bei der Programmierung dieser Roboter mit Fujaba können Teile des Programms als Methoden direkt im Dobs aufgerufen und so leichter getestet und Teilziele sichtbar gemacht werden als mit anderen Werkzeugen, bei denen das Programm vor dem Test auf den Roboter übertragen werden muss, vgl. [DGZ03].

4.5 Eine textuelle Programmiersprache erlernen

Wählt man für die Modellierungsebene eine grafische Repräsentation statt einer textuellen, erhält man die Chance das Erlernen einer textuellen Programmiersprache von dem Erlernen der Modellierungstechnik zu trennen. Will man anschließend die grafische Repräsentationsebene der Modellierung verlassen und zur textuellen Programmierung im Unterricht wechseln, entsteht ein Bruch in der medialen Repräsentation, der sich aber meiner Einschätzung nach durch die bereits vorhandenen mentalen Modelle und Modellierungsfähigkeiten der Schüler sanft vollziehen lässt, wenn man sich hier der Erkenntnisse der Fremdsprachendidaktik bedient und die Schüler die Programmiersprache aus dem Kontext heraus selbst erarbeiten lässt.

Dieses Vorgehen forderte bereits Wilhelm Viëtor 1882 bei der Einführung von fremdsprachlichen Pflichtunterricht an deutschen Gymnasien bzw. deren Vorgängern (den Realgymnasien): „Und wenn es euch gelänge, [dem Lerner] die beste Grammatik und das umfassendste Wörterbuch in den Kopf zu schaffen, so hätte er noch immer keine Sprache gelernt!“, vgl. [Vi84]. Er kritisiert hier die bis dahin im Fremdsprachenunterricht vorherrschende und mühsame erklärungsbasierte synthetisch-deduktive Unterrichtsmethode und fordert ein von der Beobachtung und dem Beispiel ausgehendes, induktives Lernen. In Bezug auf die Verarbeitungsrichtung stellte z.B. Butzkamm in [Bu93] fest, dass beim natürlichen Spracherwerb „datengeleitete bottom-up-Prozesse[1]„ (von der Wahrnehmung zur kognitiven Repräsentation) vorherrschen.

Eine Unterrichtsmethode, die diesem induktiven Lernen entspricht, könnte im Anschluss an die anderen hier vorgestellten Vorgehensweisen, die Quelltexterzeugung eines CASE-Tools wie Fujaba als zentrales Hilfsmittel einsetzen. Die Schüler erzeugen zunächst einfache Konstrukte (Klassen, Attribute inkl. get- und set-Methoden, eine 1:1-Assoziation, eine leere neue Methode, ...) und betrachten die „Übersetzung“ im Quelltext. Aus verschiedenen ähnlichen Situationen leiten sie induktiv eine Regel her (die „Grammatik“ der Programmiersprache). Diese Kognitivierung des Aufbaus der Sprache unterstützt das Lernen, da die Schüler hier aktiv Verknüpfungen zwischen der symbolischen und der bildhaften Repräsentation von Wissen innerhalb ihres mentalen Modells herstellen.

[1] Leider widerspricht hier die Begriffsbildung der Fremdsprachendidaktik der der Informatik, weil sie bottom und top mit sensorischer Wahrnehmung (z.B. Hände, unten) und Kognition im Kopf (oben) gleichsetzen. Ich würde das deduktive Lernen eher als bottom-up und das induktive als top-down bezeichnen.

5 Fazit und Ausblick

Ich habe in [Di07] ein Unterrichtskonzept entwickelt, das viele Schwierigkeiten bei der Einführung in die objektorientierte Modellierung im Unterricht vermeidet. Hierzu habe ich zunächst Kriterien in Form von Leitideen erarbeitet, um die lerntheoretischen Vorteile der objektorientierten Modellierung nutzen zu können. Auf der Basis von gezielt ausgewählten Lerntheorien habe ich für Teilbereiche der OOM im Unterricht geeignete Unterrichtsmethoden entwickelt.

Meine Erfahrungen mit den Unterrichtsmethoden aus der Praxis lassen vermuten, dass die zu Beginn aufgestellte Hypothese, dass die berichteten Schwierigkeiten mit OOM nicht aus der didaktischen Entscheidung herrühren, sondern aus der methodischen Umsetzung der OOM, richtig war. Ferner schließe ich, dass die berichteten Schwierigkeiten zu einem großen Teil mithilfe der abgeleiteten Leitideen und den daraus entwickelten Unterrichtsmethoden vermieden werden. Allerdings ist das verwendete Werkzeug Fujaba noch stark verbesserungswürdig. So fehlt dringend eine leicht zu bedienende Möglichkeit, aus dem Modell eine grafische Oberfläche zu konstruieren und eine ausführbare Datei zur Weitergabe an Freunde zu erzeugen. Auch müsste es in seiner Bedienbarkeit viel intuitiver werden, um die Kreativität von Schülern angemessen nutzen zu können. Meine Beobachtungen zeigen, dass meine Leitideen wichtige Kriterien für Unterricht in OOM aufzeigen, diese aber aufgrund des noch eingeschränkt nutzbaren Werkzeugs noch nicht in vollem Umfang umgesetzt werden konnten.

Außerdem müsste in empirischen Studien versucht werden zu klären, inwiefern die Schüler durch die hier vorgestellten Unterrichtsmethoden tatsächlich bessere Abstraktionsfähigkeiten erhalten oder ob sie nur bestimmte „Kochrezepte“ ausführen. In Zusammenarbeit mit diesen Studien könnten allgemeine Kriterien entwickelt werden, die für Lernumgebungen und IDEs für den Unterricht gelten müssen, damit diese nicht nur die hier vorgestellten Unterrichtsmethoden unterstützen.

Wenn solche Lernumgebungen in einer größeren Anzahl als eins vorliegen, stellt sich in Zukunft die Frage nach der Sinnhaftigkeit und dem Stellenwert von Quelltext im Unterricht noch viel mehr als heutzutage.

Literaturverzeichnis

[BB04] Helmut Balzert und Heide Balzert. Modellieren oder Programmieren oder beides? LOG IN, 128/129, Seiten 20–25, 2004.

[Be00] Joseph Bergin. Why Procedural is the Wrong First Paradigm if OOP is the Goal, 2000.

[BS05] Jürgen Börstler und Carsten Schulte. Teaching Object Oriented Modelling with CRC-Cards and Roleplaying Games. In 8th IFIP World Conference on Comuters in Education, Cape Town, South Africa, July 2005.

[Bu93] Wolfgang Butzkamm. Psycholinguistik des Fremdsprachenunterrichts. Francke, Tübingen, 1993.

[DGS06] Ira Diethelm, Leif Geiger und Carsten Schulte. Einführung in die Objektorientierung im Informatik-Anfangsunterricht. `www.se.eecs.uni-kassel.de/se/fileadmin/se/LehrerFortbildung/OOM-Skript061108.pdf`, 2006.

[DGZ02] Ira Diethelm, Leif Geiger und Albert Zündorf. UML im Unterricht: Systematische objektorientierte Problemlösung mit Hilfe von Szenarien am Beispiel der Türme von Hanoi. In Sigrid Schubert, Johannes Magenheim, Peter Hubwieser und Torsten Brinda, Hrsg., Erster Workshop der GI-Fachgruppe Didaktik der Informatik, Bommerholz, Germany, 2002.

[DGZ03] Ira Diethelm, Leif Geiger und Albert Zündorf. Fujaba goes Mindstorms: Objektorietierte Modellierung zum Anfassen. In Peter Hubwieser, Hrsg., Informatische Fachkonzepte im Unterricht, Informatik und Schule INFOS 03, München, Germany, 2003.

[DGZ05a] Ira Diethelm, Leif Geiger und Albert Zündorf. Mit Klebezettel und Augenbinde durch die Objektwelt. In Steffen Friedrich, Hrsg., Unterrichtskonzepte für informatische Bildung, Informatik und Schule INFOS 05, Dresden, 2005.

[DGZ05b] Ira Diethelm, Leif Geiger und Albert Zündorf. Teaching Modeling with Objects First. In 8th IFIP World Conference on Comuters in Education, Cape Town, South Africa, 2005.

[Di07] Ira Diethelm. „Strictly models and objects first“ – Unterrichtskonzept und -methodik für objektorientierte Modellierung im Informatikunterricht. Dissertation, Universität Kassel, Fachbereich Elektrotechnik / Informatik, Mai 2007. im Druck.

[Fü99] Klaus Füller. Objektorientiertes Programmieren in der Schulpraxis. In Andreas Schwill, Hrsg., Informatik und Schule, INFOS 99, Seiten 190–201. Springer Verlag, 1999.

[HGW97] Simon Holland, Robert Griffiths und Mark Woodman. Avoiding object misconceptions. SIGCSE Bull., 29(1):131–134, 1997.

[Sc95] Andreas Schwill. Programmierstile im Anfangsunterricht. In Sigrid Schubert, Hrsg., Innovative Konzepte für die Ausbildung, 6. GI-Fachtagung Informatik und Schule INFOS 95, Seiten 178–187. Springer Verlag, 1995.

[Sc04] Carsten Schulte. Lehr- Lernprozesse im Informatik-Anfangsunterricht: theoriegeleitete Entwicklung und Evaluation eines Unterrichtskonzepts zur Objektorientierung in der Sekundarstufe II. Dissertation, Universität Paderborn, Didaktik der Informatik, Fakultät für Elektrotechnik, Informatik und Mathematik, März 2004.

[Sp01] Siegfried Spolwig. Methodische Probleme mit OOP im Anfangsunterricht, 2001.

[Vi84] Wilhelm Viëtor. Der Sprachunterricht muss umkehren : ein Pamphlet aus dem 19. Jahrhundert neu gelesen. Hueber, München, 1984. erstmals in Forum Sprache, 1882.

Kriterien kreativen Informatikunterrichts

Ralf Romeike

Didaktik der Informatik, Universität Potsdam
August-Bebel-Str. 89, 14482 Potsdam
romeike@cs.uni-potsdam.de

Abstract: Kreativität spielt in der Informatik und im Informatikunterricht eine wichtige Rolle. Nur wenige Arbeiten der Fachdidaktik setzen sich dagegen mit diesem Thema auseinander. In diesem Artikel werden bezüglich der Unterrichtsvoraussetzungen, Aufgabenstellungen, Schülertätigkeiten und Unterrichtsumgebung Kriterien aufgestellt, die für einen kreativen Informatikunterricht erfüllt sein sollten. Unterrichtsskizzen aus der Zeitschrift LOG IN wurden anhand der Kriterien untersucht. Nur wenige berücksichtigen kreatives Arbeiten; viele zeigen Ansätze dafür. Mit generellen Empfehlungen für einen kreativen Informatikunterricht schließen wir diesen Beitrag ab.

1 Einleitung

Bereits 1950 fragte Guilford [Gu50], warum Schulen nicht mehr kreative Persönlichkeiten hervorbringen. Wie empirische Studien zeigen, bemängeln auch heutige Berufstätige, dass in ihrer Lehrzeit wichtige Sozialkompetenzen, insbesondere Kreativität, zu wenig gefördert wurden (vgl. [Fe96]). Aus der Wirtschaft wird ebenfalls mangelnde Kreativität bei Schulabgängern angemahnt (vgl. [Ge06]). In der Schule scheint ein besonderer Wert auf Gedächtnisleistungen und analytische Fähigkeiten gelegt zu werden, wobei doch kreative und praktische Fähigkeiten genauso wichtig für den Erfolg im Leben sind, vielleicht sogar wichtiger (vgl. [St03]). Wir denken, dass das Schulfach Informatik eine gute Plattform zur Förderung von Kreativität bietet. Auch wenn eine mögliche Übertragung von Kreativität aus einem Gebiet auf ein anderes in der psychologischen Forschung noch umstritten ist [St04], sollte auch fachspezifische Kreativität nach Möglichkeit gefördert werden. Dazu kann der Informatikunterricht einen wichtigen Beitrag leisten, vergleichbar mit Fächern wie Musik, Kunst oder Sprache, vielleicht sogar darüber hinaus.

2 Zum Kreativitätsbegriff

Kreativ ist eine Leistung im Allgemeinen dann, wenn sie neu ist, Seltenheitswert besitzt und nützlich ist[1]. Wie lassen sich solche neuen und seltenen Leistungen in der Schule vermitteln und fördern? Boden [Bo95] beschreibt zwei Ausprägungen kreativer Leistun-

[1] Dieser gemeinsame Nenner findet sich in vielen Artikeln zur Kreativität wieder [St04].

gen: Historische Kreativität (H-Kreativität), welche Ideen bezeichnet, die in der Geschichte neu sind, sowie P-Kreativität für auf eine Person bezogene originelle und neue Ideen. Für die Schule steht die persönliche Kreativität im Zentrum des Interesses. Der Unterschied zwischen einem außergewöhnlich kreativen und einem weniger kreativen Menschen ist damit „nicht irgendeine besondere Fähigkeit, sondern größeres Wissen (in der Form praktisch angewandter Kenntnisse) und der Wille, sich dieses anzueignen und es zu benutzen.“ ([Bo95], S.39) Wir wollen Denken damit als kreativ bezeichnen, wenn es zu persönlich neuen und verwendbaren Ideen, Lösungen oder Erkenntnissen führt (vgl. [RC95]).

3 Kreativität in Informatik und Informatikdidaktik

Die Informatik wird von praktizierenden Informatikern als kreatives Gebiet angesehen: „Die Konstruktion des Produkts erfordert Ideen, viele kleine und manchmal auch einige größere. Jedes Projekt ist ‚Neuland’ - denn sonst könnte man das System bereits kaufen“ ([FHM]). Betont wird, dass – im Gegensatz zu anderen Disziplinen, wie z.B. der Chemie oder Physik – die Realitäten nicht fest sind. An der eigenen Welt kann „mitgebaut“ werden. „In computer science, the limit is your imagination! The more creative you are, the further you are going to get” [EHZ].
Auch für den Informatikunterricht sehen Fachdidaktiker kreatives Potential, wobei vor allem das informatische Modellieren als kreativer Prozess herausgestellt wird (vgl. [Th02, Sc03, Br03]), welcher „Schüler[n] die Möglichkeit [bietet,] kreative Lösungen zu Problemen zu entwickeln, diese zu realisieren und zu überprüfen“ ([Th02], S. 71). Auch Schulte [Sc03] hält es für besonders wichtig, dass Softwareentwicklung, speziell Modellieren, den Schülern als kreativer Prozess deutlich wird. Fraglich bleibt, ob dieses Potential in Unterrichtsmodellen und im Unterricht auch umgesetzt wird. Ansätze und Begründungen für den Informatikunterricht beziehen sich zwar auf das (potentiell kreative) informatische Problemlösen oder Modellieren, allerdings werden bspw. nach dem informationszentrierten Ansatz (vgl. [Hu00]) zwar Modelle erstellt und mit informatischen Mitteln formalisiert, nur unterbleibt bewusst das Entwickeln von Softwareprodukten. Eine Implementierung der Modelle erfolgt wenn, dann nur zu ihrer Prüfung. Dennoch betont Hubwieser den Motivationsaspekt eines lauffähigen Produktes bei den Schülern. An verschiedenen Stellen wird der kreativitätsfördernde Charakter der - im Informatikunterricht häufig zu findenden - Projektarbeit betont. „Projekte erlauben es, die Kreativität und Gestaltungsideen der Schülerinnen und Schüler einzubinden, sie eigene Gestaltungserfahrungen machen zu lassen und deutlich werden zu lassen, dass unterschiedliche Entwürfe denkbar sind und je nach Ziel unterschiedliche Aspekte einer Situation modelliert werden müssen“ ([Sc03], S. 53). Unbeantwortet bleibt allerdings die Frage, wie die Schüler wirkliche Gestaltungserfahrungen machen sollen, wenn sie auf ein vom Lehrer erwartetes, i. d. R. funktionales, oft konvergentes Ergebnis hinarbeiten müssen. Schüler neigen in solchen Situation zu einem Problemvermeideverhalten, d. h. sie wählen sichere, ausgetretene Wege, die risikoarm und erfolgversprechend sind, aber für Kreativität nicht viel Raum lassen (vgl. [SL91, WD95]). Frühe Evaluation und Perfektionsstreben sind in diesem Kontext als Kreativitätshemmer bekannt (vgl. [Os53]).
Mangelndes Bewusstsein für kreative Unterrichtsphasen spiegelt sich auch in den in der LOG IN vorgeschlagenen Unterrichtssequenzen wider. In einer Analyse auf Möglichkei-

ten kreativer Schülertätigkeiten räumten nur wenige der vorgestellten Unterrichtsskizzen kreativen Handlungen explizit Raum ein (vgl. Kap. 5). Mitunter wird sogar gefürchtet, dass Schüler „ihre Kreativität zu viel spielen lassen“ ([Ja06], S. 65). Tatsächlich sind unkreative Schüler im Unterricht für den Lehrer „handlicher“ als kreative Schüler (vgl. [WD95, SL91]), so dass Kreativität mitunter im Unterricht sogar unerwünscht ist.
Wir halten das explizite Einbeziehen von kreativen Unterrichtsphasen für wichtig. Für den Informatikunterricht bedeutet das bspw., statt nur das Erkennen von Prinzipien von Informatiksystemen als Lernziel zu fokussieren, Freiraum für das Gestalten von Informatiksystemen zu lassen und divergente Aufgabenstellungen anzubieten. Implementieren sollte nicht nur zum Testen papiergefundener (oder vorgegebener) Modellierungen dienen, sondern in den Lernprozess aktiv mit eingebunden werden.
Informatik scheint ein Unterrichtsfach zu sein, in dem Kreativität aufgrund der Inhalte und Methoden eine besondere Rolle spielen kann. Häufig treten einzelne Schüler mit besonders kreativen Leistungen im Unterricht hervor (vgl. [Ro06]). Aus Unterrichtsbeobachtungen vermuten wir, dass kreative Phasen die Motivation der Schüler erhöhen können und dass diese selbständige, kreative Auseinandersetzung mit den Unterrichtsinhalten das Verständnis für Konzepte erhöht (vgl. [To81]), so wie im Gegenzug ein Verständnis der Fakten, Methoden und Paradigmen eines Gebiets für kreatives Arbeiten Voraussetzung ist (vgl. [SB94]); Studien zum Programmierenlernen und in anderen Gebieten deuten darauf hin [BR05, LJ05, Hi98] u. a.). Im Folgenden schlagen wir verschiedene Kriterien für die Durchführung kreativer Unterrichtsphasen vor.

4 Kriterien für einen kreativen Informatikunterricht

4.1 Voraussetzungen

Auch wenn der Informatikunterricht es den Schülern erleichtert, im Umgang mit Informatiksystemen kreativ zu sein, ist das Anwenden von kreativen Lehrmethoden im Unterricht notwendig. Wird im Unterricht nicht nur konvergentes sondern auch divergentes Denken stimuliert, werden alle Schüler zur Kreativität angeregt und kreative Schüler weiter animiert.

Relevanz. Eine zentrale Voraussetzung für einen kreativen Unterricht ist ein Unterrichtsgegenstand, der für die Schüler relevant und interessant ist und ansprechend motiviert wird. Interesse und besonders intrinsische Motivation sind entscheidende Faktoren für kreative Leistungen. Es ist demnach notwendig, dass die Schüler vom Thema, mit dem sie sich beschäftigen sollen, angesprochen werden (vgl. [Fa00]). Hierzu sollte das Thema aus der Lebenswelt der Schüler stammen und/oder so aufbereitet sein, dass der Schüler einen persönlichen Bezug dazu herstellen kann. Im Informatikunterricht sollte das leicht sein, wenn ein Ausschnitt der Realität modelliert wird. Möglichst konkret statt abstrakt bedeutet dann auch, wenn möglich auf „Zahlenbeispiele“ zugunsten von „greifbaren“ Beispielen zu verzichten.

Problemlösung oder Produkt. Gardner [Ga93] klassifiziert fünf Typen kreativer Aktivitäten, von denen zwei als informatiktypisch zu bezeichnen sind: Das Lösen eines

speziellen Problems und das Erschaffen eines Produkts. Gardner zählt dabei zum Problemlösen wissenschaftliche und mathematische Fragen genauso wie künstlerische Tätigkeiten wie das Schreiben musikalischer Arrangements. Im Mittelpunkt der Informatik steht der Prozess der Softwareentwicklung und damit auch das Erschaffen von Softwareprodukten. So ist z.B. der Katalog der fundamentalen Ideen nach Schwill am Softwareentwicklungsprozess orientiert (vgl. [SS04]). Natürlich kann sich nicht der gesamte Informatikunterricht auf Problemlösung und das Modellieren von Software beschränken: Das Erwerben von Wissen, vor allem von Konzeptwissen, bildet die Grundlage für jede kreative Tätigkeit. Unserer Ansicht nach sollten allerdings kreative Unterrichtsphasen explizit in der Planung berücksichtigt werden, um dem gelernten Wissen einen konkreten und kreativen Anwendungsbezug zu geben.

4.2 Anforderungen an Aufgabenstellungen

Kreative Leistungen sind nur in eigenständiger Arbeit möglich, welche grundsätzlich durch die zentrale Stellung von Projekten im Informatikunterricht begünstigt wird. Da selbständige Arbeitsphasen im Unterrichtskontext durch den Unterrichtsverlauf bzw. den Lehrer einen Rahmen in Form von Themen und Aufgabenstellungen oder Arbeitsaufträgen bekommen, sind diese hinsichtlich ihres kreativen Potentials genauer zu analysieren.

Subjektive Neuheit. Neuheit ist ein wichtiges Kriterium kreativer Leistungen. In der Schule dürfte allerdings nur selten eine absolut neue Lösung von einem Schüler entwickelt werden. Dennoch kann jeder Schüler *für sich* neue (P-kreative) Produkte und Lösungen entwerfen, wenn ihm bezüglich der Bearbeitung einer Aufgabenstellung kein Lösungsweg oder Muster bekannt ist oder vorliegt. Nicht erfüllt wird dieses Kriterium von der Aufgabe (Negativbeispiel): „Verschlüssele folgende Nachricht mit der (bekannten) Caesar-Verschlüsselung“. Erfüllt wird dieses Kriterium durch die Aufgabenstellung (Positivbeispiel): „Denke Dir ein eigenes Verfahren zum Verschlüsseln eines Textes aus.“

Offenheit. Charakteristisch für kreative Prozesse sind Bestandteile des Problemfindens, Explorierens und Entdeckens (vgl. [Tr80]). So kann bei einer offenen Aufgabenstellung eine ungefähre Zielvorstellung vorhanden sein, die aber nicht klar definiert ist und erst im Prozess festgelegt wird. Stattdessen sind im Informatikunterricht oft Rahmenbedingungen einzuhalten, welche die Richtung vorgeben und durch den Lehrer oder die Aufgabenstellung bestimmt sind. Der Schüler muss sich dann seine Möglichkeiten bewusst machen und sein Betätigungsfeld abstecken. Die Möglichkeit, die Aufgabe selbst mitzugestalten, wirkt zusätzlich motivierend. Positivbeispiel: „Wende deine Kenntnisse aus der Kryptologie bei der Erstellung eines Informatiksystems an“.

Abstufungen

Ist eine offene Zielstellung nicht möglich oder unpraktikabel, lassen sich Abstufungen bzgl. des Bearbeitungswegs und des Ziels vornehmen.

Offenheit. Offene Ergebniserwartung: Auch bei festgelegtem Ziel ist es möglich, unterschiedliche Ergebnisse zu erhalten, wenn diese die Anforderungen erfüllen (divergente Aufgaben). So kann bspw. den Schülern die Wahl gelassen werden, verschiedene Parameter begründet oder nach Belieben gegenüber einem empfohlenen Lösungsweg auszuwählen oder zu variieren. Dieses Vorgehen führt zu einer begrenzten Vielfalt an

Lösungen. Schülern bietet sich dadurch die Möglichkeit, auch andere Lösungen herauszufinden und den Lösungsraum zu erkunden. Beispiel: „Verschlüssele folgenden Text nach dem RSA-Algorithmus. Variiere die Parameter p und q."

Offenheit. Offener Lösungs-/Bearbeitungsweg: Zum kreativen Prozess in der Informatik gehört das Auswählen aus verschiedenen Vorgehensweisen, das Anwenden von Algorithmen, Konzepten und Modellen und das bewusste Entscheiden, wie eine Problemlösung/ein Produkt erarbeitet werden soll - nur Aufgaben, die verschiedene Wege zulassen, ermöglichen den Schülern Gestaltungserfahrungen[2]. Beispiel: „Erstelle ein Textdokument nach folgender Vorlage."

Bearbeitungstiefe. Engagieren sich Schüler kreativ, arbeiten sie selbstgesteuert. Sie entscheiden, wie umfangreich, wie lange und wie intensiv sie sich mit einem Gegenstand/einer Aufgabenstellung auseinandersetzen möchten und welche Qualität ihr Produkt besitzen soll[3]. Dabei kann eine intensive Beschäftigung bis hin zum Forschen auftreten. Um dieses Engagement nicht zu unterbinden oder durch mangelnde Vertiefungsmöglichkeiten abzuschneiden, sollte eine Aufgabe unterschiedliche Bearbeitungstiefen zulassen, z.B. indem das angestrebte Ziel Erweiterungen oder Veränderungen/Optimierungen zulässt. Bei Implementationen steigt die Bearbeitungstiefe z.B. mit der Optimierung von Algorithmen, der Berücksichtigung möglicher Fehleingaben oder umfangreicheren Ausgaben. So lassen sich auch unterschiedliche Leistungsniveaus berücksichtigen.

Konzeptwissen. Ein solides Grundwissen im Betätigungsgebiet ist Voraussetzung für jeden kreativen Prozess[4], da hierauf begründete Modellierungs- und Problemlöseentscheidungen basieren. Faktenwissen und Produktwissen sind in dem Zusammenhang zwar ebenfalls notwendig, aber erst das Anwenden von dahinter stehenden Zusammenhängen und Konzepten ermöglicht ein problemübergreifendes kreatives Denken. Folglich ist insbesondere Konzeptwissen zu vermitteln und den Schülern die Möglichkeit zu bieten, dieses anzuwenden. So ermöglicht bspw. die Kenntnis von Substitution und Transposition bei Verschlüsselungsverfahren gegenüber alleinigem Anwendungs-/ Algorithmenwissen, die jeweiligen Eigenschaften geschickt für ein eigenes Verfahren der Kryptographie anzuwenden und zu beurteilen.

Ideenanregung. Einer kreativen Leistung geht zumeist ein Stimulus voraus. Art, Inhalt, Formulierung und/oder Hintergrund einer Aufgabenstellung können eine solche Anregung darstellen. Die Bedeutung hierfür wird recht schnell deutlich, wenn man z.B. Aussagen von Komponisten betrachtet: Hier ist häufig ein Gefühl, eine Begegnung, ein Erlebnis oder ein Eindruck von außen ausschlaggebend für die Inspiration. Auch im Informatikunterricht soll die Aufgabe dem Schüler reichlich Anstoß geben, Ideen zu entwickeln, z.B. durch den Kontext, in den sie eingebettet ist, oder durch das Anknüpfen an die Erfahrungswelt der Schüler. Das bedeutet, dass der Schüler sich vorstellen kann, wozu bspw. seine zu entwerfende Software eingesetzt werden kann und welche „größeren" Probleme sie löst.

---

[2] Auch konvergente Aufgaben können hier hinzuzählen, falls verschiedene Lösungswege möglich sind.

[3] "Creativity also involves quality of work" [St03] S. 336.

[4] Ohne Wissen ist nach Meinung der Kreativitätsforscher keine kreative Leistung möglich. Um H-kreative Leistungen zu vollbringen, ist eine Expertise von i. d. R. mind. 10 Jahren im Fachgebiet nötig. Es besteht also eine enge Verbindung zwischen Kreativität und Wissen.

4.3 Schülerorientierte Anforderungen

Kreative Beschäftigung kann einen Menschen begeistern, an die Aufgabe fesseln und/oder in einen Flow-Zustand (vgl. [Cs90]) versetzen, in welchem er voll in seiner Betätigung aufgeht. Dieses Phänomen erfahren z.B. viele Programmierer, die sich in Open Source Projekten engagieren [Lu06]. Ziel kreativer Unterrichtsphasen sollte es sein, einem solchen Empfinden oder einem solchen Zustand möglichst nahe zu kommen. Hierzu zählt auch, eine positive Einstellung zur Kreativität zu etablieren. Sternberg und Lubart [SL91] identifizierten folgende Einstellungen als ausschlaggebend für Kreativität: Toleranz für Mehrdeutigkeiten, Willen, Hindernisse zu bewältigen und durchzuhalten, Willen, mit der Aufgabe zu wachsen, Risikobereitschaft und der Glaube an sich selbst. Folgende schülerorientierte Anforderungen begünstigen diese Einstellungen.

Identifikation. Um möglichst stark in seiner Arbeit aufzugehen, soll sich der Schüler mit seiner Beschäftigung identifizieren können (vgl. [Pa93, Hi98]). Der Unterrichtsinhalt muss damit für den Schüler eine Bedeutung besitzen oder zumindest eine Bedeutung erlangen können. Verantwortung für einen Teil eines Softwareprojekts übernehmen zu können, das Gefühl als Experte eingesetzt zu werden und die spätere Möglichkeit, seine Lösung auch präsentieren zu dürfen, können dies unterstützen.

Originalität. Jeder Schüler ist ein eigenes Individuum mit eigenen Ansprüchen, Vorstellungen und Vorlieben. Originalität als Kriterium kreativer Leistungen bedeutet in diesem Zusammenhang, dass sich ein Schüler einen Originalitätsanspruch setzen und erfüllen kann, bspw. indem er seiner Lösung/Bearbeitung eine „eigene Note“ verleiht. Diese kann ästhetische, funktionale, gewitzte oder andere Besonderheiten ausmachen: Z.B. durch die Gestaltung einer GUI, einmalige Programmfunktionen oder spezielle Anwendungsgebiete.

4.4 Anforderungen an die Unterrichtsumgebung

Kreatives Arbeiten ist immer auch von der Umgebung abhängig (vgl. [FT85, Da91]). Ein negatives Unterrichtsklima kann ein deutlicher Kreativitätshemmer sein. Entsprechend sollte der Lehrer (wie in jedem Unterricht) darauf achten, dass die Schüler sich akzeptiert und wohl fühlen. Darüber hinaus stellen kreative Unterrichtsphasen weitere Anforderungen an die Umgebung, um Kreativität zu stimulieren: Das „Zünden“ neuer Ideen, die Ermutigung, kreativen Ideen zu folgen, sowie das Auswerten und Wertschätzen von kreativen Ideen (vgl. [SL91, To81]). Eine Besonderheit stellen hier wieder die Informatikunterrichtslabore dar: Durch die Arbeit am Computer mit Entwicklungs- und Simulationsumgebungen wird den Schülern Experimentieren ermöglicht, sie erhalten direktes (Compiler-)Feedback und können ihre Ergebnisse meist direkt betrachten und analysieren.

Experimentieren. Kreativ tätig sein bedeutet, mit Ideen zu experimentieren, Heuristiken anzuwenden und Lösungsmöglichkeiten zu testen. Ist das Lösungsfinden durch Versuch und Irrtum auch ein Vorgehen, das methodisch im Informatikunterricht oft nicht präferiert wird, so gehört es doch zum kreativen Prozess dazu (vgl. [Gl06, Cu95]) und ermöglicht gerade in der Softwareentwicklung das Aufstellen und Testen von Hypothesen für kleinere Probleme. Experimentieren schließt hierbei nicht das „Nach-

Experimentieren“ gemäß vorgegebener Versuchsanleitungen mit ein, sondern meint das selbstständige Untersuchen und Prüfen von Ideen und Hypothesen.

Zeitlicher Raum. Kreativität ist unter Druck nur schwer realisierbar. Zur Überprüfung und Realisierung von Ideen sowie für die Illumination von Gedankenansätzen wird Zeit benötigt. Da die Schüler in kreativen Unterrichtsphasen selbstgesteuert arbeiten, teilen sie sich ihre Zeit auch selbst ein; Informatikprojekte begünstigen dieses Kriterium.

Unterrichtsklima der Vielfalt. So wie zeitlicher Druck negative Einflüsse haben kann, sind auch Konformitätsdruck (Gruppendenken), erwartete Perfektion (Suche nach der erwarteten Antwort) (vgl. [SL91, WD95]), Hierarchien und frühe Evaluation Kreativitätshemmer. Stattdessen sollte der Unterricht gegenseitige Anregungen und Inspiration ermöglichen (vgl. [Fe88]). Neue Ideen sollten willkommen sein, Misserfolge ermöglicht und vielfältige Lösungen begrüßt und respektiert werden. In kaum einem anderen Unterrichtsfach als Informatik ist es möglich, so viele verschiedene Ergebnisse zu erhalten. Diese sind vorzustellen und zu fördern, auch wenn eine Software am Ende nicht „läuft“.

Lehrerrolle. Während im traditionellen Unterricht der Lehrer den Unterricht leitet, Wissen vermittelt und die Schüler korrigiert und bewertet, besteht in kreativen Unterrichtsphasen seine Aufgabe im Coaching: Sollten Schüler ein nur schwer zu überwindendes Problem haben, nicht weiter wissen oder eine Inspiration benötigen, kann der Lehrer helfen, ansonsten hält er sich – vor allem auch mit wertenden Äußerungen – zurück. Zuspruch und Motivation sind allerdings legitim und notwendig. Damit sollte der Lehrer auch ausdrücken, dass er Kreativität begrüßt und wertschätzt (vgl. [Fa00]). Er berät, begleitet, informiert und unterstützt wo erforderlich und erwünscht.

Kreativitätsorientierte Unterrichtsphasen verlangen die Berücksichtigung dieser genannten Kriterien. Nichtsdestotrotz kann die Beachtung einzelner Kriterien kreatives Arbeiten begünstigen bzw. vorbereiten.

5 Analyse von Unterrichtsbeispielen für den Informatikunterricht

Mit Hilfe der aufgestellten Kriterien wurde untersucht, inwieweit Kreativität im Informatikunterricht bereits berücksichtigt wird. Sucht man nach einer expliziten Thematisierung von Kreativität in der deutschen Fachzeitschrift zum Ideenaustausch von Informatiklehrern LOG IN, findet sich wenig. Allein in einem Heft aus dem Jahr 1995 wurde Kreativität im Zusammenhang mit dem Computer als Werkzeug zur Musik- oder Textgenerierung thematisiert. Die informatikspezifischen Möglichkeiten kreativen Tuns, wie z.B. in der Programmierung, wurden dabei nicht berücksichtigt. Untersucht wurde nun, ob sich kreative Aspekte des Informatikunterrichts vielleicht implizit in den vielen Unterrichtsvorschlägen in LOG IN wiederfinden. Hierzu wurden 144 Unterrichtsvorschläge, die innerhalb der Jahre 1995-2006 vorgestellt wurden, auf Indizien analysiert[5].

Zum Vorgehen

Die Unterrichtsbeispiele beinhalten meist eine inhaltliche und eine methodische Ebene. Für die Untersuchung waren vor allem die methodischen Hinweise, die Unterrichtsempfehlungen sowie – wenn dargestellt – die (Teil-)Lernziele interessant. Schüleraktivität

[5] Die Untersuchung wurde mit einer Gruppe von Lehramtsstudenten des Hauptstudiums durchgeführt, nachdem die Kriterien ausgiebig besprochen wurden.

wird in der Regel durch Arbeitsaufträge oder Aufgaben gesteuert. Solche durch Unterrichtsbeispiele suggerierte Schüleraufgaben wurden genauer betrachtet. Ein großer Teil der Unterrichtsskizzen bezog sich allerdings auf Inhalte und deren didaktische Aufbereitung. Bzgl. des Aufbaus der Vorschläge stellten wir uns die Fragen: Lässt die vorgeschlagene Vorgehensweise die Erfüllung der Kreativitätskriterien zu? Bleibt im Kontext des dargestellten Unterrichts Raum für kreative Phasen? Die Analyse brachte folgende Ergebnisse[6].

Unterrichtsvoraussetzungen

Fast Dreiviertel der Unterrichtsskizzen wählten als Unterrichtsgegenstand Themen, die für die Schüler relevant sein dürften[7] und motivierten diese ansprechend. Publizierende Lehrkräfte sind sich offenbar dem wichtigen Einflussfaktor von Motivation auch auf den allgemeinen Unterricht[8] bewusst. Gezielt werden in vielen Beispielen schülerrelevante Themen ausgewählt, die von Gefahren im Internet bis hin zu gut motivierten Graphenproblemen reichen. Dieses Kriterium wurde z.B. nicht erfüllt von Unterrichtsvorschlägen zur Turingmaschine, zu Teleheimarbeit oder zur Datenmodellierung[9].
Nur gut die Hälfte der Unterrichtsvorschläge lässt den Schülern die Möglichkeit, selbst problemlösend oder gestaltend tätig zu werden. Hier scheinen viele Lehrer v. a. dem Verstehen von Informatiksystemen und dem Aneignen von Wissen einen Vorzug zu geben. Indikatoren für solche Unterrichtsvorschläge sind formulierte rezeptive Lernziele der Art: „Die SuS wissen, dass...“, „Die SuS erkennen, dass...“ oder „Die SuS vollziehen nach, wie...“. Erfüllt wurde dieses Kriterium z.B. von Unterrichtsbeispielen zur Kryptologie oder zur Gestaltung von Internetpräsentationen.

Aufgabenstellung

Subjektiv neue Aufgaben werden in 57% Prozent der Unterrichtsskizzen gestellt. Dieser Wert korrespondiert mit der Art der Aufgabenstellungen: Sollen die Schüler nicht problemlösend oder gestaltend tätig werden, besteht ihre Aufgabe wohl darin, den Lernstoff an Aufgaben zu festigen, die zu den behandelten analog sind, bzw. an ähnlichen Aufgaben nachzuvollziehen. Mitunter reichen einfache Änderungen, um einer Aufgabenstellung Neuigkeitswert zu geben, aber dennoch Gelerntes anwendbar zu machen: „Definiere ausgehend von der hier beschriebenen Ameisenwelt eine Hamsterwelt gemäß folgender Vorstellung...“ [Pr04]. Immerhin zwei Drittel der Unterrichtsskizzen zielen auf die Vermittlung und Anwendung von Konzeptwissen. Dies bedeutet bspw. die Vermittlung der Textverarbeitungsdarstellungen „kursiv“ oder „fett“ als Attribute des Objekts „Zeichen“. Die Vermittlung einer Handlungsfolge zum „Kursiv-machen“ eines markierten Wortes bewirkt dagegen prozedurales Wissen. Gut die Hälfte der Unterrichtsskizzen lässt eine variable Bearbeitungstiefe zu und immerhin die Hälfte regt die Schüler zum Einbringen eigener Ideen an. So bietet eine Unterrichtseinheit zu interaktiven Animationen eine Bandbreite an Möglichkeiten, eigene Ideen einzubringen, während „Beobachtungen an Bildschirmen“ wenig Einfluss zulassen.

[6] Da es sich um eine indizienbezogene Auswertung handelt, werden die Anteilsangaben nur grob dargestellt.
[7] Die tatsächliche Relevanz ist natürlich vom Individuum abhängig. Wir legten bei der Beurteilung die Fragestellungen zu Grunde: Stammt das Thema aus der Lebenswelt der Schüler? Ist das Thema schülerorientiert?
[8] Etwa 20 Prozent der Schülerleistung können der Motivation zugeschrieben werden! (vgl. [As94])
[9] Was nicht bedeutet, dass es unmöglich ist, diese Themen entsprechend zu motivieren.

Der Grad der Offenheit einer Aufgabe ist ein entscheidender Anhaltspunkt dafür, wie viel Kreativität verlangt wird und eingebracht werden kann. Aufgaben mit offener Zielstellung und damit mit Möglichkeiten des Problemfindens regen nur ein Fünftel der vorgestellten Unterrichtseinheiten an (i. d. R. Aufgaben zum Modellieren oder Programmieren in einem bestimmten Themenbereich). Verschiedene Ergebnisse sind nur bei 44 % der Aufgaben möglich, verschiedene Bearbeitungswege bei knapp der Hälfte. Vor dem Hintergrund, dass Modellieren und Problemlösen einen zentralen Stellenwert in der Informatik besitzt und wohl jeder dieser Prozesse in seiner Durchführung variabel ist, scheint dieses Ergebnis unbefriedigend. Konvergente Aufgaben mit strikten Lösungsmustern scheinen den Informatikunterricht zu dominieren.

Schülerbezogene Kriterien

Eine Auswertung des Kriteriums, ob sich ein Schüler mit einem Thema identifizieren kann, ist problematisch. Während der unterrichtende Lehrer ein Gefühl für die Interessen und Bedürfnisse seiner Schüler hat und diese bei der Berücksichtigung des Kriteriums in der Planung des Unterrichts einbeziehen kann, ist dies bei der Auswertung von Unterrichtsvorschlägen schülerunabhängig nur schwer durchzuführen. Die zugrunde liegende Frage bei der Auswertung war entsprechend: „Ist es grundsätzlich möglich, sich mit dem Unterrichtgegenstand bzw. der eigenen Tätigkeit im Rahmen dieses Unterrichts zu identifizieren?“ Offensichtlich ist das der Fall bei Modellierungen, Problemlösungen oder Themen, welche die Schüler direkt betreffen. Bei der Behandlung von z.B. Protokollen ist das nicht ohne weiteres möglich. 41% der ausgewerteten Themen ließen in diesem Sinn eine Identifizierung zu. Das Einbringen einer „eigenen Note“ ermöglichten 39% der Unterrichtsvorschläge, welches häufig durch das Gestalten einer GUI, einer Präsentation o. ä. geschah.

Umgebung

Gut die Hälfte der Unterrichtsskizzen gibt den Schülern die Möglichkeit zu experimentieren - hier spiegelt sich die Durchführung des Informatikunterrichts in Computerkabinetten wider, welche softwarebasierte Experimentier- und Simulationsumgebungen bereitstellen. Entsprechend beziehen sich auch einige Unterrichtsskizzen auf die Anwendung solcher Software. Ein gewisser zeitlicher Freiraum wird den Schülern in immerhin 41% der Unterrichtsskizzen zugestanden. Dieses geschieht meist durch das Einbinden von (Mini-) Projekten oder längeren Schülerarbeitsphasen. Eine Mehrzahl der Lehrer hält an der traditionellen Lehrerrolle fest – in nur 38% gibt der Lehrer phasenweise diese Rolle ab und tritt in die Rolle des Coachs. Ein Unterrichtsklima der Vielfalt ist in nur einem Drittel der Unterrichtseinheiten vorstellbar. Möglicherweise offenbart sich hier ein Bedürfnis der Lehrkräfte nach konvergenten, leicht zu überschauenden und zu bewertenden Lösungen sowie einer möglichst homogenen Schülerschaft, welche den Unterricht einfacher machen. Insgesamt ist festzustellen, dass eine Mehrheit der untersuchten Unterrichtsvorschläge einem kreativen Informatikunterricht noch wenig Beachtung schenkt. Es spiegelt sich hierin allerdings auch wider, dass Informatikunterricht einen Teil der Anforderungen für kreatives Arbeiten per se schon anwendet, sei es durch das Arbeiten an Computern, welche kreatives Arbeiten begünstigen, oder durch den zentralen Stellenwert von Problemlösen und Modellieren. In nicht wenigen Unterrichtsskizzen wäre allerdings durch eine Änderung der Aufgabenstellung und durch die Berücksich-

tung der Kriterien ein stärkeres Einbinden kreativer Unterrichtsphasen möglich. Dieses kann auch durch Anwenden folgender Vorschläge unterstützt werden.

6 Vorschläge für kreativen Informatikunterricht

Möchte man in seinem Unterricht kreative Schülerleistungen fördern, ist es vor allem notwendig, explizit kreative Unterrichtsphasen einzuplanen sowie sich selbst und die Schüler entsprechend darauf vorzubereiten. Für die Vorbereitung des Unterrichts empfiehlt sich die Berücksichtigung der o. g. Kriterien. Darüber hinaus existieren zahlreiche Empfehlungen in Aufsätzen zur Kreativität und in Kreativitätstrainings, welche Faktoren das Hervorbringen kreativer Leistungen unterstützen. Folgende Richtlinien empfehlen wir für den Informatikunterricht (vgl. [Fa00]):

1. Bereiten Sie die Schüler auf kreatives Tun vor: Vermitteln Sie Konzepte, grenzen Sie Wissen ab, arbeiten Sie mit vielen Beispielen und machen Sie Quellen verfügbar, wo weitere Informationen zu finden sind.
2. Schaffen Sie in der Thematik, im Umgang mit dem Computer als Werkzeug und mit dem notwendigen „Handwerkszeug“ Selbstvertrauen bei den Schülern.
3. Verwenden Sie in Ihrem Unterricht Kreativitätstechniken zur Ideenfindung, wie Brainstorming, Mind-Mapping, Analogienbildung und viele andere.
4. Schaffen Sie eine zündende Eingangserfahrung, um die Schüler für ein Problem, Konzept, eine Situation, Aufgabe oder Idee zu interessieren. Knüpfen Sie an die Alltagserfahrungen der Schüler mit Informatiksystemen an – Interessen können geweckt werden.
5. Stellen Sie offene Aufgaben mit denen die Schüler erfahren, dass sie Informatiksysteme selbst gestalten können.
6. Helfen Sie den Schülern, sich in ihrer Aufgabe und Lösung mit eigenen Ideen selbst wiederzufinden.
7. Schaffen Sie ein kreatives Unterrichtsklima, welches die Vielfalt informatischer Gestaltungsmöglichkeiten widerspiegelt und dafür Raum lässt.
8. Wenden Sie kreative Phasen regelmäßig und als langfristige Leitlinie im Informatikunterricht an.

7 Zusammenfassung und Ausblick

Die Informatik darf und braucht sich hinsichtlich Kreativität nicht hinter anderen Fächern zu verstecken. Es wurden Ansätze aufgezeigt, wie den Schülern ein kreativer Zugang zur Informatik ermöglicht werden kann. Während einige veröffentlichte Unterrichtsskizzen kreative Aspekte im Informatikunterricht bereits berücksichtigen, sind bei

vielen Vorschlägen die Möglichkeiten der kreativen Schülertätigkeit noch nicht ausgeschöpft. Die vorgestellten Kriterien bieten Anhaltspunkte, welche die Planung und Gestaltung kreativer Unterrichtsphasen unterstützen. Nicht zuletzt wird durch die Berücksichtigung von Kreativität gegenüber einem techniklastigen Unterricht dazu beigetragen, das Bild der Informatik zurechtzurücken. Der Einfluss kreativen Arbeitens auf Motivation und Konzeptverständnis scheint vielversprechend und soll zukünftig genauer untersucht werden. Des Weiteren sollen Programmierumgebungen bezüglich ihrer Unterstützung kreativen Arbeitens im Informatikunterricht analysiert werden. So zeigt sich Informatik, als kreatives Fachgebiet, auch in der Schule als Unterrichtsfach, das Schüler zu kreativen Leistungen befähigt und anregt.

Literaturverzeichnis

[As94] Asmus, E. P.: Motivation in Music Teach. & Learning. In Quaterly 5(4), 1994; S. 5-32.

[BR05] Bergin, S., Reilly, R.: The Influence of Motivation and Comfort-Level on Learning to Program. PPIG 17, University of Sussex, Brighton UK, 2005.

[Bo95] Boden, M. A.: Die Flügel des Geistes: Kreativität und künstliche Intelligenz. Dt. Taschenbuch-Verl., München, 1995.

[Br04] Brinda, T.: Didaktisches System für objektorientiertes Modellieren im Informatikunterricht der Sekundarstufe II, Universität Siegen, Diss., 2004.

[Cs90] Csikszentmihalyi, M.: Flow: The psychology of optimal experience. Harper and Row, New York u.a., 1990.

[Cu95] Custer, R. L.: Examining the determinants of technology. In International Journal of Technology and Design Education 5, 1995; S. 219-244.

[Da91] Davis, G. A.: Teaching creativity thinking. In N. Colangelo and G. A. Davis: Handbook of gifted education, Boston, Allyn & Bacon, 236–244, 1991.

[Fa00] Fasko, D.: Education and creativity. In Creativity Research J. 13(3-4), 2000; S. 317-327.

[FT85] Feldhusen, J. F. and D. J. Treffinger: Creative thinking and problem solving in gifted education. Kendall-Hunt, Dubuque, Iowa, 1985.

[Fe88] Feldman, D. H.: Creativity: Dreams, insights, and transformations. In R. J. Sternberg: The nature of creativity, New York, Cambridge University Press, 1988.

[Fe96] Feller, G.: Defizite der Ausbildung - Erfordernisse der Weiterbildung? Ergebnisse und Betrachtungen an der Schnittstelle Lehrabschluß aus der Sicht von Auszubildenden. In P. Diepold: Berufliche Aus- und Weiterbildung., Nürnberg,1996; 251-260.

[Ga93] Gardner, H.: Creating minds: an anatomy of creativity seen through the lives of Freud, Einstein, Picasso, Stravinsky, Eliot, Graham, and Gandhi. BasicBooks, New York, 1993.

[Ge05] Geser, H.: Mängel der Schulausbildung aus Arbeitgebersicht. Zürich, Soziolog. Institut der Universität Zürich, o.J.: http://geser.net/work/geser/05.pdf (September 2nd, 2006).

[Gl06] Glass, R. L.: Software creativity 2.0. developer .* Books, Atlanta, 2006.

[Gu50] Guilford, J. P.: Creativity. In American Psychologist 5, 1950; S. 444-454.

[Hi98] Hill, A. M.: Problem solving in real-life contexts: An alternative for design in technology education. In International Journal of Technology and Design Educ. 5(3), 1998; S. 1-18.

[Hu00] Hubwieser, P.: Didaktik der Informatik: Grundlagen, Konzepte, Beispiele. Springer, Berlin [u.a.], 2000.

[Ja06] Janneck, M.: Partizipative Systementwicklung im Informatikunterricht. In LOG IN (138/139), 2006; S. 60-66.

[LJ05] Lewandowski, G., E. Johnson, et al.: Fostering a Creative Interest in Computer Science. SIGCSE '05, St. Louis, MO, 2005.

[LP98] Lewis, T., S. Petrina, et al.: Problem Posing-Adding a Creative Increment to Technological Problem Solving. In: Journal of Industrial Teacher Education 36(1), 1998.
[Lu00] Lubart, T. I.: Models of the creative process: Past, present and future. In: Creativity Research Journal 13(3-4), 2000; S. 295-308.
[Lu06] Luthiger-Stoll, B.: Spass und Software-Entwicklung: Zur Motivation von Open-Source-Programmierern, Zürich, 2006.
[FHM] o.V.a: Informationen zum Fachbereich Informatik. Fachhochschule München, o.J.: http://www.cs.fhm.edu/broschuere/ (September 2nd, 2006).
[EHZ] o.V.b: Interview with Alonso, G, o. J.: http://www.inf.ethz.ch/news/focus/edu_focus/alonso (September 2nd, 2006).
[Os53] Osborn, A. F.: Applied imagination. Scribner, New York, 1953.
[Pa93] Papert, S.: The children's machine: rethinking school in the age of the computer. BasicBooks, New York, 1993.
[Pr04] Prätorius, P.: Virtuelle Ameisenwelt - Digitale Ameisen und Termiten als Modelle künstlichen Lebens in JAVA (Teil 2). In LOG IN(131/132), 2004; S. 81-89.
[RC95] Runco, M. A. and Chand, I.: Cognition and Creativity. In Educational Psychology Review 7(3), 1995; S. 243-267.
[Ro06] Romeike, R.: Creative Students - What Can We Learn From Them for Teaching Computer Science? Koli Calling, Koli, 2006.
[Sc03] Schulte, C.: Lehr-Lernprozesse im Informatik-Anfangsunterricht: Theoriegeleitete Entwicklung und Evaluation eines Unterrichtskonzepts zur Objektorientierung in der Sekundarstufe II, 2003.
[SB94] Scragg, G., D. Baldwin, et al.: Computer science needs an insight-based curriculum. Proceedings of the twenty-fifth SIGCSE symposium on Computer science education, Phoenix, Arizona, United States, ACM Press, 1994; 150-154.
[St01] Starko, A. J.: Creativity in the classroom. Erlbaum, Mahwah, NJ [u.a.], 2001.
[St03] Sternberg, R. J.: Creative Thinking in the Classroom. In Scandinavian Journal of Educational Research 47(3), 2003; S. 325-338.
[St04] Sternberg, R. J.: Creativity: from potential to realization. American Psychological Assoc., Washington, DC, 2004.
[SL91] Sternberg, R. J. and T. I. Lubart: Creating Creative Minds. In Phi Delta Kappan 72(8), 1991; S. 608-614.
[SS04] Schubert, S. and A. Schwill: Didaktik der Informatik. Spektrum, Heidelberg [u.a.], 2004.
[Th02] Thomas, M.: Informatische Modellbildung: Modellieren von Modellen als zentrales Element der Informatik für den allgemeinbildenden Schulunterricht, 2002.
[To81] Torrance, E. P. Creative teaching makes a difference. In J. C. Gowan, J. Khatena and E. P. Torrance: Creativity: Its educational implications, Dubuque, IA, Kendall/Hunt, 1981; S. 99-108.
[Tr80] Treffinger, D. J.: Encouraging creative learning for the gifted and talented. Ventura County Schools/LTI, Ventura, CA, 1980.
[WD95] Westby, E. L. and Dawson, V. L.: Creativity - Asset or Burden in the Classroom. In Creativity Research Journal 8(1), 1995; S. 1-10.

Das informatische Weltbild von Studierenden

Maria Knobelsdorf, Carsten Schulte

Institut für Informatik
Freie Universität Berlin
Takustr. 9
10495 Berlin
{knobelsd,schulte}@inf.fu-berlin.de

Abstract: Trotz zahlreicher Gegenmaßnahmen hat die Informatik nach wie vor mit hohen Abbruchquoten und einem steigenden Desinteresse zu kämpfen. Wir vermuten, dass Wahlmotive sich in einem biographischen Lern-Prozess entwickeln, verändern und in Bezug auf informatische Weltbilder stabilisieren. Beim Analysieren individueller Computernutzungserfahrungen haben wir die Computernutzung unter dem Aspekt des individuellen Zugangs zur Informatik analysiert. Dabei zeigte sich, dass sich die computerbezogenen Nutzungserfahrungen auf das Selbstbild im Verhältnis zum Computer, die subjektive Konzeptualisierung des Fachs (das Weltbild) und die Handlungsweisen und Reaktionsmuster bei der Computernutzung auswirken. In diesem Artikel stellen wir unseren Forschungsansatz und erste Ergebnisse dar.

1 Die Aufgabe des IU

Trotz verschiedener Maßnahmen ist das Interesse an der Informatik in den letzten Jahren weiter gesunken. Die Anzahl der Studienanfängerinnen und -anfänger in der Informatik ist zwischen 2000 und 2004 um 22% zurückgegangen [SB05]. Ferner hat die Informatik an deutschen Hochschulen mit 38% die höchsten Abbruchquoten aller Fächer [HSS05]. Dieser für die Informatik negativer Trend kann in allen angloamerikanischen und europäischen Ländern beobachtet werden. Am dramatischsten ist die Situation wohl in den USA, wo zwischen 2000 und 2004 die Immatrikulationen in Informatik um 60% zurückgegangen sind [Ve05].

Für diese Situation werden verschiedene Gründe genannt: Informatikstudierende sind nur wenig über die Inhalte der Informatik informiert und verstehen Informatik meistens als eine Art „Computerwissenschaft“, in der es darum geht, Computerbedienung zu erlernen [Hu03]. Berufsvorstellungen beschränken sich meist auf das Bild vom einsamen Programmieren am Computer [BM05], [Ca06], [Gr98]. Männliche Informatiker werden entsprechend als „Geek“ oder „Nerd“ gesehen [Ma04] und beziehen die Frage nach sozialen Fähigkeiten nicht auf die Wahl ihres Faches [Be03]. Arbeiten aus der Gender-Forschung weisen darauf hin, dass dieses Berufsbild vor allem junge Frauen abschreckt [CGA06], [FM02], [Te02]. Obwohl Mädchen sehr oft die nötigen mathematischen und

sozialen Fähigkeiten mitbringen und damit gute Voraussetzungen für ein Informatikstudium hätten, entscheiden sie sich aus den genannten Gründen gegen Informatik [SS04].

Diese während der Schulzeit entstandenen Vorstellungen und Überzeugungen darüber, was die Informatik ist, werden möglicherweise entscheidend durch Erfahrungen aus dem Informatikunterricht beeinflusst – so lautet eine gängige Meinung in der Universitätsinformatik. Richtig daran ist, dass Studieninteresse und Vorstellungen über Informatik nicht plötzlich entstehen, sondern in einem biographischen Prozess reifen. Der biographische Hintergrund wird jedoch in den Bemühungen der Universitäten um neue Informatikstudierende vernachlässigt. In unserem Forschungsansatz gehen wir daher von der These aus, dass für die Misserfolge universitärer Maßnahmen zwei bislang vernachlässigte Faktoren ausschlaggebend sind:

1. Bisherige Interventionsstrategien bauen auf situativ erfassten Merkmalen und Ursachen für das Wahlverhalten auf. Wir vermuten dagegen, dass Wahlmotive sich in einem biographischen Lern-Prozess entwickeln, verändern und in Bezug auf sogenannte informatische Weltbilder stabilisieren. Damit sind einerseits die Effekte isolierter Maßnahmen für Schülerinnen und Schüler in ihrer Wirksamkeit begrenzt und andererseits erreichen sie nur diejenigen, deren subjektives, informatisches Weltbild zu den Maßnahmen „passt".

2. Die sich in biographischen Prozessen entwickelnden, informatischen Weltbilder sind das Resultat verschiedener Erfahrungen. Die in der Literatur genannten Faktoren deuten darauf hin, dass die alltägliche Erfahrung mit dem Computer und mit anderen Informationstechnologien ein ausschlaggebender Einflussfaktor sein könnte. Bisher jedoch wurde der Zusammenhang zwischen computerbezogenen Nutzungserfahrungen und der Entstehung eines informatischen Weltbilds kaum erforscht.

Zur Untersuchung dieser beiden Faktoren wird das Datenerhebungsinstrument „Biographien der Computernutzung" eingesetzt. Bereits durchgeführte Voruntersuchungen lassen vermuten, dass eine begrenzte Anzahl biographischer Typen unterschieden werden kann. Deren genauere Untersuchung durch vertiefte Einzelfallanalysen liefert Grundlagen für die Entwicklung geeigneter Interventionsstrategien, die bereits in der Schule umgesetzt werden könnten. Diese unterscheiden sich von bisher verfolgten Ansätzen durch die Beachtung längerfristiger biographischer Entwicklungs- und Lernprozesse und ihre Adaptivität an unterschiedliche Voraussetzungen in Form von informatischen Weltbildern.

2 Informatische Weltbilder im Informatikunterricht

Die naturwissenschaftlichen Fächer (Physik, Biologie und Chemie) haben in der Vergangenheit ebenfalls unter Absolventenschwund und einem Desinteresse für den naturwissenschaftlichen Unterricht an der Schule leiden müssen. In diesem Zusammenhang wurde, angeregt durch konstruktivistische Einflüsse in den 1970er Jahren, eine breite Grundlagenforschung in den Fachdidaktiken dieser Fächer begonnen, die im „Conceptu-

al Change“–Ansatz mündete. Aus Sicht der Konstruktivistischen Didaktik ist das Lernen ein Prozess der Selbstorganisation von Wissen, der sich auf der Basis der Wirklichkeits- und Sinnkonstruktion jedes einzelnen Lernerindividuums vollzieht [Re05]. Davon ausgehend wird dem Vorwissen sowie der Vorgeschichte eines Schülers oder einer Schülerin eine große Bedeutung beigemessen.

Die Schülerinnen und Schüler kommen nicht als „unbeschriebene Blätter“ in den Unterricht. Sie besitzen bereits Vorstellungen zu den zu erlernenden Phänomenen, Begriffen und Prinzipien. In der fachdidaktischen Literatur spricht man von vorunterrichtlichen Schülervorstellungen, Präkonzepten, Laientheorien oder auch Alltagsvorstellungen. Kinder konstruieren von Geburt an sowohl kognitive Erkenntnisstrukturen als auch Wissen und Vorstellungen zu den Phänomenen und Sachverhalten ihrer Umwelt. Es hat sich gezeigt, dass Alltagsvorstellungen von Lernenden nicht einfach „ausradiert“ und durch neue Konzepte ersetzt werden können. Die Veränderungen in der Wissensstruktur von Lernenden, bei denen Alltagsvorstellungen von wissenschaftlich akzeptierten Konzepten abgelöst werden, bezeichnet man als Konzeptwechsel (Conceptual Change). Unterricht im Sinne des Konzeptwechsels knüpft an die vorhandenen Alltagsvorstellungen an [Du00].

Die Voraussetzungen für gelingende Konzeptwechsel sind vielschichtig. Sie bedingen neben vertrauensvollem Lernklima die angemessene Berücksichtigung der vorhandenen Weltbilder, ihre Aufarbeitung sowie die subjektive Erfahrung, dass die zu erlernenden, neuen Konzepte oder Weltbilder angemessen und im persönlichen Alltag erfolgreich sind. Im Fall der Informatik kommt hinzu, dass die Alltagswelt, der diese Erfahrungen entstammen, im Gegensatz zu naturwissenschaftlichen Weltbildern je nach Hintergrund und Geschlecht sehr unterschiedlich sein kann. Bisherige Studien haben es nicht geschafft bzw. nicht angestrebt, aus den unterschiedlichen Facetten die informatischen Weltbilder von Studierenden zu rekonstruieren – erst deren Kenntnis aber ermöglicht die Entwicklung effektiver Interventionsstrategien für den Unterricht in Schule und Hochschule.

Dass nur wenig über Prozesse der Entstehung eines informatischen Weltbilds untersucht wurde, liegt zum größten Teil am verwendeten Lernbegriff, der die biographische Komponente des Lernens vernachlässigt. Durch die biographische Perspektive kann das Individuum als Subjekt des Lernens in den Blick genommen werden: Der Mensch entwirft in einem Prozess der Biographisierung nicht nur ständig sich selbst neu, sondern auch die Welt aus der Perspektive einer bestimmten, individuellen Sichtweise. In biographischen Lernverläufen schichten sich Erfahrungen aufeinander, werden subjektiv in Beziehung gesetzt und gedeutet, sodass (implizite, und zum Teil auch unbewusste) Orientierungen in Form von Selbst- und Weltbildern entstehen. Veränderungen in Selbst- und Weltbild entstehen langsam und sprunghaft. Sichtbar werden sie erst retrospektiv in der Darstellung der eigenen Biographie[DM05], [Ec06].

Diese Perspektive des biographischen Lernens wurde bisher nicht genutzt, um informatische Weltbilder als Voraussetzung und Orientierungsrahmen für Studienwahlverhalten und Studienverläufe zu untersuchen. Aufgrund des Forschungsstands ist zu vermuten, dass erstens computerbezogene Nutzungserfahrungen zu einem Großteil die Genese

informatischer Weltbilder beeinflussen. Zweitens dürften diese Erfahrungen bereits im Informatikunterricht wirksam sein, also dort ebenfalls Wahlverhalten sowie Lern- und Leistungsmotivation beeinflussen. Drittens gehört unserer Meinung nach zum wissenschaftspropädeutischen Auftrag der Schule auch, ein angemessenes Weltbild über Informatik zu vermitteln. Schlussfolgerungen dafür dürften sich aus dem hier verfolgten biographischen Ansatz ergeben.

3 Der Biographische Ansatz

Die Entwicklung informatischer Weltbilder und entsprechender biographischer Lernverläufe ist meistens ein impliziter und unbewusster Prozess, der nicht einfach per Fragebogen evaluiert werden kann. Erfahrungen werden individuell verarbeitet und spontan miteinander verknüpft, unabhängig vom erlebten Zeitraum. Biographisches Lernen ist daher gleichzeitig langsam, langfristig und sprunghaft.

Zugänglich wird biographisches Lernen über verschiedene Formen biographischer Erzählungen bzw. biographischer Stegreiferzählungen. Wir verwenden ein Forschungsinstrument, das wir Biographien der Computernutzung nennen [KS05]: So genannte „Locktexte“ – kurze Ausschnitte aus anderen Biographien – regen das Nachdenken über die eigenen Erfahrungen an und verweisen auf mögliche Gesichtspunkte der Erzählung, ohne die Antwort auf bestimmte Aspekte zu erzwingen. Angeregt durch diese Vorlagen und die Bitte, ihre eigenen Computernutzungserfahrungen aufzuschreiben, verfassen die Teilnehmenden einen Text, in dem sie ihre Computernutzungsbiographie schildern. Dieser Schreibprozess dauert ungefähr eine halbe Stunde. Gegenüber dem sonst üblichen Interview ist dieses Instrument weitaus kostengünstiger und auch für größere Probandenzahlen einfach zu erheben. Allerdings können durch die fehlende Nachfragemöglichkeit eventuelle Lücken in der Erzählung oder interessante individuelle Erfahrungen jedoch nicht weiter erhoben werden. Daher ist geplant, diese nach ihrer Auswertung in einem zweiten Schritt durch eine geringe Anzahl von Einzelbefragungen zu ergänzen.

3.1 Zur Analyse von Computernutzungsbiografien

Wir vermuten, dass die Befragten zum überwiegenden Teil Informatik als Computerwissenschaft verstehen, bzw. wir sehen enge Bezüge zwischen Computernutzung und Informatik ([Hu03], S. 122), sodass die Texte Hinweise auf subjektive Theorien über Informatik, sowie typisches Nutzungsverhalten und Selbstbild enthalten. Die computernutzungsbezogenen biographischen Lernprozesse werden anhand drei miteinander verschränkter Perspektiven rekonstruiert und sichtbar gemacht (vgl. [Ti05], S. 75).

1. Aus der Strukturperspektive wird das Weltbild rekonstruiert. Das Weltbild umfasst die subjektiven Theorien über Computernutzung und Informatik, außerdem Alltagsvorstellungen und das Berufsbild von Informatik.

2. Aus der Sinnperspektive wird das Selbstbild rekonstruiert. Das Selbstbild umfasst das Fähigkeitsselbstkonzept, die Einstellungen des Subjekts zur eigenen Computernutzung und zur Informatik.

3. Aus der Handlungsperspektive werden die verfestigten Lern- und Verhaltensweisen rekonstruiert. Die verfestigten Lern- und Verhaltensweisen umfassen Lernstrategien, typische Verhaltensweisen im Umgang mit dem Computer und der Informatik sowie Reaktionsmuster auf Probleme.

Ziel ist, Erkenntnisse über biographische Prozesse der Entwicklung von informatischen Weltbildern zu gewinnen. Wir hoffen, typische Prozessstrukturen und „hot spots“ in biographischen Verläufen aufdecken zu können, welche Grundlagen für pädagogische Interventionen liefern. Weiteres Ziel ist die Entwicklung einer Typologie informatischer Weltbilder, um typische Lebensverläufe zu erfassen und zu beschreiben.

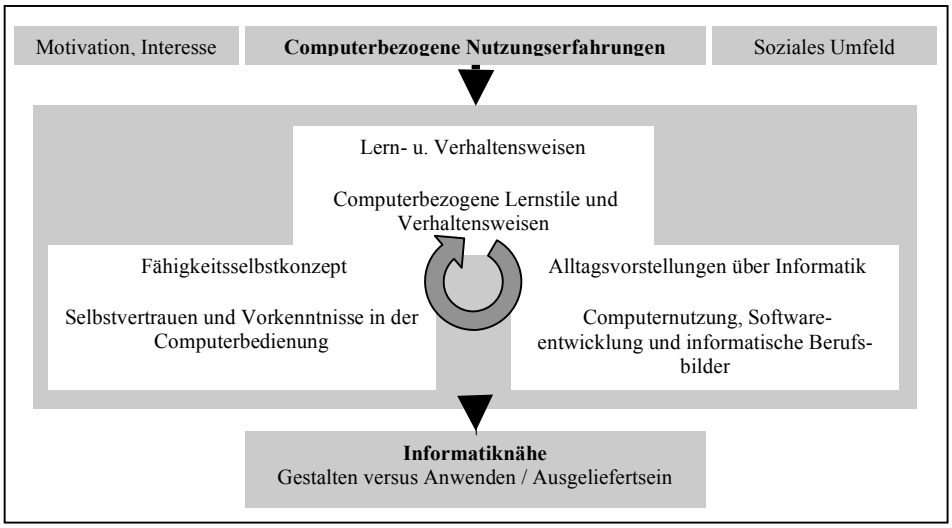


Abbildung 1: Modell der Auswirkungen der computerbezogenen Nutzungserfahrungen auf die Informatik

Im Folgenden werden die drei Perspektiven und unsere Ergebnisse genauer erläutert.

3.2 Das Selbstbild

Bei der Rekonstruktion des Selbstbildes interessiert uns die Einstellung des Subjekts zur Computernutzung und zur Informatik. Generell gehen wir dabei den Fragen nach: Was sagt die Person über sich? Was wird nicht gesagt? Wie positioniert sich die Person in Bezug auf Informatik und die eigenen computerbezogenen Nutzungserfahrungen? Werden die aus den Nutzungserfahrungen gewonnenen Kompetenzen als Anwendungswissen (=außerhalb der Informatik) oder als Informatik-Kenntnisse erlebt? Hierbei liegt der Fokus auf der Trennung zwischen Nutzen und Gestalten, ähnlich wie von Crutzen skizziert ([Cr05], S. 5), demnach fällt unter das Nutzen jede Aktivität am Computer, die

Informatik ausschließt, während das Gestalten diese einbindet. Unter anderem wird analysiert, ob sich die Befragten als Anwender bzw. Anwenderin, oder bereits als Gestalter bzw. Gestalterin sehen. Hierzu schreibt Crutzen ([Cr05], S. 5):

> „Durch eine Genderanalyse kann man feststellen, dass die Handlungen „Entwerfen“ und „Benutzen“ (Design und Use) in der Informatik Oppositionen sind, weil unterstellt wird, dass nur die Informatiker Informatikrepräsentationen entwerfen und die Abnehmer diese Produkte dann nur benutzen. Die Zuweisung der Bedeutungen „Entwerfen“ und „Benutzen“ an bestimmte Handlungen in der Informatik wurde „gendered“ durch die Dualität des passiven Nutzens und des kreativen Entwerfens.“

Diese Opposition zeigt, wie Computernutzungserfahrungen sowohl als Einstieg, aber auch als Barriere zur Informatik wirken können: Computernutzung kann als reines Benutzen bzw. als Anwenden vorgegebener Möglichkeiten aufgefasst werden, womit sich die anwendende Person als außerhalb der Informatik positioniert. Oder die eigene Computernutzung wird als aktives Gestalten und Entwerfen gesehen und damit zur Eintrittskarte in die Welt der Informatik. Wir vermuten, dass nur im letzteren Fall Interesse geweckt wird, mehr über Entwurf und Gestaltung von Informatiksystemen zu erfahren.

Interessanterweise argumentiert Crutzen, dass die Oppositionen zwischen Entwerfen und Benutzen nur scheinbare sind, da auch entwerfende Tätigkeiten wie das Programmieren auf Nutzen vorgegebener Artefakte angewiesen ist – und ebenso bedeutet Nutzen auch Entwerfen. Es gibt demnach also einen Graubereich, in dem diese scheinbaren Oppositionen ineinander fließen. Dieser Graubereich, bzw. genauer: Erfahrungs-möglichkeiten in diesem Graubereich könnten daher helfen, Interesse an Informatik zu entwickeln.

Wir vermuten, dass Erfahrungen in der Nutzung des Computers eine Voraussetzung darstellt, um sich mit dem Aspekt des Gestaltens zu beschäftigen, also der Informatik näher zu kommen. In einer ersten Studie mit Studierenden der Informatik konnten wir diesbezüglich eine Entwicklung rekonstruieren, die mit der Internetnutzung begann, sich über die Erzeugung von Web-Seiten fortsetzte und überleitete zur Erzeugung von dynamischen Web-Seiten, die Skripte und kleine Datenbanken enthielten. Zusammengefasst konnten wir einen Übergang vom puren Anwenden zum Gestalten rekonstruieren. Wichtig dabei war jedoch, dass die Studierenden diesen Übergang als solchen wahrgenommen haben, das heißt ihr Selbstbild sich beim Erlernen neuer Fähigkeiten änderte.

In einer weiteren Studie wurde dieses Phänomen bei Studierenden der Psychologie untersucht. Hierbei stellte sich heraus, dass es eine weitere Vorstufe des Übergangs gibt. Die Studierenden unterschieden zwischen Benutzen und professionellem Benutzen. Dabei bewerteten sie sich selbst als Benutzerinnen oder Benutzer mit „durchschnittlichen Fähigkeiten“. Personen, die mit dem Computer professionell umgehen können, wurden hingegen als (männliche) Informatiker bezeichnet – weil sie Fertigkeiten beherrschen wie z. B. das De- und Installieren von Programmen oder das Administrieren von Computern. Für die Studierenden der Psychologie beschäftigt sich ein solcher Informatiker vor allen Dingen mit Computern. Sie wissen nicht, dass dieser auch für Informatikerinnen und Informatiker oft nur das Medium ist, auf das sich ihre Arbeit bezieht. D.h. das Selbstbild wird durch die Abgrenzung definiert und die Computernutzungsfähigkei-

ten werden als minderwertiger bewertet. Gleichzeitig wirkt sich dieses Selbstbild auf das informatische Weltbild aus, indem es Benutzenden keinen Platz einräumt.

3.3 Handlungsstrategien

In dieser Perspektive geht es um die Rekonstruktion von Reaktionsmustern auf Probleme und von verfestigten Handlungsstrategien und –alternativen: Wie wird typischerweise auf Computerprobleme reagiert? Wie wird Computernutzung erlernt (eher passiv oder aktiv, zielgerichtet oder suchend)? Dabei interessiert uns vor allem die Rekonstruktion von Übergängen zwischen den Oppositionen Gestalten und Benutzen als mögliche Übergänge zwischen verschiedenen informatischen Weltbildern: von der Computernutzung zu angemessenen Vorstellungen über Informatik.

Bei Nichtinformatikern/-innen fallen eine große Unselbstständigkeit, ein Hang zur Passivität und der fehlende Wunsch, das zu ändern auf. Diese Haltung ergibt sich aus dem Selbstbild vom „dummen user“, der überzeugt ist, dass er diesen Zustand nicht ändern kann. Die Studierenden setzen voraus, dass um Hilfe zu fragen die einzige Reaktionsmöglichkeit ist, selbst wenn ihnen das unangenehm ist. Informatikstudierende hingegen beschreiben, dass Probleme sie herausfordern und die Suche nach einer Lösung aufregend ist, die gleichzeitig eine Wissenserweiterung ermöglicht. In ihrer Computernutzungsgeschichte haben Informatikstudierende andere Strategien entwickelt, um auf Probleme zu reagieren. Hierbei spielt nicht nur das Selbstbild eine Rolle. Es ist vor allem das Weltbild, das hier interagiert und das im nächsten Abschnitt näher untersucht wird.

3.4 Das Weltbild

Das informatische Weltbild enthält subjektive Annahmen und Alltagsvorstellungen über die Disziplin Informatik: Was ist Informatik, womit beschäftigen sich Informatiker/-innen, welcher Zusammenhang besteht zwischen Informatik und Computer, was ist die Rolle des Programmierens? Das informatische Weltbild ist ein Puzzle unterschiedlicher Wahrnehmungen und subjektiver Konzepte. Damit besitzt der Ansatz des informatischen Weltbilds Ähnlichkeiten zum oben beschriebenen Ansatz der Konzepte und des Konzep

twechsels aus den Naturwissenschaften. Konzeptwechsel beziehen sich auf Vorstellungen über eng umgrenzte und definierbare wissenschaftliche Sachverhalte oder Theorien. Ein Weltbild setzt sich aus mehreren solchen Vorstellungen zusammen, aber auch aus weiteren Dimensionen (Werturteile, Annahmen über Ziele und Perspektiven der Disziplin, ihrer Grenzen und Abgrenzungen, Berufsbilder, etc.). Im Falle der Informatik fällt es zudem schwer, eine eng abgrenzbare Definition der Disziplin und ihres Gegenstandsbereichs zu liefern [GI05].

Bei der Rekonstruktion des biographischen Lernprozesses aus der Perspektive des informatischen Weltbilds interessieren uns generell gemachte Annahmen und Wahrnehmungsmuster: Welchen Stellenwert und welche Rolle hat der Computer? Welche Vorstellungen über Informatik werden als relevant angesehen und geben eine Orientierung für die eigene Studienwahlentscheidung? Welche sozialen Beziehungen, institutionellen Erfahrungen und Lernprozesse sind in diesem Zusammenhang wesentlich?

Der Computer als Teil des informatischen Weltbilds, wird von vielen Informatikstudierenden als eine Art „Wundertüte" gesehen [Tu84]. Studierende, deren Fach inhaltlich von Informatik und/oder von den Naturwissenschaften abweicht, sehen im Computer mehr ein Arbeitsgerät oder ein Medium für die Freizeitgestaltung. Bereits aus dieser Wahrnehmung heraus entwickeln sich beispielsweise das Nutzungsverhalten und die damit einhergehenden Handlungsstrategien. Bei vielen Informatikstudierenden, die den Computer als „Wundertüte" mit unbegrenzten Möglichkeiten wahrnehmen, geht eine spielerische Herangehensweise mit dem Wunsch, die „Wundertüte" und ihre Möglichkeiten auszuprobieren und zu erforschen, einher. Daraus entwickeln sich aktive und selbstständige Lern- und Handlungsstrategien, die wiederum das Selbstbild beeinflussen. Erlebt man sich selbst bei der Lösung von Problemen als eigenständig und souverän, so schätzt man nicht nur seine Fähigkeiten als positiv ein, sondern erlebt sich selbst als kompetentes, gestaltendes Mitglied der „Computer-Welt", womit sich der Kreis schließt.

Wird wie bei den Studierenden der Psychologie der Computer als Arbeitsgerät, als „Mittel zum Zweck" wahrgenommen, so erscheint jede Computer-Tätigkeit, die keinen offensichtlichen pragmatischen Nutzen hat, als irrelevant. Häufig berichten Studierende, dass sie das Programmieren im Informatikunterricht an der Schule als völlig zweckfrei empfanden und erst mit dem Beginn des Studiums der Computer für sie interessant wurde. Das geht dann damit einher, dass im Studium verschiedene Programme und Anwendungen benötigt werden, deren Nutzen dem Studium dient und so einen Zweck erfüllt.

Wir konnten beobachten, dass die Studierenden der Psychologie keine Vorstellungen von Informatik haben und deren Lerninhalte nebulös bleiben. Worte wie „Mysterium" oder „Buch mit sieben Siegeln" treten immer wieder auf. Damit einher ging in der Schulzeit dann die Abwahl des Informatikunterrichts oder es kam erst gar nicht dazu. Gleichzeitig äußern die Studierenden, dass sie sich wünschen würden, mehr über das „Phänomen Computer" zu wissen und um dieses besser verstehen zu können, ohne jedoch bewusst zu bedauern, dieses Wissen aus dem IU nicht mitgenommen zu haben. Der erlebte Informatikunterricht in der Schule wird dementsprechend als völlig unzureichend und weltfremd beurteilt und abgewählt, weil seine Lerninhalte an das informatische Weltbild und das sich daraus ergebende Selbstbild nicht anknüpfen können.

Zukünftige Informatikstudierende entwickeln auf Grundlage ihres Weltbildes von Computern Neugier und Interesse, das ihnen ermöglicht, erfolgreiche Handlungsstrategien bei der Nutzung zu entfalten, die wiederum ihr Selbstbild prägen. Auf dieser Grundlage erfolgt der Übergang vom Nutzer zum Gestalter. Das informatische Weltbild der Nichtinformatiker/-innen führt dazu, dass sie sich selbst innerhalb der „Computerwelt" als Außenstehende wahrnehmen.

4 Didaktische Schlussfolgerungen

Die eigene Computernutzung hat einen grundlegenden Einfluss auf das eigene Selbstbild die damit verbundenen Handlungsweisen und auf das informatische Weltbild. Diese Bereiche hängen zusammen und beeinflussen sich gegenseitig. Auch die Computernutzung – in Form von Nutzen oder Gestalten – gehört dazu. Interventionsstrategien greifen

zu kurz, wenn sie sich nur auf eine Perspektive beschränken. Der beispielsweise selbstbewusste Umgang mit dem Computer bei Frauen, wie ihn Frauenförderungsmaßnahmen zu verwirklichen versuchen, kann nicht allein durch Einwirken auf das Selbstbild realisiert werden. Erfolgreiche Handlungsstrategien gehören ebenso dazu, wie ein informatisches Weltbild, das die Realität wiedergibt und den Frauen eine Rolle innerhalb der Informatik anbietet. Erst die Berücksichtigung aller drei Perspektiven schafft eine Grundlage, um sich selbst innerhalb der „Computerwelt“ neu zu definieren.

Ein denkbarer Ansatzpunkt ist hierbei die Handlungsstrategie des unabhängigen Entdeckens, die wir bei denjenigen beobachten, die sich als Informatiker oder als Informatikerinnen wahrnehmen: Informatikstudierende erwähnen in ihren erfolgreichen Computernutzungsbiographien immer wieder die Fähigkeit, zu entdecken und auszuprobieren. Sie benutzten nicht einfach nur eine Anwendung, sie erforschten ihre Möglichkeiten und ihre Grenzen. Diese Entwicklung wurde schließlich gekrönt durch den Wunsch (und dessen Umsetzung) zu programmieren, um die Anwendung so weiter zu gestalten und vor allem umzugestalten. Den Studierenden gelang der Übergang vom Nutzen zum Gestalten auf Grund ihrer Fähigkeit und Motivation, selbstständig zu entdecken.

Beschränkt sich der Unterricht jedoch auf die Vermittlung bestimmter Handlungsweisen, wird der Übergang nicht gelingen. Wenn sich jemand als „dummer user“ wahrnimmt und dann im Unterricht Fertigkeiten wie den Umgang mit „Word“ erlernt, wird er sich danach als „dummer user mit Word-Kenntnissen“ wahrnehmen. Der im ITG-Unterricht erworbene „Computer-Führerschein“ kann daher nicht als Ersatz für ein grundlegendes Verständnis von und Handeln in einer informatischen Welt herhalten.

Wir glauben, dass der Unterricht Übergänge oder vielmehr Brücken vom Nutzen- zum Gestaltenkonzept provozieren bzw. bauen muss. Lehrende, sollten die Fehlvorstellungen ihrer Schülerinnen und Schüler akzeptieren und mit ihnen im Sinne des Konzeptwechsels arbeiten. Der IU sollte den Schwerpunkt darauf setzen, den Lernenden ihre Perspektiven in der Informatik auf Grundlage ihrer Fähigkeiten zu eröffnen. Ihnen sollte am Ende der Schulzeit der allgemeinbildende Charakter der Informatik zugänglich gemacht worden sein, so dass sie sich selbstständig entscheiden können, innerhalb der Informatik tätig zu werden.

5 Ausblick

Die hier vorgestellten ersten Ergebnisse innerhalb unseres Forschungsansatzes ergänzen die bisherigen Forschungsergebnisse zu Wahlverhalten und Interesse für die Informatik. In weiteren Schritten sollen Zusammenhänge zwischen Computernutzungserfahrungen und informatischen Weltbildern weiter aufgedeckt und in Form einer Typologie von informatischen Weltbildern (oder genereller gefasst typischer biographischer Lernprozesse innerhalb der Informatik) konsolidiert werden.

Warum erleben und lernen manche Kinder oder Jugendliche den Computer als „Wundertüte“ und andere als „Arbeitsmittel“ kennen? Inwiefern haben der Umgang mit Computern von Familienmitgliedern und Peers aber auch Erfahrungen mit ersten Anwendungen

einen Einfluss? Welche Ereignisse können als so genannte „Hot spots“ identifiziert werden und welche Auswirkungen haben sie? Diese und andere Fragen müssen im nächsten Schritt untersucht werden. Da unser Datenerhebungsinstrument keine Nachfragemöglichkeit bietet, möchten wir im weiteren Forschungsverlauf narrative Interviews führen. Ein Interview ermöglicht, Momente der biographischen Steggreiferzählung zu vertiefen, die in schriftlicher Form nicht weiter ausgeführt worden wären.

Auf Grundlage unserer Forschung wird es uns möglich sein, angedeutete „hot spots“ oder generell interessante Begebenheiten zu erkennen und nachzufragen. Das so noch weitaus detailliertere Datenmaterial sollte zu weiteren Erkenntnissen im biographischen Lernprozess führen. Mit unseren Ergebnissen möchten wir Lehrenden in Schule und Hochschule eine Möglichkeit anbieten, den Schülern informatischen Lernstoff besser näher zu bringen. Die Zusammenarbeit zwischen Hochschule und Schule sollte dadurch enger gestaltet werden können.

Literaturverzeichnis

[Be03] Beaubouef, T. Why computer science students need language. SIGCSE Bull., 35(4):51–54, 2003.

[BM05] Beaubouef, T. and Mason, J. Why the high attrition rate for computer science students: some thoughts and observations. SIGCSE Bull., 37(2):103–106, 2005.

[Ca06] Carter, L. Why students with an apparent aptitude for computer science don’t choose to major in computer science. In SIGCSE 06, 37:27–31, 2006.

[CGA06] Cohoon, J. McGrath and Aspray, W. Women and information technology : research on underrepresentation. MIT Press, 2006.

[Cr05] Crutzen, C. K. M. „IKT ist ein Werkzeug und vielleicht ein Spielzeug“ – Setzen Frauen andere Akzente im technologischen Innovationsprozess? Webdokument: http://www.cecile-crutzen.de/Downloads/2005-IKT-ist-ein-Werkzeug-und-vielleicht-ein-Spielzeug.pdf

[DM05] Dick, M. and Marotzki, W. Biographie und Lernen. In: ZBBS, Seite 5-9, Heft 1/2005.

[Du00] Duit, R. Konzeptwechsel und Lernen in den Naturwissenschaften. In R. Duit and C. von Rhöneck (Hrgs.). Ergebnisse fachdidaktischer und psychologischer Lehr-Lern-Forschung. Institut für die Pädagogik der Naturwissenschaften (IPN), 2000.

[Ec06] Ecarius, J. Biographieforschung und Lernen. In: Krüger/Marotzki (Hrsg.): Handbuch zur erziehungswissenschaftlichen Biographieforschung, 2006.

[FM02] Fisher, J. and Margolis, J. Unlocking the clubhouse: the Carnegie Mellon experience. SIGCSE Bull., 34(2):79–83, 2002.

[GI05] GI (Gesellschaft für Informatik): Was ist Informatik? Positionspapier der Gesellschaft für Informatik. 2005

[Gr98] Greening, T. Computer science: through the eyes of potential students. ACSE 98: 145–154, 1998.

[HSS05] Heublein, U. and Schmelzer, R. and Sommer, D. u. a. Studienabbruchstudie 2005. HIS (Hochschul-Informations-System), 2005.

[Hu03] Humbert, L. Zur wissenschaftlichen Fundierung der Schulinformatik. Diss. Univ. Siegen, 2003.

[IF] Informatica Feminale. Offizielle Homepage, http://www.informatica-feminale.de.

[KS05] Knobelsdorf, M; Schulte, C .: Computer Biographies - A Biographical Research Perspective on Computer Usage and Attitudes Towards Informatics. Kolin Kolistelut - Koli Calling 2005.

[Ma04] Martin, C. D. Draw a computer scientist. ITiCSE-WGR ’04: Working group reports from ITiCSE on Innovation and technology in computer science education, Seite 11–12, 2004.
[MW06] Maaß, S. and Wiesner, H. Programmieren, Mathe und ein bisschen Hardware...Wen lockt dies Bild der Informatik? Informatik Spektrum, 2006.
[Re05] Reich, K. Konstruktivistische Didaktik. Beltz Verlag, 2005.
[RS04] Romeike, R. and Schwill, A. Das Studium könnte zu schwierig für mich sein - Langzeitbefragung zur Studienwahl Informatik am Institut für Informatik der Universität Potsdam. Institut für Informatik der Universität Potsdam, 2004.
[SB05] Statistisches Bundesamt Deutschland. Hochschulstandort Deutschland 2005. http://www.destatis.de/presse/deutsch/pk/2005/aktuelle ergebnisse.pdf.
[SN00] Sleumer, N. and Nievergelt, J. Erfahrungen und Gedanken zur Frauenförderung in der Informatik. Informatik Spektrum 23, No. 6, 2000.
[SS04] Schubert, S. and Schwill, A. Didaktik der Informatik. Spektrum Lehrbuch, 2004.
[Te02] Teague, J. Women in computing: what brings them to it, what keeps them in it? SIGCSE Bull., 34(2):147–158, 2002.
[Ti05] Tiefel, S. Kodierung nach der Grounded Theory lern- und bildungstheoretisch modifiziert: Kodierungsleitlinien für die Analyse biographischen Lernens, ZBBS 1/2005.
[Tu84] Turkle, S. Die Wunschmaschine, Rowohlt 1984.
[Ve05] Vegso, J. Interest in cs as a major drop among incoming freshmen. Computing Research News, 17:126–140, Mai 2005.

„KOMA“ – Das Konzept einer Fortbildung

Helmar Fischer
Mittelschule Weixdorf
Alte Dresdner Str. 22
01108 Dresden
helmar-fischer@web.de

Heiko Neupert
Goethe Mittelschule Heidenau
E.-Thälmann-Str. 22
01809 Heidenau
hneupert@web.de

Thomas Knapp
98. Mittelschule Dresden
Berthelsdorfer Weg 2
01279 Dresden
98ms-knapp@gmx.de

Klaus Thuß
Hans-Erlwein-Gymnasium Dresden
Eibenstocker Str. 30
01277 Dresden
thuss@gmx.de

Steffen Friedrich
Technische Universität Dresden
Fakultät Informatik
Institut Software und Multimediatechnik
Didaktik der Informatik / Lehrerbildung
01062 Dresden
steffen.friedrich@tu-dresden.de

Abstract: Mit der Einführung neuer Lehrpläne in Sachsen kam es auch im Fach Informatik an Mittelschulen und Gymnasien zu einer inhaltlichen Neuorientierung. Um daraus erwachsenden Anforderungen an den Unterricht gerecht zu werden, sind umfangreiche Fortbildungen unerlässlich. Der Beitrag will am Beispiel von bereits mehrfach erfolgreich durchgeführten Veranstaltungen zur Modellierung in der Sekundarstufe I zeigen, wie es gelingen kann, dass sich Lehrende mit der Modellierung von Klassen, Objekten, Methoden und Attributen (deshalb: KOMA) aktiv auseinandersetzen.

1 Ausgangslage zur Informatikbildung in Sachsen

Nach den Jahren erfolgreichen Informatikunterrichts an Mittelschulen und Gymnasien in Sachsen, insbesondere im Sekundarbereich I, kam es im Rahmen der Überarbeitung aller Lehrpläne der allgemein bildenden Schulen zu einer Neuorientierung der informatischen Bildung. Auf der Basis verschiedener breit diskutierter und abgestimmter Eckwertepapiere, auch zur informatischen Bildung [EIB04], entstanden für die einzelnen Schularten die Lehrpläne sowohl für das Fach Informatik als auch für die anderen Fächer. Durch eine weite Sicht auf unterschiedliche Facetten und Realisierungen von informatischer Bildung im Schulalltag gelang es, das Verhältnis von Unterrichtsfach Informatik, der Nutzung von Informatiksystemen im Fachunterricht anderer Fächer sowie möglichen

Ergänzungsangeboten in einen Gesamtrahmen einzuordnen [Be05]. Es wurden folgende Hauptpunkte informatischer Bildung herausgestellt:

- Informatische Vorbildung
- Systematische wissenschaftsbezogene Grundlagenbildung
- Verpflichtende Anwendungen in anderen Fächern
- Weiter führende neigungs- und leistungsdifferenzierende Bildungsangebote

Die informatische Vorbildung ist von den verschiedenen Fächern im Bereich der Grundschulen und dem Fach „Technik und Computer“ der Mittelschulen und Gymnasien zu leisten. Hier sollen die Schülerinnen und Schüler an die sachgerechte Benutzung der Computer herangeführt werden, mit dem Ziel, grundlegende Bedienkompetenzen zu erwerben. Die wissenschaftsbezogene Grundlagenbildung findet später im Fach Informatik (mit je einer Pflicht-Wochenstunde) statt. Parallel dazu sind alle Fächer aufgefordert, fachspezifisch informatische Bildung zu verwirklichen, indem informatische Inhalte mit den Inhalten des Faches verbunden werden. Darüber hinaus können in Neigungskursen (wahlobligatorische Kurse an der Mittelschule von Klassenstufe 7 bis 9 und Vertiefungsangebote in Klassenstufe 10) oder Grundkursen in der gymnasialen Oberstufe, die unter Beachtung der KMK-Vereinbarungen in das mathematisch-naturwissenschaftliche Aufgabenfeld eingeordnet sind, weitere informatische Inhalte vertieft werden.

Der Lehrplan des Faches Informatik formuliert für die Mittelschule folgende fachliche Ziele:

- Aneignen von Strategien und Methoden des Umgangs mit Informationen und Daten
- Nutzen von Informatiksystemen und Auseinandersetzen mit deren Wirkung auf Individuum und Gesellschaft
- Verwenden von informatischen Modellen und Modellierungstechniken
- Nutzen von Problemlösestrategien [LIM04]

Neben dem Pflichtfach Informatik in den Klassenstufen 7 und 8 wird eine fachsystematische Bildung in den Klassenstufen 9 und 10 am Gymnasium profilspezifisch durchgeführt. Die systematische und wissenschaftsorientierte Grundlagenbildung zur Informatik am Gymnasium vom Unterrichtsfach in der Sekundarstufe I, über eine integrative profilorientierte Bildung bis zum Grundkurs in der Oberstufe basiert auf folgenden allgemeinen fachlichen Zielen:

- Umgehen mit Daten und Informationen
- Kennen lernen von Aufbau und Funktionalität ausgewählter Informatiksysteme
- Modellieren von Zuständen und Abläufen

- Realisieren von Problemlöseprozessen
- Bewerten von gesellschaftlichen Aspekten der Informatik [LIG04]

Der Lehrplan selbst hat diese Zielkategorien mit konkreten Inhalten untersetzt, schreibt allerdings an keiner Stelle vor, mit welcher Anwendung bzw. Programmierumgebung gearbeitet werden soll. Für eine solche Entscheidung ist neben der Kenntnis der jeweiligen Werkzeuge eine didaktische Reflektion zu den Möglichkeiten und Grenzen von Modellierungen im Informatikunterricht der Sekundarstufe I erforderlich. Obwohl an sächsischen Schulen ein großer Teil der Informatiklehrerinnen und -lehrer ein abgeschlossenes Hochschulstudium und eine entsprechende Staatsprüfung für die jeweilige Schulart besitzen, gibt es eine Anzahl von Lehrenden, die lediglich einen 360 Stunden umfassenden Kurs zu Grundlagen der Arbeit mit dem, und Nutzung des Computers absolviert haben oder auch ohne Ausbildung das Fach Informatik unterrichten.

Durch die Neuorientierung der Lehrpläne und die neuen Schwerpunkte im Informatikunterricht der Sekundarstufe I an den sächsischen Schulen entstand nicht nur bei den unausgebildeten Lehrkräften ein enormer Fortbildungsbedarf. Waren es früher Inhalte, die die Grundlagen des Faches oder die Nutzung bestimmter Anwendungen betrafen, so sind es heute vor allem Aspekte der unterrichtlichen Realisierung von grundlegenden Aspekten des Modellierens und Problemlösens sowie die geeignete Nutzung von Arbeitsmethoden der Informatik. Aus unserer Erfahrung heraus ergeben sich dabei besonders positive Effekte, wenn das in praktischen Sequenzen für die Lehrenden erlebbar wird. Mit diesem Anliegen wurde die Fortbildung für das "Modellieren in der Sekundarstufe I – KOMA" konzipiert und bereits mehrfach erfolgreich durchgeführt.

2 Zielstellungen der Fortbildung

Unter den dargestellten Voraussetzungen und Rahmenbedingen ergaben sich folgende Schwerpunkte als Ziele der zu konzipierenden Fortbildung:

- Vermitteln von Grundlagen zur Modellierung unter besonderer Sicht auf die informatische Bildung in einem Fachunterricht Informatik
- Vorstellen eines didaktischen Konzeptes für die Umsetzung des Modellierens im Informatikunterricht in der Sekundarstufe I
- Verstehen der Modellierungen von Anwendungen unter dem Aspekt der Objektorientierung
- Einführen und Benutzen von “standardisierten Schreibweisen” bei einer objektorientierten Modellierung verschiedener Sachverhalte und Anwendungen

Es sollte erreicht werden, dass die Teilnehmer an dieser Veranstaltung:

- Informatische Modelle entsprechend ihrer Verwendung und verschiedene Darstellungsformen der objektorientierten Modellierung kennen

- einheitliche Darstellungsformen für die Modellierung im Unterricht verwenden
- Darstellungsformen altersspezifisch einsetzen lernen
- Anwendungen sinnvoll und zweckgebunden modellieren

Diese Fortbildung ist wegen der Durchführbarkeit während des Schuljahres als Tagesveranstaltung mit einem Stundenumfang von 8 Unterrichtsstunden konzipiert und gliedert sich in die folgenden vier Teile.

2.1 Vermitteln von Grundlagen zur informatischen Modellierung

In einem einführenden Vortrag wird ausgehend von den Erfordernissen des Unterrichtsfaches Informatik und den bei Schülerinnen und Schülern zu entwickelnden Kompetenzen der Blick auf den Stellenwert der Modellierung in der informatischen Bildung gerichtet. Insbesondere geht es darum, die Modellierungen in der Informatik bewusst zu machen und unterschiedliche Modelle zu unterscheiden [Th02]. Modellieren im Informatikunterricht soll helfen, die Umwelt – dazu gehören auch Anwendungen – strukturiert wahrzunehmen. Es wird dabei besonders betont, dass Modellieren im Informatikunterricht nicht Selbstzweck ist, nicht der Gesamtdarstellung ausgewählter Anwendungen dient und auch keine Fertigkeiten zu deren Benutzung ausprägen kann. In Verbindung zu den anderen Fächern soll bei den Lernenden (und natürlich vorher bei den Lehrenden) das Verständnis für Arbeitsmethoden der Informatik und deren bewusste Verwendung gefestigt werden. Besondere Bedeutung wird den für diese Alterstufe möglichen Darstellungsformen für Ergebnisse der Modellierung beigemessen [FK05]. Neben einer Vergleichbarkeit der Arbeitsergebnisse wird dadurch erreicht, dass sich die Schülerinnen und Schüler im Informatikunterricht sofort an die Einhaltung von Standards gewöhnen, um diese Arbeitsweise im weiteren Verlauf, beispielsweise bei der Nutzung von Programmierumgebungen, kompetenter einsetzen zu können.

2.2 Verwenden einheitlicher Darstellungsformen beim Modellieren einer bekannten Anwendung

Um das Modellieren einer Anwendung zu erlernen, ist es notwendig, die Anwendung zu kennen. Erfahrungen haben gezeigt, dass die gleichzeitige Einführung einer Anwendung und das Modellieren dieser Anwendung die Lernenden in dieser Altersstufe überfordern ([FK05]). Deshalb nutzen wir für das Erlernen des Modellierens eine bekannte Anwendung. So ist den Schülerinnen und Schülern die Tabellenkalkulation z. B. aus dem Mathematikunterricht bekannt. Damit liegt der Schwerpunkt des Informatikunterrichts in dem Modellieren und nicht auf der Einführung oder einer sicheren Beherrschung der Anwendung.

In einem ersten Teil der praktischen Arbeit im Rahmen der Fortbildung haben die Lehrerinnen und Lehrer die Aufgabe, Objekte einer – auf einem Arbeitsblatt – gegebenen Tabelle zweckgebunden zu modellieren und dabei die vorgegebene einheitliche Darstellungsform zu verwenden.

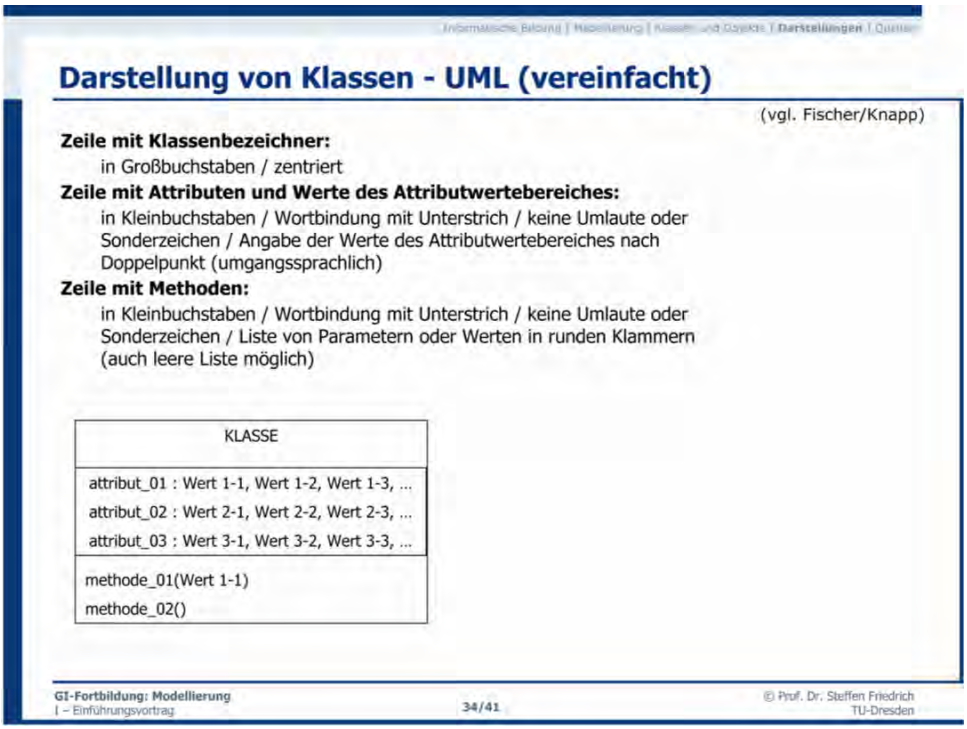


Abbildung 1: Eine Folie des Einführungsvortrages

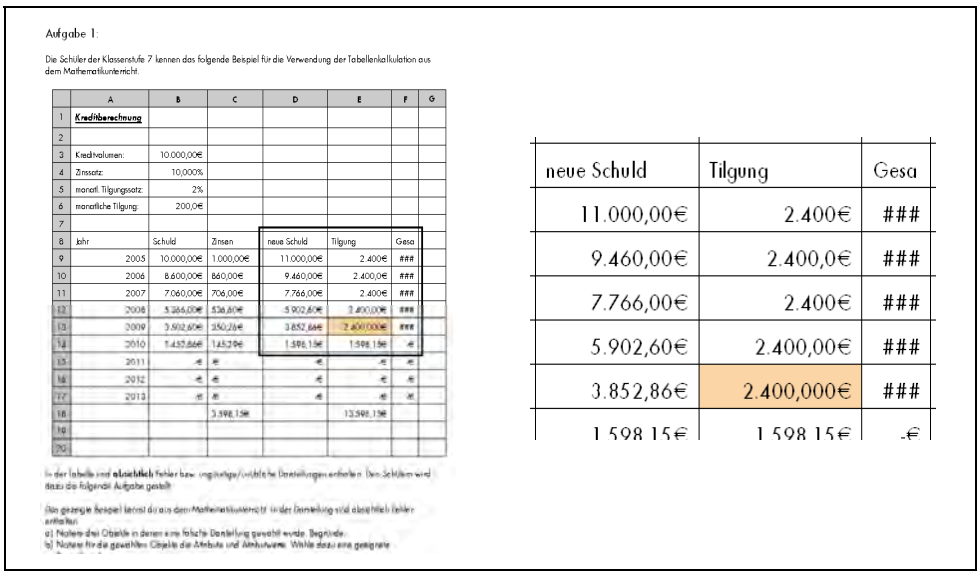


Abbildung 2: Aufgabenstellung mit einem ausgewählten Objekt (Zelle E13)

wörtliche Formulierung

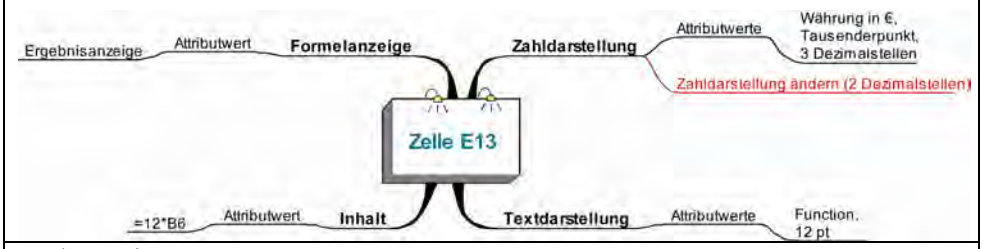
Die Zelle E13 enthält eine Zahl. Diese Zahl ist mit Tausenderpunkt, 3 Dezimalstellen dargestellt und als Währung in € formatiert.

Die Zahlen in E9:E14 sind mit unterschiedlicher Anzahl von Dezimalstellen dargestellt. Die Rechnung ist nur schwer nachprüfbar, da die Kommas nicht untereinander stehen.

vereinfachte UML-Schreibweise

zelle_e13	
inhalt =	=12*B6
zahldarstellung =	Währung in €, Tausenderpunkt, 3 Dezimalstellen
textdarstellung =	Function, 12 pt
formelanzeige =	Ergebnis
zahldarstellung_aendern(2 Dezimalstellen)	

Mindmap



Punktnotation

```
zelle_e13.inhalt = =12*B6
zelle_e13.zahldarstellung_waehrung = €
zelle_e13.zahldarstellung_dezimalstellen = 3
zelle_e13.zahldarstellung_tausenderpunkt = ja
zelle_e13.textdarstellung_schriftart = Function
zelle_e13.textdarstellung_schriftgroesse = 12 pt
zelle_e13.formelanzeige = Ergebnisanzeige
zelle_e13.zahldarstellung_aendern(2 Dezimalstellen)
```

Abbildung 3: Ergebnisse der Modellierung des oben dargestellten Objektes (Zelle E13) in vier Darstellungsformen.

2.3 Bearbeiten eines Problems durch zweckgebundenes Modellieren

Zu einem gleichen Sachverhalt, z. B. einem Rezept für Klöße, werden unterschiedliche Aufgabenstellungen formuliert. Diese Aufgabenstellungen können inhaltlich an verschiedene Fächer angebunden sein (z. B. Mathematik – Kalkulation, Deutsch – Beschreibung, Kunst – Gestaltung). Die Umsetzung mit Hilfe einer entsprechend passenden Softwareapplikation kann ebenfalls im Unterricht des entsprechenden Faches erfolgen.

Im Informatikunterricht werden die Objekte, die zur Problemlösung verwendet werden, modelliert. Dabei erfolgt eine Schwerpunktsetzung hinsichtlich der Modellierungstiefe entsprechend dem Zweck. Das heißt, Text-Objekte werden bei der Tabellenkalkulation weniger im Detail modelliert als bei Verwendung der Textverarbeitung. So erfolgen Modellierungen bei der Präsentation meist weniger detailliert als bei einer Bildbearbeitungsapplikation. Die Lernenden erkennen, dass das gewählte Modell stark vom eigentlichen Zweck einer Aufgabe abhängig ist, auch wenn scheinbar der Sachverhalt oder das

Problem gleich ist. Die Darstellung der gefundenen Modelle erfolgt mit den bekannten Darstellungsformen, die durch diese Übung gefestigt werden.

Perspektivisch werden die Lernenden (und das sind zuerst die Lehrerinnen und Lehrer) in die Lage versetzt, zweckgebunden zu modellieren, also zu erkennen, worauf Prioritäten zu setzen sind. Damit ist es letztlich völlig egal, mit welchen Applikationen gearbeitet wird, gerade weil zu erwarten ist, dass diese in Zukunft sicher noch stärker miteinander verschmelzen werden.

(1)
„Ich möchte Klöße für eine beliebig wählbare Personenanzahl kochen“

Im (Mathematik)-Unterricht haben die Schüler mithilfe der Tabellenkalkulation die folgende Aufgabe zu lösen:
Ermittle die Preise für die Zutaten in handelsüblichen Mengen.
Entwirf eine Kalkulation für die Menge der Zutaten in Abhängigkeit von der Personenanzahl. Kalkuliere die Gesamtkosten für die Beilage.

Im Informatikunterricht (Klasse 7) ist anschließend von den Schülern die folgende Aufgabenstellung zu behandeln:
Finde (mindestens 2) Objekte der Tabellenkalkulation, die für die Lösung des Problems wichtig sind und beschreibe sie durch Ihre Attribute. Nutze dazu eine geeignete Darstellungsform.

(2)
„Ich möchte ein Rezept verschenken“

Im Deutschunterricht haben die Schüler mithilfe der Textverarbeitung die folgende Aufgabe zu lösen:
Gliedere den im Sachverhalt dargestellten Text und fertige dazu eine Kochbuchseite.

Im Informatikunterricht (Klasse 7) ist anschließend von den Schülern die folgende Aufgabenstellung zu behandeln:
Finde (mindestens 2) Objekte der Textverarbeitung, die für die Lösung des Problems wichtig sind und beschreibe sie durch Ihre Attribute. Nutze dazu eine geeignete Darstellungsform.

(3)
„Ich möchte in einer Kindereinrichtung das Rezept präsentieren“

Im WTH[1]-Unterricht haben die Schüler mithilfe eines Präsentationsprogrammes die folgende Aufgabe zu lösen:
Stelle von den im Sachverhalt beschriebenen Zutaten und Hilfsmitteln Fotos oder Grafiken bereit.
Plane und erstelle eine Bildfolge, für Kinder (die noch nicht lesen können)

Im Informatikunterricht (Klasse 7) ist anschließend von den Schülern die folgende Aufgabenstellung zu behandeln:
Finde (mindestens 2) Objekte der Präsentation, die für die Lösung des Problems wichtig sind und beschreibe sie durch Ihre Attribute. Nutze dazu eine geeignete Darstellungsform.

Abbildung 4: Aufgabenstellungen zum Verdeutlichen der Zweckgebundenheit der Modellierung

2.4 Übertragen des Modellierens auf eine unbekannte Anwendung

Nach dem Erlernen und Festigen der einheitlichen Darstellungsformen und des sinnvollen und zweckgebundenen Modellierens bekannter Anwendungen sind die Lernenden in der Lage, selbst zu modellieren. Dazu eignen sich besonders Anwendungen, die im Klassen- und Methodenumfang beschränkt sind. Geeignete Beispiele dafür sind Spiele, wie Minesweeper, Tetris oder SimTower.

[1] Wirtschaft – Technik – Haushalt/Soziales (Unterrichtsfach an Mittelschulen)

Gegeben ist die kleine Anwendung MineSweeper.

Aufgaben:

1. Machen Sie sich mit dem Spiel vertraut. Spielen 1 bis 2 Runden. Untersuchen Sie mit Hilfe des Menüs verschiedene Einstellungsmöglichkeiten und Eigenschaften. Beschreiben Sie schriftlich kurz das Spiel und das Ziel des Spiels.
2. Erstellen Sie ein mögliches Klassenmodell für die Anwendung MineSweeper. Veranschaulichen Sie die verschiedenen Klassen, deren Attribute und den Attributwerteumfang in zwei verschiedenen Darstellungsformen. (Tipp: Vergessen Sie nicht die klassenspezifischen Methoden)
3. Erzeugen Sie von jeder Klasse ein Objekt und beschreiben Sie es in einer von Ihnen gewählten Darstellungsform. (Tipp: Speichern Sie die Objekte so, dass sie zur Präsentation mit der Objektbeschreibung gezeigt werden können)

Abbildung 5: Aufgabenstellung zur Modellierung einer unbekannten Anwendung

Bei der Lösung der Aufgabe sind mehrere Klassenmodelle denkbar und eigentlich auch zu erwarten. Eine Modellierung dieser Anwendung ist beispielsweise mit den folgenden Klassen darstellbar:

- SPIEL
- MINENFELD
- PARZELLE

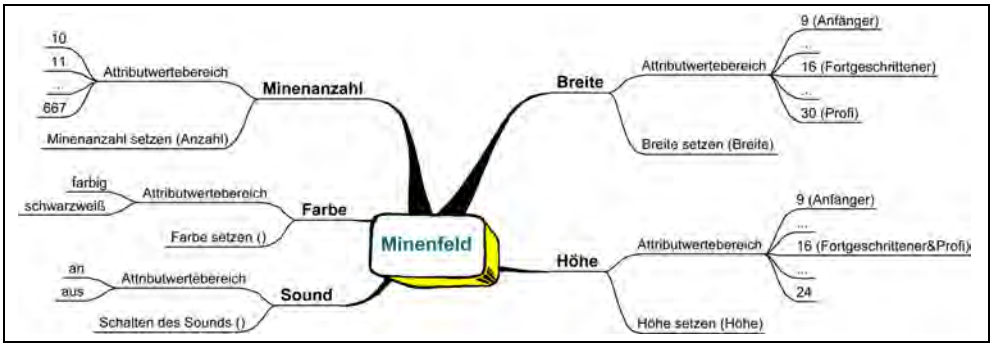


Abbildung 6: Modellierung der Klasse MINENFELD in der Darstellungsform Mindmap

3 Resonanz

In Gesprächen mit Kolleginnen und Kollegen, bei Unterrichtsbesuchen durch die Fachberater und in anderen Fortbildungsveranstaltungen oder Fachgruppentreffen wurde die Fortbildung ausgewertet. Die Teilnehmer berichteten über ihre Erfahrungen aus der Veranstaltung. Die Fachberater beobachteten die Veränderungen im Unterricht.

3.1 Feedback der Teilnehmer

Aus den Gesprächen während und nach den Fortbildungsveranstaltungen ergaben sich die folgenden Aspekte:

- Die Zweckgebundenheit der Modellierung ist für viele Lernende ein neuer Gesichtspunkt und hilft, krampfhafte Versuche der vollständigen Modellierung im Unterricht zu vermeiden. So bleibt mehr Zeit für die Arbeit mit der jeweiligen Anwendung.
- Bezüglich Schulwechsel und Durchlässigkeit der Schularten ist eine einheitliche Festlegung auf bestimmte Darstellungsnormen wichtig. Gerade durch die Anwendungsfreiheit der Lehrplaninhalte kann den Schülerinnen und Schülern mit einer einheitlichen Modellierungsgrundlage ein Wechsel erleichtert werden.
- Vielen Kolleginnen und Kollegen wurde die Einheit von Modellieren und Modellierungsgegenstand deutlich. Unterrichtseinheiten mit einem Theorieteil „Modellierung“ und darauf folgenden praktischen Arbeiten mit einer Anwendung können jetzt didaktisch und methodisch besser zu einer Einheit verbunden werden.
- Unterrichtenden im Fach Informatik ohne Fachausbildung sind Grundlagen zur informatischen Modellierung oft unbekannt. Inhalte werden dann formal nach Lehrplan abgehandelt, ohne den praktischen Bezug zur jeweiligen Anwendung und zum Zweck der Modellierung zu beachten.

3.2 Unterrichtsbeobachtungen

Die Einführung neuer Lehrpläne wird durch die Fachberater evaluiert und begleitet. Mit einem Fragebogen werden Schwerpunkte, Probleme und Fortbildungsbedarf untersucht ([Ne01]). In vielen Unterrichtsbesuchen ist das Bemühen zur objektorientierten Modellierung zu beobachten. Schwierigkeiten hingegen treten dabei in der Verbindung von Modellierung und Anwendung auf. Bei den Schülerinnen und Schülern ist die Kompetenz zur Darstellung der gefundenen Modelle gut ausgeprägt. Während verschiedener Tests zur Modellierung „neuer“ Probleme konnte beobachtet werden, dass selbständig die gewohnte Darstellungsform aufgegriffen und auf den neuen Inhalt angewendet wurde. Es konnte die Verwendung einer Reihe schüler- und praxisnaher Beispiele zur Modellierung beobachtet werden:

- vom Schülerhandy zur Klasse HANDY
- der Informatikhefter von Schüler Fritz H. als Objekt
- von der Klasse CD zum Klassendiagramm einer CD-Datenbasis
- Beschreibung geometrischer Figuren als Klassen und Objekte
- Klassen, Methoden und Objekte, in der Programmierumgebung KARA

Interessant ist, dass neben den fachlichen Inhalten auch angewendete Methoden der Fortbildung im Unterricht eingesetzt wurden. Beim Modellieren arbeiteten die Schülerinnen und Schüler in Gruppen, bildeten Expertenrunden und diskutierten ihre Resultate untereinander.

3.3 Auswirkungen auf den Unterricht in Klassenstufe 9 und 10

In den Klassenstufen 9 und 10 wird entsprechend den Lehrplaninhalten die bereits gefestigte Modellierungstechnik und die Modelldarstellung auf weitere Anwendungsbeispiele übertragen. Bei der Erarbeitung von Datenmodellen für eine Datenbasis zu konkreten Problemen wird auf die Darstellung von Objekten und Klassen zurückgegriffen ([FKN06] S. 12f).

In dem Lernbereich „Komplexe Anwendungssysteme“ der Klassenstufe 10 bieten sich Erweiterungen bereits behandelter Anwendungen an. Beim Problemlösen mit DTP Systemen kann zum Beispiel die Klasse TEXT weiter modelliert werden ([FKN06] S. 146).

Lehrerinnen und Lehrer berichteten, dass sie mit ihren Schülerinnen und Schülern erfolgreich ganz neue Problembereiche nach der gelernten Methode modelliert und dargestellt haben, wie CNC - Steuerungen, CAD von historischen Kleidungsstücken, Modellierung eines Parkhauses, Organisationsabläufe bei Schulsportfesten, statistische Erfassungen und Auswertungen.

Literaturverzeichnis

[EIB04] Eckwerte zur informatischen Bildung, Sächsisches Staatsministerium für Kultus, Comenius-Institut, 2004

[FKN06] Fischer, H.; Knapp, T.; Neupert, H.: Grundlagen der Informatik II, Oldenbourg Schulbuchverlag GmbH, München, 2006

[FK05] Fischer, H.; Knapp, T.: Modellieren im Informatikunterricht der Sek I. In: LOG IN 135, LOG IN Verlag GmbH, Berlin, 2005, S. 69-73

[Ne01] Neupert, Heiko, Evaluation der Einführung der neuen Lehrpläne Informatik in Sachsen, Praxisband INFOS07, Universität Siegen, 2007

[Be05] Bechstädt, T. u.a.: Informatische Bildung im Freistaat Sachsen - ein Gesamtkonzept. In: Friedrich S. (Hrsg.). Unterrichtskonzepte für informatische Bildung. LNI, Band P-60, S.11-26

[LIG04] Lehrplan Informatik Gymnasium, Sächsisches Staatsministerium für Kultus, Dresden, 2004

[LIM04] Lehrplan Informatik Mittelschule, Sächsisches Staatsministerium für Kultus, Dresden, 2004

[Th02] Thomas, M.: Informatische Modellbildung. Modellieren von Modellen als ein zentrales Element der Informatik für den allgemeinbildenden Schulunterricht. Potsdam, Dissertation, 2002

Lehrarrangements in der Informatiklehrerausbildung

Peter K. Antonitsch, Ulrike Lassernig, Andreas Söllei

Institut für Informatiksysteme/Informatik Fachdidaktik
Alpen-Adria Universität Klagenfurt
Universitätsstraße 65-67
A 9020 Klagenfurt
Peter.Antonitsch@uni-klu.ac.at
ulassern@edu.uni-klu.ac.at
asoellei@edu.uni-klu.ac.at

Abstract: Lehramtsstudien sollen auf die spätere Tätigkeit als Lehrer vorbereiten. Diese Vorbereitung darf sich nicht im Vermitteln fachlicher Kenntnisse erschöpfen, sondern muss auch das praktische Agieren des Lehrers berücksichtigen. Eine Möglichkeit dazu bieten Lehrarrangements, die im universitären Umfeld Unterrichtssituationen nachbilden. Simulation von Informatikunterricht unter Verwendung informatischer Mikrowelten ist ein derartiges Lehrarrangement, in dem die Studierenden sowohl Aspekte des Unterrichtens als auch grundlegende Konzepte der Informatik erfahren können.

1 Informatik und Unterrichten – Informatikunterricht

Das Lehramtsstudium Informatik an Universitäten hat das Ziel, Studierende auf ihre spätere Aufgabe, das Gestalten von Informatikunterricht vorzubereiten (vgl. z.B. [St04], S. 20ff). Dabei wird der informatische Blick primär in Fach-Lehrveranstaltungen vermittelt, die Sicht auf das Unterrichten im schulischen Kontext ist eher Domäne der pädagogischen Ausbildung. Die Vermittlung zwischen „Informatik“ und „Unterrichten“ obliegt der Informatik Fachdidaktik. Bedingt durch die Struktur universitärer Lehre erfolgt aber die didaktisch-methodische Aufbereitung informatischer Inhalte weitestgehend abgekoppelt von der Unterrichtswirklichkeit, d.h. praxisfern[1]. Dies fällt umso stärker ins Gewicht, als der im Unterricht zu vermittelnde Inhalt ein wesentliches interaktives Moment des Lehrens und Lernens beinhaltet [Da05]: Zum einen sollen die Teilnehmer am Lernprozess über den Inhalt miteinander in Interaktion treten, zum anderen wird auch die Aneignung der Inhalte durch die innerhalb der Lerngruppe stattfindende Interaktion beeinflusst. Eine bloß theoretische Analyse von Unterricht kann dieser Dynamik aber kaum gerecht werden.

[1] Die (in Österreich) derzeit einzige Gelegenheit für Studierende, sich im Rahmen ihres Studiums praktisch mit ihrer zukünftigen Lehrerrolle im Informatikunterricht auseinanderzusetzen, besteht in einem zwölfwöchigen Schulpraktikum, vgl. [St04], S. 11.

Microteaching [Al76] [Jü98] ist ein handlungsorientierter Ansatz zur Vorbereitung der Praxisphase an Schulen, der auch in der Informatiklehrerausbildung angewandt wird[2]. Komplexe Handlungsstrategien der Lehrperson werden in möglichst einfache Fertigkeiten zerlegt, die isoliert geübt, auf Video aufgezeichnet, reflektiert und (beliebig oft) wiederholt werden können. Das Üben und Reflektieren erfolgt in kleinen Gruppen von angehenden Lehrkräften, die wechselweise die Lehrer- und die Schülerrolle einnehmen. Ein derartiges „praxisnahes" Setting im Rahmen universitärer Lehrveranstaltungen, in dem auch der Aspekt der Lehrer-Schüler Interaktion abgebildet werden kann, wird nachfolgend als Lehrarrangement bezeichnet. Microteaching berücksichtigt damit bis zu einem gewissen Grad die Forderung seitens systemisch-konstruktivistischer Pädagogik nach Selbsterfahrung von pädagogischen Beziehungen ([Re05], S. 268). Allerdings liegt der Fokus von Microteaching-Sequenzen primär auf der lehrerzentrierten Vermittlung von Inhalten und den sich daraus ergebenden Lehrer-Schüler Interaktionen. Dadurch ermöglichen sie nur begrenzt das Erleben der (und damit die Vorbereitung auf die) Komplexität des Unterrichts ([GG83] S. 20). Dieser Aspekt benötigt einerseits das Ineinandergreifen mehrerer Phasen von Unterricht und macht auch das Einbeziehen anderer, eher schülerzentrierter Sozialformen notwendig.

„Simulation von Unterrichtssituationen im universitären Umfeld" [An06] versteht sich diesbezüglich als alternatives bzw. ergänzendes Lehrarrangement: Studierende des Lehramtes sollen im komplexitätsreduzierten Rahmen einer studentischen Lerngruppe eine längere Sequenz zu einem vorgegebenen Thema gestalten und reflektieren, sodass eigene Referenzerfahrungen für die spätere schulische Praxis ermöglicht werden. Im Rahmen des Schulpraktikums soll dann die Rückkoppelung mit diesen Referenzerfahrungen der Unterrichtssimulation erfolgen.

Hinsichtlich des Designs derartiger Unterrichtssimulationen für die Informatiklehrerausbildung existieren allerdings zwei prinzipielle Problemfelder: Damit die Referenzerfahrungen mit Interaktionen in der Schulrealität vergleichbar werden, muss es einerseits gelingen, dass die Studierenden die Lehrer- und insbesondere die Schülerrolle möglichst echt annehmen. Andererseits muss ein Ansatz gewählt werden, der in der Simulation des Informatikunterrichts tatsächlich Interaktion zulässt, d.h. eher menschen- als computerzentriert ist.

2 Mikrowelten als Lehrarrangements

Mikrowelten (auch: „Mini-languages") haben in der Informatik Fachdidaktik eine lange Tradition. Reduzierte und visualisierende Systeme wie Logo [Pa85], Karel der Roboter [Pa95] oder Kara der Marienkäfer [RNH04] werden als ideale Lernumgebungen für das Programmieren beschrieben und reflektiert (vgl. z.B. [Br97], [BOM99]). Auch der Transfer dieses Konzepts zur Verwendung von Tabellenkalkulationssoftware im Programmierunterricht wurde bereits angedacht [An05].

[2] Vgl.z.B. http://ddi.cs.uni-potsdam.de/Lehre/SPS/ bzw.
http://ddi.cs.uni-potsdam.de/Lehre/SPS/MicroteachingBeispiele.htm (25. 01. 2007).

Mikrowelten eignen sich aber auch als Basis für Lehrarrangements zur Durchführung von Unterrichtssimulationen in obigem Sinne. Erste diesbezügliche Erfahrungen gehen auf eine informatikdidaktische Lehrveranstaltung mit 10 Studierenden an der Alpen-Adria Universität Klagenfurt im Wintersemester 2004/05 zurück. Dabei waren von einem bzw. zwei der Studierenden als „Lehrer[3]“ mit Hilfe von Mikrowelten[4] zwei simulierte Unterrichtseinheiten von ca. 50 Minuten Dauer zu gestalten. Die anderen sollten jeweils als „Schüler“ agieren, d.h. die gestellten informatischen Probleme lösen[5].

Ausgehend von ihren subjektiven Theorien über (Informatik-) Unterricht hatten die „Lehrer“ den Unterricht eigenständig zu planen und durchzuführen und sollten über den „Stoff“ mit den „Schülern“ in Beziehung treten. Dabei zeigten sich angesichts der zuvor angesprochenen Problemfelder zwei wesentliche Vorteile der Verwendung von Mikrowelten:

- Face-to-face Interaktionen werden im Informatikunterricht durch das Werkzeug Computer eingeschränkt (siehe auch Abschnitt 3.2). Mikrowelten der betrachteten Art können aber leicht real nachgebaut werden[6], die „Schüler“ (oder auch der „Lehrer“) können/müssen die Rolle des „Akteurs“ in dieser „Welt“ übernehmen. Dadurch wird einerseits das Miteinander-in-Beziehung-Treten wesentlich erleichtert, andererseits entsteht dadurch eine für die Studierenden ungewohnte informatische Perspektive: Nicht das individuelle Codieren sondern das gemeinsame (!) Problemlösen wird zur primären Tätigkeit des Programmierens (vgl. Abb. 1).

Abbildung 1: „Schülerin“ als „Akteur“ beim „Problemlösen“ in der Karel-Mikrowelt (links) bzw. „Lehrerin“-„Schüler“ Interaktion am realen Modell der Kara-Mikrowelt (rechts)

- Andererseits waren die Studierenden mit den Konzepten des Programmierens wohl vertraut, die verwendeten Mikrowelten waren aber unbekannt, sodass die als „Leh-

[3] Im Sinne besserer Lesbarkeit wird bei Sammelbezeichnungen stets das grammatikalisch männliche Geschlecht verwendet. Dies meint aber selbstverständlich stets beide biologischen Geschlechter!
[4] In diesen Unterrichtssimulationen wurden Kara der Marienkäfer bzw. eine Version von Karel dem Roboter verwendet.
[5] Neben den beiden Co-Autoren Ulrike Lassernig und Andreas Söllei waren dies Ernst Aschbacher, Gudrun Egger, Michael Gyarmati, Irmgard Hausharter, Andreas Lukasser, Peter Moser, Rene Scheriau und Rainer Swatek. Ihnen sei an dieser Stelle für ihr engagiertes Mittun in den Unterrichtssimulationen und ihre kritischen Diskussionsbeiträge gedankt.
[6] Dieser Aspekt der „methodischen Skalierbarkeit“ von Mikrowelten wurde bereits in [An05] beschrieben.

rer“ agierenden Studierenden aufgrund ihrer Unterrichtsvorbereitung einen Informations- bzw. Erfahrungsvorsprung erwerben konnten, der zur Herausbildung einer asymmetrischen (und daher schulähnlichen) „Lehrer“-„Schüler“ Beziehung ausreichte[7].

In den durchgeführten Einheiten der Unterrichtssimulation folgte auf eine kurze Phase des „Lehrervortrages“, in dem die jeweils verwendete Mikrowelt erklärt wurde, das Lösen von vorgegebenen Problemstellungen in der Mikrowelt durch die „Schüler“. Lehrerzentrierter Unterricht und kooperatives Lernen, in dem die Lehrperson eine aktivierende bzw. beratende Funktion hat, lösten einander also ab. Dadurch konnten sich die Unterrichtssimulationen bis zu einem gewissen Grad als „Netz gegenseitiger Beziehungen (Erwartungen, Verhaltensweisen, Wahrnehmungen)“ entwickeln[8], sodass trotz gründlicher Vorbereitung durch die „Lehrer“ stets ein Rest an Unsicherheit bei der Durchführung des „Unterrichts“ verblieb. Die Unterrichtssimulationen stellten so tatsächlich ein Modell für reales Unterrichtsgeschehen dar.

3 Erfahrungen

Die Unterrichtssimulationen wurden für die Reflexionsphase videographiert, um erste Anhaltspunkte über mögliche Lernfelder zu erhalten. In einer zweiten Phase wurden zwei der als „Lehrer“ agierenden Studierenden – Frau Lassernig und Herr Söllei – bei ihrem Schulpraktikum begleitet (und ihre „realen“ Unterrichtsauftritte ebenfalls auf Video aufgezeichnet), um Aufschluss darüber zu erhalten, inwieweit Referenzerfahrungen gemacht werden konnten. Die folgenden Ausführungen berücksichtigen die Ergebnisse beider dieser Phasen.

3.1 Unterrichten

Die Aufgabenstellung, nicht bloß Informationsinput in Form eines Lehrervortrages vorzubereiten, sondern eine gesamte „Unterrichtseinheit“ zu gestalten, verlangte von den „Lehrenden“, Unterricht als Prozess zu sehen. Zusätzlich zur inhaltlichen Planung mussten unter den gegebenen Rahmenbedingungen auch das Initiieren von Lernprozessen, die Organisation von Gruppenprozessen, das Interagieren mit den „Lernenden“ und das Abschließen der Einheit vorgedacht werden. Dadurch wurden Erfahrungen nicht nur im Hinblick auf das informatische Thema, sondern auch auf das Unterrichten (vor dem Hintergrund eines informatischen Themas!) ermöglicht:

Ein wesentlicher Referenzbereich war die für die Studierenden geänderte Wahrnehmung der Lehrerrolle. In der Reflexionsphase getätigte Äußerungen wie: „(...) man kennt eben nur seine eigene Schulzeit (...) wo man als Schüler nur die Durchführung des Unterrichts wahrnimmt]“ bzw. „vorher (...) die Lehrerrolle einfacher gesehen: Tests, Hausübung geben, prüfen (...) nachher: Lehrer in unterschiedlichen Rollen [als Leiter und als Begleiter/Berater]“ weisen auf Referenzerfahrungen bezüglich der tatsächlichen Dimensionen

[7] Vgl. [An06] zum zu Grunde gelegten Bild der Lehrer- und Schülerrolle.
[8] Vgl. [AP98], S. 78 zur systembezogenen Sichtweise von Unterricht.

von Lehrerhandeln im Unterricht hin. Besondere Beachtung fand dabei die gegenseitige Beeinflussung von Unterrichtsplanung und Unterrichtsdurchführung, entsprechend der Planung von Lernarrangements und den intendierten Interaktionen.

Ein zweiter Referenzbereich betraf die Wahrnehmung der Schüler durch den Lehrer. In Unterrichtssimulationen kennen die „Lehrer“ die „Schüler“ auch als Kollegen, d.h. in anderem sozialen Kontext, wissen daher um deren (in diesem Fall: informatisches) Können. Eine Referenzwahrnehmung ergab sich aus unterschiedlichem Verhalten zweier als „Schüler“ agierende Studienkollegen, die beide als „ausgezeichnete Informatiker“ geschätzt wurden, von denen der eine aber sehr zurückhaltend, der andere hingegen aktiv an der Unterrichtssimulation teilnahm. Das Beobachten der Schüler im Unterricht allein ließe offenbar nur bedingt Schlüsse auf deren Können zu, so die Schlussfolgerung der Studierenden. Vielmehr bedürfe es auch der verstärkten Auseinandersetzung mit dem individuellen Lernhintergrund von „auffälligen“ Schülern. Dafür wurde in der Reflexionsphase der Begriff der „Subjekt-sensiblen Didaktik“ geprägt.

Derartige Erfahrungen und deren Reflexion zeigten den im Rahmen der Unterrichtssimulationen “Lehrenden” die Bedeutung der Beziehungsebene auch für den Informatikunterricht. Durch den Kontrast zu den ursprünglich vorhandenen subjektiven Theorien über Unterricht gelang damit zusätzlich eine erste Orientierung hin zu einem „beruflichen Selbstbild“ zwischen den Polen des „Fachexperten“, des „didaktischen Experten“ und des „pädagogischen Experten“ [BVV00].

3.2 Informatik

Die aktive Interaktion/Kommunikation[9] zwischen Lehrern und Schülern ist Grundlage jedes Unterrichts [AT06]. Trotzdem fehlt dieser Aspekt im vorangegangenen Abschnitt, obwohl das Erzielen möglichst realitätsnaher Interaktion ein Hauptanliegen beim Design der Unterrichtssimulationen war. Aus Sicht der Autoren ist aber Kommunikation im Informatikunterricht spezifisch von den Gegebenheiten im Informatikunterricht bestimmt, damit aus informatischem Blickwinkel zu betrachten.

Informatikunterricht in der Schule findet häufig (üblicherweise?) in Computerräumen statt. Aufgrund ergonomischer Überlegungen (Lichteinfall, gerade Sitzhaltung,...), und um die Aufmerksamkeit der Schüler auf den Vortragenden/die Tafel/den Beamer zu fokussieren, werden dort Computer meist so positioniert, dass sie als „natürliche“ Barriere zwischen Lehrer und Schüler bzw. auch zwischen den in verschiedenen Reihen sitzenden Schülern stehen (vgl. Abb. 2a).

In diesem Punkt ergab sich eine Differenz zwischen der Schulwirklichkeit und den Unterrichtssimulationen. Letztere wurden ja über weite Strecken ohne Computer durchgeführt, die Studierenden konnten als Akteure in einer Mikrowelt die gestellten Probleme durch Darstellen und Diskutieren – eben kommunikativ – lösen. Gerade aus diesem Kontrast zwischen Kommunikation in einer face-to-face-Situation und der Kommunikationssituation „am Computer“ erwuchs eine weitere Referenzerfahrung:

[9] Die Begriffe Kommunikation und Interaktion werden hier synonym verwendet.


Abbildung 2 a: Struktur eines „typischen“ Computerraums mit Computern als „Barriere“
Abbildung 2 b: Beispiel für „indirekte Mensch-Computer-Mensch Kommunikation“

Bei einer Gesprächssituation, in der von einer Schülerin Hilfe bei der Lösung eines Codierungsproblems erbeten wurde, wurde die Kommunikation über den Computerbildschirm vermittelt: Die Frage (Aussage!) der Schülerin: „Das da [zeigt und blickt auf den Bildschirm] funktioniert nicht (...)“ wurde von der Praktikantin („Lehrerin“) mit „Da musst Du diese beiden Werte [zeigt und blickt auf den Bildschirm] verknüpfen (...)“ beantwortet und „gelöst“ (vgl. Abb. 2b). Obwohl sich also die beiden Kommunikationspartner direkt nebeneinander befanden, lief die Kommunikation ohne direkten Blickkontakt ab. Wir bezeichnen dieses Handlungsmuster[10] im Informatikunterricht als „indirekte Mensch-Computer-Mensch Kommunikation“.

Was bedeutet dies für das Lernen von Informatik? Eine von der Schülerin offenbar als problemhaltig erkannte Situation wurde anhand der vorhandenen Problemrepräsentation am Bildschirm „gemeinsam mit der Lehrerin bearbeitet“, die Darstellung des Problems dabei aber nicht verändert. Dies unterschied sich deutlich von den Prozessen der Problembearbeitung im Rahmen der Unterrichtssimulationen, wo das Umformulieren des Problems stets Teil der Problemlösung war. Aufgabe des Lehrers im Informatikunterricht müsse es daher sein, die Schüler auch zur jeweils individuellen Darstellung eines wahrgenommenen Problems anzuleiten, so eine Überlegung bei der Reflexion dieser beiden Situationen[11]. Dadurch ergab sich aus einer vom Standpunkt der Kommunikation problematisch erlebten Situation eine Referenzerfahrung für didaktisches Handeln im Informatikunterricht.

Problemrepräsentation ist im Zusammenhang mit dem (Erlernen von) Programmieren allerdings aus zwei unterschiedlichen Blickwinkeln zu sehen: Einerseits wird Programmieren häufig (auch von Informatikstudierendenen) mit dem Erlernen einer Programmiersprache bzw. dem Codieren des „fertigen Programms“ gleichgesetzt. Die Darstellung des Problems und dessen Aufbereitung für die maschinelle Lösung durch den Computer wird demnach als dem Programmieren vorgelagerte Tätigkeit verstanden.

[10] Dabei ist anzumerken, dass dieses Handlungsmuster auch bei „erfahrenen“ Informatiklehrern immer wieder zu beobachten ist (jeder beobachte sich in derartigen Situationen selbst!).
[11] Vgl. z.B. [PG90], S. 48 oder [Br94], S. 204 zum Zusammenhang zwischen Problemlösekompetenz und der Fähigkeit, das Problem „passend“ darzustellen)

Durch das Fehlen von Computerarbeitsplätzen in der Unterrichtssimulation war dort aber gerade diese Tätigkeit der wesentliche Inhalt der Programmiererfahrung, sowohl für „Lehrer“ als auch für „Schüler“. Dadurch wurden die Studierenden sensibilisiert, den Begriff des Programmierens zumindest unter didaktischen Erwägungen weiter zu fassen und als durchgängigen Problemlösungsprozess zu verstehen. Das Codieren als „Überantworten der Lösung an den Computer“ wurde damit zum Test, ob die Problemlösung „richtig“ ist.

Dies weist auf den zweiten Aspekt der Problemrepräsentation beim Programmieren hin: Die Darstellung des Problems orientiert sich dabei immer an den Möglichkeiten der verwendeten Software. Im Kontext von Mikrowelten sind dies die Fähigkeiten des jeweiligen Akteurs und die Beschränkungen, die ihm durch seine „Lebenswelt“ auferlegt werden[12]. Diese Sichtweise findet sich bei Duchâteau [Du92], der Programmieren definiert als das Delegieren einer Aufgabe[13] an einen Ausführenden/Akteur unter Mitgabe einer Gebrauchsanweisung (Programm), die ihn zum Bewältigen der Aufgabe befähigt. Diese Sichtweise zeigte sich auch implizit in den Problemlösungen der Studierenden, z.B. als in der verwendeten Karel-Roboterwelt die Fähigkeiten dadurch eingeschränkt waren, dass nur Linksdrehungen, nicht jedoch Rechtsdrehungen möglich waren. Dadurch wurde eine (wenn auch nur geringfügige) Änderung der ursprünglichen Problemaufbereitung notwendig. „Implizit“ meint dabei, dass wohl die Änderung bewusst durchgeführt (und auch verbalisiert) wurde, dass die diesbezügliche Bedeutung der Software-Umgebung aber erst in der Reflexion gesehen wurde.

4 Resümee und Ausblick

In [An06] wurde darauf hingewiesen, dass sich das Konzept der Unterrichtssimulation als Lehrarrangement nicht nur mit informatischen Mikrowelten, sondern auch mit anderen, methodisch adaptierbaren Themen umsetzen lässt. Die methodische Adaptierbarkeit ermöglicht dabei in der beschriebenen Weise das Annehmen der Lehrer- bzw. der Schülerrrolle und so das Entstehen von realitätsnahen Interaktionen im Rahmen der Unterrichtssimulation. Gleichzeitig können durch die methodische Adaption etablierter Themen auch Kontrasterfahrungen zur Situation im realen Schulunterricht hergestellt werden. Sowohl Situationen, die direkt auf die Schulpraxis übertragen werden können, als auch solche, die im Kontrast zur gängigen Praxis stehen, scheinen nämlich für das Herausbilden von Referenzerfahrungen wichtig zu sein.

Vom Standpunkt der Informatikdidaktik wird mit der Forderung nach methodischer Adaptierbarkeit zwar die Auswahl von Themen für die Simulation von Informatikunterrichts eingeschränkt, Unterrichtsentwürfe nach dem Muster der Informatik-Werkstätten[14] zeigen aber auch hier Möglichkeiten auf, Unterrichtssimulationen im universitären Umfeld mit einer anderen als der vorgestellten thematischen Ausrichtung umzusetzen. Den-

[12] Aufgrund dieser Sichtweise wird in diesem Artikel auch der Begriff „Mikrowelt“ und nicht „Mini-Sprachen“ verwendet. Die Sprache definiert einen wesentlichen Teil der Umgebung des Akteurs, seine „kleinen Welt“.
[13] Duchâteau spricht von Aufgaben anstelle von Problemen und argumentiert, das speziell im Anfangsunterricht die kleinen Herausforderungen die Bezeichnung „Problem“ wohl nicht verdienen.
[14] Vgl. z.B. http://www.swisseduc.ch/informatik/werkstatt/ (18. 04. 2007)

noch bleibt auch das Programmieren in diesem Zusammenhang interessant. Die geschilderten Ergebnisse deuten darauf hin, dass durch die Erfahrungen im Rahmen der durchgeführten Unterrichtssimulationen Studierende zum Nachdenken über den „didaktischen Kern“ des Programmierens angeleitet werden können: Ein Programm bewirkt, dass „etwas“ in einer bestimmten „Arbeitsumgebung“ eine Aufgabe löst.

Im Fall von Mikrowelten ist diese Arbeitsumgebung eine „conceptually simple notional machine“ [BOM99], im Fall professioneller Programmiersprachen (die dennoch in Schulen zum Erlernen des Programmierens verwendet werden[15]) die jeweilige virtuelle Maschine bzw. der Computer selbst. Damit erfordert aber das Arbeiten mit einer derartigen Programmierumgebung – wie der Name schon sagt – auch das Kennen und (einfach) Beschreiben-Können der Umgebung. Die Umgebung eines Programms ist – in letzter Instanz – die Hardware inklusive der durch die Programmiersprache auferlegten Beschränkungen. Die zugehörige „conceptually simple notional machine“ kann das Modell einer Register- oder Turingmaschine sein. Programmieren bedeutet somit mehr als „nur“ Codieren, mehr als „nur“ Problemlösen. Programmieren ist ganz im Sinne von [De03] „ein Fenster in die Welt der Computational Mechanics“ und verknüpft Bereiche des Informatikstudiums, die von Studierenden nicht selbstverständlich als zusammengehörig erkannt werden.

Zwar konnte dieser „Transferschritt“ von „Mikrowelten“ zur „Computerwelt“ beim ersten Durchgang von Unterrichtsimulationen (noch) nicht gemacht werden, die Erfahrungen der Studierenden und mit den Studierenden deuten aber darauf hin, dass derartige Lehrarrangements bei geeigneter Begleitung die skizzierte Sichtweise bei zukünftigen Informatiklehrern fördern können. Dass dabei zusätzlich individuell wertvolle (Vor-) Erfahrungen über das Unterrichten gemacht werden (können), ist für die Ausbildung von Lehrern der Informatik wohl mehr als nur beiläufiges Beiwerk.

Literaturverzeichnis

[Al76] Allen D. (1976): Microteaching – ein Überblick. In: (Zifreund W. Hrsg.): Training des Lehrverhaltens und Interaktionsanalyse. Beltz, Weinheim und Basel 1976; S. 1-28

[An05] Antonitsch P.K.: Standard Software as Microworld?. In (Mittermeir R.T. Ed.): From Computer Literacy to Informatics Fundamentals. Proc. 1st Int. Conf. On Informatics in Secondary Schools – Evolution and Perspectives (ISSEP), Klagenfurt 2005. Springer, Berlin Heidelberg, 2005; S. 189-197

[An06] Antonitsch P.K.: Simulation von Unterrichtssituationen im universitären Umfeld als Beitrag zum Erwerb von Praxiskompetenz. In (Hilligus A.H., Rinkens H.-D. Hrsg.): Standards und Kompetenzen – neue Qualität in der Lehrerausbildung? LIT Verlag, Berlin 2006; S. 391-397

[AP98] Altrichter H., Posch P.: Lehrer erforschen ihren Unterricht. 3. Aufl. Klinkhardt, Bad Heilbrunn 1998

[AT06] Antonitsch P.K., Tischer K.: Kommunikation im sozialen System Schule. In (Antonitsch P., Delanoy W., Palencsar F., Theuermann A., Tischler K. Hrsg.): Ich + Du ≠ Wir. Wege zur Kommunikation in der LehrerInnenbildung. dravaDiskurs, Klagenfurt 2006; S. 15-28

[15] Die verwendeten Programmiersprachen überdecken dabei – zumindest in Österreich – ein weites Spektrum von der Objektorientierung (Java, C#) über Skriptsprachen bis hin zu „maschinennahen“ Sprachen wie C.

[Br94] Brusilovsky P.: Explanatory Visualization in an Educational Programming Environment: Connecting Examples with General Knowledge. In: (Blumenthal B., Gornostaev J., Unger C. Eds.): Human-Computer Interaction. Selected Papers of the 4th International Conference, EWHCI '94, St. Petersburg 1994. Springer, Berlin Heidelberg 1994, S. 202-212

[Br97] Brusilovsky P., Calabrese E., Hvorecky J., Kouchnirenko A., Miller P.: Mini-languages: A Way to Learn Programming Principles. In: Education and Information Technologies 2 (1) 1997; S. 65-83

[BOM99] du Boulay B., O'Shea T., Monk J.: The black box inside the glass box: presenting computing concepts to novices. In: International Journal of Human-Computer Studies, 51(2) 1999; S. 265-277

[BVV00] Beijaard D., Verloop N., Vermunt J.D.: Teacher's perceptions of professional identity: an exploratory study from a personal knowledge perspective. In: Teaching and Teacher Education, 16 2000, S. 749-764

[Da05] Dauber H.: Der Lehrer in der Schule der Zukunft: Coach oder Pädagoge? In: (Dauber H., Krause-Vilmar D. Hrsg.): Schulpraktikum vorbereiten. Pädagogische Perspektiven für die Lehrerbildung. Klinkhardt, Bad Heilbrunn 2005, S. 23-37)

[De03] Denning P.J.: Great Principles of Computing. In: Communications of the ACM Vol. 46, No. 11 November 2003

[Du92] Duchâteau C.: From "DOING IT…" to "HAVING IT DONE BY…": the heart of programming. Some didactical thoughts. In: Preproceedings NATO ARW Cognitive Models and Intelligent Environments for Learning Programming. S. Margherita Ligure, Genova 1992

[GG83] Grell J., Grell M.: Unterrichtsrezepte. Beltz, Weinheim und Basel 1983

[Jü98] Jürgens B.: Wie lernen Lehrer Lehrerverhalten? In: Praxis Schule 5–10, Heft 4/1998, S. 38-43

[Pa85] Papert S.: Gedankenblitze. Kinder, Computer und Neues Lernen. Rororo, Reinbek bei Hamburg 1985

[Pa95] Pattis R.E.: Karel the Robot. A Gentle Introduction to the Art of Programming. 2nd ed. John Wiley & Sons, New York 1995

[PG90] Pennington N., Grabowski B.: The Tasks of Programming. In: (Hoc J.-M., Green T.R.G., Samurçai R., Gilmore D.J. Eds.): Psychology of Programming. Academic Press, London 1990, S. 45-62

[Re05] Reich K.: Systemisch-konstruktivistische Pädagogik. Beltz, Weinheim und Basel, 5. Aufl. 2005

[RNH04] Reichert R., Nievergelt J., Hartmann W.: Programmieren mit Kara. Ein spielerischer Zugang zur Informatik. Springer, Berlin Heidelberg 2004

[St04] Studienplan Lehramt an der Fakultät für Wirtschaftswissenschaften und Informatik an der Universität Klagenfurt 2004; verfügbar unter: http://www.uni-klu.ac.at/ home/stplaene/wiinfo/lawiinfo04.pdf (25. 01. 2007)

Lauschen am Internet – Experimente mit einem Nachrichten-Rekorder im Informatikunterricht

Ute Heuer

Didaktik der Informatik
Fakultät für Informatik und Mathematik
Universität Passau
94030 Passau
heuer@fim.uni-passau.de

Abstract: Der Themenbereich Protokolle/Rechnernetze hat Einzug gehalten in einige Lehrpläne für die Sekundarstufe I. In der Unterrichtspraxis scheint die Versuchung recht groß, den Schülern gerade in diesem Themenbereich in relativ kurzer Zeit viele Fakten, Modelle und Veranschaulichungen zum Repetieren vorzulegen. Schließlich ist das Gebiet Rechnernetze so wichtig wie umfangreich. Die Autorin möchte mit diesem Beitrag ein Gegensteuern unterstützen. Es werden Schülerexperimente mit einem „Nachrichtenrekorder" skizziert. Dabei wird der Aufruf von Seiten aus dem Internet „belauscht", der zugehörige Netzverkehr wird beobachtet, hinterfragt. Vermutungen können mit weiteren „Lauschexperimenten" geprüft werden. Zugrunde liegende Konzepte wie Client-Server-Rollenverteilung und Schichtung von Zuständigkeiten werden angesprochen, wenn die Analyse der „Lauschaufnahmen" dies nahe legt und ein Verständnis bzw. eine Einordnung der beobachteten Vorgänge erleichtert.

1 Einleitung

Es existieren eine Reihe kleiner einfacher Simulationen zum Gebiet Rechnernetze, die sich auch für den Schulunterricht eignen, z.B. in [KR05] oder [Si03]. Typischerweise stellen sie für einzelne Mechanismen Visualisierungen bereit. So kann sicherlich ein Verständnis dieser Kommunikationsmechanismen gut unterstützt werden. Jedoch bieten diese Simulationen häufig nur recht beschränkte Interaktionsmöglichkeiten. Weiter gibt es eine Simulation, in der man ein Rechnernetz konfigurieren, und damit experimentieren kann [SS04].
Schülerexperimente sind nicht ohne Grund fester Bestandteil des naturwissenschaftlichen Unterrichts. Diese „hands on – minds on" Vorgehensweise bietet sich auch für manche Themengebiete des Informatikunterrichts an. Die Autorin gibt in diesem Beitrag Anregungen für Schülerexperimente im Themenbereich Protokolle/Rechnernetze im Unterricht der Mittelstufe.
Dabei wird folgendes Ziel verfolgt: Schüler sollen selbst einige (nur einige!) Aspekte von Protokollen und Mechanismen der Kommunikation von Rechnern im Experiment beobachten, hinterfragen und versuchen, diese einzuordnen. Sie sollen lernen, ihre Vor-

stellungen in weiteren Experimenten einer Prüfung zu unterziehen und ggf. zu revidieren.
Die hier vorgestellten Experimente gehen von der Beobachtung und Analyse von tatsächlichem Netzverkehr in das und aus dem Internet aus. Sie beschränken sich dabei auf eine klassische Anwendungssituation: das Laden von Seiten aus dem Netz. Diese Lauschexperimente am Internet werfen viele weiterführende Fragen auf und fordern zur Prüfung von Vermutungen und Antworten heraus. Nach und nach schauen die Lernenden dabei weiter unter die Motorhaube der ausgetauschten Daten. So werden einige Aspekte des http Protokolls erforscht. Bei anstehenden Fragen werden Einordnungshilfen in Form von Lehrerinputs gegeben. Schüler werden auch eigene Versuchsabläufe planen und dadurch ihr Wissen und ihre Fähigkeiten weiter vertiefen. Z.T. können die so entstehenden Experimente jedoch mehr Fragen aufwerfen als sie beantworten. Diskutiert man den sinnvollen Umgang mit dieser Situation, so kann ein wichtiger Beitrag zur Orientierung in einem komplexen Problembereich geleistet werden.
Die Schüler experimentieren mit einem frei zugänglichen Werkzeug aus dem universitären Umfeld. Es ist ein Rekorder (engl.: Packet Sniffer), der ins Netz einlaufende und auslaufende Pakete aufnehmen und übersichtlich darstellen kann. Dieser wird beispielsweise von Fachleuten wie Systemadministratoren genutzt oder in manchen Veranstaltungen an der Universität von Studierenden der Informatik eingesetzt [KR05]. Das Werkzeug bietet entsprechend viele Funktionen an. Für die Arbeit mit den Schülern ist es günstig, sich auf wenige dieser Funktionen und eine sehr kleine, übersichtliche Anzeige zu beschränken. So wurde der Rekorder für die hier angesprochenen Experimente geeignet voreingestellt: Er nimmt nur tcp-Segmente auf (die mein Rechner versendet bzw. die an meinen Rechner adressiert sind) und zeigt nur http-Nachrichten an. In diesem Zusammenhang sprechen wir mit Schülern von einem „Nachrichten-Rekorder“ (kurz Rekorder), da wir uns hauptsächlich für das http-Protokoll interessieren.

2 Vorbereitung der Arbeit mit einem Nachrichtenrekorder

Schüler sollten zunächst eine grundsätzliche Vorstellung von Informationsnetzen haben. Diese wird meist in der Unterstufe erworben. Eine schöne Darstellung findet sich in den neueren Schulbüchern in diesem Bereich, exemplarisch sei verwiesen auf [Br04]. Günstig ist es weiter, wenn Schüler einige wenige typische html-Syntaxelemente wieder erkennen können.
Ein Unterrichtsgespräch/-spiel im Sinne der „Sendung mit der Maus“ (bzw. der Netzseiten dazu) kann ggf. als Einstieg in die Thematik dienen [Ma07]. Dann kann im Laufe der Lauschexperimente manchmal Bezug zu diesem Spiel und den Rollen der einzelnen Personen bzw. Institutionen genommen werden. Dies kann helfen, neue Begriffe und Zusammenhänge einzuordnen.
Bevor einige Lauschexperimente durchgeführt werden, wird Funktionsweise und Handhabung des Nachrichten-Rekorders gezeigt. Dabei kann gut mit einem DVD-Rekorder verglichen werden. Ein solcher Rekorder besitzt Knöpfe zur Kanalwahl, zum Aufnehmen und zum Abspielen. Einen Aufnahmeknopf finden wir auch bei unserem Nachrichten-Rekorder. Nach der Aufnahme wird diese automatisch angezeigt, so dass sich ein Abspielknopf erübrigt. Wir stellen uns vor, dass unser Rekorder direkt an der Netzkabelbuchse lauschen kann. Da mehrere zur Verfügung stehen können, müssen wir analog zur

Kanalwahl beim DVD-Rekorder eine Netzverbindung auswählen. Was dort ein- und ausläuft, kann dann aufgenommen werden. Ähnlich wie bei einem DVD-Rekorder bekommen wir die Aufnahme nicht als Reihe von 0en und 1en angezeigt, sondern in einer Art und Weise, die für uns interessanter und aufschlussreicher anzusehen ist. Beim DVD-Rekorder sehen wir „laufende Bilder“, bei unserem Nachrichten-Rekorder sehen wir übersichtlich dargestellten lesbaren Text.
Wir verwenden das Werkzeug Ethereal/Wireshark [EW07] als Nachrichtenrekorder. Er bietet ein übersichtliches Bedienfeld. Weiter kann für eine Überblickssicht voreingestellt werden, welche Attribute einer Nachricht man sich anzeigen lassen möchte. Möglichst wenige auszuwählen erleichtert sicherlich die Arbeit mit dem Rekorder für Schüler, vgl. Abb. 1.

3 Experimente mit http Nachrichten

3.1 Texte und Bilder

Es ist weiter vorteilhaft, anfangs ausschließlich mit der Überblickssicht zu arbeiten. Die Schüler belauschen zunächst den Aufruf einiger „Miniseiten“. Diese sind beim Experimentieren mit dem Rekorder einfacher zu handhaben als die meisten Seiten im Netz, denn es werden nur wenige Nachrichten ausgetauscht (Abb. 1).
Erste Fragen können gestellt werden: Woran erkennt man in dieser knappen Überblickssicht die Nachricht, die die Netzseite anfordert? In welcher Nachricht wird dann vermutlich die Seite geliefert? Was wird noch geliefert?
Den Wunsch, ein kleines Bild zu laden, habe ich als Benutzer des Browsers gar nicht geäußert, aber ich bekomme es trotzdem? Wo liegt dieses „favicon“, wie sieht es aus? Hier kann man auf die Idee kommen, unter lehramt.fmi.uni-passau.de/favicon.ico einmal nachzuschauen. Das kleine Favoritensymbol kann im Browserfenster angezeigt werden. Vorgesehen ist ein Symbol für die Adresse lehramt.fmi.uni-passau.de. Wird das auch angezeigt, wenn man Unterordner anwählt? Welche Nachrichten werden dann ausgetauscht? Auch das kann man anhand der mitgegebenen Links (Abb. 1) gleich ausprobieren. Im Experiment zeigt sich, dass das Favoritensymbol zwar vom Browser angezeigt, aber weder erneut angefordert, noch erneut geschickt wird. Woran kann das liegen?
Welche Vorteile bringt das Zwischenspeichern von Information für den Benutzer? (Man kann lokal und damit ggf. schneller auf Information zugreifen. Der Nachrichtenverkehr ins Netz wird reduziert.) Gibt es auch Nachteile? (Man hinterlässt viele Spuren. Ggf. muss geprüft werden, ob sich die Seite im Netz inzwischen geändert hat, d.h. ob der Zwischenspeicher noch aktuell ist.)
Solche Fragen führen zu Versuchen mit dem Zwischenspeicher des Browsers (Cache). Man kann sich anzeigen lassen, was alles momentan im Zwischenspeicher liegt. Man kann diesen löschen, und eine erneute Anfrage einer Seite belauschen. Bei Interesse kann auch mit einer Seite mit großem Bild experimentiert werden. Wie lange dauert das erste Laden des Bilds? Aus der zugehörigen Lauschaufnahme kann die Zeit ausgelesen werden, die vom Abschicken der Anfrage bis zur Ankunft des Bildes vergeht, vgl.

Abb. 2. Wo findet sich das Bild im Zwischenspeicher des Browsers?[1] Welche Nachrichten werden beim erneuten Laden der Seite ausgetauscht? In der Überblickssicht fallen sofort die Antworten „not modified“ ohne Text/Bild auf, vgl. Abb. 2. Nach dem Favoritensymbol wurde nicht erneut gefragt. Warum fragt der Browser aber erneut nach dem Bild, obwohl er es im Zwischenspeicher hat und von dort doch gleich laden könnte?[2] Am Beispiel einer Auktionsseite kann geklärt werden, dass eine alte aus einem Zwischenspeicher nutzlos sein kann, wohingegen es auf das Favoritensymbol wohl nicht ganz so ankommt.

Mit einem Zwischenspeicher haben Schüler schon im Zusammenhang mit Standardsoftware gearbeitet. Welche Gemeinsamkeiten, welche Unterschiede fallen auf? Das Konzept Zwischenspeicher kann nun vertieft werden. Wenn mein Browser über einen Zwischenspeicher verfügt, dann verwendet vielleicht auch mein „Gesprächspartner“, der Webserver, einen Zwischenspeicher. Was könnte da vielleicht festgehalten werden? Den Zwischenspeicher meines Browsers kann ich löschen. Den Zwischenspeicher von Webservern im Netz kann ich mir weder zeigen lassen, noch kann ich etwas löschen.

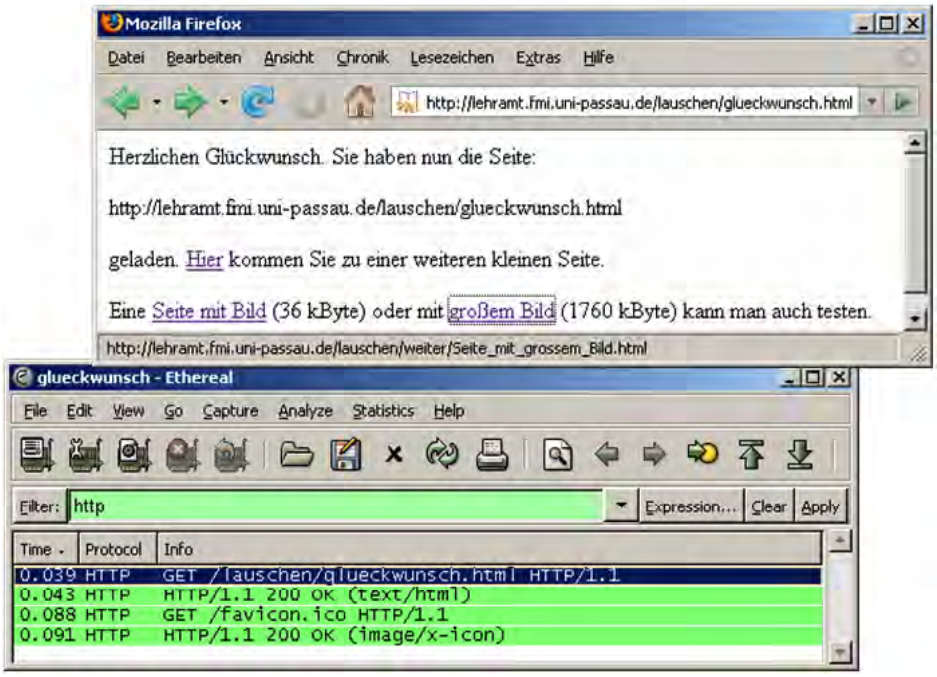


Abbildung 1: Der Aufruf einer kleinen Seite mit einem Browser wird belauscht.

[1] Weiterführende Fragen wären hier: Warum ist es sinnvoll, einen Teil des Zwischenspeichers im Arbeitsspeicher und einen Teil auf der Festplatte des Rechners zu halten. Unter welchen Umständen „wandert“ ein Bild vom Arbeitsspeicher auf die Festplatte.

[2] Ggf. werden Schüler stutzig und überlegen, ob „beim zweiten Mal“ auch die Anfragen des Browsers anders gestellt sind. Ja, es gibt ein entsprechendes Attribut der Anfragenachricht (if modified since), welches in unserer Überblickssicht nicht angezeigt wird. Der Rekorder stellt bei Bedarf eine zweite Sicht bereit, in der Details der ausgewählten Nachricht zu sehen sind. Dort findet man u.a. auch das gesuchte Attribut.

Falls die „Sendung mit der Maus“ eingangs thematisiert wurde, kann man an dieser Stelle daran anknüpfen und ansprechen, dass auch Verbindungspunkte (im Spiel werden Sie z.T. Wegweiser genannt, aber auch der im Spiel genannte Provider fällt darunter) Zwischenspeicher halten. [3]
Möchte man die Datenschutz-Problematik an dieser Stelle vertiefen, so bietet sich z.B. Material aus dem entsprechenden Kapitel eines Schulbuchs der 9.Klasse des Gymnasiums an [Hu07]. Dort findet sich die interessante Geschichte „Spurensicherung für einen Kriminalfall in einem Zwischenspeicher eines Servers“. Weiter gibt es schöne, handlungsorientierte Aufgaben, z.B. die Übung „Neue Informationen durch Verknüpfung von Daten – Anonymität von Fragebögen???“, in welcher die Schüler mit einer Beispieldatenbasis arbeiten.
Erste Beobachtungen zum Austausch von Nachrichten zwischen den beiden Gesprächsteilnehmern „Browser“ und „Webserver“ beim Aufruf einer Seite können nun festgehalten werden. Dazu bietet sich eine Notation, wie sie Abb. 3 zeigt, an. Man kann von Unwichtigem abstrahieren, wie die genaue Aufnahmezeit einer Nachricht und sieht auf einen Blick, dass sich hier Anfragen von meinem Browser und Antworten von meinem Gegenüber abwechseln.
Manchmal unterscheidet man die Gesprächsteilnehmer je nach der Rolle, die sie im Gespräch spielen. Stellt ein Partner Anfragen und erhält er Antworten, so spricht man von einem Dienstnutzer oder Client. Derjenige, der Anfragen entgegennimmt und Antworten gibt heißt dann Dienstanbieter oder Server.[4]

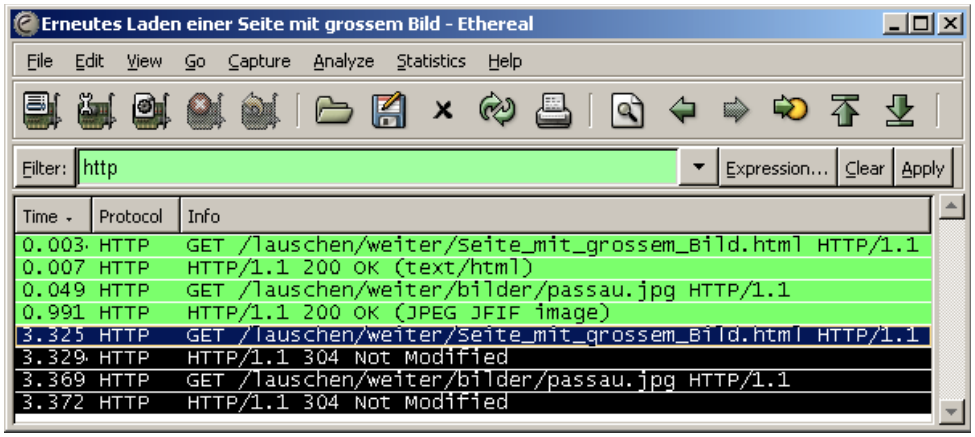


Abbildung 2: Beim erneuten Laden erhält man andere Antwortnachrichten. Die Antwort auf die Anfrage nach dem Bild erfolgt prompt. Das Bild wird nicht erneut geliefert.

[3] Während des Experimentierens im Unterricht werden Seiten, Bilder etc. oft nur von einem Zwischenspeicher (Proxy-Server) der Schule geladen. Auch dieser Sachverhalt kann bei Interesse mit Schülern diskutiert werden.
[4] Wurde die Funktion des Zwischenspeichers der Schule (Proxy) angesprochen, so hat man ein Beispiel eines Gesprächsteilnehmers, der sowohl als Server als auch als Client arbeitet.

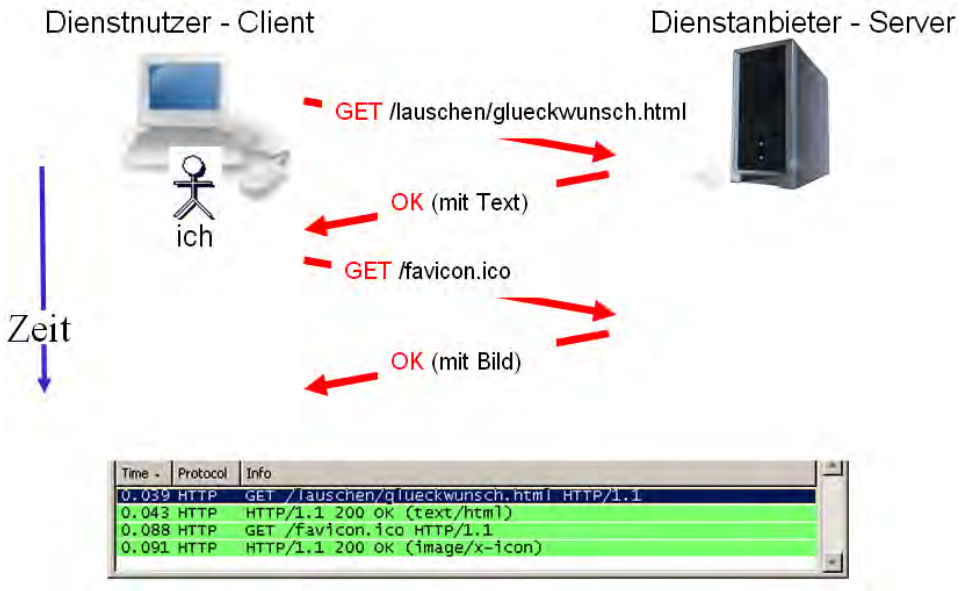


Abbildung 3: Die beim Aufruf einer kleinen Seite ausgetauschten Nachrichten im Überblick (Client-Server-Interaktionsdiagramm)

Jetzt folgt eine Information zum Begriff Protokoll, dem der Rekorder eine eigene Spalte widmet. Menschen kommunizieren, indem Nachrichten gesendet und empfangen werden. Der Empfänger einer Nachricht, versucht angemessen zu reagieren. Dafür gibt es Regeln. Typisches Beispiel: (Ich frage.) Wieviel Uhr ist es bitte? (Der Gefragte versteht mein Anliegen und liest die Zeit auf seiner Armbanduhr ab und antwortet.) Es ist drei Uhr.

Ähnlich kommunizieren Rechner. Sie verstehen sich, wenn Sie das gleiche Protokoll „sprechen“. Im Protokoll sind z.B. Befehlswörter für Anfragen (wie GET) oder bestätigende Wörter (wie OK) festgelegt.

Weiter ist festgeschrieben, was nach den Befehlswörtern zu stehen hat. Und es ist festgelegt, welche Reaktion erwartet wird (GET heißt ich erwarte, dass mir die Seite gesendet wird). Das Protokoll, was sowohl mein Browser, als auch sein Gegenüber, ein Webserver, spricht, heißt hypertext transfer protocol (http). Es gibt noch eine Vielzahl weiterer Protokolle, die die Kommunikation zwischen Rechnern regeln können.

Nun wird es Zeit, etwas genauer in die http-Nachrichten hineinzuschauen. Kann man dort den gesendeten Quelltext der Seite wieder finden? Findet man auch die Bilder wieder? Dazu arbeiten die Schüler mit einer geänderten Voreinstellung des Rekorders. Es erscheint eine zweite Sicht, in der Details der ausgewählten http-Nachricht zu sehen sind (Abb. 4).

Die Schüler können bei Interesse auch in einer dritten Sicht die Nachricht im Rohzustand betrachten. Sie sehen dort sozusagen die 0en und 1en, die in den Computer hinein und hinauslaufen, allerdings als Hexadezimalzahlen gebündelt. Man kann in der Antwortnachricht, die das Bild enthält, diesen Teil markieren, mit Hilfe des Rekorders exportieren und erhält so tatsächlich das gesuchte Bild zum Anschauen. Misstrauische können sich das exportierte Bild zusätzlich in einem geeigneten einfachen Editor (Hex-

Editor) ansehen und einige Zahlen mit den Zahlen der Lauschaufnahme vergleichen (Abb. 5).

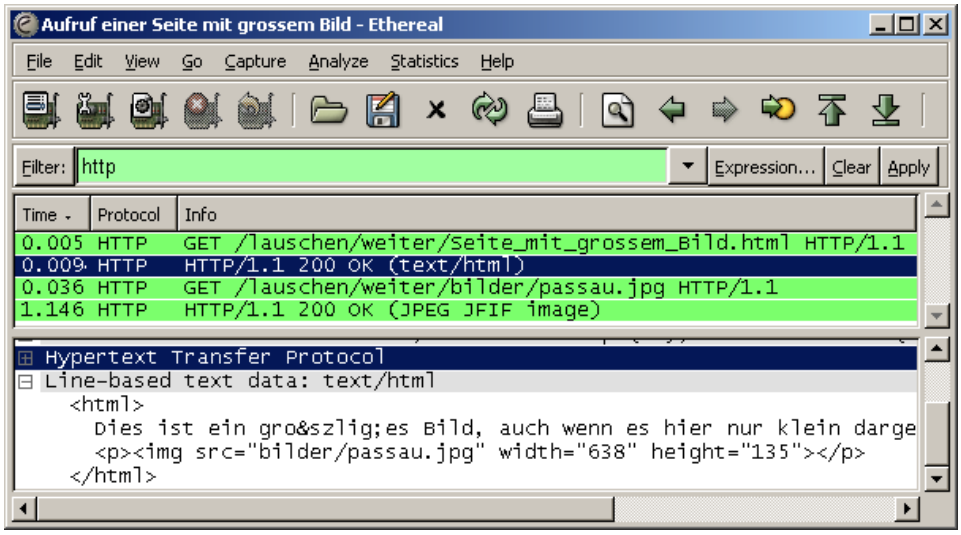


Abbildung 4: Die markierte Nachricht enthält den Quelltext der angeforderten Seite

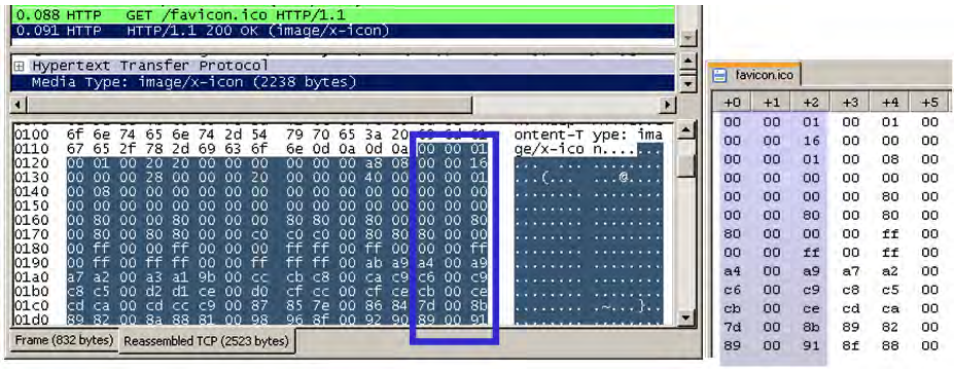


Abbildung 5: Diese Nachricht enthält das angeforderte Bild, es besteht aus nichts anderem als den markierten Zahlen. Misstrauische können einige Zahlen vergleichen, z.B. die doppelt markierten.

3.2 Was mein Browser über mich verrät

Wenn man im Netz Seiten aufruft, wundert man sich manchmal, dass automatisch Software für ein bestimmtes Betriebssystem zum Laden vorgeschlagen wird. Oder man bekommt eine deutsche Seite, obwohl unter der angewählten Adresse auch englische, französische oder italienische Seiten bereitstehen. Ein Beispiel ist die europäische Mozilla-Seite. Um zu klären, was für weitere Informationen in ihrer Anfrage nach einer Seite

übermittelt werden, können sich Schüler eine Anfrage nach einer Seite im Rekorder genauer ansehen. Sie sehen so, dass im Nachrichtenkopf der Anfrage typischerweise Informationen über das verwendete Betriebssystem, den Browser, Sprachpräferenzen und die Netzseite, von der ich kam, eingetragen sind.
Nun kann mit den Spracheinstellungen experimentiert werden. Diese können im Browser geändert werden. Prompt bekommt man die Seite in der gewünschten Sprache. In der Lauschaufnahme ist diese Sprache dann auch im entsprechenden Attribut als erstes vermerkt.
Bei Interesse können auch Seiten im Netz besucht werden, die direkt anzeigen, welche Daten man via Anfrage gerade an den Webserver übermittelt [Re07] [Ze07].

3.3 Passwörter

Interessant ist weiter, eine Seite zu laden, die mit einem einfachen Passwortschutz versehen wurde. Mit Hilfe des Rekorders erkennen die Schüler, dass das eingegebene Passwort sich im Klartext in der Nachricht befindet (Abb. 6). Wird die Nachricht auf dem Weg durchs Netz belauscht, kann dieser Passwortschutz ausspioniert werden. In diesem

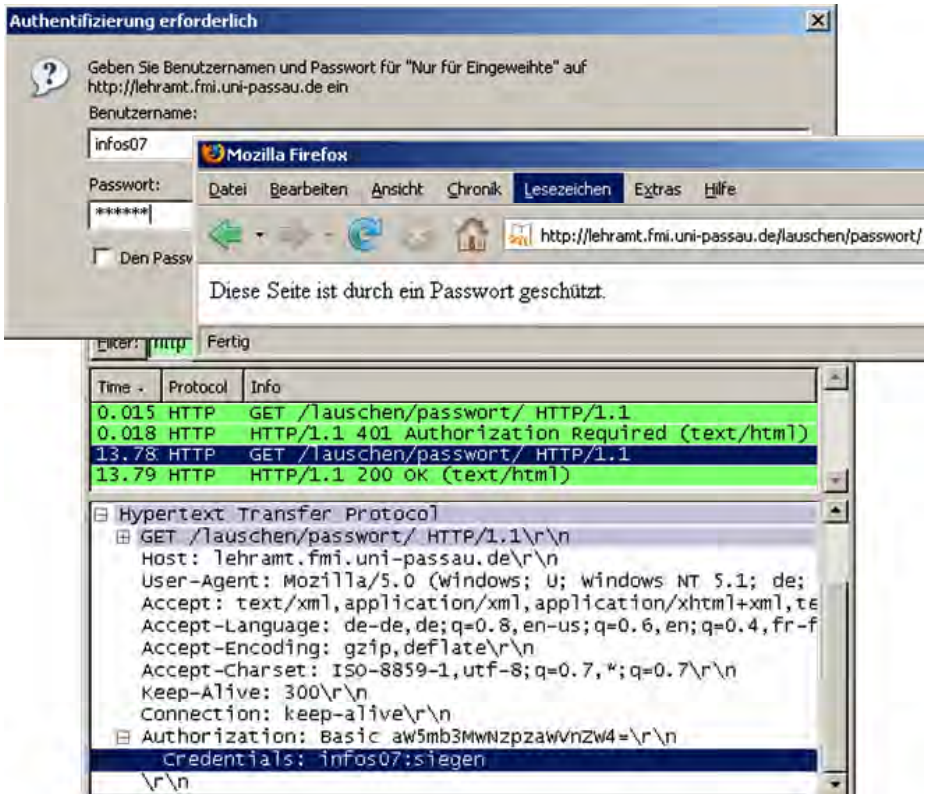


Abbildung 6: Ein einfacher Passwortschutz wird belauscht. Der Benutzername lautet hier infos07, das Passwort siegen. Es kann direkt aus dem Kopf der Nachricht ausgelesen werden.

Zusammenhang kann auf Verschlüsselungsmöglichkeiten hingewiesen werden. Ob diese nötig sind, hängt von der Sensibilität der Daten ab, die übertragen werden.
Ein kleiner Vergleich mit altbewährten Kommunikationsmitteln ist aufschlussreich. Entgegen anfänglicher Unkenrufe (im Jahr 1865) hat sich die Postkarte neben dem Brief etabliert. Unverschlüsselte Nachrichten auszutauschen ist vergleichbar mit dem Schreiben von Postkarten. Meine Kreditkartendaten schreibe ich nicht auf eine Postkarte, Urlaubsimpressionen schon.

3.4 Ethischer Aspekt

Als Einstieg in eine sich anschließende Diskussion mit Schülern kann ein Magazinartikel der Art „Ethereal Rekorder deckt Liebesaffäre auf“ gewählt werden, siehe z.B. [Re06]. Es öffnet sich ein klassisches Spannungsfeld zwischen technisch möglichem und ethisch bzw. juristisch vertretbarem. Schülern wird weiter bewusst, dass es schützenswerte Daten gibt (nicht nur in Liebesdingen). Die Verschlüsselung von Daten bietet einen ziemlich guten, wenngleich keinen absolut sicheren Schutz vor dem Zugriff unberechtigter Dritter. Auch dieses deutet der oben zitierte Artikel an.

4 Unsere HTTP Nachrichten haben einen TCP Kopf

In Seiten im Netz sind normalerweise eine größere Anzahl von Bildern und andere Dokumente integriert. So zeigt die Lauschaufnahme zum Aufruf der europäischen Mozillaseite eine ganze Latte von Anfragen und Antworten. Es fällt auf, dass manchmal zwei Anfragen hintereinander losgeschickt werden bzw. zwei Antworten hintereinander eintreffen, vgl. Abb. 9.
Aber wie können dann auf Nutzerseite Frage und Antwort einander zugeordnet werden? Die Lauschaufnahme zeigt, dass die Antworten den angeforderten Dateinamen in keiner Weise enthalten. Wir wählen als Beispiel die Anfrage nach dem Bild „firefox-title.png“ aus und suchen nach der Antwort mit diesem Titelbild. Die gute Idee, einfach einerseits alle Anfragen, andererseits alle Antworten durchzuzählen, führt bei dieser Lauschaufnahme zur Antwort mit der Nummer 51. Ein Test der Art von Abb. 5 zeigt aber, dass die Nachricht 51 nicht das gesuchte Titelbild enthält.
Wir veranschaulichen das Problem mit einer kleinen Demonstration. Der Lehrer bietet einen Stiftservice an. Ein Schüler mit verbundenen Augen nutzt ihn und fordert erst einen und dann noch einen Stift an. In welcher Reihenfolge könnten die Stifte eintreffen?

Abbildung 7: Drei mögliche Abläufe der Demonstration „Stiftservice“ (Kapitel 4)

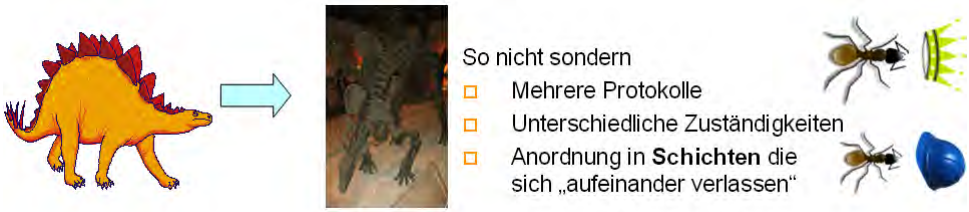


Abbildung 8: Warum nicht nur ein Protokoll, das alles macht?

Der Nutzer ärgert sich zu Recht über den Stiftservice, weil Buntstift und Bleistift so gleich geformt sind und kein Hinweis gegeben wird, was nun welcher Stift ist. (Abb. 7)
Nun kann thematisiert werden, warum es sinnvoll ist, mit einer Protokollhierarchie anstelle eines Protokolls zu arbeiten. Man kann mit einem Bild einsteigen (Abb. 8). Dinosaurier sind ausgestorben, sie waren zu wenig anpassungsfähig. Ameisenstaaten kennen klare Aufgabenverteilungen. Arbeiterameisen stellen Aufbaudienste zur Verfügung, die die Königin nutzt. Dieses Prinzip findet sich in vielen Gebieten der Informatik wieder. Auf unseren Themenbereich angewendet heißt das, es gibt eine Protokollhierarchie und nicht nur ein riesiges Protokoll, das alles regel
Die Schüler steigen in der Protokollhierarchie etwas hinab, um Licht in die Ansammlung von verschiedenen Anfragen und Antworten zu bringen. Unsere http-Nachrichten haben einen tcp-Kopf. Betrachtet man beides zusammen, spricht man oft von tcp-Segmenten. (Eine Aufteilung der Nachricht auf mehrere Segmente wird nicht weiter angesprochen.)

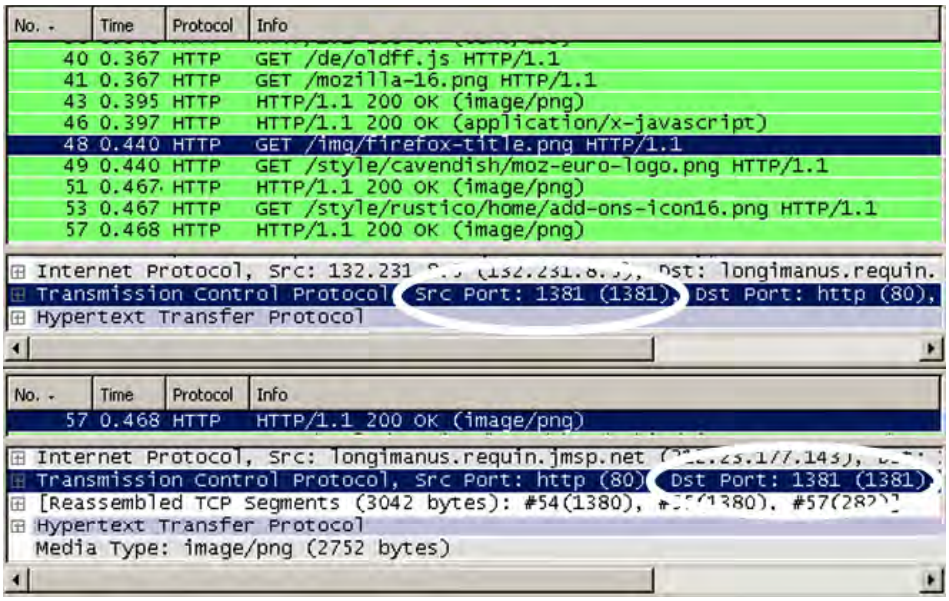


Abbildung 9: Ausschnitt aus einer Lauschaufnahme zum Aufruf der Mozilla-Homepage.
Die markierten Portnummern der Frage 48 und Antwort 57 stimmen überein.

Die Segmente haben übrigens wiederum einen Kopf, den ip-Kopf (mit den Absender- und Empfängeradressen). Segmente mit ip-Kopf heißen ip-Pakete. Mit Hilfe des Rekorders wird der tcp-Kopf eines tcp-Segments inspiziert. Er enthält Informationen, die uns weiterhelfen (Abb. 9).
Man kann sich die beobachtete Situation wie in Abb. 10 gezeigt vorstellen.

Abbildung 10: TCP stellt Kommunikationskanäle bereit, HTTP nutzt sie

Mit Hilfe der Filtermöglichkeiten des Rekorders lassen sich alle Nachrichten die „über den Kanal 1381“ bzw. alle Nachrichten die „über den Kanal 1382“ gelaufen sind, ansehen (Abb. 11). Betrachtet man die Kanäle einzeln ist eine Zuordnung von Anfrage und Antwort offensichtlich.
Wenn mein Browser verschiedene Kanäle zur Kommunikation mit dem Webserver verwendet, dann verwundert es, dass der Webserver scheinbar alle Anfragen über diese Portnummer 80 abwickelt. Enden die Kanäle tatsächlich beim Webserver alle in einem großen „Anschluss 80“. Nein, das ist nicht der Fall. Ähnlich wie der Browser verwendet auch der Webserver mehrere Kanäle. Die werden jedoch intern entsprechend der Absenderadresse und des Absenderports der einlaufenden ip-Pakete angeschlossen. Von außen können sie alle als „Anschluss 80“ erreicht werden.

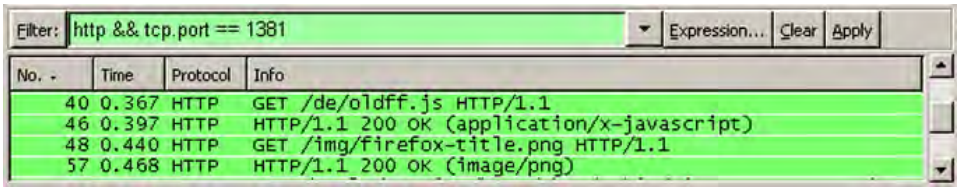


Abbildung 11: Kommunikation auf einem Kanal

5 Bemerkungen

Einige Grundzüge der hier skizzierten Experimente wurden von der Autorin und einem studentischen Mitarbeiter bereits auf regionalen Lehrerbildungsmaßnahmen diskutiert. Diese wurden im Rahmen eines Drittmittelprojekts aus BMBF-Mitteln mitfinanziert.
Bisher wurde der Rekorder in einer 9. Realschulklasse technischer Ausrichtung eingesetzt. Unser Eindruck war sehr positiv. Schülerinnen und Schüler fasziniert die Vorstellung, hineinschauen zu können in Datenpakete, die zum einen aus ihrem Rechner hinaus ins Netz laufen und zum anderen von entfernten Rechnern zu ihrem eigenen geschickt werden. Sie interessiert, was beim Surfen im Netz so alles an Daten ausgetauscht und an

Aktionen angestoßen wird. Die Handhabung des Rekorders stellte kein Problem dar. Kleine Client-Server-Interaktionsdiagramme haben sich als Diskussionsgrundlage bei auftretenden Fragen bewährt.
Es ist geplant, in diesem Jahr Schülerexperimente mit dem Rekorder für die Sekundarstufe I im Rahmen von zwei Staatsexamensarbeiten auszuwerten. Dazu kooperieren wir mit einer Realschule und einem Gymnasium. In diesem Rahmen ist auch eine Erweiterung der skizzierten Experimente auf weitere Protokolle bzw. Protokollhierarchien (ggf. HTTP/TCP/IP bzw. DNS/UDP/IP) und ein Vergleich dieser angedacht.

Literaturverzeichnis

[EW07] http://www.ethereal.com/ bzw. http://www.wireshark.org/

[Br04] Brichzin, P.; Freiberger, U.; Reinold, K.; Wiedemann, A.: Ikarus Natur und Technik Schwerpunkt: Informatik 6/7, Oldenbourg 2004

[Hu07] Hubwieser, P.; Spohrer, M.; Steinert, M.; Voß, S.: Informatik 2, Ernst Klett Verlag, Stuttgart 2007

[KR05] Kurose, J.; Ross, K.: computer networking, 3. Aufl.. Addison Wesley, 2005

[Ma07] http://www.die-maus.de/sachgeschichten/internet/

[Re06] http://www.theregister.co.uk/2006/03/30/ethereal_relationship_break-up/

[Re07] http://daten.rehbein.net/

[Si03] http://www.didaktik-der-informatik.de/simba

[SS04] Schubert, S.; Schwill, A.: Didaktik der Informatik. Spektrum Akademischer Verlag, Heidelberg 2004

[Ze07] http://www.zendas.de/service/browserdaten.html

Erfahrungen bei der Vermittlung algorithmischer Grundstrukturen im Informatikunterricht der Realschule mit einem Robotersystem

Bernhard Wiesner, Torsten Brinda

Didaktik der Informatik
Universität Erlangen-Nürnberg
Martensstr. 3
91058 Erlangen
{wiesner,brinda}@informatik.uni-erlangen.de

Abstract: Informatische Unterrichtsinhalte lassen sich erfolgreicher vermitteln, wenn Anwendungsbezüge zur Erfahrungswelt und zum Interessenfeld der Lernenden geschaffen werden. In diesem Zusammenhang richtet sich das Interesse auf einfache Robotersysteme als Unterrichtsmedium und als Unterrichtsgegenstand. Vorgestellt werden die Ergebnisse einer Fallstudie, die exemplarisch untersucht, inwieweit ein Robotersystem als Unterrichtsmedium die Vermittlung von Informatikinhalten der Sekundarstufe I unterstützen kann.

1 Motivation

Der Algorithmusbegriff gilt als fundamentale Idee der Informatik. Dementsprechend ist es ein Ziel des Informatikunterrichts, bei Schülerinnen und Schülern Verständnis für das Entwickeln von Algorithmen zur Lösung von Problemen bzw. Problemklassen zu wecken [SS04], [Hu04]. Dieses Vorhaben wurde in der Vergangenheit unter anderem über das Modellieren einfacher Abläufe und anschließendes Implementieren in einer Programmiersprache umgesetzt. Es zeigte sich dabei oft, dass das Erlernen der dazu erforderlichen Programmiersprache einen Großteil der zur Verfügung stehenden Zeit verschlang, ebenso wie die Einarbeitung in die notwendige Programmierumgebung. Darüber hinaus ging die Motivation der Schülerinnen und Schüler für das Entwerfen, Codieren und Fehlersuchen im Lauf der Zeit innerhalb der Klasse erheblich auseinander. Während auf der einen Seite Lernende mit den Programmierwerkzeugen motiviert arbeiteten und über den Unterricht hinaus kreativ mit der neuen Technik umgingen, kam eine große Gruppe über den rein reproduzierenden Umgang mit dem Thema kaum hinaus.

Eine Möglichkeit solche Schwierigkeiten zu vermeiden besteht darin, algorithmisierende Vorgehensweisen in andere Kontexte zu verlegen. Strukturierte Handlungsvorschriften werden in verschiedenen Unterrichtsfächern entwickelt, beispielsweise im Mathematikunterricht der Sekundarstufe I bei der Determinantenberechnung von linearen Gleichungssystemen (8. Jgst) oder im Informatikunterricht beim Umgang mit Tabellenkalku-

lationsprogrammen. Die Berechnungsvorschriften beschränken sich hierbei jedoch auf das Berechnen von Formelwerten. Ein Verständnis für die Strukturierung von Abläufen wird damit noch nicht erreicht.

Ein weiterer Ansatz verfolgt den Gedanken, statt Standard-Programmiersprachen „Mini-languages“ im Unterricht zu verwenden, die mit auf das Wesentliche beschränkter Syntax und Entwicklungsumgebung bereitgestellt werden. Damit erwartet man ein grundlegendes Verständnis der Schülerinnen und Schüler für die Entwicklung von Algorithmen und andererseits eine Verminderung der Misserfolge beim Implementieren. Konkret handelt es sich dabei um Umgebungen, die Variationen der Grundidee des programmierbaren, virtuellen Roboters „Karel“ [Pa81] darstellen, z. B. [Fr04]. Vorteile sind die einfache Inbetriebnahme des Systems und geringe bzw. keine Kosten, nachteilig die Tatsache, dass es sich dabei um abstrakte Simulationen mit reinem Übungscharakter handelt.

Für die Realschule, die bei der Auswahl der Unterrichtsmedien und -methoden grundsätzlich Praxisbezug und Handlungsorientierung anstrebt (z. B. [BUK01]), erscheinen Lösungen besonders wertvoll, bei denen reale Gegenstände als Medien den Lernprozess unterstützen. In diesem Zusammenhang sind Robotersysteme von besonderem Interesse, speziell solche, die nicht nur Eigenschaften und Verhaltensweisen realer technischer Geräte repräsentieren, sondern auch eine didaktisch gut aufbereitete Entwicklungsumgebung mitbringen.

2 Robotersysteme als Unterrichtsmedien

2.1 Überblick

Mit der Verwendung von Robotersystemen im schulischen Umfeld werden unterschiedliche Ziele verfolgt. Zum einen sollen sie dazu beitragen, Jugendlichen einen Zugang zur Technik in ganz allgemeinem Sinn zu verschaffen. Dieser Grundgedanke manifestiert sich in unterschiedlicher Ausprägung. Abhängig von Alter und Interessenlage werden Lösungen mit Baukastensystemen[1], mit festen Selbstbausätzen[2] bis hin zu kompletten Selbstbauten [PPA06] favorisiert. Dabei geht es außer dem Konstruieren eines technischen Geräts, unter Umständen auch mit viel Feinwerkarbeit, vor allem um das Ergründen der Möglichkeiten, die Roboter generell bieten können. Die Schülerinnen und Schüler sollen erkennen, wie Roboter mit Sensoren und Aktoren und einer programmierten Steuerung in Wechselwirkung zu ihrer Umwelt treten.

Eine wesentliche Erweiterung dieses Leitgedankens ist in den vielfältigen Formen von Roboterwettbewerben zu finden. Wettbewerbe fördern über den Umgang mit technischem Gerät hinaus zusätzliche Erziehungsziele wie Teamfähigkeit und Wettbewerbs-

[1] Z. B. LEGO Education – Mindstorms, http://www1.lego.com/eng/education/mindstorms/default.asp; Fischer Technik RoboPro, http://www.fischertechnik.de/de/computing/index.html (beide zuletzt geprüft am 26.01.2007).
[2] Z. B. DLR_School_Lab – Robotik: Experimente, http://www.dlr.de/schoollab/desktopdefault.aspx/tabid-1991/2841_read-4404/ (zuletzt geprüft am 26.01.2007); Conrad Electronic GmbH: Hauptkatalog 2006/2007, Hirschau; S. 1060.

verhalten [Kr05]. Sie sind allerdings keine spezifische Domäne der Robotik, sondern in vielen Disziplinen zu finden.

In der Hochschullehre werden Robotersysteme immer häufiger eingesetzt, um die Verflechtung von Technik und Informatik am konkreten Beispiel zu vermitteln und exemplarisch Projektarbeiten in fachübergreifenden Teams zu erstellen [KQ04, S. 3f] sowie um Studierenden Anwendungsfälle für das Modellieren und Implementieren in höheren Sprachen zu geben [Fa00]. Zum letztgenannten Zweck werden sie auch im Informatikunterricht der Sekundarstufe II verwendet. Motivation für den Einsatz von Robotersystemen in der Sekundarstufe I ist die Förderung naturwissenschaftlich-technischer Bildung, beispielsweise in Verbindung mit dem Projekt „Roberta“ (vgl. [Te05]), wobei meist Mädchen als Zielgruppe im Vordergrund stehen, und die Vermittlung informatischer Inhalte im Rahmen des regulären Informatikunterrichts (z. B. [DR01], [Mü02]).

2.2 Umsetzung didaktischer Prinzipien

Grundsätzlich unterstützt das Arbeiten mit Robotersystemen eine Reihe von didaktischen Prinzipien guten Unterrichts [Sa03, S. 3ff].

- Situiertheit: Die Lernenden gehen unmittelbar mit dem Roboter als Gegenstand der Aufgabe um und sind dadurch stets mit einem konkreten Fall befasst. Die hoch motivierende Wirkung wird mehrfach bestätigt. Insbesondere gelingt es damit Schülerinnen wie Schüler gleichermaßen anzusprechen [MRH00, S. 4], [Mü05, S. 147].

- Selbststeuerung: Die Modellierungs- und Implementierungsaufgaben lassen sich beim Arbeiten mit Robotersystemen oft so gestalten, dass die Lernenden ihre Produkte, ihre Vorgehensweise und ihr Arbeitstempo selbst bestimmen. Sie können aufgrund des System-Feedbacks ihren eigenen Arbeitsfortschritt feststellen und eignen sich das grundlegende Prinzip „Aus (Programmier-)Fehlern lernen“ an [MRH00, S. 4].

- Soziale Einbettung: Die Lernenden erarbeiten den überwiegenden Teil der Lerninhalte kooperativ in ihren Teams und tauschen während des Testens ihrer Entwürfe Erfahrungen mit anderen Teams aus. Die Lehrperson ist überwiegend beratend tätig.

- Vielfältige Anschlussmöglichkeiten: Der direkte Anwendungsbezug von Modellierung und Programmierung sowie der Umgang mit den Geräten sprechen verschiedene Sinne an und berücksichtigen vielfältige Lernertypen. Das problemorientierte Lernen ermöglicht die Entwicklung nützlicher Lernstrategien. Die Lernenden erwerben Wissen und wenden es gleichzeitig an.

Es ist unmittelbar ersichtlich, dass die genannten Vorteile nur bei einer soliden didaktisch-methodischen Unterrichtskonzeption zum Tragen kommen.

2.3 Robotersysteme für den Informatikunterricht

An Robotersysteme für den Informatikunterricht sind andere Anforderungen als beispielsweise an Geräte zu stellen, die in Wettbewerben erfolgreich sein sollen. Schnelligkeit und Präzision der Mechanik sind weniger entscheidend als eine gut strukturierte Programmiermethodik und eine Realisierung mit vertretbarem Zeit- und Organisationsaufwand. Gerade dieser Punkt stellt immer wieder ein Hemmnis für den Einsatz im Pflichtunterricht dar.

Aus didaktischen wie auch finanziellen Gründen ist es vorteilhaft, wenn das System in mehreren Jahrgangsstufen eingesetzt werden kann. Es sollte als Werkzeug den Schülerinnen und Schülern vertraut sein und im Idealfall eine Vielzahl von Lernzielen in unterschiedlichen Altersgruppen unterstützen.

Das Angebot an entsprechenden Systemen ist gegenwärtig klein. Für den Unterricht im Klassenverbund scheiden Geräte aus, zu deren Erstellung Lötarbeiten an kleinen Elektronikbauteilen notwendig sind. Die Bewältigung solcher Arbeiten ist nicht Unterrichtsziel der Sekundarstufe I. Ganz abgesehen davon wäre der organisatorische Aufwand nicht vertretbar. Andererseits müssen die Geräte von den Lernenden auf einfache Weise programmierbar sein. Damit kommen manche Fertigsysteme nicht in Betracht, die lediglich Robotereigenschaften demonstrieren oder nur fernsteuerbar sind.

Zu den Baukastensystemen zählen die Robotersysteme LEGO Mindstorms Robotic Invention System (RIS) und dessen Nachfolger LEGO Mindstorms NXT, die nach dem LEGO-Prinzip durch einfaches Zusammenstecken den Bau von Modellen mit Robotereigenschaften erlauben. Damit bleibt einerseits das Selbstbauprinzip mit seinen oben genannten didaktischen Vorteilen erhalten, andererseits bleiben den Lernenden mechanisch aufwändige Konstruktionen und Lötarbeiten erspart.

LEGO Mindstorms RIS lässt sich visuell mittels RCX-Code programmieren. Die grafischen Symbole der Programmierumgebung repräsentieren stets vollständige algorithmische Grundelemente wie etwa Wiederholung oder bedingte Anweisung. Entsprechend der gestellten Aufgabe ordnen die Lernenden die notwendigen Symbole auf der Oberfläche an und stellen gegebenenfalls erforderliche Parameter in geeigneter Weise ein, beispielsweise Motordrehrichtungen. Damit können Syntaxfehler, wie sie aus der Skriptprogrammierung bekannt sind, etwa fehlende Zeichen, nicht entstehen. Die Programme sind grundsätzlich lauffähig und die Lernenden können sich auf die semantischen Probleme ihrer Konstruktion konzentrieren. Ein zusätzlicher Vorteil der grafischen Programmierung ist die Ähnlichkeit der Programme mit Struktogrammen. Diese Eigenschaften lassen RIS für den Einsatz in der Sekundarstufe I als geeignet erscheinen.

Weiterführende Möglichkeiten der Programmierung bieten RoboLab[3], das auf LabVIEW basiert, und NQC (Not Quite C) [Ba03]. Nach entsprechender Änderung der Roboter-Firmware lässt sich das RIS-System auch in Java, FORTH, ADA, C oder LISP programmieren [Fa00, S. 148ff], [Ba03, S. 363ff], [Kl04].

[3] LEGO Education Mindstorms: LMfS Concept http://www.lego.com/eng/education/mindstorms/home.asp?pagename=lmfsc&l2id=4_4 (zuletzt geprüft am 28.01.2007)

2.4 Vermittelbare Informatik-Kompetenzen

Roboter sind Informatiksysteme, die dazu geeignet sind, um mit ihnen Problemlösekompetenzen im weitesten Sinn zu erwerben. Gegenüber reinen Softwareprodukten zeichnen sie die mechanisch-technischen Komponenten aus, die auf jugendliche Lernende attraktiv und motivierend wirken (vgl. [MRH00, S. 4]).

Programmierbare Robotersysteme sind dazu geeignet, die Prozesskette Analysieren, Modellieren, Implementieren und Reflektieren handlungsorientiert durchzuführen, wobei sich dies zielgruppenspezifisch variieren lässt: In der Sekundarstufe I beispielsweise analysieren Lernende Aktionen, welche die Roboter ausführen sollen, zerlegen sie in elementare Handlungsschritte und gelangen damit auf natürliche Weise zur Modellierung von Abläufen. Weiterhin konnte bereits exemplarisch gezeigt werden, wie sich objektorientiertes Modellieren [DR01], ereignisgesteuerte Abläufe und zustandsorientiertes Modellieren [RNH04] mit Robotersystemen als Medien verknüpfen lassen.

In Verbindung mit der grafischen Programmiersprache RCX-Code bzw. dem Nachfolger NXT Software erscheint das Mindstorms-System speziell zur Einführung des Algorithmusbegriffs geeignet. Die Lernenden analysieren die gegebene Aufgabe, die ihr Roboter bewältigen soll, entwickeln und beschreiben Lösungen, zerlegen den geplanten Ablauf in elementare Schritte, stellen ihn in Diagrammform dar und generieren Programme aus algorithmischen Grundbausteinen. Anschließend testen sie, korrigieren Fehler und beurteilen ihre Produkte.

Zusätzlich gewinnen die Schülerinnen und Schüler bei der Arbeit mit den Robotern eine Reihe weiterer Kompetenzen. Dazu zählen die Fähigkeit zum Entwurf von Systemen, Komponenten und Prozessen, die vorgegebene Eigenschaften besitzen, die Fähigkeit zur effektiver Kooperation und Kommunikation im Team sowie die Fähigkeit, mit neuartigen Geräten und Softwareanwendungen umzugehen (vgl. [KQ04, S. 4]).

2.5 Probleme und offene Fragen

Es ist denkbar, dass der angestrebte Lernerfolg bei der Verwendung von Robotersystemen durch Randbedingungen beeinträchtigt wird, die der Einsatz von Baukästen im Unterricht mit sich bringt. Hier ist vor allem der erhöhte Zeitbedarf zu nennen, den das Bauen der Roboter und das Einarbeiten in ihre Programmierung erfordern.

Aufgrund ihrer Konstruktion sind Roboter, die durch Zusammenstecken einzelner LEGO-Steine gebaut wurden, mechanisch anfällig. Es ist die Frage zu stellen, in welchem Maß das Erhalten eines verlässlichen Betriebszustands sowie das immer erforderliche Anpassen von Sensorwerten und Motordrehzeiten den Unterricht behindern [Ku04, S. 7], [FM02, S. 13].

Ferner ist ungewiss, ob sich die hohe Motivation, mit der Schülerinnen und Schüler das Thema „Roboter“ beginnen, über eine längere Unterrichtssequenz erhalten lässt [PMT03, S. 5] und inwieweit Roboter als Medien die Aufmerksamkeit der Lernenden dominieren und damit die eigentlichen Lernziele zu sehr in den Hintergrund drängen. Um zu diesen Fragen erste Antworten zu erhalten, wurde an einer bayerischen Realschule nachfolgende Fallstudie durchgeführt.

3 Fallstudie

3.1 Organisatorisches Konzept

Informatik ist an der Realschule in Bayern ein Wahlpflichtfach mit einem Stundenmaß von 2 Wochenstunden von der 8. bis zur 10. Jahrgangsstufe. Als Variante wird in einem seit 2001 dauernden Modellversuch das Wahlpflichtfach Informationstechnologie gelehrt, das je nach Wahlpflichtfächergruppe Inhalte aus Informatik, Textverarbeitung, Technischem Zeichnen (CAD) und Rechnungswesen enthält. Für den IT-Unterricht wird eine Klasse üblicherweise in zwei Gruppen geteilt. Die unterrichtete Gruppe bestand aus 4 Schülerinnen und 10 Schülern, die für die Unterrichtssequenz 7 Zweiergruppen zugeordnet wurden, da insgesamt 7 Arbeitsplätze mit je einem Rechner und je einem Mindstorms-Baukasten zur Verfügung standen.

Im Rahmen eines studienbegleitenden fachdidaktischen Praktikums wurde die Fallstudie in der Zeit von Oktober 2006 bis Januar 2007 an einer Realschule im Raum Erlangen-Nürnberg zusammen mit 3 Informatik-Lehramtsstudierenden durchgeführt. Die Unterrichtssequenz umfasste 9 Unterrichtsstunden (Doppelstunden) in einer 9. Jahrgangsstufe. Die Unterrichtsstunden wurden gemeinsam mit den Studierenden in einem Begleitseminar konzipiert und von jeweils einem Studierenden im wöchentlichen Wechsel gehalten. Die anderen Praktikumsteilnehmer, die Autoren und eine weitere Lehrkraft beobachteten den Unterricht als Hospitierende. Bei der Evaluation im Anschluss an den jeweiligen Unterricht wurden Verbesserungsvorschläge entwickelt, die im nachfolgenden Erfahrungsbericht eingearbeitet sind.

Zusätzlich zur Beobachtung und Dokumentation des Unterrichtsgeschehens wurden die Leistungen der Schülerinnen und Schüler erfasst. Zu Beginn der dritten Unterrichtseinheit fand eine schriftliche Lernerfolgskontrolle im Umfang von 10 Minuten statt. Am Ende der Reihe bearbeiteten die Lernenden schriftliche Aufgaben im Rahmen eines Übungszirkels. Die Beobachtungsergebnisse sowie die Ergebnisse der Lernerfolgskontrollen wurden für die Auswertung verwendet.

3.2 Inhaltliche und methodische Konzeption

Die Ziele der Unterrichtssequenz orientierten sich am Lehrplan. Die Schülerinnen und Schüler sollten wissen, was Roboter sind und verstehen, dass ihr Verhalten von einem Programm gesteuert wird und dass Sensoren den Programmablauf beeinflussen. Sie sollten die algorithmischen Strukturen Sequenz, Wiederholung, Verzweigung kennen und im Zusammenhang mit dem benutzten Roboterfahrzeug zur Lösung von Aufgaben verwenden können. Ferner sollten sie den Algorithmusbegriff als genau definierte Handlungsvorschrift zur Lösung eines Problems oder einer bestimmten Art von Problemen kennen und einfache Aktionen in Struktogrammform beschreiben können.

Die Inhalte und ihre Verteilung auf die Unterrichtseinheiten (UE) werden in Tabelle 1 stichwortartig wiedergegeben. Eine vollständige Dokumentation der Reihe ist unter http://ddi.informatik.uni-erlangen.de/Lehre/WS200607/ddifapra/material/ zu finden.

1	Einsatzbereiche und Eigenschaften von Robotern; Definition des Roboterbegriffs; Präsentation von LEGO-Robotern; Bau des Roboterfahrzeugs; Ausführung eines vorgegebenen Programms und Anfertigung einer genauen Beschreibung des Bewegungsablaufs
2	Simulation eines Roboters in Rollenspielform zur Einführung der Begriffe „Anweisung“ und „Befehlssatz“; Definition des Begriffs „Sequenz“; Einführung in die Programmierung mit RCX-Code; Erstellung eines Programms mit rein sequentieller Struktur; Einführung der Wiederholung als verkürzendes Verfahren
3	Simulation eines Roboters in Rollenspielform zur Einführung der bedingten Anweisung; Erweiterung des Rollenspiels in mehreren Varianten; Einbau eines Lichtsensors in das Roboterfahrzeug; Modellierung und Programmierung der Aufgabenvarianten aus dem Rollenspiel; Einführung des Begriffs „bedingte Anweisung“; Erweiterung der Aufgabe um ein Lenkmanöver beim Unterschreiten eines bestimmten Lichtwerts; Einführung des Begriffs „Verzweigung“
4	Aufgabe, den Roboter so lange jeweils 1 Sekunde geradeaus fahren zulassen, wie der Messwert des Lichtsensors einen hellen Untergrund meldet; Erklärung der Verknüpfung von Sensormesswerten mit der Wiederholungsanweisung in RCX-Code; Erklärung der „bedingten Wiederholung“; Vergleich dieses Verfahrens mit der ähnlich lautenden Aufgabe aus der Vorstunde
5	„Übersetzung“ des RCX-Programms aus der Vorstunde in eine umgangssprachliche Beschreibung; Erzeugung eines Struktogramms zum Programm aus der Vorstunde; Definition des Begriffs Struktogramm und seiner Elemente; Erzeugung eines Struktogramms zu einer umgangssprachlichen Beschreibung einer Folge von Roboterbewegungen; Realisierung mittels RCX-Code
6, 7	Wiederholung und Festigung der Lernziele in Form eines Übungszirkels
8	Aufgabenstellung zum Erreichen eines Labyrinth-Ausgangs; Diskussion mehrerer Lösungsverfahren; vergleichende Abschätzung des Zeitaufwands; Definition des Algorithmusbegriffs; Analyse und Detailplanung eines Lösungsverfahrens; Programmierung und Test; Demonstration der Lösungen mit Wettbewerbscharakter
9	Erstellung einer schriftlichen Zusammenfassung der Unterrichtssequenz in Form von bearbeiteten, ausführlich kommentierten Aufgaben durch die Schüler

Tabelle 1: Unterrichtseinheiten und Lerninhalte

Das Bauen und Programmieren eines Roboters wird häufig, wie auch im Lehrplan vorgeschlagen, als Projektarbeit im Informatikunterricht durchgeführt. Dabei planen, bauen und programmieren die einzelnen Schülerteams ihre Geräte, wobei ihnen ein verhältnismäßig großer gestalterischer Freiraum gewährt wird. Die Lernenden arbeiten über einen längeren Zeitraum selbstständig, dokumentieren ihre Arbeit und präsentieren am Ende ihre individuellen Ergebnisse. Da in dieser Unterrichtssequenz jedoch bestimmte, fest umrissene Inhalte vermittelt werden sollten, wurden die Phasen der Gruppenarbeit kürzer gefasst und in andere Unterrichtsformen, wie Lehrerpräsentation und Unterrichtsgespräch, eingebettet.

Die Vermittlung neuer Lerninhalte erfolgte, indem den Lernenden zunächst Aufgaben gestellt wurden, zu deren Lösung neue Kenntnisse oder neue Fähigkeiten erforderlich waren. Im Plenum oder in den Teams wurden dann Lösungsansätze gesucht und diskutiert. Anschließend machten sich die Schülerinnen und Schüler im Team an die Bewältigung der Aufgaben. Die Lehrperson leistete bei Gruppen, die längere Zeit vergeblich an der Lösung arbeiteten, Hilfestellung. Im Anschluss daran wurden jeweils die neuen Erkenntnisse schriftlich festgehalten. Diese Methode erlaubte den Lernenden, in gewissem Umfang selbst aktiv neue Inhalte zu erarbeiten (vgl. auch [HNR06, S. 92]).

Zur Festigung der Lerninhalte wurde ein Übungszirkel gestaltet (Stunde 6 und 7). Die Schüler hatten insgesamt sechs Aufgaben, die sich jeweils auf unterschiedliche Themen der Unterrichtssequenz bezogen, sowohl schriftlich als auch am Rechner zu bearbeiten (siehe http://ddi.informatik.uni-erlangen.de/Lehre/WS200607/ddifapra/material/). Dieses Verfahren diente der Förderung des didaktischen Prinzips der Selbststeuerung, was die Reihenfolge, das Arbeitstempo und auch in gewissem Umfang die Auswahl der Aufgaben anging.

4 Ergebnisse und Erfahrungen

4.1 Formales Beschreiben von Abläufen, Ablaufdiagramme

In der ersten Stunde wurde von den Schülerinnen und Schülern die detaillierte Beschreibung einer vom Roboter ausgeführten Aktion erwartet. Die Bewegungen des Roboterfahrzeugs waren klar voneinander zu unterscheiden und wurden meist streng sequentiell ausgeführt. Der Detaillierungsgrad der Beschreibungen fiel allerdings sehr unterschiedlich aus. Eine Drehbewegung des Fahrzeugs wurde beispielsweise so beschrieben: „Die linken Räder drehen sich kurz rückwärts und die rechten Räder drehen sich kurz vorwärts“, bis hin zu „Der Roboter fährt nach links.“ Dies trat nicht mehr auf, sobald die Lernenden die Programmierumgebung und die Symbole des RCX-Codes kennen gelernt hatten, da dann der Detaillierungsgrad der Anweisungssymbole übernommen wurde.

Um deutlich zu machen, dass ein Roboter nur eine beschränkte und fest definierte Menge an Methoden hat, erwies sich die Simulation in Form eines Rollenspiels als sehr hilfreich: Einem Schüler wurde die Rolle eines Roboterfahrzeugs zugewiesen. Ein zweiter Schüler bekam die Rolle des Anweisungsgebers. Er musste unter Beschränkung auf die zur Verfügung stehenden Methoden, die „Befehlssatz des Roboters“ genannt wurden, den Roboter leiten. Die gestellte Aufgabe lautete etwa: Der Roboter steht zwischen den Tischen des Klassenzimmers und soll bis zur Tür gehen. Der Befehlssatz ist beschränkt auf „Gehe einen Schritt vorwärts“, „Drehe dich um 90 Grad nach rechts“, „Drehe dich um 90 Grad nach links“. Der Spielverlauf zeigte, dass sich sowohl Anweisungsgeber als auch Roboter nicht immer an den vorgegebenen Befehlssatz hielten. Dem wurde entgegengesteuert, indem ein dritter Schüler die Einhaltung der Formulierungen überwachte. Er war mit einer Hupe ausgestattet und intervenierte, sobald eine nicht vorhandene Methode angefordert wurde.

Die Darstellung von Abläufen mit Diagrammen wurde erst behandelt, als die Lernenden einen ausreichenden Überblick über die algorithmischen Grundelemente und die entsprechenden RCX-Icons gewonnen hatten. Die verwendete Programmierumgebung erwies sich an dieser Stelle als vorteilhaft, da die grafische Programmdarstellung große Ähnlichkeit mit Struktogrammen aufwies. Daraus folgte zum Ersten die Entscheidung für die Verwendung von Struktogrammen als formale Darstellung für die Aufgaben der Roboter, zum Zweiten die Nutzung der Programm-Analyse als einführende Methode. Auf die Modellierung mit Hilfe von Aktivitätsdiagrammen wurde verzichtet, da die Transferanforderungen beim Übergang vom Diagramm zum Programmcode wesentlich größer sind.

Unter Hinweis auf eine produktunabhängige grafische Generalisierung erfolgte der Übergang von der grafischen Darstellung im RCX-Code zu einer ersten Version des Struktogramms. Einfache Anweisungen und bedingte Anweisungen (Verzweigungen) bereiteten dabei keinerlei Schwierigkeiten, da ihre Darstellungen fast direkt aus RCX in ein Struktogramm übertragen werden konnten. Wiederholungen dagegen erforderten eine gewisse Umgestaltung, die mit dem Hinweis auf Darstellungsnormen (DIN 66261) eingeführt wurde.

4.2 Anwendung der algorithmischen Grundelemente

Die erste Programmieraufgabe für das Roboterfahrzeug und Hinführung zum Begriff der Sequenz war das Umfahren eines Quadrats von ca. 30 cm Kantenlänge. Nach der Vorbereitung durch das Rollenspiel (vgl. Tab. 1, UE 2) gelang den Schülerinnen und Schülern die Zerlegung der Aufgabe in die einzelnen Anweisungen ohne Probleme. Die Aufgabe bereitete auch gleichzeitig den nächsten Lerninhalt vor, die Wiederholung als algorithmisches Grundelement. Hierzu wurde das bisher erstellte Programm untersucht und festgestellt, dass sich einzelne Teile der Sequenz, nämlich die Geradeausfahrt und anschließende Drehung genau viermal wiederholen. Nachdem die Wiederholungsanweisung im RCX-Code identifiziert war, wurde die Aufgabe damit neu gelöst. Der daraus resultierende Vorteil war für die Schüler unmittelbar einsichtig: Eventuell nötige Änderungen an den Fahranweisungen oder Drehungen müssen bei Anwendung der Wiederholungsstruktur nur einmal für alle vier Quadratseiten vorgenommen werden.

Nach der Erklärung des Symbols für die bedingte Anweisung programmierten die Teams die Abläufe des Rollenspiels selbstständig im RCX-Code (vgl. Tab. 1, UE 3). Es erwies sich als wichtig, die Abfolge der gespielten Varianten genau einzuhalten. Teams, die mit den komplexen Varianten begonnen hatten, verloren den Zusammenhang zwischen dem Ablauf der Aufgabe und der hieraus folgenden notwendigen Anordnung der Programmelemente.

Bei der Einführung der bedingten Wiederholung (vgl. Tab. 1, UE 4) wurde auf die Unterscheidung zwischen Wiederholung mit Anfangs- und Endebedingung verzichtet. In Musterlösungen wurde ausschließlich die Wiederholung mit Anfangsbedingung verwendet.

4.3 Hard- und Software

Das Bauen nach einer genauen Anleitung erwies sich als vorteilhaft, da damit die Funktionssicherheit erreicht wurde, die im Unterrichtsbetrieb erforderlich ist (vgl. [Ku04, S. 12]). Da die Produkte alle den gleichen Fahrzeugaufbau besaßen, mussten gemeinsam entwickelte Programmkomponenten nicht an individuelle Konstruktionen angepasst werden.

Zum Bau der Fahrzeuge wurden an die Gruppen genau die Bauteile herausgegeben, die für die Konstruktion nötig waren. Das vorherige Abzählen der Bauteile bedeutete zwar einen erhöhten Vorbereitungsaufwand, führte aber zu einer geordneten Konstruktionsphase, da die Schülerinnen und Schüler die benötigten Bauteile nicht aus einer großen Menge heraussuchen mussten und keine Bauteile verloren gingen. Die Fahrzeuge blieben den einzelnen Teams während der gesamten Unterrichtssequenz zugeordnet.

Die Programmierumgebung bietet mehrere Symbole, die recht ähnlich aussehen und daher für den Anfänger in ihrer Funktionalität schlecht unterscheidbar sind. So gibt es beispielsweise das Symbol „Motor(en) ein“ und das Symbol „Motor(en) ein für x Sekunden“. Im zweiten Fall wird die Programmausführung für die Zeit x angehalten, im ersten Fall nicht, was zu unerwarteten Reaktionen führt. Hier wäre es vorteilhaft, die zur Verfügung stehenden Symbole per Voreinstellung einschränken zu können.

Es zeigte sich, dass in der Basisausstattung des RIS-Systems aufgrund fehlender Rotationssensoren nur Zeitdauern und Motorleistungen für den Betrieb der Motoren angegeben werden können, nicht etwa Umdrehungszahlen. Damit wurde das Programmieren von Drehungen um feste Winkel zu einem Geduldsspiel, da vor allem die Reibung der Räder auf dem Boden schlecht kalkulierbar war. Dieser Nachteil ist in der Nachfolgeversion Mindstorms NXT behoben.

Mit der Verwendung paralleler Programmstränge lassen sich im verwendeten System Reaktionen auf Sensormeldungen eleganter als bisher beschrieben realisieren. Dabei wird eine Bedingung zu einem Sensorwert formuliert, bei deren Eintreten das Programm im Parallelstrang (z. B. Lenkmanöver) Priorität vor dem Programm im Hauptstrang (z. B. Geradeausfahrt) erhält. Anschließend kehrt die Ausführung wieder zum Hauptprogramm zurück. Da sich die beschriebene Unterrichtssequenz aber ausschließlich mit der Behandlung grundlegender algorithmischer Strukturen befasste, wurde diese Methode nicht verwendet. Auch hier würde es die Unterrichtspraxis erleichtern, wenn sich die entsprechenden Symbole, wie schon oben geschildert, ausblenden ließen.

4.4 Unterrichtsmethoden

Die Simulation von Handlungsabläufen in Form von Rollenspielen hat sich als ausgesprochen hilfreich erwiesen. Das schrittweise „Durchspielen“ der Algorithmen brachte das notwendige Verständnis für die genaue Abfolge der Anweisungen und deckte dabei Verständnisschwierigkeiten der Lernenden auf, die der Lehrperson beim Formulieren der Aufgaben zum Teil nicht bewusst waren.

Erfahrungsgemäß besteht bei Gruppenarbeit, besonders bei größeren Gruppen, die Gefahr, dass sich Einzelne aus der gemeinsamen Bearbeitung „ausklinken“. Dies war im

vorliegenden Fall nicht zu beobachten. Die Schülerinnen und Schüler arbeiteten in Zweierteams. Zudem wechselte die Unterrichtsform innerhalb einer Stunde mehrfach zwischen Gruppenarbeit und Unterrichtsgespräch, in das alle Schülerinnen und Schüler gleichermaßen eingebunden wurden.

Der gegen Ende durchgeführte Übungszirkel erwies sich als wirksame Methode zur Festigung aller Lerninhalte. Die Teams arbeiteten selbstständig und mit wachsendem Engagement am Aufgabenmaterial. Die Gruppenarbeit ermöglichte es der Lehrperson, während des Unterrichts die Teams bei Fragen zu beraten und ihre Lösungen zu beurteilen.

5 Fazit und Ausblick

Mit der Fallstudie konnte am Thema „Algorithmische Grundstrukturen“ exemplarisch gezeigt werden, wie die Verwendung von Robotersystemen die Vermittlung informatischer Inhalte im Unterricht der Realschule unterstützt. Mit dem Hinzufügen dieses „begreifbaren“ Unterrichtsmediums werden unterschiedliche Lerntypen gefördert und ein Leitgedanke der Realschule umgesetzt. Gleichzeitig wird ein Praxisbezug für die zu vermittelnden informatischen Sachverhalte geschaffen, der die Auseinandersetzung der Lernenden mit der Thematik deutlich intensiviert. Das verwendete System zeichnet sich zusätzlich dadurch aus, dass es die Entwicklung des Modellierungsprozesses, im dargestellten Fall die Modellierung von Abläufen, sehr einfach macht. Es konnte gezeigt werden, dass sich Bau und Verwendung der Roboter durch das in Kap. 4.3 dargestellte Verfahren problemlos in den Unterrichtsbetrieb einfügten. Die in Kap. 2.5 genannten mechanischen Schwächen traten auch hier zutage, wurden aber von den Lernenden nicht als hinderlich empfunden. Das Interesse der Schülerinnen und Schüler am Thema wurde von den Hospitierenden über den gesamten Zeitraum als sehr hoch beurteilt. Ein Nachlassen war nicht zu beobachten.

Die Ergebnisse der Lernerfolgskontrollen zeigten ein Bild, das den sonstigen Leistungen der Klassengruppe entsprach. Der Mehrwert des geschilderten Verfahrens ist zum großen Teil darin zu sehen, dass sich auch Schülerinnen und Schüler zur Mitarbeit motivieren ließen, die dem Fach Informatik nach Aussagen ihres Informatiklehrers bisher eher ablehnend gegenüberstanden. Im Übungszirkel zeigte sich, dass die Lernenden auch einfache Algorithmen darstellen konnten, die nichts mit den Robotersystemen zu tun hatten, mithin also ein weiterführendes Verständnis für die Thematik entwickelten, das Transferleistungen ermöglicht.

Davon ausgehend erscheint es sinnvoll, die Erprobung des Konzepts weiterzuführen und auszudehnen auf die in Kap. 2.4 genannten Themenbereiche. Voraussetzung für ein erfolgreiches Arbeiten ist sicher eine didaktisch wie technisch ausgereifte Programmierumgebung. Es wäre allerdings verkehrt anzunehmen, dass allein das Benutzen des Robotersystems bereits den Lernerfolg herbeiführt. Erforderlich sind ein klar strukturiertes inhaltliches Konzept und ein methodisch gut durchdachter Unterrichtsablauf, in den die Roboterpraxis eingebunden ist.

Literaturverzeichnis

[Ba03] Baum, D.: Definitive guide to LEGO MINDSTORMS. Apress, Berkeley Calif., 2003.

[BUK01] Bayerisches Staatsministeriums für Unterricht und Kultus (Hrsg.): Lehrplan für die sechsstufige Realschule in Bayern, Maiß, München, 2001.

[DR01] Dietzel, R.; Rinkens, T.: Eine Einführung in die Objektorientierung mit Lego Mindstorms Robotern. In: Keil-Slawik, R.; Magenheim, J. (Hrsg.): Informatikunterricht und Medienbildung, INFOS 2001. Köllen, Bonn, 2001; S. 193–199.

[Fa00] Fagin, B. S.: Using Ada-based robotics to teach computer science. In: Proceedings of the 5th Annual Conference on Innovation and Technology in Computer Science Education. ACM Press, New York, 2000; pp. 148-151.

[FM02] Fagin, B. S.; Merkle, L. D.: Quantitative Analysis of the Effects of Robots on Introductory Computer Science Education. In: JERIC 2 (2002) 4; pp. 1-18.

[Fr04] Freiberger, U.: Robot Karol, 2004, URL: http://www.schule.bayern.de/karol/ (zuletzt geprüft am 22.01.2007).

[HNR06] Hartmann, W.; Näf, M.; Reichert, R.: Informatikunterricht planen und durchführen. Springer, Berlin, 2006.

[Hu04] Hubwieser, P.: Didaktik der Informatik. Berlin, Springer, 2004.

[KQ04] Kitts, C.; Quinn, N.: An interdisciplinary field robotics program for undergraduate computer science and engineering education. In: JERIC 4 (2004) 2; Art. 3.

[Kl04] Klassner, F.: Enhancing lisp instruction with RCXLisp and robotics. In: Proceedings of the thirty-fifth Technical Symposium on Computer Science Education. ACM Press, New York, 2004; pp. 214–218.

[Kr05] Kraetzschmar, G. K.: Motivation for Free - Zum Motivationsfaktor von internationalen RoboCupJunior-Wettbewerben. In: Cremers, A. B.; Manthey, R.; Martini, P.; Steinhage, V. (Hrsg.): INFORMATIK 2005 - Informatik LIVE! Band 1. Köllen, Bonn, 2005; S. 148–152.

[Ku04] Kumar, A. N.: Three Years of Using Robots in an Artificial Intelligence Course - Lessons Learned. In: JERIC 4 (2004) 3; Art. 1.

[MRH00] Magenheim, J.; Reinsch, T.; Hirsch, M.: Zugänge zur Informatik mit Mindstorms. In: LOG IN 20 (2000) 2; S. 34–46.

[Mü02] Müller, W.: Algorithmik mit dem LEGO-Roboter, 2002, URL: http://www.lehrer-online.de/url/lego-roboter (zuletzt geprüft am 24.01.2007).

[Mü05] Müllerburg, M.; Börding, J.; Theidig, G.; Petersen, U.: Informatikausbildung, Roboter und Mädchen. In: Cremers, A. B.; Manthey, R.; Martini, P.; Steinhage, V. (Hrsg.): INFORMATIK 2005 - Informatik LIVE! Band 1. Köllen, Bonn, 2005; S. 143–147.

[Pa81] Pattis, R. E.: Karel the Robot - A Gentle Introduction to the Art of Programming. 1st Edition, Wiley, New York, 1981.

[PMT03] Petersen, U.; Müllerburg, M.; Theidig, G.: Girls and Robots - A Promising Alliance. URL: http://alex.ais.fraunhofer.de/zeno/web?action=content&journal=16919&rootid=15465 (zuletzt geprüft am 28.01.2007)

[PPA06] Professur für Prozessautomatisierung der TU Chemnitz (Hrsg.): Roboking 2007 Wettbewerbsdokumentation, 2006, URL: http://www.tu-chemnitz.de/etit/proaut/rk/fileadmin/user_upload/downloads/RoboKing2007_Wettbewerbsdoku.pdf (zuletzt geprüft am 22.01.2007).

[RNH04] Reichert, R.; Nievergelt, J.; Hartmann, W.: Programmieren mit Kara. Ein spielerischer Zugang zur Informatik. Springer, Berlin, 2004.

[Sa03] Sacher, W.: Neue Medien - neuer Unterricht?. Schulpädagogische Untersuchungen Bd. 19, Nürnberg, 2003.

[SS04] Schubert, S.; Schwill, S.: Didaktik der Informatik. Spektrum, Berlin, 2004.

[Te05] Tempelhoff, A.: Robotik in der Sekundarstufe I. In: LOG IN 25 (2005) 134; S. 23–29.

Informatik – Mensch – Gesellschaft im Schulunterricht

Jochen Koubek, Constanze Kurz

Humboldt-Universität zu Berlin
Institut für Informatik
Informatik in Bildung und Gesellschaft
Unter den Linden 6
10099 Berlin
jochen.koubek@hu-berlin.de
kurz@informatik.hu-berlin.de

Abstract: Ein selbstbestimmter, verantwortungsvoller und sicherer Umgang mit Informatiksystemen bedingt neben technischem Sachverstand auch Kenntnisse um gesellschaftlichen Wechselwirkungen dieser Techniken. Zu diesem Zweck wurde im Schuljahr 2005/2006 von den Autoren eine Schüler-Arbeitsgruppe „Computer – Mensch – Gesellschaft" durchgeführt und die Ergebnisse in Form von Unterrichtsentwürfen dokumentiert.

1 Einleitung: Die AG „Computer – Mensch – Gesellschaft"

Seit der Gründung der informatischen Fakultäten, Institute oder Arbeitsgruppen an Deutschen Universitäten in den frühen 70er Jahren gehört die Reflexion gesellschaftlicher Wechselwirkungen der Informationstechnologien zur akademischen Ausbildung irgendwie dazu. Dazu gehört sie, weil es in den stark politisierten Nach-68ern undenkbar erschien, ein neues Fach ohne gesellschaftspolitische Sekundanten einzurichten. Was in den etablierten Gründerfächern Mathematik und Elektrotechnik nicht mehr möglich war, sollte zumindest ihrem gemeinsamen Kind nicht fehlen. Dass die Dazugehörigkeit der gesellschaftlichen Dimensionen lange Zeit ein „irgendwie" war, zeigt sich sowohl an der reservierten Einstellung vieler Studierenden zu diesen „weichen" Themen, als auch an der Stellenpolitik der meisten Hochschulen. Erfolgreiche Spin-Offs wie Fragen der Sicherheit von Computersystemen oder der Gestaltung von Mensch-Maschine-Schnittstellen haben sich allerdings als eigenständige Forschungszweige etablieren können. Zurück blieben Klassiker wie „Computer-Ethik" und „Datenschutz" oder „Geschichte der Informatik".

Seit dem allgemein sichtbaren Übergang der Industrie- zur Informationsgesellschaft haben seit Mitte der 90er Jahre neben diesen Evergreens zahllose große und kleine Themen den gesellschaftskulturellen Horizont der Informatik erweitert. Zu nennen seien hier nur die digitale Wissensordnung zwischen Allgemeingut und geistigen Eigentumsansprüchen, die ökologisch nachhaltige Entwicklung von Computersystemen oder der Einsatz des Computers als Medium. Sie bergen Fragen von erheblicher gesellschaftlicher

Bedeutung, und nicht zuletzt zeigen sich an den Aushandlungen um Urheber- und Patentrecht, um Energiepolitik und Datenschutz die Demarkationslinien einer globalisierten Gesellschaft, die sich hier in irgendeiner Form positionieren muss.

Ein selbstbestimmter, verantwortungsvoller und sicherer Umgang mit Informatiksystemen bedingt daher neben technischem Sachverstand auch Kenntnisse um gesellschaftlichen Verknüpfungen und Wechselwirkungen dieser Techniken [Ko04]. Konsequenterweise sind diese Kenntnisse eine der tragenden Säulen informatischer Bildung im Schulunterricht [GI00]. Dennoch ist die Situation an den Schulen ähnlich zu denen der Hochschulen: Während niemand im offenen Diskurs die Bedeutung der gesellschaftlichen Dimensionen anzweifelt, werden sie in der Ausbildung allzu oft vernachlässigt. Solange IMG-Themen nicht verpflichtend in den Lehrplänen stehen, ist es der Entscheidung der Lehrer überlassen, sie in den Unterricht zu integrieren, weswegen vielerorts darauf verzichtet wird. Auf Rückfragen werden von Lehrern für diesen Umstand regelmäßig fünf Argumente angeführt:

1. Es fehlt an Methodenkenntnis auf Lehrerseite. Wer nicht weiß, in welcher Form, in Hinblick auf welche Kompetenzen und mit welchen Medien ethische Fragen in der Klasse angesprochen werden, kann nicht über Raubkopien, Netikette oder Hacking sprechen.
2. Es fehlt an Sachkenntnis auf Lehrerseite. Wer glaubt, dass jede Privatkopie eine Raubkopie ist, kann nicht zu einem mündigen Umgang mit digitalen Medien erziehen.
3. Es fehlt an Unterrichtsmaterial. Die Grundlagen müssen geeignet didaktisch aufbereitet werden, um sie den Vorkenntnissen und Voraussetzungen der Schüler entsprechend anzupassen.
4. Es fehlt an Verknüpfungen. Die vereinzelt auffindbaren Unterrichtsentwürfe müssen in größere Unterrichtseinheiten mit technischem Bezug eingebaut werden, damit sie nicht als fakultative Angebote an den Jahresrand gedrängt werden, bzw. dort hinunter fallen.
5. Dafür sind andere Fächer zuständig, vor allem der PW- oder der Sozialkundeunterricht. Wer so spricht, weiß allerdings nicht, was diese Fächer leisten können und wollen und was nicht.

Um zumindest einige dieser Mängel zu lindern, planen die Autoren ein mehrstufiges Programm, dessen erste Schritte bereits erfolgreich umgesetzt wurden. Langfristig soll es Gegengewichte zu den Argumenten 1-4 liefern, indem Materialien zur Sach- und Methodenschulung in ausreichender Menge, modularem Aufbau, vertikaler Differenzierung sowie in verknüpfbarer Form geschaffen und bereitgestellt wird.

- „Ausreichende Menge" bedeutet, dass möglichst viele Dimensionen berücksichtigt werden, um ein Reservoir anbieten zu können, aus dem je nach Bedarf geschöpft werden kann.

- „Modularer Aufbau" bedeutet, dass die verschiedenen Themen nicht voneinander abhängen, sondern unabhängig voneinander bearbeitet werden können.
- „Vertikale Differenzierung" bezieht sich auf die Einsatzmöglichkeit im gesamten Bildungssystem in jeder Kompetenzstufe.
- Die „verknüpfbare Form" meint schließlich die Möglichkeit, die Unterrichtsmaterialien in technische Einheiten eingliedern zu können, um die Vieldimensionalität informatischer Bildung zu betonen.

Die genannten Anforderungen definieren ein Forschungsprogramm für mehrere Jahre. Die einzelnen Projektabschnitte erfolgen nicht notwendig in der genannten Reihenfolge, wichtiger erschien es zunächst, Substanz zu schaffen, die im Folgenden verarbeitet und aufbereitet werden soll.

Zu diesem Zweck wurde im Schuljahr 2005/2006 von den Autoren eine Schüler-Arbeitsgruppe „Computer – Mensch – Gesellschaft" geplant, ausgerichtet und durchgeführt. Die Ziele der AG waren auf zwei Ebenen angesiedelt, einerseits auf der Unterrichts-, andererseits auf der Projektebene.

- Die Unterrichtsziele orientieren sich an den IMG-Kompetenzen der zurzeit entwickelten Leitlinien für informatische Bildung, wie sie im Rahmen dieser INFOS vorgestellt werden sollen. Aus diesem Grund soll an dieser Stelle nicht weiter darauf eingegangen und auf spätere Veröffentlichungen verwiesen werden.
- Auf Projektseite sollte einerseits ein Proof-of-Concept erreicht werden, um zu zeigen, dass eine didaktische Reduktion der wichtigsten gesellschaftlichen Dimensionen möglich ist und zu durchführbarem Unterricht führt. Damit wurde ausreichend Material geschaffen, das im weiteren Projektverlauf zu den erwähnten verknüpfbaren Modulen führen soll. Mit dem angesammelten Material ist ein einjähriger Informatik-Kurs mit zwei Stunden pro Woche ausschließlich mit Inhalten im Bereich IMG möglich. Das ist mehr als diesen Themen üblicherweise in einem Leistungskurs in der gesamten Oberstufe zugestanden werden.

Bevor die Unterrichtseinheiten vorgestellt werden, sei auf die Rahmenbedingungen der AG verwiesen, die bei den weiteren Überlegungen nicht außer Acht gelassen werden dürfen.

Die AG wurde zu Beginn des Schuljahres 2005/2006 an Berliner Schulen ausgeschrieben mit dem Hinweis:

> „Informatik ist nach unserem Verständnis ein disziplinübergreifendes Fach, das insbesondere gesellschaftliche, kulturelle und technische Dimensionen umfasst. Die Auswahl der AG-Themen richten sich danach, wie viel Raum sie in diesen Dimensionen einnehmen können. Geplant sind Einheiten zu den Themen Geschichte der Rechentechnik, Computer & Ethik, Informationsrecht, Datenschutzrecht und Informationelle Selbstbestimmung, Urheberrecht und Geistiges Eigentum, P2P-Netze, Kryptographie, Digitale Ökonomie, Computer und Umwelt und kulturelle Produktionen. Die Themen werden über ihre gesellschaftlich-kulturelle

Bedeutung motiviert und diese Kontexte werden gleichrangig zu den technischen Inhalten behandelt. Technische Kompetenzen werden dabei vermittelt, sie stehen aber nicht alleine im Mittelpunkt. Die AG orientiert sich am Vertiefungsgebiet Informatik & Gesellschaft des Berliner Rahmenlehrplans Informatik. Darüber hinaus wird ein besonderes Augenmerk auf die Methodik des wissenschaftlichen Arbeitens sowie auf Techniken der Wissenspräsentation und -darstellung gelegt."

Insgesamt haben 10 Schülerinnen und Schüler aus gymnasialen Oberstufen regelmäßig an der AG teilgenommen. Wie bei einem freiwilligen Kurs nicht anders zu erwarten, handelte es sich bei den Teilnehmern keineswegs um durchschnittliche (Informatik-) Schüler, sondern vielmehr um engagierte und motivierte Jugendliche, die ganz bewusst das ausgeschriebene Angebot wahrgenommen haben, die nicht-technische Seite der Informatik näher kennen zu lernen. Insofern können die sehr positiven Erfahrungen und Rückmeldungen nur bedingt auf beliebige Lerngruppen übertragen werden. Nicht zuletzt die technikzentrierten Informatik-Freaks fehlten, wenngleich grundsätzlich eine stabile informationstechnische Grundkenntnis vorhanden war.

Allerdings gibt es Anzeichen dafür, dass mit den behandelten Themen mehr Schülerinnen für informatische Themen interessiert und begeistert werden können. Insbesondere Mädchen, die verschiedenen Beobachtungen zufolge verstärkt Fragen nach dem Sinngehalt und Mehrwert von Technik stellen und für die Computer weniger Selbstzweck als Mittel zum Zweck sind. Die Einbettung von Informationstechnologien in gesellschaftliche Sinnzusammenhänge bedient gerade diese Neugier und befördert den Wunsch, den technischen Hintergrund besser verstehen zu können.

Anzeichen für diese These zeigen Rückmeldungen der weiblichen Teilnehmerinnen der AG, die sich durchweg von den behandelten Themen angesprochen fühlten und gern mehr technischen Hintergrund bekommen hätten.

Dass diese Themen aber keine Selbstläufer für die gezielte Förderung von Mädchen sein müssen, zeigen andere Beobachtungen. So blieb das Diskussionsverhalten innerhalb der typischen geschlechtsspezifischen Unterschiede. Während die Jungen schnell den Finger oben hatten oder einfach mal das eine oder andere Argument ausprobierten bzw. ihr Hintergrundwissen zu dieser oder jener Frage in den Raum stellten, waren die Mädchen tendenziell reflektierter und überlegter, begnügten sich dadurch oft mit der Rolle der Zuhörerinnen. Auch blieben sie bei Gruppenarbeiten häufig unter sich, konnten dabei aber natürlich mit den Jungen mithalten. Diese Beobachtungen decken sich also mit systematischeren Untersuchungen zum koedukativen Unterricht.

2 Unterrichtseinheiten

Geschichte der Informatik: Zwei Methoden haben wir erfolgreich erproben können: Einerseits eignen sich historische Themen sehr gut für Schülerreferate. Es gibt inzwischen viele zum Teil ausgezeichnete Online-Quellen, die Lehrern und Schülern mit vertretbarem Rechercheaufwand zur Verfügung stehen. Bei freier Recherche dürfen aller-

dings Hinweise zum kritischen Umgang mit Quellen nicht fehlen, denn nicht notwendigerweise ist der erste Google-Treffer auch der beste.

Andererseits kann die Geschichte der Informatik eindringlich in einer Ausstellung zur Computergeschichte vermittelt werden, in der Geräte, Apparate und Maschinen noch den Geist ihrer Zeit ausstrahlen (vgl. [Pe05]). Nicht in jeder Stadt gibt es hierzu die Gelegenheit, wir waren zu diesem Zweck im Technikmuseum Berlin, das sich zwar auf Zuse-Rechner beschränkt, aber auch hier die wichtigsten Etappen der Computergeschichte vor dem PC dokumentiert.

Geistiges Eigentum: Die Auseinandersetzung um die Wissensordnung der Informationsgesellschaft ist durch eine komplexe Durchmischung unterschiedlicher Interessenlagen geprägt. Als geeignete Methode, diese Komplexität zumindest in Grundzügen zu verdeutlichen, wurde daher das gelenkte Rollenspiel gewählt. Jeder Schüler übernahm dabei die Rolle einer Interessengruppe, vom Autor über den Verleger und Verwerter bis zu privaten und öffentlichen Rezipienten. Jede Interessengruppe sollte ihre Vorstellungen über den Umgang mit Werken vertreten und geeignete Maßnahmen zur Durchsetzung vorschlagen. Gelenkt wurde das Spiel durch die Lehrer, die als Ideen- und Impulsgeber auftraten und dafür verantwortlich waren, das Spiel innerhalb der gewünschten Bahnen zu halten. Die Ergebnisse des Spiels wurden in den folgenden Stunden ausgewertet. Insgesamt umfasste die Einheit zum geistigen Eigentum acht Unterrichtsstunden. Eine ausführliche Diskussion dieses Urheberrecht-Rollenspiels würde den hier zur Verfügung stehenden Rahmen überschreiten, es wird noch in einer eigenen Publikation vorzustellen sein. Zwei Anmerkungen sollen genügen:

1. Auf Schülerseite zeigte sich ein ausgeprägtes Gespür für die verschiedenen Interessenlagen, bereits nach kurzer Zeit waren sie in der Lage, die Position der anderen zu übernehmen und zu vertreten. Überrascht waren sie bei Hinweisen, wie weit ihre Ideen zur Durchsetzung der Interessen der von ihnen übernommenen Positionen in der öffentlichen Diskussion bereits vorangeschritten sind, von rechtlichen bis zu technischen Maßnahmen.

2. Ein gelenktes Rollenspiel erscheint im Rückblick als ideale Methode, gesellschaftspolitisch relevante Inhalte sowohl zu simulieren als auch durch die Lenkung in ihrer Komplexität zu reduzieren. Der erfolgreiche Spagat zwischen Simulation und Reduktion setzt allerdings Hintergrundwissen zu den Positionen der einzelnen Interessengruppen voraus, um im richtigen Moment mit den nötigen Impulsen die Ideen der Schüler zu verstärken oder zu bremsen. Es ist daher methodisch anzusiedeln zwischen lehrerzentriertem Unterrichtsgespräch und handlungsorientiertem Projekt.

Datenschutz: Die Unterrichtseinheit zum Datenschutz orientierte sich am Planspiel Jugend im Datennetz, das ursprünglich von Brandt, Heinzerling und Kempny für das hessische Institut für Bildungsplanung und Schulentwicklung entworfen wurde [BHK91], und seither von verschiedenen Seiten weiter entwickelt wird (u.a. in [DGW05]).

Ökologie: Zwei Doppelstunden wurden in dieser Einheit angeboten: (i) Die Sensibilisierung für den Energieverbrauch von Computern, durchgerechnet am geschätzten

Verbrauchs eines Computer-Pools. Die Angaben zur Leistungsaufnahme auf der Geräterückseite oder -unterseite ermöglichen in Zusammenhang mit der geschätzten Betriebsdauer eine Abschätzung des Energieverbrauchs, der leicht in anfallende Kosten umgerechnet werden kann. (ii) Ein weiteres wichtiges Thema im Bereich Computer und Ökologie ist der Elektroschrott. Hier bietet sich eine Exkursion in ein nahe gelegenes Recycling-Unternehmen an. Der Verantwortliche vor Ort verdeutlichte die Spielregeln der Abfallwirtschaft, die Kosten, die ein einzelner PC aufwirft, und beschrieb die ökologischen Probleme, die aus unsachgemäßer Produktion, Verwendung und Entsorgung entstehen können.

Kryptographie: Die Unterrichtseinheit umfasste drei Doppelstunden mit verschiedenen Schwerpunkten: (i) In der einführenden Stunde zu kryptographischen Verfahren wurden als Motivation der Einheit die wichtigsten Anwendungsgebiete erörtert. Anhand eines Rollenspiels erlernten die Schüler grundlegende Protokolle für den Schlüsselaustausch und erprobten mögliche Schwachstellen kryptographischer Kommunikation. Die Schüler versuchten im Rahmen des Rollenspiels anschließend, die zunächst vorgegebenen Protokolle zu verbessern. (ii) Als neues Thema wurde in der zweiten Unterrichtsstunde die E-Mail-Verschlüsselung thematisiert. Die Sensibilität für die Gefahren bezüglich der Vertraulichkeit von E-Mails musste bei der Mehrheit der Schüler erst entwickelt werden. Gemeinsam wurde also herausgearbeitet, welche typischen Sicherheitsdefizite beim Versenden von Nachrichten auftauchen. In der Unterrichtsstunde bot es sich an, verfügbare freie Programme zur E-Mail-Verschlüsselung wie GnuPG vorzustellen. Das Programm ermöglicht sowohl das Signieren als auch das Verschlüsseln von Nachrichten. Welche kryptographischen Algorithmen das Programm verwendet und wie man ein Schlüsselpaar erzeugt, wurde mit Hilfe von GnuPG vorgeführt. Die Hausaufgabe der Unterrichtsstunde war die selbständige Installation des Programms auf dem heimischen Rechner sowie die Dokumentation des Vorgehens. Eine verschlüsselte E-Mail sollte quasi als Beweis an den Lehrenden gesendet werden. Diese Hausaufgabe gingen alle Schüler mit Begeisterung an, bis auf eine Ausnahme installierten alle erfolgreich das Programm und verschickten die geforderte Nachricht. In der Folge verwendeten die Schüler verschlüsselte E-Mails für die Abgabe weiterer Hausaufgaben sowie für Fragen, die sich nach dem Unterrichtseinheiten ergaben. (iii) Die dritte Unterrichtsstunde erweiterte das Wissen um die kryptographischen Protokolle aus der ersten Stunde. Mit vorgegebener Notation wurden komplexere Protokolle von den Schülern eigenständig nachvollzogen. Bereits nach kurzer Zeit waren sie in der Lage, die neu erlernte Notation auf neue Protokolle anzuwenden. In Gruppen von zwei oder drei Schülern stellten sie ihre Ergebnisse den übrigen Schülern vor.

Sicherheit: Die Schwachstellen und Sicherheitslücken heutiger Betriebssysteme und Anwendungssoftware nimmt in der Hochschul-Informatik einen größer werdenden Raum ein. Die Angriffe durch Schadcode verlagern sich derzeit vom Betriebssystem auf die Anwendungen, beispielsweise auf E-Mail-Programme, Webbrowser oder Antiviren-Software. Eine Doppelstunde mit einem gelenkten Unterrichtsgespräch verdeutlichte die Angriffsvektoren beim Ausnutzen von Sicherheitslücken und stellte die wichtigsten Schadprogramme vor: Viren, Würmer und Trojaner. Mit den Schülern wurden dabei auch geeignete Maßnahmen zum Schutz des heimischen Computers erörtert. Inhaltlich unterstützt wurde das Unterrichtsgespräch durch Folien. Die Schüler begeisterte das

Thema sichtlich, sie stellten viele Fragen. Einige der noch offenen Fragen zu Phishing und Spyware konnten erst in der folgenden Stunde ausgewertet werden. Die Unterrichtseinheit zur Sicherheit von Computern eignet sich aufgrund der Aktualität des Themas und der Fülle an Material auch für zwei Doppelstunden.

Ethik: Zwei Doppelstunden wurden zum Thema Informatik und Ethik angeboten: (i) Die Sensibilisierung der Schüler für ethische Dilemmata beim Einsatz von Computern wurde mit Hilfe zweier Fallbeispiele, welche die GI-Fachgruppe Informatik & Ethik erarbeitet hat, erstmals erprobt. Die anonymisierten Beispielfälle ethischer Konflikte sollten den Diskurs über die Verantwortung und die Verantwortlichkeit der Folgen des Einsatzes von IuK-Technologie einleiten. Keine andere Unterrichtsstunde hat die Schüler so intensiv zur Diskussion und zum Mitdenken angeregt, dass sie sogar das Ende der Doppelstunde ignorierten und in der Folge in zahlreichen E-Mails die Fallbeispiele weiter kontrovers diskutierten. (ii) In der zweiten Doppelstunde wurde als weiteres Thema die sogenannte Hackerethik diskutiert. Die den Schülern noch unbekannten Begriffe „White Hat, Grey Hat, Black Hat" wurden unterschieden und die Genese der Hackerethik sowie die zugrunde liegenden Wertvorstellungen und der moralische Anspruch von Hackern, die sich freiwillig einer Hackerethik unterwerfen, vorgestellt.

Multimediarecht: Nie war es für Schüler so einfach, mit dem Gesetz in Konflikt zu kommen, wie in Zeiten des Internet. Neben dem Umgang mit geistigem Eigentum gibt es gerade bei Veröffentlichungen auf Webseiten zahlreiche juristische Rahmenbedingungen zu beachten. Zu den in der AG behandelten Themen gehörten (i) der Umgang mit Hyperlinks und die Frage, ab wann man für Inhalte auf fremden Webseiten verantwortlich ist; (ii) das Recht auf freie Meinungsäußerung und seine Grenzen in den Persönlichkeitsrechten, die u.a. vor Bedrohung, Beleidigung und übler Nachrede schützen; (iii) die Verpflichtung, für seine Äußerungen namentlich einzustehen, wie sie in der Impressumspflicht des Teledienstegesetzes bzw. des Mediendienstestaatsvertrags formuliert ist. Methodisch orientierte sich diese Unterrichtseinheit an den rechtsdidaktischen Hinweisen für den Informatik-Unterricht [Ko05b].

Digitale Medien: Die letzte Unterrichtseinheit behandelte die digitalen Medien, gegliedert an den Basismedien Ton, Schrift, Bild, Film, Netz. Aus dem Angebot mediendidaktischer Konzepte (vgl. [Tu02], S. 122 ff.) wurde abwechselnd die Ästhetisch-kulturorientierte (Stichwort: Bilder lesen) und die Handlungs-, interaktionszentrierte Methode (Stichwort: Filme drehen und schneiden) eingesetzt. Projektseitiges Ziel war auch hier zu zeigen, dass sich jedes Basismedium mit einer beliebigen mediendiaktischen Methode behandelt werden kann. Langfristig wird für die Unterrichtseinheit „digitale Medien" eine Matrix aus Unterrichtsentwürfen entstehen, in der die verschiedenen methodischen Herangehensweisen mit den Basismedien kombiniert sind.

3 Unterrichtsentwürfe

Auf der AG-Website [KK06] gibt es zu jeder Stunde einen kurzen Unterrichtsentwurf mit Angaben zu

- Thema der Stunde im Bezug zur jeweiligen Unterrichtseinheit.
- Datum und Uhrzeit.
- Methodische Angebote der Stunde. Diese Angaben werden zu einem späteren Zeitpunkt mit vertiefenden Informationen und Schulungsunterlagen zu den einzelnen Methoden verknüpft, sofern es sich nicht um unterrichtsübergreifende Standards handelt. Hervorzuheben seien die bereits erwähnten Plan- und Simulationsspiele, mit deren Hilfe komplexe Situationen sehr überzeugend veranschaulicht werden können, die aber in der Vorbereitung und Durchführung sowohl Fingerspitzengefühl als auch Bereitschaft zur Improvisation erfordern.
- Verlaufsplanung. Hier wird die Doppelstunde in aufeinander aufbauende Blöcke eingeteilt.
- Verwendetes Material in Form von Arbeitsblättern, Publikationen, Links etc. Alle im Rahmen der AG erstellten Arbeitblätter stehen zum Download bereit und können beliebig verwendet werden.

Ein deutlicher Mangel der Unterrichtsentwürfe ist ihre fehlende Sachanalyse. Sie unterstützen die Lehrkräfte in der zurzeit dargebotenen Form noch nicht bei der Einarbeitung in ein Thema. Das ist aber auch nicht der Sinn eines Unterrichtsentwurfs. Hier sind vielmehr eigenständige Publikationen in Arbeit, in der die gesellschaftlichen Dimensionen der Informatik mit der gebotenen Schärfentiefe diskutiert werden. Aus diesem Grund wird auch in diesem Beitrag auf sachbezogene Quellenangaben verzichtet, die zu jedem der angegebenen Themen natürlich in großer Anzahl zur Verfügung stehen, wenngleich deutlich seltenster mit Bezug auf Schulunterricht. Bis zu dieser schulbezogenen didaktischen Aufarbeitung sind die Unterrichtsentwürfe nur mit Vorkenntnissen von Nutzen. Insofern sind die vorliegenden Ausführungen neben ihrer praxisbezogenen Anwendbarkeit auch als Präsentation eines Bausteins für einen allgemein bildenden Informatik-Unterricht zu lesen. Der aber darf sich natürlich nicht in gesellschaftlichen Dimensionen erschöpfen.

Und so ist eine weitere Leerstelle die Verknüpfung mit technischen Kerninhalten. Wenngleich sich jedes der vorgestellten Themen zwanglos an technische Schwerpunkte angliedern ließe, fehlt das runde Gesamtkonzept für einen mehrdimensionalen Informatik-Unterricht. Das konnte und sollte die AG allerdings auch nicht leisten. Hier sind weitere Bausteine vonnöten, ehe die Angebote zu einem lebendigen und zeitnahen Informatikunterricht verbunden werden können, ein Unterricht, der das Prädikat verdient, auf das Leben und Arbeiten in der Informationsgesellschaft vorzubereiten.

Literaturverzeichnis

[KK06] Koubek, J., Kurz, C.: Computer – Mensch – Gesellschaft. Website der Informatik-AG. http://waste.informatik.hu-berlin.de/Lehre/informatik-ag/index.html

[BHK91] Brandt, F.; Heinzerling, H.; Kempny, G.: Jugend im Datennetz – Ein Planspiel. Reihe „Materialien im Unterricht“, Heft 105, Wiesbaden: Hessisches Institut für Bildungspla-

nung und Schulentwicklung, 1991. http://ddi.informatik.hu-berlin.de/schule/unterrichtsmaterial/JugendImDatennetz.pdf

[DGW05] Dorn, R.; Gramm, A.; Wagner, O.: Planspiel zum Datenschutz. In: LOG IN 136/137, 2005, S. 72-75.

[GI00] Fachausschuss 7.3 „Informatische Bildung in Schulen" der Gesellschaft für Informatik e.V.: Empfehlungen für ein Gesamtkonzept zur informatischen Bildung an allgemein bildenden Schulen. Beilage zu LOG IN 20 (2), 2000.

[Ko05a] Koubek, J.: Recht und informatische Bildung Rechtsdidaktische Hinweise für den Informatikunterricht. In LOG IN 136/137, 2005, S. 36-40.

[Ko05b] Koubek, J.: Informatische Allgemeinbildung. INFOS 2005.

[Pe05] Peters, I.-R.: Im Museum. In: LOG IN 136/137, 2005, S. 47-50.

[Tu02] Tulodziecki, Gerhard: Computer & Internet im Unterricht. Medienpädagogische Grundlagen und Beispiele. Berlin: Cornelsen Scriptor, 2002.

Lernortkooperation in der IT-Ausbildung – Kompetenzentwicklung in Projekten

Stephan Repp, Christoph Meinel
Hasso-Plattner-Institut für Softwaresystemtechnik GmbH
Prof.-Dr.-Helmert-Str. 2-3
D-14482 Potsdam
stephan.repp@hpi.uni-potsdam.de
meinel@hpi.uni-potsdam.de

Ralf Ziegler
Fachbereich Pädagogik
Universität Trier
Universitätsring 15
54286 Trier
zieg1101@uni-trier.de

Abstract: An der Berufsbildenden Schule für Gewerbe und Technik in Trier werden Anwendungsentwickler an konkreten Projekten ausgebildet. In einer Lernortkooperation werden reale Aufgabenstellungen, die bei einem Softwareentwicklungsunternehmen anfallen, genutzt, um die Themen objektorientierte Programmierung und Design zu erarbeiten, zu reflektieren und zu vertiefen. Ein Unternehmen aus dem benachbarten Ausland gestaltet aktiv die Ausbildung und gibt darüber hinaus einen Anstoß zur interkulturellen Bildung der Lernenden. Es werden Organisation, Durchführung und gewonnene Erfahrungen des Unterrichtskonzeptes vorgestellt.

1 Einleitung

Die duale Ausbildung zum Fachinformatiker stellt eine Säule für die Heranbildung von IT-Fachkräften in Deutschland dar. Fachinformatiker arbeiten in der Softwareentwicklung, Administration von Rechnersystemen und in der IT-Beratung. Diese Ausbildung ist gekennzeichnet durch den hohen Praxisanteil in den Betrieben und durch die Vermittlung der theoretischen Lerninhalte in der Berufsschule. Aufgrund der engen Verzahnung des Berufsausbildungssystems mit den Wirtschaftsbetrieben gilt dieses System weltweit als vorbildlich und als eine der Stärken der deutschen Wirtschaft. In der Vergangenheit zeigte sich aber, dass dieses System Schwächen besitzt. Einerseits sind dies die hohen Kosten für die Unternehmen, die mangelnde Ausbildungsfähigkeit vieler Schulabgänger und die oft unzureichende Ausstattungen der Schulen mit zeitgemäßer Technik sowie die Verzahnung der praktischen Inhalte in den Betrieben mit den theoretischen Lerninhalten in der Berufsschule, die oft nur schwer oder gar nicht zu erreichen ist. Hier fordert die

Bund-Länder-Kommission eine wesentlich engere Verzahnung von Schule und Betrieb [Be06].

Eine zukünftige Möglichkeit die Wissensunterschiede der Lernenden individuell zu fördern stellt der in [Li07] vorgestellte „Virtual Tele-TASK Professor“ dar.

Aufgrund des unterschiedlichen Vorwissens der Lernenden sind die Klassen oft sehr heterogen zusammengesetzt und stellen damit hohe Anforderungen an die Unterrichtsgestaltung. So besteht die Gefahr, dass sich einzelne Lernende schnell über- bzw. unterfordert fühlen. In einer Klasse sind z.B. Lernende mit bereits umfangreicher Erfahrung in der Softwareentwicklung, während andere Lernende diesbezüglich ihre ersten Erfahrungen erst nach dem Start ihrer Ausbildung erlangen. Ziel dieses Tagungsbeitrages ist es nicht, ein Patentrezept für die Lösung der oben geschilderten Probleme zu liefern, sondern die Darstellung eines Unterrichtskonzepts, um die Interaktion der Betriebe mit der Schule zu verbessern und die Vermittlung von Fach- und Handlungskompetenzen an realen Projekten in der Berufsschule zu ermöglichen. Darüber hinaus stellt er ein Konzept der individuellen Kompetenzförderung für die Lernenden dar.

2 Unterrichtskonzept

2.1 Fundament

Unser Unterrichtskonzept basiert auf der Annahme, dass ein Wissenserwerb in einem aktiven und Wissen anreichernden Prozess erfolgt, der vom Lernenden ausgeht. Lerngegenstände müssen danach in einer konkreten Situation oder einem Handlungszusammenhang stehen. Wissen lässt sich nicht vom Lehrenden auf den Lernenden „übertragen“. Es baut sich vielmehr vor dem Hintergrund eigener Erfahrungswelten auf. Der Mensch erkennt die Welt nicht, wie sie wirklich ist, sondern wie sie ihm erscheint. Lernen ist ein selbstreferenzieller, subjektiver Entwicklungsprozess und somit eine Reinterpretation von bereits Bekanntem [AS98, Sc00, Bo01]. Individuen können zwar durch externe Anstöße lernen, die Resultate der angestoßenen Lernprozesse sind allerdings von ihren bereits vorhandenen und entwickelten kognitiven Eigenstrukturen geprägt. Folgende Merkmale sind für einen konstruktivistischen Unterricht kennzeichnend [Sc00]:

- Lernen erfolgt unter aktiver Beteiligung der Lernenden. Diese müssen motiviert sein und an dem, was oder wie es zu tun ist, Interesse haben oder entwickeln.
- Die Lernenden steuern und kontrollieren ihre Lernprozesse selbst. Der Ausprägungsgrad dieser Selbststeuerung kann je nach Lernsituation variieren.
- Lernen wird konstruktiv durchgeführt. Der Erfahrungs- und Wissenshintergrund der Lernenden findet Berücksichtigung. Subjektive Interpretationen finden statt.
- Lernen findet in einem spezifischen Kontext statt.
- Lernen ist sozial ausgerichtet, indem es interaktiv geschieht und den soziokulturellen Hintergrund berücksichtigt.

Diese Merkmale bedingen eine Lehrerrolle, bei der die Beherrschung des Wissens nur eine Basisqualifikation darstellt. Der Lehrer ist Anbieter des Wissens, nicht „Überträger" des Wissens. Lehrende haben sich aus der Instruktionsrolle zu verabschieden - zugunsten der Aufgabe, Lernräume zu schaffen, Lerngegenstände zu inszenieren und so das selbstbestimmte Lernen zu fördern, zu moderieren und zu begleiten [Ho98, Gu02, Bo01]. Koubek beschreibt in [KF05] zwei Stufen für Kompetenzen in der Informatikerausbildung, die Stufe Technik und die Stufe Diskursanalyse. Bei der Kompetenzvermittlung Technik stehen die zu vermittelnden Techniken (z.B.: objektorientierte Softwareentwicklung) im Vordergrund. Diese Technologien sind bei unserem Konzept eingebunden in den Diskurs der Aufgabenstellung aus dem Unternehmen. In [SB05] werden Ansätze für ein Kompetenzmodell des informatischen Modellierens beschrieben. Hierin heißt es: „.in der die Anwenderperspektive der Lernenden in die Sichtweise eines Informatikerexperten überführt werden soll…." [SB05, S. 138]. Die Lernenden vollziehen einen Rollenwechsel hin zum IT-Experten, womit der Lernprozess unterstützt wird.

Aufbauend auf diesem theoretischen Fundament wird im Folgenden das Unterrichtskonzept dargestellt.

2.2 Konzept

Die Berufswelt eines/einer Anwendungsentwicklers/in ist stark von Projekten geprägt. Eine Anforderung eines Kunden, eines firmeninternen Kunden oder eines Vorgesetzten steht am Anfang einer jeden Arbeit eines Entwicklers. Dies spiegelt sich auch in der Struktur der praktischen Abschlussprüfung wider, in der die Auszubildenden ein Projekt bearbeiten und dokumentieren müssen [Br00]. Es ist daher naheliegend, dies als Unterrichtsmethode bei der Umsetzung unseres Unterrichtskonzeptes einzusetzen. Dabei unterscheidet sich natürlich das Projekt in einem Unternehmen sehr stark von einem Projekt in der Schule. Während in den Unternehmen verschiedene Vorgehensmodelle existieren und der Projektablauf sehr dynamisch und auf die Kundeninteressen abgestimmt ist, wird ein Projekt im Unterricht oft auf den reinen Ablauf fokussiert betrachtet [VWF05]. Frey skizziert für den strukturierten Ablauf eines Unterrichtsprojekts fünf Komponenten einer Synopse (Projektinitiative, Projektskizze, Projektplan, Projektdurchführung und -abschluss) die durchlaufen werden [Fr02]. Einige dieser Phasen spiegeln die verschiedenen Stufen des Softwareentwicklungszyklus im Unternehmen wider und lassen sich gut in unser Unterrichtskonzept integrieren. In unserem Projekt erfolgt die Projektinitiative in einem Vorgespräch mit der Klasse, in dem Ideen und Anregungen aus den Ausbildungsunternehmen gesammelt, vom Lehrenden bewertet und ein didaktisch passendes Projektthema ausgewählt wird. Da die Lernenden in unterschiedlichsten Unternehmen beschäftigt sind, stellt dies einen breiten Fundus an potenziellen Projekten und Themengebieten dar.

Der ausgewählte Projektauftrag wird vom beteiligten Unternehmen aus seinem Projektalltag generiert. Die Lernenden nehmen die Rolle der IT-Experten ein und übernehmen die Verantwortung für die Projektrealisation. Die Verzahnung eines Schulprojektes mit einem „realen" Softwareentwicklungsprojekt aus einem Unternehmen stellt weitere Anforderungen an die Gestaltung des Unterrichts. Die Kundenanforderungen sind oft

vage, was eine intensive Kommunikation der beiden Projektpartner erfordert. Ebenso müssen neue Technologien bewertet und in vorhandene Infrastrukturen integriert werden.

Bei der Planung des Projektablaufs in der Klasse entsteht ein gravierendes Problem. Da einige fachliche Kompetenzen den Lernenden am Anfang des Projektes fehlen, können diese die zeitlichen, organisatorischen und fachlichen Entscheidungen in der Planungsphase nur schwer oder gar nicht treffen. Im Gegensatz dazu werden im Unternehmen die Planung und das Design von erfahrenen Mitarbeitern durchgeführt, um ein eventuelles Projektscheitern zu vermeiden. Im Unterricht ist es daher nötig, flexibel auf die vorhandenen Kompetenzen der Lernenden einzugehen und eventuell Tutorien oder Hilfen in dieser Planungsphase anzubieten. Der Lehrende tritt als Moderator und Coach auf und steuert ggf. bei Fehlentwicklungen gegen um ein Scheitern zu vermeiden. Ein Scheitern wäre für die Lernmotivation fatal, obwohl es durchaus zu einem Kompetenzzuwachs bei den Lernenden beiträgt. Die Abschlusspräsentation und ein Feedback schließen das Konzept ab.

Der Unterrichtsablauf stellt sich im Detail wie folgt dar:

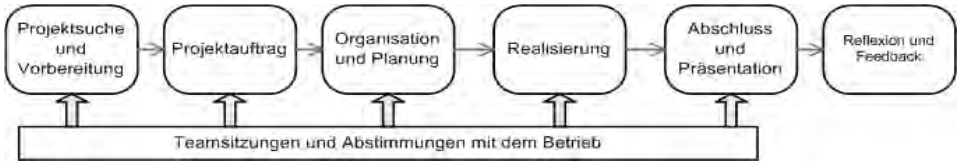


Abbildung 1: Unterrichtskonzept

Projektsuche und Vorbereitung

In dieser Phase unseres Unterrichtskonzeptes steht die Sammlung von Projektideen im Vordergrund. Die Betriebe werden durch den Lehrenden/Lernenden kontaktiert und der zeitliche und fachliche Rahmen abgeklärt. Zum Beispiel in unseren durchgeführten Projekten richtete sich die Wahl des Themas an folgenden technologischen Punkten: Einsatz einer objektorientierten Programmiersprache, objektorientiertes Design, breite Nutzung von grundsächlichen Technologien und Verfahren (z.B.: Datenbanken, ODBC, JDBC XML, etc…), Unabhängigkeit von proprietären Lösungen in den Unternehmen, kein finanzieller Vorteil für die Unternehmen (Sparen von Entwicklerkosten). Fokus ist die Erarbeitung und Vertiefung von objektorientierter Softwareentwicklung und Modellierung mit UML- Diagrammen.

Wünschenswert ist ebenso ein Projektthema, das über die reine Technologieumsetzung hinaus einen Diskurs zu anderen Themen, eine breite gesellschaftliche Relevanz und eine für die Arbeitswelt hohe Bedeutsamkeit besitzt. Die zu erreichenden Handlungskompetenzen im Projekt sind Teamwork, soziale Interaktion in Gruppen, Kommunikationstechniken anwenden, zielgerichtetes Arbeiten, fächerübergreifendes Umsetzen von Fähigkeiten/Kenntnissen, Hilfsbereitschaft und kundenfreundliches Auftreten.

Projektauftrag

Die Projektübergabe erfolgt im Unternehmen. Damit wird erreicht, dass die Lernenden einen Bezug zum Projekt erlangen und den Nutzen für den Kunden erkennen. Das Unternehmen stellt sich kurz dar und erläutert die Projektanforderung. Die Anforderungsbeschreibung und das Lastenheft werden vom Unternehmen bereitgestellt. Zusätzlich werden Hilfen zu speziellen Technologien geliefert (z.B.: Links, Bücher, eventuelle online-Kurse…). In einem offenen Gespräch werden Unklarheiten ausgeräumt, Fragen diskutiert und die jeweiligen Ansprechpartner festgelegt.

Abbildung 2: Übergabe des Projektes im Betrieb

Organisation und Planung

Nachdem das Projektziel der Gruppe bekannt ist, erfolgt die weitere Bearbeitung des Projektes in der Schule. In einem offenen Klassengespräch unterteilt die Klasse die Aufgabe in einzelne Arbeitspakete. Diese Aufgabenpakete werden im Klassenverbund detailliert definiert und beschrieben. Im nächsten Schritt müssen diese Aufgabenpakete den entsprechenden Gruppen und Lernenden, basierend auf den individuellen Kompetenzen und noch zu fördernden Kompetenzen, zugeordnet werden. Hierbei stellt sich die Frage, ob es sinnvoll ist, einem Erfahrenen ein entsprechendes Aufgabenpaket zu überlassen oder einen gänzlich Unerfahrenen in diesem Bereich einzusetzen. Ebenso sollen die persönlichen Neigungen und Interessen der Lernenden berücksichtigt werden, um deren Motivation zu stärken. Für die Lösung dieses Dilemmas wählen wir folgendes Vorgehen:

1. Feststellung der Kompetenzen der einzelnen Lernenden in den einzelnen Arbeitspaketen, basierend auf einem Klassengespräch und den jeweiligen Erfahrungen und Leistungen der Lernenden aus vergangenen Unterrichten.
2. Feststellung der Interessen der Lernenden und Wünsche für die Wahl eines Arbeitspaketes. Die Lernenden führen eine Priorisierung der gewünschten Arbeitspakete durch.

Nach dieser Analyse zeigte sich in den von uns durchgeführten Projekten, dass kein Lernender in allen Arbeitspaketen die entsprechenden Kompetenzen besaß, jedoch jeder Lernende mindestens in einem Bereich die entsprechenden Kompetenzen hatte. Die Lernenden wählten nicht nach ihren Kompetenzstärken die jeweilige Gruppe aus, son-

dern ließen sich von ihren Interessen leiten. Die Gruppen werden jetzt basierend auf dieser Analyse zusammengestellt. Falls bei einem Lernenden schon die jeweiligen Kompetenzen vorhanden sind (z.B.: bei einem erfahrenen Entwickler) so wird der Lernende in die nächste von ihm priorisierte Gruppe zugeordnet (z.B.: Zuordnung zur Gruppe „Präsentation“). Die Abbildung 3 zeigt für das Projekt „Rescouse Bundles“ die Gruppenaufteilung. Eine Gruppe entwickelt das Softwaredesign in Kooperation mit der Gruppe „Programmierung“, die die Klassen in JAVA realisiert und dokumentiert. Eine andere Gruppe entwickelt die Oberfläche der Anwendung mit Hilfe eines Eclipse-Plugins. Eine Gruppe ist für die Flip-Charts und die Abschluss-Präsentation verantwortlich, während eine andere Gruppe die Aufgabe erhält, den Entwicklungsprozess zu dokumentieren, die Kommunikation mit dem Unternehmen zu gewährleisten und zu steuern.

Abbildung 3: Beispiel der Aufteilung der Gruppen in der Klasse IuK F1a

Realisierung und Implementierung

Wie bereits erwähnt ist ein Schulprojekt nicht mit einem Projekt in einem Betrieb zu vergleichen. Ebenso ist ein Schulprojekt, wie es [Fr02] vorschlägt, in Bezug auf ein Softwareprojekt in der Industrie nicht anwendbar. Das Problem bei der Bearbeitung und vor allem bei der Planung eines Projektes ist, dass die Lernenden nicht die notwendigen Informationen und Kompetenzen im Vorfeld besitzen. Diese sollen erst im Projekt erlangt werden. Nachdem die Gruppen eingeteilt sind, wird daher jeder Gruppe eine individuelle Förderung durch Tutorien, Literaturangaben und Gesprächen angeboten.

Die grundlegende Softwarearchitektur und das Softwaredesign werden, je nach Kompetenzstand der Klasse, gemeinsam erarbeitet. Die regelmäßigen Teammeetings stellen für diese Hilfestellungen den nötigen Rahmen dar. Eventuelle Rückfragen, Unklarheiten und Detailfragen werden mit den Ansprechpartnern des Unternehmens abgesprochen und geklärt.

Abschluss und Präsentation

Nach der Planung, Realisierung und entsprechenden Test wird das Projekt bei der Firma präsentiert und die entwickelte Software übergeben. In einem Abnahmeprotokoll wird der Stand der Zielerreichung festgehalten und auf Fragen des Kunden/Unternehmens eingegangen.

Abbildung 4: Realisierung in der Berufsschule

Reflexion und Feedback

Nach der erfolgreichen Übergabe an das Unternehmen geht es daran, das Projekt innerhalb der Klasse zu reflektieren. Dabei wird ein anonymisierter Feedbackbogen eingesetzt, um die Akzeptanz dieses Projektes bei den Lernenden zu ermitteln, Kritik zu erhalten, die Motivation und den persönlichen Nutzen für die Lernenden festzustellen und zu hinterfragen, ob solche Projekte als sinnvoll erachtet werden.

Kompetenzermittlung

Auf Grund der unterschiedlichen Gruppenarbeiten und individuell zu unterscheidenden Leistungen wird auf die Transparenz der Kompetenzermittlung und der Notenfestlegung besonderen Wert gelegt. Auch geschieht die Benotung der Arbeiten im Hinblick auf die Abschlussprüfung. Die praktische Abschlussprüfung der IuK Prüfungen besteht aus einem Projekt, das die Prüfungskomponenten Projektdokumentation, Präsentation und Fachgespräch enthält [Br00]. Diese drei Komponenten spiegeln sich auch in der Bewertung unseres Projektes wider. Nachdem die Gruppen eingeteilt sind, werden die zu erbringenden Leistungen und die zu erreichenden Fachkompetenzen individuell mit der Gruppe besprochen, gewichtet und schriftlich festgehalten.

Hierbei wird zwischen den zu erarbeitenden Fachkompetenzen und den zu erbringenden Handlungskompetenzen unterschieden. So wird z.B. für die Gruppe „Präsentation“ und „Dokumentation“ die zu bewertende Fachkompetenz nach der Bewertungsmatrix der IHK bewertet. Diese Bewertungsmatrix wird bei der praktischen Abschlussprüfung bei der Bewertung von Projektarbeit und Präsentation eingesetzt [Br00].

Bei der Gruppe „Programmierung“ könnten die Fachkompetenzen „Funktion des Programms“, „Test“ und „Dokumentation“ ein Leistungsmerkmal sein. Der Gewinn, eines Einzelnen, an Handlungskompetenz in einem Projekt, ist dagegen schwerer zu bewerten. Die Einsatzfreude, die Kommunikationsbereitschaft mit den Teammitgliedern, Annahme der Tutorien, gegenseitige Hilfestellungen, selbstständige Lösungsfindung und die Übernahme von Verantwortung können Leistungsmerkmale sein, die am Anfang des Projekts mit der Klasse ausgehandelt werden. Am Ende des Projekts erfolgt ein Diskurs zwischen Eigen- und Fremdbewertung.

3 Durchgeführte Unterrichte

Aufbauend auf diesem Konzept wurden in der Vergangenheit zwei Projekte in zwei unterschiedlichen Klassen mit der FERNBACH-Software S.A. Luxemburg durchgeführt.[1] Der zeitliche Rahmen umfasste 2-4 Unterrichtsstunden in einem Zeitraum von 3 Monaten. Das kooperierende Unternehmen ist eine Softwarefirma, die sich auf die Entwicklung bankenspezifischer Produkte und Dienstleistungen spezialisiert hat.

Unsere durchgeführten Projekte liegen im 2. Lehrjahr, so dass sichergestellt ist, dass die Lernenden gewisse Grundkompetenzen in der Gruppenarbeit besitzen und der zeitliche Abstand zur Abschlussprüfung groß genug ist, um eventuelle fehlende Kompetenzen bis dahin aufzuarbeiten. Im 2. Ausbildungsjahr steht die objektorientierte Softwareentwicklung im Vordergrund. Elementare Grundlagen der strukturierten und der objektorientierten Programmierung sind bei den Lernenden vorhanden. Die Projekte werden im Folgenden kurz dargestellt.

3.1 Projekt: Grafische Darstellung eines Zahlungsplanes

Aufgabe des Projektes war es, für ein Kreditinstitut die Zahlungsströme eines Kredites zu visualisieren. Die in einem Zahlungsplan vorliegenden tabellarischen Daten sollten aus einer Datenbank gelesen und grafisch dargestellt werden (Abbildung 5). Das Thema Kredit stellt einen breiten Bezug zur Lebenswelt der Jugendlichen dar, da hier die oft angepriesenen Ratenkredite thematisiert wurden. Dies stellt gleichzeitig einen gesellschaftlichen Diskurs her. Eine Recherche der zur Verfügung stehenden Java-Bibliotheken für die Visualisierung von Diagrammen ging der Entwicklung voraus.

Ziel war es unter anderem auch, fächerübergreifend das Thema Kredit und Kreditvergabe zu thematisieren und in das Projekt zu integrieren. Diese theoretischen Erkenntnisse wurden durch die Lernenden in einer Webseite integriert. (Abbildung 6)

3.2 Projekt: Editor für die Resource-Bundle-Dateien

In diesem Projekt ging es um das Erstellen von sprachunabhängigen Java-Programmen mit Hilfe der so genannten Resource-Bundle-Dateien. Resource-Bundle-Dateien werden für eine Sprache erstellt und dann komplett in eine andere Sprache übersetzt. Läuft nun die Anwendung in der neuen Sprache, so müssen nur diese Resource-Bundle-Dateien ausgetauscht werden (Abbildung 7). Bei einer späteren Änderung oder Pflege der Software musste die ganze Resource-Bundle-Datei übersetzt werden. Damit bei einer späteren Änderung des Programms nur die veränderten Stellen übersetzt werden müssen, wurde ein Editor entwickelt, der die Sprachen Englisch, Deutsch und Französisch verwaltet (Abbildung 8). Die Oberfläche wurde als Plugin in die Eclipse-Umgebung eingebunden. Hierbei waren verschiedene gesellschaftliche Aspekte wie Globalisierung, Öffnung der Märkte und der härtere Wettbewerb der gesellschaftliche Bezug. Des Weiteren konnten die Lernenden ihre englischsprachige Kompetenz weiterentwickeln.

[1] Die Flip-Charts, die Präsentationen und die Diagramme können unter Info2007@repp.eu bezogen werden.

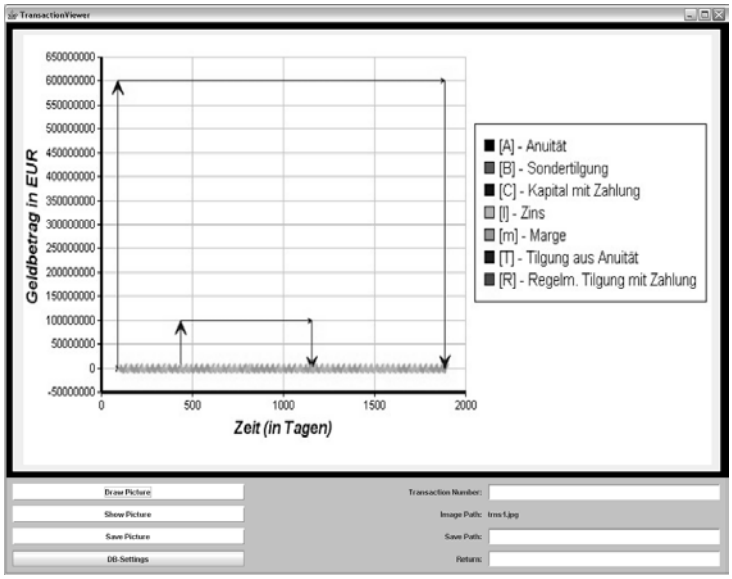


Abbildung 5: Anwendung für die „Visualisierung des Zahlungsverkehrs“

Abbildung 6: Theoretische Inhalte zusammengefasst in einer Web-Seite

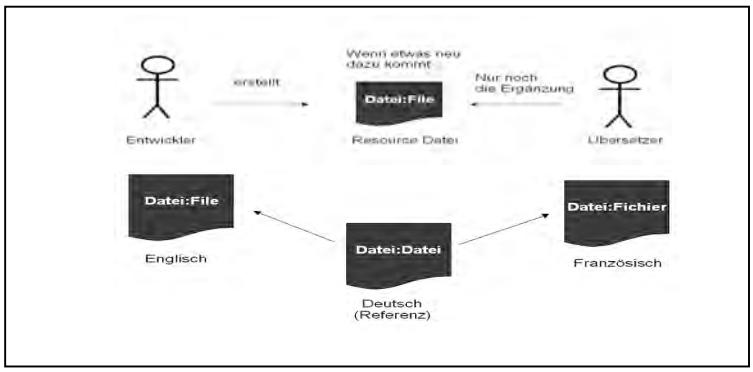


Abbildung 7: Prinzip des Editors für Resource-Bundle-Dateien

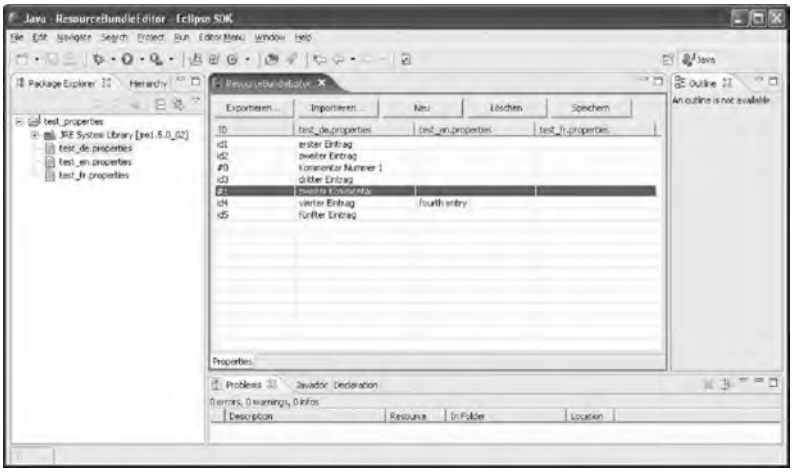


Abbildung 8: Oberfläche des Resource-Bundle-Editors

4 Schlussfolgerung und Ausblick

Hauptkennzeichen unseres Konzeptes ist die Integration einer Aufgabe aus dem Betrieb in den Unterrichtsalltag. Ein Projekt wird von einem Betrieb in Auftrag gegeben und intensiv und reflektiert in der Schule bearbeitet. Dies grenzt sich von einem reinen Schulprojekt und von einem reinen Wirtschaftsprojekt ab, da einerseits auf die Schwächen der Projektmitglieder in der Schule eingegangen werden kann und auf der anderen Seite ein enger Bezug zur Arbeitswelt besteht. Ein entscheidender Vorteil dieses Vorgehens ist die Motivation der Lernenden und die Viabilität für die Lernenden. Beim Feedbackgespräch und bei der Auswertung der Bewertungsbögen war dies bei allen Lernenden positiv beurteilt worden. Die Lernenden nahmen es begeistert auf, den Schulalltag zu verlassen und eine „praktische“ Aufgabe zu bearbeiten.

Das Projekt stellte einen Bezug zur Arbeitswelt der Lernenden her. Aktuelle technologische und organisatorische Entwicklungen in den Unternehmen konnten so nachvollzogen und in den Unterricht integriert werden. Nicht nur der Lernende entwickelt sich und lernt, sondern auch der Lehrende behält den Bezug zur Praxis und kann auf die Bedürfnisse der Unternehmen reagieren und die Anforderungen in seinen Unterricht einbringen. Die Verzahnung von Betrieb und Schule erfolgt hier sehr eng. Ebenso ist das breite Spektrum an Themen und Querbeziehungen zu gesellschaftlichen Themen von Vorteil.

Nachteilig waren die Kommunikationsprobleme auf Grund der zeitlichen Beschränkung auf 2 Wochenstunden und die örtliche Entfernung zwischen Auftraggeber und Schule. In den Feedbackbögen wurde der Kommunikationsfluss als der größte Kritikpunkt angesehen. Die Lernenden kritisierten, dass die Aufgabenstellung des Unternehmens nicht ganz klar war, die Kommunikation innerhalb der Gruppen schwierig war und die Abstimmung nicht optimal verlief. Dies stellt noch einen Lerneffekt dar, weil erfahrungsgemäß die Kommunikation der Projektpartner zwischen Erfolg und Misserfolg entscheidet. Kritik kam von anderen nicht beteiligten Firmen. Diese Firmen hatten Bedenken, dass die Auszubildenden im Namen der Schule für andere Unternehmen kostenlos arbeiten. Hier ist zu bedenken, dass der Aufwand und die Kosten für das kooperierende Unternehmen in keinem Verhältnis zum direkten wirtschaftlichen Nutzen des Projektes standen.

Einige Kompetenzen können auf Grund der Gruppenstrukturen nicht gleichmäßig bei allen Lernenden erarbeitet werden, aber im Rahmen dieses Projektes ist es erst möglich auf die individuellen Kompetenzschwächen und Wünsche der einzelnen Lernenden intensiv einzugehen. Wir sehen diese Projekte als sinnvolle Ergänzung zum Schulalltag, als unterstützende Hilfe für das berufliche Projektgeschäft und als wertvolle Vorbereitung auf die Abschlussprüfung an.

Literaturverzeichnis

[AS98] Arnold, R.; Schüßler, I.: Wandel der Lernkultur. Wissenschaftliche Buchgesellschaft, Darmstadt, 1998; S. 76 ff.

[Be06] Becker, M; Spöttl, G.; Dreher, R.; Doose C.: Berufsbildende Schulen als eigenständig agierende lernende Organisationen. – Forschungsbericht der Bund-Länder-Kommission, Heft 135, Bonn, 2006, S. 28, S.99-100, S.127.

[Bo01] Bonz, B.: Didaktik der beruflichen Bildung, Berufsbildung konkret. Schneider Verlag GmbH, Hohengehren, 2001; S. 186 und S. 220ff.

[Br00] Breuer, K.U.: Umsetzungshilfen für die neue Prüfungsstruktur der IT - Ausbildung, Abschlussbericht. Bundesministerium für Bildung und Forschung (BMBF), Bonn, 2000.

[Fr02] Frey, K.: Die Projektmethode. 9. Auflage, Beltz Verlag, Weinheim und Basel, 2002.

[Gu02] Gudjons, H.: Krisen als Wandlung im Lehrerberuf. In Pädagogik: Heft 11/2002; S. 6-12.

[Ho98] Hoffmann, N.: Selbstorganisiertes Lernen in (berufs-)biographischer Reflexion. Julius Klinkhardt – Verlag, Bad Heilbrunn, 1998; S. 7.

[KF05] Koubek J., Friedrich, S. (Hrsg.): Informatische Allgemeinbildung. In Tagungsband der 11. GI Fachtagung Informatik und Schule – Infos 2005, Springer-Verlag, Berlin Heidelberg New York, 2005, S. 57-66.

[Li07] Linckels S., Repp S., Karam N., Christoph Meinel: The Virtual Tele-TASK Professor - Semantic Search in Recorded Lectures. ACM SIGCSE'07, Covington, Kentucky, USA, 2007, S. 50 - 54

[Ot00] Ott, B.: Grundlagen des beruflichen Lernens und Lehrens. Cornelsen Verlag, Berlin, 2000; S. 38 ff.

[SB05] Schulte, C.; Brinda, T.: Beiträge der Objektorientierung zu einem Kompetenzmodell des informatischen Modellierens. In Tagungsband der 11. GI Fachtagung Informatik und Schule – Infos 2005, Springer-Verlag, Berlin Heidelberg New York, 2005, S. 137-148.

[Si03] Siebert, H.: Pädagogischer Konstruktivismus. Luchterhand, München, 2003.

[Sc00] Schelten, A.: Begriffe und Konzepte der berufspädagogischen Fachsprache. Franz Steiner Verlag, Stuttgart, 2000; S. 100-103.

[VWF05] Vocke, H., Woigk, U., Friedrich, S. (Hrsg.): Software- Engineering in der beruflichen Ausbildung- Simulation realer Projektsituationen. In Tagungsband der 11. GI Fachtagung Informatik und Schule – Infos 2005, Springer-Verlag, Berlin Heidelberg New York, 2005, S. 297- 307.

Lernzielgraphen und Lernzielerfolgsanalyse

Markus Steinert[1]

Didaktik der Informatik
TU-München
Boltzmannstr. 3
85748 Garching
markus.steinert@in.tum.de

Abstract: Die vorliegende Arbeit beschreibt eine Methode, wie eine konzeptuell-didaktische Gliederung der Informatik empirisch ermittelt werden kann. Die durch diese Methode gewonnene Klassifizierung unterscheidet sich von bekannten derartigen Strukturierungen dadurch, dass Lernziele als zentrale Elemente verwendet werden. Gegenüber abstrakten Taxonomien hat dies den Vorteil größerer Praxisnähe. Sofern die Lernziele darüber hinaus mit konkreten Aufgabenstellungen verknüpft sind und für diese Aufgaben Leistungserhebungen vorliegen, lässt sich analysieren, inwieweit die Lernziele von den Lernenden erreicht wurden. Da umfangreiche Leistungserhebungen bisher nur aus dem Hochschulbereich vorliegen, werden die Erstellung des Lernzielgraphen und die Lernzielerfolgsanalyse im Hochschulumfeld erläutert. Die Übertragung auf den schulischen Bereich ist jedoch bei entsprechender Datenlage problemlos möglich.

1 Einführung

Nach wie vor ist eine konzeptuell-didaktische Klassifizierung der Inhalte der Informatik ein Gegenstand der didaktischen Forschung. Mit zu den bedeutendsten Publikationen in diesem Bereich zählen die Arbeiten von A. Schwill [Sc93], sowie von P. Denning [De99] [De03]. Während die Taxonomie von A. Schwill auf den fundamentalen Ideen von J.S. Bruner [Br60] basiert, definiert P. Denning ohne Angabe einer detaillierten Methode sogenannte „Great Principles“ als bestimmend für die Informatik. Dennoch sind die Klassifikationen von A. Schwill und insbesondere die neueste Klassifikation von P. Denning rein konzeptuell sehr ähnlich. Obwohl die Notwendigkeit einer konzeptuellen Klassifizierung der Informatik außer Frage steht, werden vor allem die fundamentalen Ideen von einigen Autoren (z.B. [Ba98]) durchaus kritisch gesehen. Insbesondere die Frage, ob alle fundamentalen Ideen wirklich fundamental im Sinne von J.S. Bruner sind, ist durchaus nicht in allen Fällen empirisch gesichert (siehe dazu etwa [Sc06a], [Sc06b]).
Darüber hinaus hat sich auch gezeigt [Sc06a] [Sc06b], dass die Verwendung von Konzepten ein prinzipielles Problem nach sich zieht: Die Konzepte, seien es die fundamenta-

[1] geb. Schneider

len Ideen oder die „Great Principles“, sind Abstraktionen verschiedener konkreter Inhalte. Welche Inhalte das aber konkret sind, bleibt in vielen Fällen offen. Durch diese fehlende Spezifizierung verlieren die Konzepte, trotz ihres intuitiv „fundamentalen“ Charakters, vieles von ihrer Wertigkeit, da es in der Praxis oft schwer ist einen Inhalt einem bestimmten Konzept zu zuordnen.
Man könnte vermuten, dass von der Seite der Hochschulinformatik eine Klärung der Frage, was denn die zentralen Inhalte der Informatik seien, gegeben werden könnte. Insbesondere mit der Einführung von Bachelor / Master-Studiengängen müssten die Vorlesungen zur Einführung in die Informatik vergleichbare zentrale Konzepte vermitteln und diese Konzepte könnten dann die Grundlage einer detaillierten Modellierung eines konzeptuellen Netzwerkes bilden. Bereits ein kurzer Blick auf die Webseiten namhafter deutscher Universitäten [Tu07] [Tb07] [Ra07] zeigt, dass bereits in den Einführungsvorlesungen die Ansichten über die wesentlichen Inhalte der Informatik weit auseinandergehen.
Angesichts der geschilderten Probleme scheint ein praxisnäherer Ansatz zur inhaltlichen Strukturierung der Informatik empfehlenswert. Deshalb wird in der vorliegenden Arbeit ein lernzielorientierter Ansatz gewählt. Lernziele sind konkreter als Konzepte und lassen sich unter Verwendung geeigneter psychologischer Taxonomien nicht nur inhaltlich, sondern auch lernpsychologisch klassifizieren. Konkrete Problemstellungen, wie etwa Klausuraufgaben, werden auf die zugrundeliegenden Lernziele analysiert. Diese Lernziele werden anschließend in einem Lernzielgraphen unter Verwendung inhaltlicher und didaktischer Relationen vernetzt. Da für Klausuraufgaben im Hochschulumfeld umfangreiches Datenmaterial in Form von Prüfungsleistungen vorliegt, lassen sich durch geschickte Kombination der Leistungen in verschiedenen Aufgaben Rückschlüsse auf die Leistungsfähigkeit der Lernenden in einzelnen Lernzielen ziehen

2 Lernzielgraph und revidierte Taxonomie von Bloom

Die revidierte Bloomsche Lernzieltaxonomie [AK01] erweitert die bekannte Bloomsche Klassifikation [Bl56] kognitiver Lernziele um eine Wissensdimension. Neben den kognitiven Dimensionen (Wissen, Verständnis, Anwendung, Analyse, Synthese, Beurteilung) werden orthogonal dazu folgende Wissensstufen eingeführt: Faktenwissen, konzeptuelles Wissen, prozedurales Wissen und metakognitives Wissen. Letzteres beschreibt domänenübergreifendes Transferwissen und ist für die vorliegende Arbeit nicht relevant.

2.1 Die revidierte Lernzieltaxonomie und der Lernzielgraph im Beispiel

Das folgende Beispiel dient der Verdeutlichung der revidierten Bloomschen Taxonomie. Außerdem wird daran die Methode der Konstruktion des Lernzielgraphen vorgestellt.
Im Modulplan der RWTH-Aachen [Ra07] ist für die Veranstaltung „Praktische Informatik“ unter anderem folgendes „Lernziel“ angegeben:

> „Beherrschung der wesentlichen Konzepte imperativer und objektorientierter Sprachen, sowie wichtiger Programmiertechniken in diesen Sprachen“

Hinter dieser Formulierung verbergen sich insgesamt vier Lernziele:

1. Wesentliche Konzepte imperativer Sprachen beherrschen (C/3)
2. Wesentliche Konzepte objektorientierter Sprachen beherrschen (C/3)
3. Wichtige Programmiertechniken imperativer Sprachen beherrschen (P/3)
4. Wichtige Programmiertechniken objektorientierter Sprachen beherrschen (P/3)

In dieser Form lassen sich die Lernziele in die revidierte Bloomsche Taxonomie einordnen. Wie bei Anderson & Krathwol [AK01] beschrieben, erfolgt die Zuordnung eines Lernziels sehr systematisch: Aus den Verben resultiert die Zuordnung zu der kognitiven Dimension, aus den Substantiven die Zuordnung zu der Wissensdimension. Da die oben genannten Lernziele 1 – 4 verlangen, dass die entsprechenden Inhalte „beherrscht“ werden, sind alle in die dritte kognitive Dimension einzuordnen (Anwendung). Bei den Lernzielen 1 und 2 handelt es sich offensichtlich um konzeptuelles Wissen. Im Gegensatz dazu handelt es sich bei den Lernzielen 3 und 4 um die Beherrschung von „Techniken“; ein eindeutiger Hinweis, dass man es mit prozeduralem Wissen zu tun hat. Die

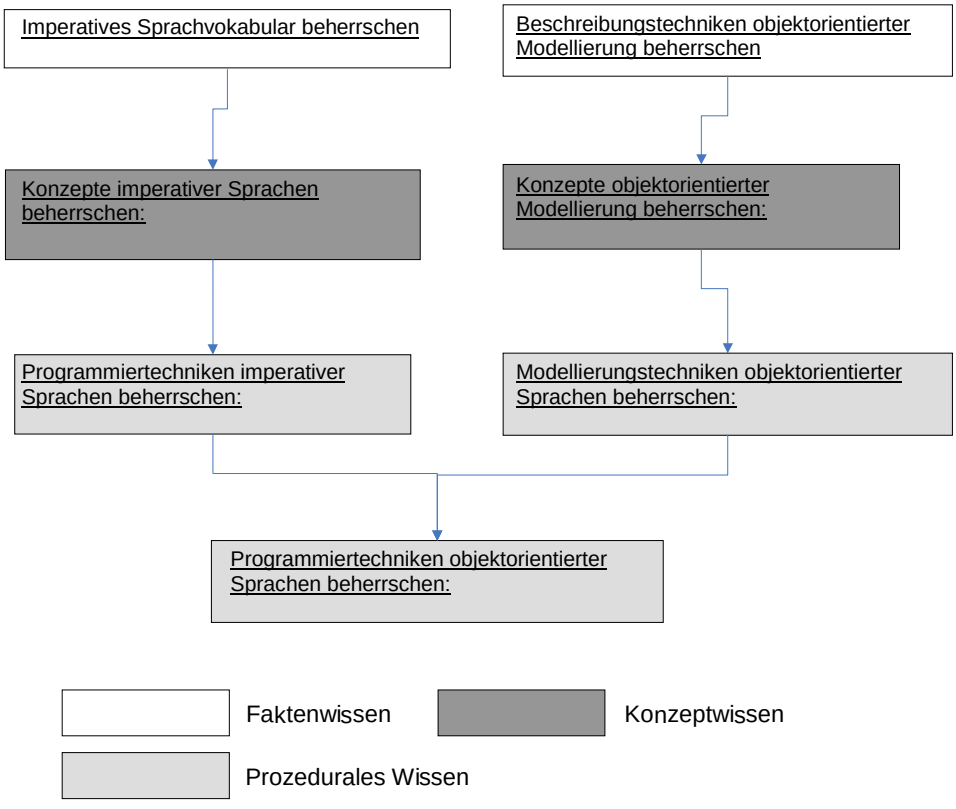


Abbildung 1: Lernzielgraph

soeben erläuterte Zuordnung ist bei den Lernzielen 1 – 4 am rechten Rand angegeben. Die Zeichen F, C und P symbolisieren Faktenwissen, kognitives oder prozedurales Wissen; die Zahlen das jeweilige kognitive Niveau: 1 → Verständnis, … 6 → Beurteilung.
In den Lernzielgraphen fließen nun die Relationen zwischen den einzelnen Lernzielen ein: In obigem Beispiel ist das konzeptuelle Lernziel 1 (imperative Konzepte) sicherlich eine Voraussetzung für das Erreichen des prozeduralen Lernziels 3 (imperative Programmiertechniken). Das konzeptuelle Lernziel 1 hat jedoch andererseits auch Faktenwissen als Voraussetzung, nämlich die Beherrschung des imperativen Sprachvokabulars. Analog benötigt man auch für die Beherrschung der Konzepte objektorientierter Sprachen entsprechendes Faktenwissen: Etwa die Beherrschung der Modellierungssprache UML. Die Beherrschung der Programmiertechniken objekt-orientierter Sprachen setzt jedoch nicht nur die Beherrschung objektorientierter Konzepte voraus; sie setzt voraus, dass objektorientierte Modellierungstechniken, -ein prozedurales Lernziel-, und die imperativen Programmiertechniken beherrscht werden. Fasst man diese Überlegungen in einem geeigneten Graphen zusammen, so ergibt sich der in Abbildung 1 dargestellte Lernzielgraph. Der Pfeil zwischen zwei Lernzielen symbolisiert dabei eine Vorrangrelation zwischen zwei Lernzielen, d.h. ein Lernziel von dem eine Kante ausgeht, ist Voraussetzung des Lernzieles, an dem diese Kante endet.
Die in Abbildung 1 verwendete Vorrangrelation für Lernziele wurde erstmals von A. Staller [St06] eingesetzt.

3 Lernzielgraphen von Klausuraufgaben

Das soeben angegebene Beispiel zeigt die grundsätzliche Vorgehensweise bei der Erstellung des Lernzielgraphen. Im weiteren werden nun konkrete Problemstellungen, insbesondere Klausuraufgaben, aus Einführungsveranstaltungen in die Informatik analysiert und der zugehörige Lernzielgraph erstellt.

3.1 Aufgabe 1: Objektorientierte Modellierung

Die erste Aufgabe, deren Lernzielgraph hier ermittelt wird, wurde im WS 2000/2001 an der TU-München als Klausuraufgabe zur Vorlesung „Einführung in die Informatik“ gestellt. Sie thematisiert einen Sachverhalt, der sich durch das Composite – Pattern beschreiben lässt:

> Aufgabe 1: Eine Prüfung kann entweder mündlich oder schriftlich abgelegt werden oder als mehrteilige Prüfung stattfinden. Jede Prüfung besitzt außerdem eine Note.
> a) Modellieren Sie diesen Sachverhalt in graphischer Darstellung und verwenden Sie dabei Ihre Kenntnisse über Entwurfsmuster.
> b) Geben Sie graphisch ein Instanzdiagramm der Prüfung „Einführung in die Informatik I“ an, die Sie gerade ablegen.

Eine mögliche Lösung ist in Abbildung 2 angegeben.

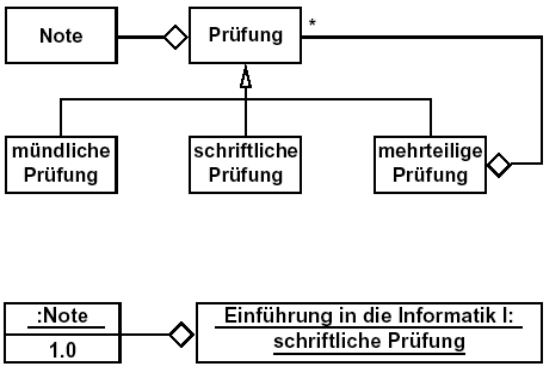


Abbildung 2: Lösung zu Aufgabe 1a und 1b

Die Formulierung der Aufgabenstellung und der Lösung bilden die Grundlage der Ermittlung des Lernzielgraphen. Dabei lassen sich zwei grundsätzlich unterschiedliche Kategorien von Lernzielen definieren: Sogenannte explizite Lernziele, deren Inhalt speziell durch die Aufgabenstellung bedingt ist und Lernziele, die implizit in der Aufgabe enthalten sind. Explizite Lernziele schlagen sich direkt in der Formulierung des Aufgabentextes nieder und bestimmen wesentlich die Formulierung der einzelnen Teilaufgaben. In Aufgabe 1 wären dies etwa folgende Lernziele:

- Rekursive Strukturen mithilfe des Composite Pattern beschreiben
- Das Composite Pattern instanzieren

Daneben gibt es implizite Lernziele, ohne die das Erreichen der expliziten Lernziele nicht möglich ist. Sie lassen sich etwa aus der Musterlösung herausarbeiten, indem man untersucht, welche Inhalte der Informatik auf welchem kognitiven Niveau benötigt werden, um die Lösung zu finden. In obigem Beispiel wäre dies alle Konzepte objektorientierter Klassenmodellierung: Klasse, Instanz, Aggregation, Vererbung, aber auch die Konzepte aus dem Bereich rekursiver Datenstrukturen. Diese Konzepte müssen jedoch in der vorliegenden Aufgabe in einem neuartigen Kontext identifiziert werden. Wir haben es deshalb mit Lernzielen der kognitiven Stufe 4 zu tun. Daneben ist jedoch auch Faktenwissen im Bereich der UML notwendig; allerdings lediglich auf Anwendungsniveau. Setzt man diese Überlegungen in einem Lernzielgraph um, so ergibt sich ein Graph wie in Abbildung 3. Die einzelnen Wissensstufen in Abbildung 3 sind dabei durch die Zahlen 1, …, 6 bei dem jeweiligen Lernziel codiert.

3.2 Aufgabe 2: Referenzgeflecht und Instanzdiagramm

Für die im nächsten Abschnitt erläuterte Methode der Lernzielerfolgsanalyse wird der Lernzielgraph einer weiteren -thematisch verwandten- Aufgabe benötigt. Es handelt sich ebenfalls um eine Klausuraufgabe, bei der für ein gegebenes objektorientiertes Programm das Instanzdiagramm oder Referenzgeflecht angegeben werden soll. Auf die Präsentation der Aufgabendetails wird hier aus Platzgründen verzichtet.

3.3 Kombination der Lernzielgraphen und lernpsychologische Auswertung

Da es sich bei den Aufgabenstellungen in den Aufgaben 1 und 2 um Problemstellungen mit dem Themenschwerpunkt „Objektorientiertes Modellieren“ handelt, haben die beiden Lernzielgraphen gemeinsame und unterschiedliche implizite Lernziele. Diese verschiedenen Mengen von Lernzielen werden im Weiteren von Interesse sein. Den kombinierten Lernzielgraphen, der die Lernziele beider Problemstellungen umfasst, zeigt Abbildung 4: Hierbei grenzt die gestrichelte Linie die Lernziele ein, die beiden Aufgaben gemeinsam sind; die durchgezogene Linie begrenzt Lernziele, die ausschließlich der Aufgabe 1 zu zuordnen sind und die gepunktete Linie diejenigen, die nur für Aufgabe 2 relevant sind.
Aus den Vorrangrelationen sowie den Zuordnungen der Lernziele zu der revidierten Bloomschen Taxonomie lassen sich Rückschlüsse hinsichtlich des Schwierigkeitsgrades einer Aufgabe ziehen. Sinnvoll erscheint es dabei für jede Wissensstufe, also für Faktenwissen, konzeptuelles Wissen und prozedurales Wissen getrennt, jeweils in entspre-

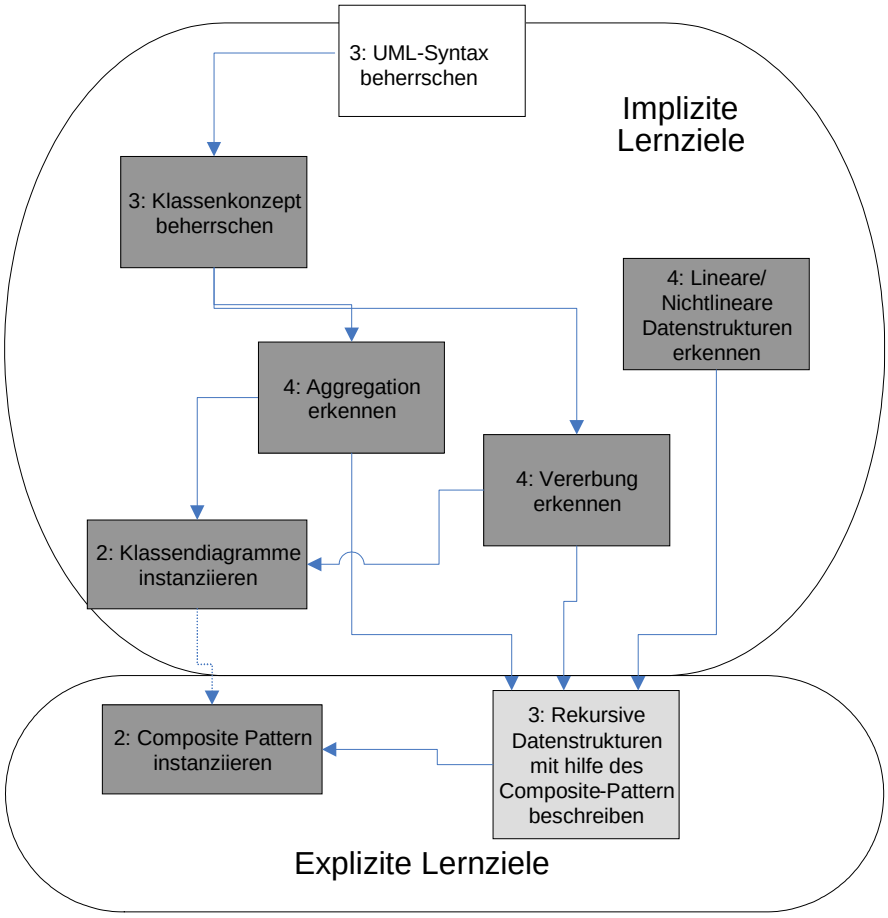


Abbildung 3: Lernzielgraph der Aufgabe 1

chende Schwierigkeitsbewertung anzugeben. Folgende Vorgehensweise bietet sich an: Die Komplexität einer Teilaufgabe und damit eines expliziten Lernziels auf einer bestimmten Wissensstufe wird bestimmt durch das Maximum der kognitiven Stufen der Lernziele derselben Wissensstufe, die von diesem impliziert werden und der kognitiven Stufe des Lernziels selbst; die Schwierigkeitsstufe wird somit nach einer Art Flaschenhalsprinzip ermittelt.
Beispielsweise hat das Lernziel der Aufgabe 1b (Composite Pattern instanziieren) unter Berücksichtigung von Transitivitäten 5 konzeptuelle Lernziele als Voraussetzung; von diesen weisen zwei die Wissensstufe 4 auf. Somit ergibt sich für dieses Lernziel die Schwierigkeitsbewertung 4. Tabelle 1 zeigt die Schwierigkeitsstufen von Aufgabe 1 und 2 im Überblick.

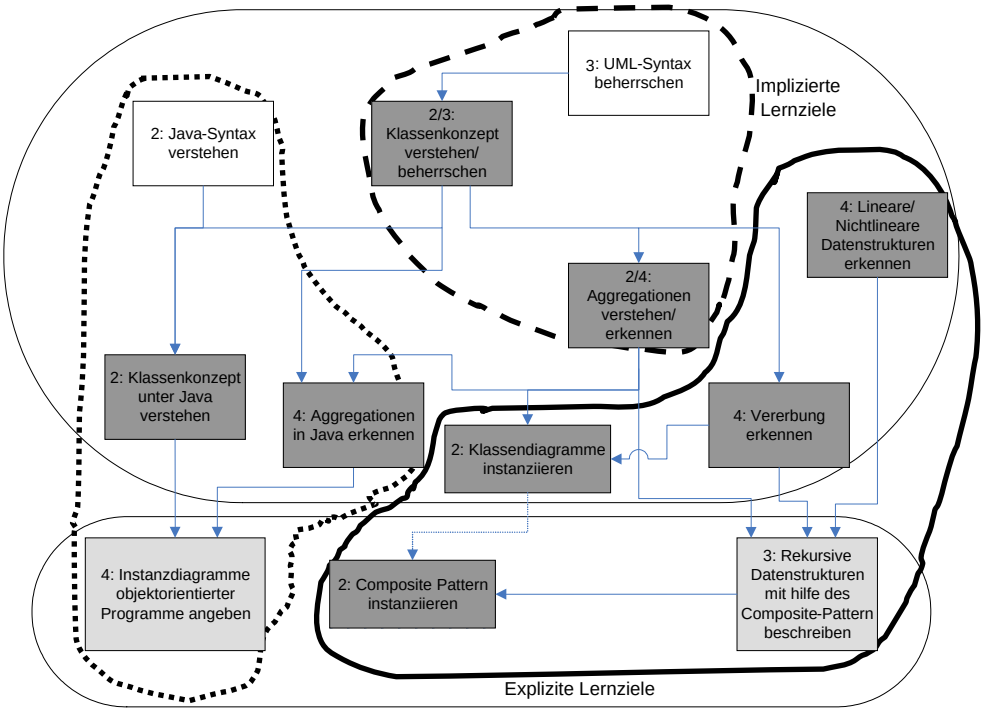


Abbildung 4: Kombinierter Lernzielgraph der gemeinsamen und exklusiven Lernziele

Wissenstufe	**Aufgabe 1 / Kognitive Stufe**	**Aufgabe 2 / Kognitive Stufe**
Faktenwissen	Stufe 3	Stufe 3
Konzeptuelles Wissen	Stufe 4	Stufe 4
Prozedurales Wissen	Stufe 3	Stufe 4

Tabelle 1: Schwierigkeitsstufen gemäß dem Lernzielgraph in Abbildung 4

Eine grobe Einschätzung der in Tabelle 1 angegebenen Schwierigkeitsstufen macht deutlich, dass beide Beispiele auf den verschiedenen Wissensstufen durchaus ähnliche Schwierigkeitsstufen aufweisen. Der objektive Schwierigkeitsgrad dürfte deshalb ver-

gleichbar sein und die Studenten sollten in den beiden Aufgaben ähnliches Leistungsvermögen zeigen.

4 Die Lernzielerfolgsanalyse

4.1 Grobanalyse

In der Lernzielerfolgsanalyse soll untersucht werden, inwieweit die Studierenden die expliziten und impliziten Lernziele der oben erläuterten Problemstellungen erreicht haben. Da beide Aufgaben lernpsychologisch ähnliche Schwierigkeitsanforderungen aufweisen, sollten sie von den Studenten mit ähnlichem Erfolg gelöst worden sein. Eine einfache Mittelwertanalyse der Prüfungsleistungen dieser Klausuraufgaben, --siehe Tabelle 2--, zeigt jedoch, dass die Studierenden insgesamt, aber auch einzelne Gruppen, ein sehr unterschiedliches Leistungsvermögen in den beiden Aufgaben haben.
Auffällig ist, dass die Studierenden allgemein die Aufgabe zum Referenzgeflecht deutlich schlechter bewältigt haben als die Aufgabe zur objektorientierten Modellierung. Dies deutet darauf hin, dass die Studierenden mit den Lernzielen, die ausschließlich der Aufgabe zum Referenzgeflecht zu zuordnen sind (Java-Syntax, Umsetzung von Klassenmodellen in Java, Interpretation von OO-Programmen), mehr Schwierigkeiten haben als mit denjenigen Lernzielen, die nur für die Modellierung relevant sind.

	Modellierung	Referenzgeflecht
Alle (703)	50,85 %	33,91 %
Frauen (132)	47,92 %	23,96 %
Männer (567)	51,57 %	36,20 %
Mathematik Diplom (21)	59,52 %	54,27 %
Informatik Diplom (570)	52,68 %	35,46 %
Informatik Bachelor (66)	35,61 %	21,59 %

Tabelle 2: Mittlere relative Punktezahl der Studierenden bei den untersuchten Klausuraufgaben

Von Interesse ist auch der Unterschied zwischen weiblichen und männlichen Studierenden: Zwar erreichen die weibliche Studierenden in beiden Aufgaben eine niedrigere durchschnittliche Punktzahl als männliche, jedoch sind die Unterschiede in den Aufgaben zum Referenzgeflecht deutlich größer. Weibliche Studierende dürften somit bei den Lernzielen, die ausschließlich der Aufgabe zum Referenzgeflecht zu zuordnen sind, größere Lücken aufweisen als männliche.
Auffällig ist weiterhin die mittlere Punktzahl der Studierenden der Mathematik, die Informatik nur als Nebenfach studieren; sie bilden die Gruppe mit der höchsten mittleren Punktzahl und haben in beiden Aufgaben ein ähnliches Leistungsvermögen. Sie bilden somit eine Gruppe, die der oben angeführten lernpsychologischen Schwierigkeitsbewertung am Nächsten kommt.

4.2 Differenzanalyse

Anstatt wie in der Grobanalyse eine bestimmte Standardgruppe zu wählen und deren Leistungsfähigkeit in den beiden Aufgaben zu bestimmen, lässt sich die Vorgehensweise auch invertieren. Man stellt die Frage, wie viele Studierende haben beispielsweise beiden Aufgaben gut gelöst, oder welche haben die Modellierungsaufgabe gut, jedoch die Aufgabe zum Referenzgeflecht schlecht gelöst. Hintergrund dieser Strategie ist wiederum der kombinierte Lernzielgraph: Studenten, die etwa genau eine der beiden Aufgaben mangelhaft gelöst haben, haben die Lernziele, die ausschließlich der mangelhaft gelösten Aufgabe zu zuordnen sind, nicht erreicht; es ist jedoch sehr wahrscheinlich, dass sie die Lernziele, die beiden Aufgaben gemeinsam sind, erreicht haben. (Vereinfachend gehen wir im weiteren davon aus, dass eine Aufgabe als mangelhaft gelöst gilt, wenn weniger als 40% der Punkte erreicht werden).
Im Detail ergeben sich folgende Fragen:

1. Welche Studenten haben beide Aufgaben mangelhaft gelöst und damit in erster Linie die beiden Aufgaben gemeinsamen Lernziele (UML-Syntax beherrschen, Klassen- und Aggregationskonzept beherrschen) nicht erreicht?

2. Welche Studenten haben genau eine der beiden Aufgaben mangelhaft gelöst und somit gerade die Lernziele, die exklusiv der mangelhaft gelösten Aufgaben zu zuordnen sind, nicht erreicht?

3. Welche Studenten haben beiden Aufgaben zufriedenstellend gelöst und damit alle Lernziele im wesentlichen erreicht?

Die Auswertung dieser Fragen liefert folgende Ergebnisse:

1. Etwa 30% der Studenten haben beide Aufgaben mangelhaft gelöst. Es ist somit sehr wahrscheinlich, dass diese Gruppe große Lücken bei den gemeinsamen Lernzielen aufweist. Die Studierenden dieser Gruppe beherrschen somit weder das Klassen- und das Aggregationskonzept noch können sie die UML-Syntax anwenden

2. Genau eine der beiden Aufgaben mangelhaft:

   (a) Etwa 33% der Studenten haben die Aufgabe zum Referenzgeflecht mangelhaft gelöst, jedoch die Aufgabe zur Modellierung zufriedenstellend. Man kann somit davon ausgehen, dass diese die gemeinsamen Lernziele erreicht haben, nicht jedoch die Lernziele, die exklusiv der Aufgabe zum Referenzgeflecht zu zuordnen sind: Somit verstehen etwa 33% der Studenten ein Java-Programm nicht oder können das Klassenkonzept nicht in Java umsetzen, obwohl sie das Klassen- und Aggregationskonzept verstanden haben.

   (b) Etwa 4% der Studenten haben die Aufgabe zur Modellierung mangelhaft gelöst, jedoch die Aufgabe zum Referenzgeflecht zufriedenstellend. Somit können etwa 4% der Studenten rekursive Strukturen nicht mit Hilfe des Composite-Pattern darstellen, obwohl sie das Klassen- und Aggregati-

onskonzept verstanden haben, die UML-Syntax beherrschen und das Klassenkonzept in Java umsetzen können.

3. Etwa 32% der Studenten haben beide Aufgaben zufriedenstellend gelöst und somit die vorausgesetzten Lernziele erreicht

Die Auswertung zeigt nochmals deutlich die Hauptschwierigkeiten der Studierenden: Bei etwa 30% der Studierenden liegen absolute Grundlagenprobleme vor: Basisbegriffe der Objektorientierung (Klasse, Objekt, Aggregation) wurden nicht verstanden. Weitere 30% haben große Schwierigkeiten mit dem Verständnis von Java-Programmen. Nur ca. 30% haben die Lernziele der Aufgaben wirklich erreicht. Diese Zahlen sind natürlich besorgniserregend angesichts der Tatsache, dass die Vorlesung, deren Klausuraufgaben hier analysiert werden, einen rein objektorientierten Ansatz wählte. Sie müssen jedoch auch vor dem Hintergrund der Studiensituation im WS 2000/2001 gesehen werden: Es war die Zeit als die TU München etwa 1200 Studienanfänger mit Hauptfach Informatik hatte. Unter diesen dürfte der Anteil der Studierenden, die ihre Studienentscheidung mit einem geringen Maß an Reflektion trafen, relativ hoch gewesen sein.
Sehr aufschlussreich ist es, für die soeben präsentierten Gruppen (1, 2a, 2b, 3) die Prüfungsleistungen in der gesamten Klausur zu ermitteln. Tabelle 3 zeigt die relativen Punktezahlen aller Aufgaben der Abschlussklausur im WS 2000/2001 für diese Studentengruppen (die Titel der Aufgaben sollen einen groben Anhaltspunkt hinsichtlich des Inhalts der jeweiligen Aufgabe geben)

Aufgabentitel	1	2 a	2 b	3
Signaturen und Terme	63,78%	81,62%	87,50%	90,17%
Boolesche Funktionen	35,74%	50,43%	72,99%	76,74%
Markov-Algorithmus	31,51%	58,17%	68,53%	79,56%
Rekursive Programmierung	28,37%	51,01%	78,45%	87,56%
Sortieren von Sequenzen von Zahlen	21,72%	37,87%	60,34%	72,50%
Reihungen	5,76%	14,37%	33,19%	55,50%
Binärbäume	8,42%	22,39%	50,69%	62,84%
Modellierung	17,15%	61,22%	21,98%	76,00%
Referenzgeflecht/Instanzdiagramm	6,10%	9,83%	70,69%	80,78%

Tabelle 3: Prüfungsleistungen in der Abschlussklausur zu „Einführung in die Informatik 1“ im WS 2000/2001

Die Studentengruppe 1 (Modellierung und Referenzgeflecht mangelhaft) weist in allen Aufgaben, die eng mit dem Klassenkonzept oder objektorientierter Programmierung verknüpft sind, also den Aufgaben zu Binärbäumen und Reihungen, im Durchschnitt eine sehr geringe Punktezahl auf. In konzeptuell anderen Aufgaben, wie beispielsweise den Aufgaben zu Markov-Algorithmen oder Booleschen Funktionen, hat diese Studentengruppe deutlich bessere Leistungen. Offensichtlich kann durch das beschriebene Ver-

fahren also der Lernzielerfolg in einer bestimmten Aufgabenklasse ermittelt werden. Entsprechende Beobachtungen lassen sich auch bei den Studentengruppen 2a und 2b machen, wo genau eine der beiden untersuchten Aufgaben mangelhaft gelöst wurde: So weist etwa die Gruppe 2a große Lücken im Bereich der objektorientierten Programmierung nicht jedoch im Bereich der Modellierung auf. Demzufolge ist das Leistungsvermögen dieser Gruppe auch in den Aufgaben zu Binärbäumen und Reihungen niedrig. Umgekehrt zeigt die Gruppe 2b, die genau die Aufgabe zum Referenzgeflecht zufriedenstellend gelöst hat nicht jedoch die Modellierungsaufgabe, in den Aufgaben zu Binärbäumen und Reihungen höhere Punktezahlen. Die Leistungen der Gruppe 3 schließlich liegen bei beiden Aufgaben im selben Bereich; dies unterstützt die oben erläuterte lernpsychologische Schwierigkeitsbewertung, wonach beide Aufgaben in allen drei Wissensdimemsionen ähnliche Ansprüche aufweisen.

5 Ausblick

Die hier vorgestellte Technik der Lernzielerfolgsanalyse stellt eine erste Iteration des Verfahrens dar. Es wird in Zukunft verfeinert werden. Insbesondere sollte der Prozess der Erstellung des Lernzielgraphen so weit wie möglich formalisiert werden. Darüber hinaus ist die Methode der Lernzielerfolgsanalyse im Hinblick auf statistische Gründlichkeit zu verbessern: Insbesondere bedarf die Verwendung simpler Mittelwerte oder die hier willkürlich gesetzte Grenze von 40% der Punkte für eine mangelhafte Leistung einer Modifizierung. Ebenso dürften sich durch Kombination der Lernzielgraphen von mehr als zwei Klausuraufgaben Aussagen über einzelne Lernziele gewinnen lassen.
Grundsätzlich ist das Verfahren jedoch geeignet eine Vielzahl dringender didaktischer Fragestellungen zu bearbeiten. Naheliegend ist es, die hier auf zwei Aufgaben begrenzte Diskussion auf ganze Klausuren zu erweitern und somit den Lernzielgraph der betreffenden Vorlesung zu ermitteln. Dadurch lässt sich der Lernerfolg der Studierenden in den Lernzielen der Vorlesung quantitativ angeben. Die damit einhergehende Identifizierung von Lernproblemen spezifischer Gruppen von Studierenden, könnte zur Optimierung der Vorlesungsstrategie beitragen.
Eine sehr viel weiter reichende Untersuchung betrifft die konzeptuelle Strukturierung der Informatik: Durch Erstellung der Lernzielgraphen von Vorlesungen verschiedener repräsentativer Lehrstrategien können die Lernziele dieser Vorlesungen sowie deren gegenseitige Abhängigkeiten dargestellt werden. Mithilfe geeigneter Schnittmengenbildung lassen sich die zentralen Lernziele und somit letztendlich auch die zentralen Konzepte der Informatik empirisch ermitteln. Das Eingangs geschilderte Problem der mangelnden Praxisnähe konzeptueller Taxonomien ist bei der beschriebenen Methode naturgemäß behoben.
Nicht zuletzt im schulischen Bereich ist der Einsatz denkbar. Zwar fehlen im Moment die hierzu notwendigen umfangreichen Datenerhebungen, jedoch wird dieses Problem in Zukunft, wenn beispielsweise an bayerischen Gymnasien flächendeckend der Unterricht im Pflichtfach Informatik (Jahrgangstufe 9 und 10) einsetzt, lösbar sein. Es ließe sich damit nicht nur der Lernzielerfolg in einer bestimmten Jahrgangstufe, sondern im Rahmen einer Langzeitstudie die Entwicklung des Lernzielerfolgs in Abhängigkeit von der Lehrstrategie untersuchen.

Literaturverzeichnis

[AK01] Anderson, L.; Krathwol, D.: A taxonomy of Learning, Teaching, and Assessing, Addison Wesley Longman 2001

[Bl56] Bloom, B.S. (Ed.), Engelhardt, M.D., Furst, E.J., Hill, W.H., & Krathwol, D.R. (1956). Taxonomy of educational objectives: Handbook I: Cognitive domain. New York: David McKay.

[Ba98] Baumann, R.: Fundamentale Ideen der Informatik – gibt es das?“; in „Informatische Bildung in Deutschland, Perspektiven für das 21 Jahrhundert“, Koerber, B. und Peters I.

[Br60] Bruner, J.S.: "The process of education", Cambridge Mass. 1960 (dt. Übers.: "Der Prozeß der Erziehung", Berlin 1970)

[De99] Denning, P.:“Computer Science: The Discipline,” Encyclopedia of Computer Science, A. Ralston and D. Hemmendinger, eds., Nature Publishing Group, 2000, pp. 405-419.

[De03] Denning, P.: “Great Principles of Computing,” Comm. ACM, Nov. 2003, pp. 15-20

[Ra07] RWTH Aachen; Modulhandbuch als Anlage zur Prüfungsordnung des Bachelorstudiengangs Informatik; (p. 6); geprüft am 24.1.2007; http://www.informatik.rwth-aachen.de/Studierende/Bachelor+Master/Studium/Bachelor-Master-Informatik-Modulhandbuch2006-10-17.pdf

[Sc93] Schwill, A.: Fundamentale Ideen der Informatik. In: Zeitschrift für Didaktik der Mathematik, 1993/1

[Sc06a] Schneider, M.: Begriffliche Strukturen der Informatik: Ein empirischer Zugang, 3. Workshop der GI-Fachgruppe "Didaktik der Informatik", Juni 2006, Potsdam

[Sc06b] Schneider, M.: Functional Modelling, Fundamental Ideas and Threads in the Subject Informatics. In (Dagiene, V.; Mittermeir, R.): Proceedings of the Second International Conference „Informatics in Secondary Schools: Evolution and Perspectives“; Vilnius 2006; pp. 413 – 423

[St06] Staller,A.: Merging domain knowledge and task analysis in an ontology, in A. Méndez-Vilas, A. Solano Martin, J. Mesa González & J.A. Mesa González (Eds.), Current Developments in Technology-Assisted Education (Vol. 3), pp. 1585-1589, 2006

[Tm07] Technische Universität München; Modulbeschreibung der Vorlesung zur „Einführung in die Informatik 1“, geprüft am 24.1.2007: http://www.in.tum.de/studium/cm.html?id=IN0001

[Tb07] Technische Universität Braunschweig; Beschreibung des konsekutiven Bachelor-/Masterstudiengangs Informatik (p. 60), geprüft am 24.1.2007: http://wiki.cs.tu-bs.de/images/e/eb/MHB-3voll.pdf

AtoCC – didaktischer Ort und erste Erfahrungen

Michael Hielscher, Christian Wagenknecht

Fachbereich Informatik
Hochschule Zittau/Görlitz (FH)
Brückenstraße 1
D-02826 Görlitz
mail@atocc.de
c.wagenknecht@hs-zigr.de

Abstract: AtoCC (http://www.atocc.de) ist eine modular aufgebaute Lehr-/ Lernumgebung für ausgewählte Inhalte der theoretischen Informatik (formale Sprachen und Automaten) und Grundlagen des Compilerbaus. Sie unterstützt Schülerinnen und Schüler bei der Anwendung von Theorie-Kenntnissen in einem Projekt-bezogenen Umfeld (Compilerkonstruktion). Das inzwischen etablierte Werkzeug AutoEdit [HW05, Wi06] wurde dafür mit weiteren Softwarekomponenten angereichert, um ein kompaktes System für den Unterricht zu schaffen, das durch abgestimmte Handhabung den Blick auf die eigentlichen Lerninhalte richtet.

1 Einleitung

Sowohl an Universitäten und Hochschulen als auch an Gymnasien[1] werden Kerninhalte der theoretischen Informatik gelehrt. Die im hessischen Lehrplan festgelegten Inhalte kommen einem Theorie-Grundkurs eines Informatik-Studiums recht nahe, vgl. [Sc92].

Begriffe und Methoden der theoretischen Informatik sind sehr abstrakt und stellen gegenüber eher ingenieurwissenschaftlichen Arbeitsformen anderer Teilgebiete der Informatik eine besondere didaktische Herausforderung dar. Von daher ist die Entwicklung geeigneter Lehr/Lernumgebungen besonders motiviert. So gibt es zur Unterstützung der Behandlung von Algorithmen entsprechende Visualisierungssoftware, wie [RAK06] und [Rö05], die ggf. durch Parametervariationen einen what-if-Arbeitsstil ermöglicht. Im Allg. ist Software dieser Art jedoch auf einen eng begrenzten Schwerpunkt beschränkt und bietet keinen Raum für Konstruktion neuen Wissens und Kontext-Erweiterung. Außerdem ist die auf das Lernen gerichtete Aktivität bei Visualisierungen relativ gering.

[1] Themen, wie formale Sprachen, formale Grammatiken, Automatentheorie und Übersetzerbau, sind beispielsweise im hessischen Lehrplan der Jahrgangsstufe 13 im Pflichtteil von Grund- und Leistungskurs „Konzepte und Anwendungen der Theoretischen Informatik“ [LIH] und auch in den Lehrplänen anderer Bundesländer zu finden.

Im Bereich der Automatentheorie findet man eine Vielzahl von Applets und Simulationstools, wie JFLAP [RF06] oder Kara [RNH04]. Eine Abgrenzung von AtoCC gegenüber dem bekannten Werkzeug JFLAP findet sich in [HW05].

Automatentheorie wird im Übersetzerbau angewendet. Neben dem fachsystematischen Aspekt ist dies aus didaktischer Sicht (Verstehen abstrakter Inhalte durch Anwendung/Instanziierung) von besonderer Bedeutung. Hierfür werden gern einfache Scanner und Parser für überschaubare (LL(1)-)Sprachen in einer bereits bekannten Programmiersprache implementiert. Im Informatiklehrplan des Saarlands wird beispielsweise auf das Verfahren des rekursiven Abstiegs zur Syntaxanalyse Bezug genommen [LISL].

Zur konzeptionellen vs. informationstechnischen Behandlung des Übersetzerbaus ist ein Top-Down-Vorgehen angezeigt. Dies entspricht genau dem, was man unter automatisierter Compiler-Entwicklung (kurz: compiler construction – das CC in AtoCC) versteht. Statt ein Übersetzungsprogramm in einer bestimmten Programmiersprache zu schreiben, werden aus der Grammatik-Definition (sprachliche) Beschreibungen für einen Übersetzer-Generator gewonnen. Die hierfür in der Berufspraxis verwendeten Werkzeuge Lex und Yacc[2] sind für didaktische Zwecke ungeeignet, da sie gegenüber den zugrunde liegenden Automaten-Modellen einen Paradigmenbruch hinnehmen und auf Performance angelegtes technisches Beiwerk erfordern. Eine Ableitung der Compiler-Beschreibung aus einem vorher entwickelten Automatenmodell ist in Verbindung mit diesen Werkzeugen nicht möglich. AtoCC unterstützt die Entwicklungskette vom Automat zum Compiler in einem modular strukturierten System mit vereinheitlichter Bedienung.

Die im Compilerbau auftretenden Prozesse werden gern durch T-Diagramme beschrieben und visualisiert. Schon auf Papier eignen sich T-Diagramme hervorragend, um Konzepte wie Bootstrapping, mehrstufige und Cross-Compiler zu beschreiben. Allerdings finden Papierdarstellungen keine Fortsetzung in der tatsächlichen Compilerentwicklung. Diesen Medienbruch (Papier – Softwaresystem) überwindet die TDiag-Komponente von AtoCC. Dabei wird nicht nur die grafische Darstellung nach Baustein-artigem Zusammensetzen einzelner T-Diagramme von der Software übernommen, sondern darüber hinaus findet eine Prüfung der Passfähigkeit benachbarter Bausteine statt. Auf diese Weise entstehen also stets korrekte Übersetzungsketten. Hiermit wird selbstgesteuerter Wissenserwerb durch zielgerichtetes Explorieren / Konstruieren tutoriell begleitet.

2 **AtoCC –** „Vom **A**utomaten zum **C**ompiler **C**ompiler“

Bisherige Beobachtungen zeigen, dass die Verwendung von AtoCC in der Hand von Lehrenden und Schüler/innen eine sehr kurze Einarbeitungszeit erfordert. Dies ist auf die einheitliche Benutzungsoberfläche aller Systemkomponenten zurückzuführen und war von Beginn an ein Entwicklungsziel. Integrierte AtoCC-Assistenten mit einfachen Einführungsbeispielen tragen zusätzlich dazu bei.

[2] Lex ist als Scannergenerator und Yacc als Parsergenerator auch unter den Weiterentwicklungen Flex und Bison bekannt, s. auch http://dinosaur.compilertools.net/.

Im Gegensatz zu zersplitterten Systemarchitekturen stellt die auf einander abgestimmte Weiterverwendbarkeit von Aufgabenlösungen (z.B.: Generierung eines Compilers aus einem Automaten oder Generierung eines T-Diagramms für einen erstellen Compiler) in zuständigen System-Komponenten einen wichtigen konzeptionellen Schwerpunkt von AtoCC dar. Das Zusammenspiel aller Komponenten im Hinblick auf einen einzigen programmiersprachlichen Träger[3] beschleunigt die „Produktherstellung“ (Compiler) und reduziert den in anderen Systemen beklagten Ballast. AtoCC sollte aber keineswegs als eine einfache, thematisch abgestimmte Tool-Sammlung verstanden werden. Durch vielseitige Beziehungen zwischen den Komponenten entsteht ein komplexes System, s. Abb. 1. und [HW06].

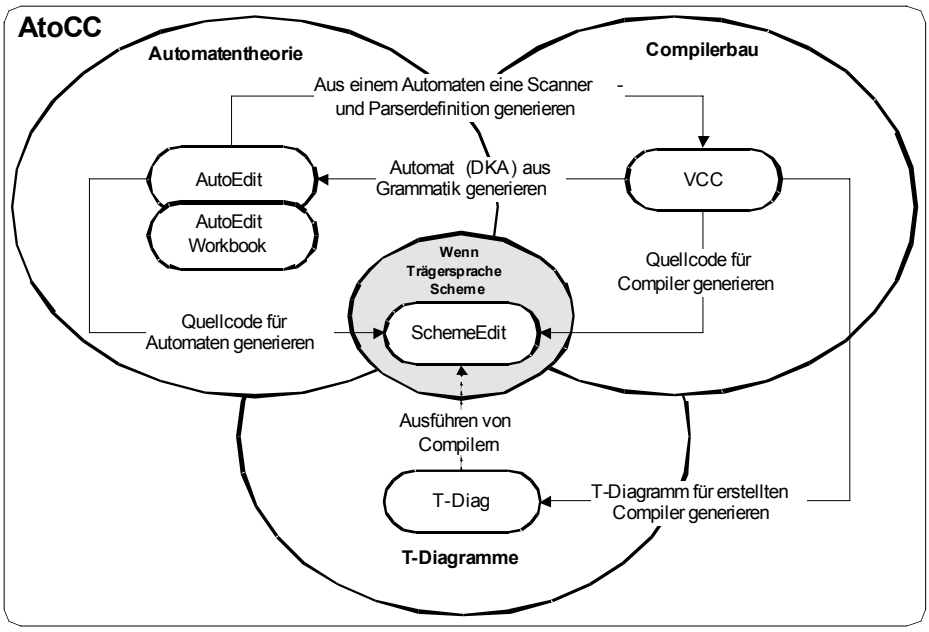


Abbildung 1: AtoCC-Komponenten und ihre Wechselwirkungen

Folgende Grundaufgaben können mit AtoCC bearbeitet werden:

- Darstellung, Simulation und Transformation von Automaten (deterministische und nichtdeterministische endliche Automaten – DEA, NEA; deterministische und nichtdeterministische Kellerautomaten – DKA, NKA; Turingmaschinen; Mealy- und Moore-Automaten)
- Automatisierte Konstruktion von Compilern (Scanner, Parser, Codegenerator)
- Konstruktion von T-Diagrammen zur konzeptionellen Planung von Übersetzungsprozessen und deren Visualisierung (Modell-getriebener Ansatz)

[3] Hierfür können in AtoCC derzeit die Programmiersprachen Scheme, Java und C# .NET gewählt werden.

Die dazu im Allg. im Verbund eingesetzten Module werden im Folgenden zusammen mit einer kurzen didaktischen Ortsbestimmung vorgestellt.

2.1 AutoEdit und AutoEdit Workbook

An der Hochschule Zittau/Görlitz wird im Fach „Theoretische Informatik“ bereits seit Jahren AutoEdit als Darstellungs-, Simulations- und Transformationswerkzeug für Automaten in Seminaraufgaben eingesetzt. Durch viele Vorschläge von Anwendern im Schul- und Hochschulbereich konnte die didaktische Qualität und der Leistungsumfang des Werkzeugs maßgeblich gesteigert werden. Mit dem Lehrbuch [Wa06] wird AutoEdit auch erstmals in entsprechenden Unterrichtsinhalten verankert.

AutoEdit Workbook, als Erweiterung zu AutoEdit, bietet Übungsaufgaben zur selbstständigen Bearbeitung und anschließenden automatischen Bewertung an. Dieses Konzept für Übungsaufgaben ist den Projekten SchemeGrader [Wa05] und Exorciser [TLN02] entlehnt, bei denen ebenfalls eine direkte Überprüfung einer Aufgabenlösung erfolgt. Eine Webdatenbank stellt diese Übungsaufgaben allen Lehrenden und Lernenden zur Verfügung[4]. Die jeweils aktuelle Aufgabenbasis kann von Lehrer/innen mit Hilfe eines integrierten Assistenten jederzeit erweitert werden. Auch Schüler/innen können für gute Ideen durch deren Eintrag ins Workbook gewürdigt werden. Die Aufgabenstellungen können mit Hinweisen (Notizzettel), Musterlösungen und bestimmten Vorgaben (Hinweise, Teillösungen, die zu vervollständigen sind) versehen werden.

2.2 VCC – Visual Compiler Compiler

VCC hilft bei einer Projekt-basierten Vermittlung ausgewählter Konzepte des Compilerbaus. Dabei steht weniger die Effizienz des Produkts, sondern vielmehr der didaktische Zugang im Vordergrund: Schüler/innen können mit VCC einen Compiler nach dem Baukastenprinzip visuell erstellen. Definierte Bausteine, wie die für die Tokens des Scanners, dienen als Grundlage für die spätere Konstruktion des Parsers. Das von AutoEdit bekannte Prinzip Software-geleiteter Arbeitsschritte wird auch hier eingesetzt.

Die visuelle Konstruktionstechnik vermeidet syntaktische und Schreibfehler weitestgehend. Da VCC intern auf einer angepassten Version von YACC beruht, ist sowohl inhaltlich (Tokendefinitionen, S-Attribut-Grammatiken) als auch technisch (Kellerautomat) ein mit Lex und Yacc vergleichbares Vorgehen garantiert. VCC vereint Scanner-, Parser- und Codegenerator in einem einzigen Werkzeug und legt erstellte Compilerdefinitionen in einem vielseitig verwendbaren XML-Format ab.

2.3 TDiag – T-Diagramme

T-Diagramme sind geeignet, um Compilerprozesse zu planen (Entwurfsperspektive) und zu visualisieren (Verifikationsperspektive). Aus den vier Grundbausteinen Compiler,

[4] Der Aufgabenautor kann ggf. festlegen, dass die jeweilige Aufgabe nur über ein Passwort zugänglich ist.

Interpreter, Ein-/Ausgabe und Programm lassen sich nahezu alle anfallenden Prozesse modellieren[5].

TDiag bietet ein einfaches Interface, um diese Diagramme aus einzelnen Bausteinen zu konstruieren. Dem Lernenden wird durch entsprechende Einfärbungen der grafischen Bausteine angezeigt, ob und wie einzelne Komponenten zusammengesetzt werden können. Auf diese Weise entstehen stets korrekte Prozessmodelle.

Um komplexe Kompilationsprozesse abzuarbeiten werden Stapelverarbeitungsdateien[6] verwendet. In TDiag kann ein entworfenes Diagramm mit einem einzigen Klick in eine Batch-Datei, die dann automatisch abgearbeitet wird, übersetzt werden. Die Schüler/innen können somit direkt überprüfen, inwieweit das entwickelte Diagramm dem gewünschten Prozess entspricht. Um dies zu ermöglichen, bietet jeder Baustein zusätzliche Runtime-Attribute, die im Diagramm nicht direkt sichtbar sind.

3 Ein kommentiertes Unterrichtsbeispiel

Das folgende Beispiel wurde einer realen Unterrichtssituation entnommen. Es handelt sich keineswegs um ein Einführungsbeispiel, sondern ist von mittlerem Schwierigkeitsgrad und illustriert die medien- und paradigmenbruchfreie Entwicklung eines Compilers innerhalb von AtoCC beginnend beim Automatenmodell.

Aufgabenstellung:

> Entwickeln Sie einen DEA[7], der alle Wörter akzeptiert, die eine durch 4 teilbare[8], natürliche Zahl repräsentieren.
>
> "182342340" wird akzeptiert, da 40 durch 4 teilbar ist.
>
> "234523173" wird nicht akzeptiert, da 73 nicht durch 4 teilbar ist.
>
> Wählen Sie zunächst ein passendes Eingabealphabet Σ.
>
> Um die Funktionsweise Ihres Automaten zu überprüfen, generieren Sie 10 Zufallswörter über Σ unter „Mehrere Eingaben prüfen“ (auf der Simulationsseite).

Eine mögliche Schülerlösung, die mit AutoEdit erarbeitet wurde, zeigt Abb. 2. Vom Schüler wird allerdings die vollständige Tupel-Definition des Automaten gefordert.

Der Entwurfsprozess des Automaten wird vom AtoCC-Programm in bestimmten Arbeitsschritten gesteuert („weiter“-Knopf – unten rechts): Hiernach beginnt die Definition eines Automaten stets mit der Festlegung des Eingabealphabets usw.

---

[5] Eine Ausnahme bildet der Compiler Compiler, für den bislang kein entsprechender Baustein definiert wurde.
[6] Unter UNIX/Linux verwendet man Shell-Scripts um Folgen von Befehlen automatisiert abarbeiten zu lassen. Vergleichbar erfüllen MS-DOS-Batchdateien unter Microsoft Betriebsystemen diese Aufgabe.
[7] Deterministischer endlicher Automat
[8] Hinweis: Es gilt, dass alle mehrstelligen Zahlen durch 4 teilbar sind, wenn die Zahl aus den letzten beiden Stellen durch 4 teilbar ist.

Nach Fertigstellung des Automatenentwurfs verwenden die Schüler den Simulationsmodus: Neben der freien Eingabe eines Eingabewortes und dessen automatischer bzw. Einzelschritt-Verarbeitung (visualisierte Durchwanderung der betreffenden Knoten des Graphen) hat sich die Verarbeitung generierbarer Zufallswörter wählbarer Länge als sehr nützlich erwiesen. Auf diese Weise kann die Verifikation des entwickelten Automaten wesentlich befördert werden.

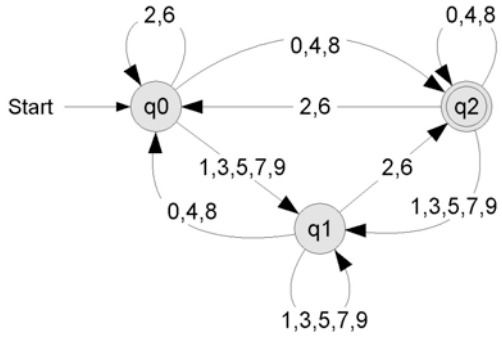


Abbildung 2: DEA für alle durch 4 teilbare Zahlen

Handelt es sich um eine Aufgabe aus dem Workbook, so kann eine Musterlösung hinterlegt werden. Diese dient der Selbstkontrolle der erarbeiteten Lösung durch die Schüler. Ein Notizzettel kann Lösungshinweise zur Aufgabe bereitstellen. Für schwierigere Aufgaben werden Lösungsrudimente in Gestalt einer mehr oder weniger ausgearbeiteten Vorgabelösung zur Verfügung gestellt. Auf diese Weise ist eine Binnendifferenzierung erreichbar. Auch für Hausaufgaben ist dies eine wertvolle Hilfe.

Bei der Darstellung der vollständigen Automatendefinition wird auch eine zugehörige formale Grammatik angegeben. Dies bietet eine zusätzliche Lernmöglichkeit.

```
G = (N,T,P,s)
G = ({q0,q2,q1},{0,1,2,3,4,5,6,7,8,9},P,q0)
P = {
   q0 -> 0q2 | 0 | 4q2 | 4 | 8q2 | 8 | 1q1 | 3q1 | 5q1 | 7q1 | 9q1 | 2q0 | 6q0
   q2 -> 0q2 | 0 | 4q2 | 4 | 8q2 | 8 | 1q1 | 3q1 | 5q1 | 7q1 | 9q1 | 2q0 | 6q0
   q1 -> 0q0 | 4q0 | 8q0 | 1q1 | 3q1 | 5q1 | 7q1 | 9q1 | 2q2 | 2 | 6q2 | 6
}
```

Der betrachtete Automat kann von AutoEdit automatisch in eine Scanner- und eine Parserdefinition für VCC transformiert werden. Dabei werden die Terminale der zugehörigen Grammatik (s. o.) in Tokens für den Scanner überführt. Die Produktionsregeln dieser Grammatik werden für den Parser verwendet. Die generierten Definitionen für Scanner und Parser zeigt Abb. 3.

Sowohl Automat, Grammatik als auch Parser- in Kombination mit der Scanner-Definition sind äquivalente Beschreibungsmittel für die in der Aufgabenstellung beschriebene Sprache.

Der Parser liefert einen Syntaxfehler, wenn das aktuelle Eingabewort eine Zahl repräsentiert, die nicht durch 4 teilbar ist. Der Parser realisiert damit das Verhalten eines Akzeptors. Je nach gewünschter Zielsprache ist es möglich, mit AtoCC einen vollwertigen Compiler zu generieren. Der Compiler liegt dann als ausführbares Programm vor[9].

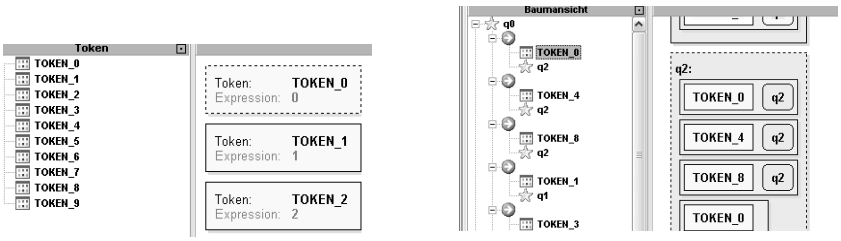


Abbildung 3: VCC generierter Scanner und Parser

Nachfolgend wird ein Aufrufbeispiel in Scheme sowohl für den Scanner als auch für die Verkettung von Scanner und Parser angegeben. Der Scanner liefert dabei eine Liste von Token-Value-Paaren. Der Parser gibt entweder „wahr“ oder „falsch“ zurück und folgt damit dem oben angesprochenen Akzeptor-Verhalten:

```
> (scanner "1234")
((token_1 . "1") (token_2 . "2") (token_3 . "3") (token_4 . "4"))
> (parser (scanner "12348082345678876544564"))
#t
> (parser (scanner "123480823456788765445642321"))
Syntax Error
#f
```

Ein Aufruf der Form (compiler "Eingabedatei.txt" "Ausgabedatei.txt") wird verwendet, um die gesamte Übersetzung (mit Zielcodegenerierung) durchzuführen.

Jede einzelne Regel in VCC kann hierfür mit einem ausführbaren Code-Schnipsel verbunden werden (S-Attributgrammatik). Diese werden während des Parsens bei erfolgreicher Regelanwendung ausgeführt. Um dies zu demonstrieren wird in unserem Beispiel ein neues Spitzensymbol „Start“ angelegt. Die Regel Start→q0 wird mit der Ausgabe "Die Zahl " $1 " ist durch 4 teilbar." belegt, s. Abb. 4.

In die Datei „eingabe.txt“ wird die Zeichenkette "12344" geschrieben und im Compilerverzeichnis abgelegt. Durch den nachfolgenden Aufruf wird der Eingabetext in eine sehr einfache Zielsprache übersetzt:

```
> (compiler "eingabe.txt" "ausgabe.txt")
#t
```

[9] Wahlweise handelt es sich um ein Scheme-, Java oder C#-Programm. Detaillierte Kenntnisse dieser Sprachen sind nicht erforderlich. Lediglich die jeweilige Ausführungsumgebung muss (einmalig) bereitgestellt werden.

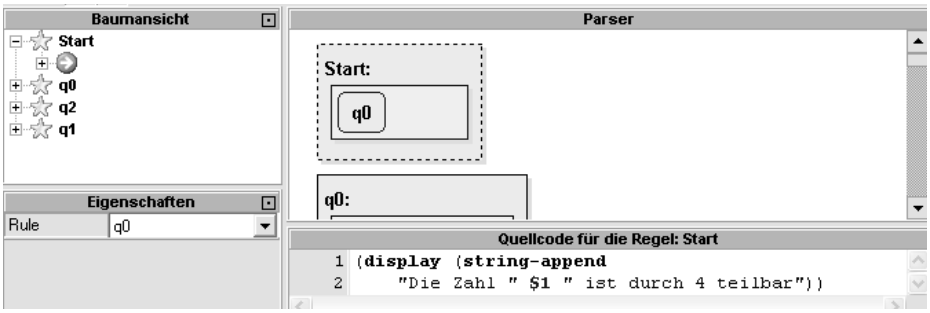


Abbildung 4: S-Attribute in VCC

In der Datei „ausgabe.txt“ steht nun der Inhalt: „Die Zahl 12344 ist durch 4 teilbar.“ Hierfür wurden Platzhalter wie $n für einzelne Regelelemente verwendet.

Für einen erstellten Compiler kann jederzeit per Mausklick ein zugehöriges T-Diagramm generiert werden (Abb. 5).

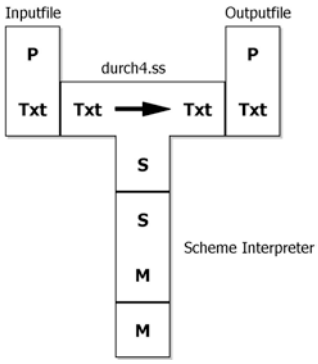


Abbildung 5: T-Diagramm für Beispiel-Compiler

Die (vom System generierte) Beschreibung des Übersetzungskonzepts durch ein T-Diagramm ist bei Wieder- bzw. Weiterverwendung „abgelegter“ Lösungen durchaus sinnvoll: Hier sollen sich Schülerinnen und Schüler am Modell orientieren können.

Der typische Arbeitsstil verläuft jedoch umgekehrt, nämlich als Modellierungsaufgabe der Architektur des zu entwickelnden Übersetzers, wie z. B. Bootstrapping, mehrstufige und Cross-Compiler. Hierfür bieten T-Diagramme eine hervorragende Entwurfsunterstützung und machen konkrete Lehrplanziele [LIH] erreichbar.

In Abb. 6 ist ein T-Diagramm für einen mehrstufigen Compilerprozess dargestellt. Eine Sprache MyLang wird zunächst von einem zu entwickelten Compiler in die Sprache TASM (Turbo Assembler) übersetzt, anschließend von TASM kompiliert (Bildmitte) und von TLINK in ein lauffähiges Programm überführt. Die Programme TASM.exe und TLINK.exe werden zur Bearbeitung der Aufgabe zur Verfügung gestellt. Die eigentliche

Aufgabe besteht also im Entwurf von MyLang→TASM. Der Rest besteht aus technischem Beiwerk, was fast vollständig von AtoCC übernommen wird.

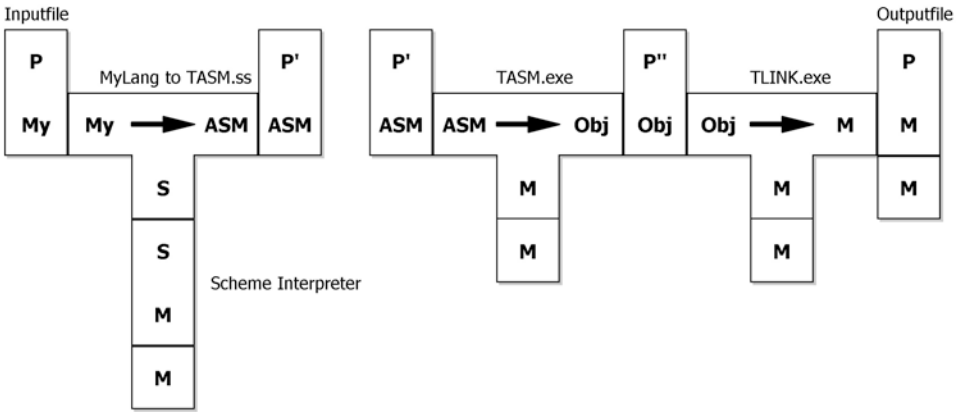


Abbildung 6: T-Diagramm für MyLang → TASM Compiler

Schließlich wird eine Stapelverarbeitungsdatei generiert, die die einzelnen Übersetzungsschritte in einem Aufruf zusammenfasst. TDiag liefert die Ausgabe wie in Abb. 7.

```
;Starting batch file that will execute the diagram.
;=================================================
Starting: petite --script tasm.ss quelltext.txt out.asm
...

Done with: petite --script tasm.ss quelltext.txt out.asm
;=================================================
Starting: tasm.exe out.asm out.obj
...
Turbo Assembler  Version 2.0  Copyright (c) 1988, 1990 Borland International

Assembling file:   out.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  433k


Done with: tasm.exe out.asm out.obj
;=================================================
Starting: tlink.exe out.obj
...
Turbo Link  Version 3.01 Copyright (c) 1987, 1990 Borland International

Done with: tlink.exe out.obj
;=================================================
Starting: out.exe
...
**********************
 Zähler
```

Abbildung 7: Ausgabe bei „Ausführung des T-Diagramms“

Die generierte Stapelverarbeitungsdatei kann unabhängig von AtoCC beliebig oft verwendet werden.

4 Erfahrungen im praktischen Einsatz von AtoCC

AtoCC hat die zweite Entwicklungsphase abgeschlossen. Während in der ersten Phase der System-Entwurf auf der Basis von Lehrerfahrungen im Hochschulbereich im Mittelpunkt stand, widmete sich die zweite Phase der Systemmodifikation in Reaktion auf entsprechende Rückkopplungen von Lehrenden an Hochschulen und Gymnasien sowie von Studierenden. Die hier berichteten Erfahrungen stammen aus dieser ca. zweijährigen Phase und haben in den meisten Fällen zu didaktisch motivierten Systemverbesserungen bzw. -erweiterungen beigetragen. Einige Beispiele:

- eine an typischen IDEs und den taktilen Gewohnheiten orientierte Maussteuerung kombiniert mit der Eingabemöglichkeit textueller Attribute beim Automaten- und T-Diagramm-Entwurf
- Ergänzung eines Notizblocks, um einem Automaten auch einfache Zusatzinformationen anheften zu können
- Modifikation der Simulationskomponente für nichtdeterministische Automaten
- Sequentielle Simulation mehrerer Zufallswörter wählbarer Länge → Testunterstützung und Vorgabemöglichkeit von Eingabewortlisten für Automaten
- Aufnahme von Automaten mit Ausgabe – Mealy und Moore (s. [LIH])
- Ausbau der Aufgabensammlung (Workbook-Komponente) mit erprobten Beiträgen von Lehrerinnen und Lehrern
- Konsolidierung der Lernendenführung bei der schrittweisen Entwicklung eines Automaten bzw. eines Übersetzers

In der nächsten Phase stehen die didaktisch-methodischen Aspekte im Vordergrund. Hierzu ist es notwendig, AtoCC weiter bekannt zu machen und an der Einführung mitwirkende Lehrende zu gewinnen.

Wir schon weiter oben festgestellt, fordert der hessische Lehrplan [LIH] in der 13. Jahrgangsstufe eine sehr breite inhaltliche Palette dieser Themen. Aufgrund der dort geforderten ‚Simulation realer Automaten, wie etwas Getränkeautomaten', müssen neben den bekannten Akzeptoren sogar Automaten mit Ausgabe (Mealy, Moore) thematisiert werden. Auch dies ist inzwischen mit AtoCC umsetzbar.

Zur Sicherung einer Lehrplan-konformen Wissensvermittlung und Kompetenzentwicklung kann AtoCC eine enorme Hilfe sein. Bisher empfohlenen Systemen ist AtoCC im Allg. überlegen, siehe z. B. die „Simulationsumgebung" in [LIH]. Dies untermauern zahlreiche positive Rückmeldungen, wie: „Von dem Programmpaket AtoCC kenne ich bisher nur die AutoEdit Komponente. Ich habe damit im vergangenen Schuljahr das erste Mal gearbeitet. Das Programm hat mir sehr gut gefallen und ich möchte weiter damit arbeiten.

Das Programm ist sehr bedienerfreundlich und veranschaulicht die Inhalte der in Frage stehenden Themen hervorragend. Durch die geringen Systemanforderungen und automa-

tisierte Installation der Software (von nur wenigen ca. 1.5 MB) war es auch Schülerinnen und Schülern mit älteren Systemen (wie Win98, WinME) möglich, AtoCC für Hausaufgaben zu verwenden.“ Nach Aussagen befragter Schüler/innen ist die Bedienung von AtoCC intuitiv. Sowohl Schüler/innen als auch Lehrende überzeugte dabei besonders die visuell und konzeptionell einheitliche Gestaltung der einzelnen Module, die nahtlos „zugeschaltet“ werden können. Die Modularisierung von AtoCC erlaubt eine gezielte Auswahl einzelner Werkzeuge zur Behandlung unabhängiger Schwerpunkte.

Andererseits wurde auch deutlich, dass LehrerInnen aus verschiedenen Gründen nicht in der Lage sind, die didaktischen Möglichkeiten des Systems im Selbststudium vollständig zu erschließen. Zur Unterstützung gibt es eine Plattform, die Tutorials sowie Aufgaben mit Musterlösungen bereithält. Darüber hinaus wird die Durchführung von Workshops im Rahmen geeigneter Fortbildungsmaßnahmen angestrebt. Hat man erst einmal die richtige Erwartungshaltung aufgebaut, ist die Einarbeitungszeit sehr kurz.

Die Betrachtung von Übersetzungsprozessen ist für das Verständnis der Verarbeitung höherer Programmiersprachen unentbehrlich. Dies gilt nicht nur für das Informatik-Studium, sondern ebenso für den Pflichtbereich des Informatik-Unterrichts in der gymnasialen Oberstufe natürlich mit angemessener Betrachtungstiefe. In mehreren Bundesländern findet sich dieser Themenschwerpunkt im Wahlpflichtbereich, z. B. NRW (Siegen). Sind Lehrplanziele aus dem Übersetzerbau ohne AtoCC wirklich erfüllbar? Der Einsatz von VCC/TDiag würde auch dem hessischen Lehrplan eine echte Realisierungschance einräumen: Dort werden Delphi bzw. Java als Basissprachen für die Grundlagen der Programmierung empfohlen, Prolog für den Übersetzerbau. Da VCC optional Java-Code generiert, kann ein Paradigmenbruch (objekt-orientiert vs. logik-basiert) vermieden werden.

5 Ausblick

Die Systementwicklung ist grundsätzlich abgeschlossen, so dass die nun verstärkt einsetzende didaktische Erprobungsphase nicht durch immer neue Versionen beeinträchtigt wird. Dennoch arbeiten wir an internen Verbesserungen, wie beispielsweise an der Erzeugung minimaler regulärer Ausdrücke als Ergebnis der Transformation aus endlichen Automaten. Die theoretischen Grundlagen für eine geeignete Approximation in vertretbarer Zeit liegen inzwischen vor [GS05]. Unter Verwendung dieser minimierten regulären Ausdrücke kann auch die Transformation deterministischer in nichtdeterministische endliche Automaten deutlich verbessert werden.

Ab sofort soll AtoCC vor allem in Workshops, die sich an den entsprechenden Lehrplaninhalten orientieren, stärker bekannt gemacht werden. Dies gilt sowohl national als auch international, was durch die Herstellung einer englischen und einer polnischen Version vorbereitet wurde. Begleitet wird dies durch den Ausbau des Workbooks und Verbreiterung der AtoCC-Kommunikationsplattform, die bereits jetzt Tutorials und Handreichungen bereithält.

Literaturverzeichnis

[Wa05] Wagenknecht, Chr.: Mediendidaktische Begleitung im Informatikunterricht mit SchRepo, SchemeNet und SchemeGrader. In (Ed.: H. Rohland): Unterrichtskonzepte für informatische Bildung -Praxisband-. INFOS'05 28.-30.09.05 Dresden , Seite 21-24.

[Wa06] Wagenknecht, Chr.: Theoretische Informatik.- In (Engelmann, L. Hrsg.): Informatik: Lehrbuch S II - ISBN 3-89818-622-9, Berlin: DUDEN PAETEC, 2006.

[GS05] Gramlich, G.; Schnitger, G.: Minimizing NFA's and Regular Expressions. In (Diekert, Volker Hrsg.): STACS 2005, 22nd Annual Symposium on Theoretical Aspects of Computer Science Stuttgart, Germany 24.-26. February 2005). Bd. 3404. Heidelberg: Springer-Verlag, 2005, S. 399-411.

[HW06] Hielscher, M.; Wagenknecht, Chr.: AtoCC - Learning Environment for Teaching Theory of Automata and Formal Languages. ITiCSE'06, June 26-28, 2006, Bologna, Italy. ACM 1-59593-055-8/06/0006

[HW05] Hielscher, M.; Wagenknecht, Chr.: AutoEdit - ein Werkzeug zum Editieren, Simulieren, Transformieren und Publizieren abstrakter Automaten. In (Ed.: H. Rohland): Unterrichtskonzepte für informatische Bildung -Praxisband-. INFOS'05 28.-30.09.05 Dresden, Seite 25-27.

[LIH] Informatik-Lehrplan Hessen: http://lernarchiv.bildung.hessen.de/reposit2/12226/LPGymInformatik.pdf?user_id=Anonymous+User&is_viewable=1&digest=6fa80232cfce34daa0499dbfdf175e11

[LINR] Informatik-Lehrplan NRW: http://www.learn-line.nrw.de/angebote/abitur-gost-07/download/inf-vorgaben-2007.pdf

[LISH] Informatik-Lehrplan SH: http://lehrplan.lernnetz.de/intranet1/links/materials/1107165545.pdf

[LISL] Informatik-Lehrplan Saarland: http://www.saarland.de/dokumente/thema_bildung/INFeb2006.pdf

[RNH04] Reichert, R.; Nievergelt J.; Hartmann W.: Programmieren mit Kara. Ein spielerischer Zugang zur Informatik. 2. erweiterte Auflage, ISBN: 3540238190, Springer, Berlin Dezember 2004.

[RF06] Rodger, S.; Finley, T.: JFLAP - An Interactive Formal Languages and Automata Package, ISBN 0763738344, Jones and Bartlett, 2006

[Rö05] Rößling, G.: Visualisierung von dynamischen Systemen. thema Forschung. E-Learning 1/2005. pp. 78-82, TU Darmstadt, 2005. ISBN 1434-7768 (ISSN).

[RAK06] Rößling, G.; Ackermann, T.; Kulessa, S.: Visualisierung von Algorithmen und Datenstrukturen. – In (Mühlhäuser, Rößling, Steinmetz Hrsg.): DeLFI 2006 4. e-Learning Fachtagung Informatik 11. - 14. September 2006 in Darmstadt, Seite 231-242.

[Sc92] Schöning, Uwe: Theoretische Informatik kurz gefasst.- Mannheim u.a.: BI Wissenschaftsverlag, 1992.

[TLN02] Tscherter, V.; Lamprecht, R.; Nievergelt, J.: Exorciser: Automatic Generation and Interactive Grading of Exercises in the Theory of Computation. Proceedings of ICNEE 2002, Lugano, Switzerland, May 2002.

[Wi06] Wikipedia; http://de.wikipedia.org/wiki/AtoCC; 21.01.2007

Pragmatische Empfehlungen zur Entwicklung von interaktiven Lernumgebungen

Ruedi Arnold
Institut für Pervasive Computing
ETH Zürich
CH-8092 Zürich
rarnold@inf.ethz.ch

Werner Hartmann
Zentrum für Bildungsinformatik
PH Bern
CH-3012 Bern
werner.hartmann@phbern.ch

Abstract: Computergestützten interaktiven Lernumgebungen wird seit den 70er Jahren ein grosses Potential vorausgesagt. Die Realität sieht meist anders aus. Viele Lernprogramme beschränken sich auf „Drill & Practice“, sie sind kaum wirklich interaktiv. Andere sind trotz hohem Entwicklungsaufwand oft schon nach wenigen Jahren aus technischen Gründen gar nicht mehr betriebsfähig.
Eine nachhaltige Entwicklung interaktiver Lernsoftware muss gleichzeitig sowohl der Schulpraxis als auch den Regeln der Softwareentwicklung und dem Wissensstand der Lehr- und Lernforschung Rechnung tragen. Eine solche interdisziplinäre Vorgehensweise nimmt bewusst eine wissenschaftliche Unschärfe in Kauf und setzt sich damit der Kritik der Scientific Communities der einzelnen Fachbereiche aus.
Am Beispiel von InfoTraffic, einer neuen Sammlung von kleinen Lernumgebungen zur Logik, zu Warteschlangen und zu dynamischen Systemen, illustrieren wir diesen ingenieurwissenschaftlich geprägten Ansatz, der sich in der Praxis bewährt hat.

1 Interaktive Lernumgebungen – Erwartungen und Enttäuschungen

Mitte der 20er Jahre des vergangenen Jahrhunderts gingen die ersten Schulfunksendungen in den Äther. Die Erwartungen an das neue Medium waren gross. Patrik Wülser [Wü06], Journalist beim Schweizer Radio DRS schreibt:

> Vorträge, Sprachkurse und Beratungen für fast alle Lebenslagen prägten von Beginn an die Programme, wie beispielsweise ein Blick in die Sendewoche vom 5. bis 11. Februar 1927 zeigt. Radio Zürich bot Kinder-, Jugend-, Schüler- und Frauenstunden an, gab eine Englischlektion, liess den Vortragsdienst der Volkshochschule zweimal sprechen und veranstaltete einen bunten Strauss

von Vorträgen wie „Klassische und moderne Bildhauerei“, „Winterschnitt am Kernobst“, die „Physik des Mondes“ oder „Zürcher Verkehrsfragen“.

Begleitet waren die Schulfunksendungen durch Initiativen zur Ausrüstung von Schulen mit Empfangsgeräten, quasi „Schulen ans Radio“.

Abbildung 1: Schulen ans Radio. (Quelle: Archiv Radio DRS, Zürich.)

Mit dem Aufkommen des optischen Mediums Fernsehen verlor das Radio seine Faszination, Schulfernsehen war angesagt, der Schulfunk spielte nur noch eine marginale Rolle. Mit dem Aufkommen des Computers in den 60er Jahren erwuchs auch dem Schulfernsehen ernsthafte Konkurrenz, welche durch den Boom des Internet noch weiter verstärkt wurde. Lernsoftware, interaktive Lernumgebungen in Form web-basierter Applets bis hin zu Edutainment versprachen kostengünstiges, individualisiertes, orts- und zeitunabhängiges Lernen samt adaptivem Benutzerfeedback. Inzwischen ist klar, dass sich viele dieser Erwartungen nicht erfüllt haben.

Betrachtet man die Nutzung der drei Medien Radio, Fernsehen sowie Computer und Internet zu Unterrichtszwecken im Rückblick, fallen verschiedene Parallelen auf:

Mangelnde Interaktivität: Die Lernenden finden sich beim Einsatz aller drei Medien zu oft in der Rolle passiver Konsumenten wieder. Bei Lernsoftware muss die Interaktion über reine Navigation hinaus gehen. Lernende sollen aktive Kontrolle über den Inhalt, Ablauf und Aspekte der Präsentation besitzen. Brenda Laurel [La93] bringt es sehr schön auf den Punkt: „You either feel yourself to be participating in the ongoing action of the representation or you don't."

Interessant in diesem Zusammenhang ist, dass die Fernsehanstalten heute zwecks stärkerer Kundenbindung vermehrt auf interaktives Fernsehen setzen, etwa mit Video on Demand oder Teleshopping.

Hoher Entwicklungsaufwand, geringe Halbwertszeit: Bei diesen drei Medien ist der Aufwand für die Erstellung von Lehr- und Lernmaterialien sehr gross. Der Entwicklungsaufwand rechtfertigt sich deshalb nur bei langfristig bedeutungsvollen Themen und einem breiten Zielpublikum.

Für das Erstellen professioneller Radio- und Fernsehbeiträge reichen Amateurkenntnisse nicht aus. Das Gleiche gilt für computergestützte Lernumgebungen. Die Jugendlichen haben aufgrund ihres von dauerndem Mediengeriesel geprägten Lebensumfelds hohe Erwartungen und mit bescheidenen Mitteln erzeugtes Edutainment verkommt schnell zu „Edupainment".

Auch die Wartung und Aktualisierung von radio-, fernseh- oder computergestützten Unterrichtseinheiten ist anspruchsvoller als von klassischen gedruckten Lehrmitteln. Im Fall von computergestützten Lernumgebungen führen die kurzen Entwicklungszyklen bei Soft- und Hardware in schöner Regelmässigkeit dazu, dass Unterrichtseinheiten nach wenigen Jahren rein technisch nicht mehr betriebsfähig sind.

Hohe Anforderungen an die Infrastruktur: Papier und Schreibzeug, sowie Lehrmittel in Buchform können „just in time and anywhere" eingesetzt werden. Auch bei der Wandtafel handelt es sich um eine relativ günstige, mehrfach verwendbare und langlebige Unterrichtshilfe.

Im Unterschied dazu sind der Einsatz von Radio, Fernseher sowie Computer und Internet mit grossen Investitionen in die Infrastruktur verbunden. Für Lehrpersonen ist die Handhabung dieser Werkzeuge zudem nicht einfach und mit Fallstricken versehen. Bei computergestützten Lernumgebungen muss oft noch zusätzliche Software installiert werden.

Im Folgenden legen wir das Schwergewicht auf die Entwicklung computergestützter Lernumgebungen für den Informatikunterricht. Themen aus der Informatik sind naheliegende Unterrichtsgegenstände. Zum einen lassen sich viele informatische Sachverhalte einfach formalisieren und damit überhaupt erst für einen Computer fassbar machen, zum anderen sind Informatiklehrer auch mit der Entwicklung und dem Einsatz von Softwaresystemen vertraut.

Die Informatik hatte auch eine Vorreiterrolle beim Einsatz des Computers im Unterricht. Zu den wichtigen frühen Projekten zählen Seymour Papert's Aktivitäten rund um die Programmiersprache LOGO für Kinder oder John Kemeny and Thomas Kurtz's Program-

miersprache BASIC. Noch einen Schritt weiter ging in den 70er Jahren das System PLATO (Programmed Logic for Automated Teaching Operations), welches während rund zwei Jahrzehnten regelmässig bei Tausenden von Studierenden zum Einsatz gelangte.

Die teils euphorischen Berichte aus den 70er Jahren über das Potential von CAI (computer aided instruction) unterscheiden sich nur wenig von heutigen Einschätzungen. Jürg Nievergelt [Ni75], selbst im Projekt PLATO engagiert, warnte aber bereits 1975 vor übertriebenen Erwartungen, unter anderem:

> Restriction to a few fixed teaching strategies appeared to be unreasonable. Programmed instruction and drill in particular, with their rigid control of the dialog by the program, should yield to (or at least not exclude) modes where the user controls the dialog, such as inquiry and simulation. [...]
>
> Resources had been diluted into too many projects of insufficient size; CAI research and development should be carried out by sizable groups of system designers and authors.

Nievergelt schliesst mit folgenden Empfehlungen:

> I came to the conclusion that there is no systematic body of knowledge which is of relevance to such a task. I am afraid that this paper does not change this situation at all. The advice I might give to someone intent on building a computer-based instructional system could be summed up in a few phrases: get the best terminals you can pay for, good programmers, try everything out in actual instruction as soon as possible, and follow your nose.

2 Das Dilemma: Wissenschaftlichkeit oder Wirkung?

Man mag die vorangehenden, 30 Jahre zurückliegenden Empfehlungen von Nievergelt belächeln, sie widerspiegeln aber nur das Spannungsfeld zwischen einem wissenschaftlichen Anspruch und gleichzeitiger Wirksamkeit auf die Schulpraxis.

In den letzten Jahren entstand in der Schweiz eine ganze Reihe interaktiver Lernumgebungen für den Informatikunterricht. Der programmierbare Marienkäfer Kara [RNH04] zum Einstieg in die Programmierung ist das wohl prominenteste Beispiel. Weitere Beispiele sind Graphbench [BN04] (NP-Vollständigkeit und Graphen-Algorithmen) oder Soekia (eine didaktische Suchmaschine). Die Lernumgebungen sind frei und inklusive Begleitmaterialien auf dem Bildungsserver SwissEduc [Sw07] verfügbar.

Bei der Entwicklung dieser Umgebungen haben wir das Spannungsfeld zwischen sogenannter Wissenschaftlichkeit und Bildungsinnovation mehrfach erlebt. Aus Sicht der Lehr- und Lernforschung fehlt den Lernumgebungen ein wissenschaftlich begründeter Wirkungsnachweis. Die Komplexität interaktiver Lernumgebungen mit ihren vielen Variablen setzt aber einer methodisch abgestützten Evaluation enge Grenzen. Gabi Reinmann führt in [Re06] aus, wie Bildungsforscher zu fast zwanghafter Differenzierung und Kontrolle im methodischen Design von Studien neigen, um so zu besser verallgemeinerbaren

Aussagen zu kommen. Die Folge seien artifizielle Lernumgebungen, die für den Unterrichtsalltag bedeutungslos sind.

Aus der Sicht der Ingenieurwissenschaft Informatik fehlt bei schultauglichen Lernumgebungen ebenfalls der Forschungsaspekt. Das behandelte Thema wird durch das Curriculum der Schulen festgelegt, wodurch es sich naturgemäss nicht um ein aktuelles Forschungsgebiet handelt. Zudem gilt bei der Entwicklung von Lernumgebungen der Grundsatz „so einfach wie nur möglich“. Die Benutzerschnittstelle soll sich auf das Allernötigste beschränken, damit im Unterricht keine Zeit mit der Einarbeitung in die Systembedienung verloren geht. Auch die Softwarearchitektur soll schlank sein, damit die spätere Wartung der Lernumgebung im Rahmen der meist nur sehr beschränkt vorhandenen Ressourcen möglich bleibt.

Fassen wir das Dilemma zusammen: Wer Lernumgebungen entwickelt, die das Lehren und Lernen nachhaltig beeinflussen, wird fast zwangsmässig von den Erziehungswissenschaften als unwissenschaftlich abgestempelt. In den Augen der Informatik auf der anderen Seite fehlt solchen Lernumgebungen die Innovation. Wer funktionierende interaktive Lernumgebungen entwickelt, die im Unterricht auf breiter Basis und über einen längeren Zeitraum zum Einsatz kommen, begibt sich aus Sicht der Scientific Community also fast zwangsmässig ins Abseits.

3 InfoTraffic – Fallbeispiel interaktiver Lernumgebungen

Der Entwurf praxistauglicher computergestützter Lernumgebungen setzt unserer Ansicht und Erfahrung nach die interdisziplinäre Mitwirkung von drei verschiedenen Personengruppen voraus: Themen, Aufgabenstellungen und Einsatzszenarien von Lernumgebungen müssen von tätigen Lehrerinnen und Lehrern mit grosser Schulpraxis festgelegt werden. Das effiziente Erstellen von qualitativ hoch stehenden interaktiven Lernumgebungen mit vertretbaren Kostenaufwand und einer grossen Halbwertszeit ist eine Ingenieuraufgabe und gehört in die Verantwortung von Softwareentwicklern. Das Evaluieren der Wirksamkeit interaktiver Lernumgebungen ist eine schwierige Aufgabe und bedarf einer besonderen Methodik, welche nur die Bildungsforschung leisten kann.

Ist in einem Entwicklungsteam eine der obigen Personengruppen nicht vertreten, sind Fehlentwicklungen vorprogrammiert. In den letzten zehn Jahren wurden immer wieder mit grossem Aufwand E-Learning Systeme entwickelt, für die es in der Schulpraxis gar keinen Bedarf gab. Wurden Schulpraktiker oder Erziehungswissenschaftler selbst als Programmierer tätig, bestand das Ergebnis nicht selten in benutzerunfreundlichen und technisch nur bedingt lauffähigen Programmen.

Im Folgenden beschreiben wir anhand von InfoTraffic [ALH07], [AH07] unseren pragmatischen Ansatz beim Erstellen interaktiver Lernumgebungen für die Informatik. InfoTraffic besteht aus drei Programmen zu den Themen Aussagenlogik (LogicTraffic), Warteschlangentheorie (QueueTraffic) und dynamische Systeme (DynaTraffic) und ist inklusive Unterrichtsmaterialien online unter [Sw07] frei verfügbar.

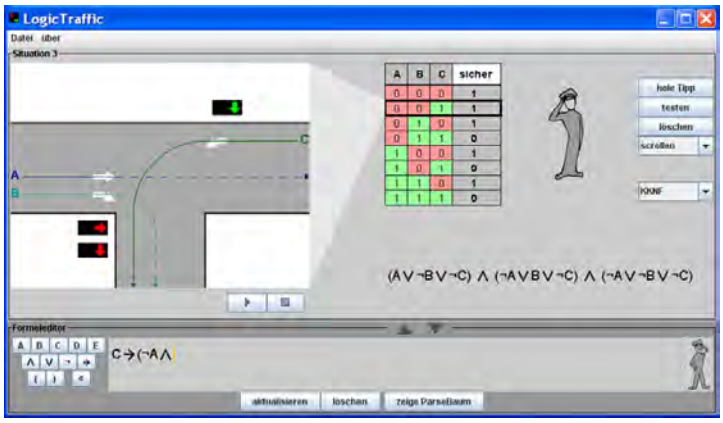


Abbildung 2: Lernumgebung LogicTraffic.

3.1 LogicTraffic – Aussagenlogik und Sicherheit bei Strassenkreuzungen

LogicTraffic bietet eine alltagsnahe Einführung in die Aussagenlogik, z.B. Boole'sche Operatoren, Wahrheitstabellen und Normalformen. Die Grundidee besteht darin, mittels aussagenlogischer Formeln die Ampeln einer Strassenkreuzung so zu steuern, dass zwischen einzelnen Spuren keine Kollisionen auftreten können. Aussagenlogik wird nicht abstrakt und formal als Formelsprache eingeführt, sondern spielerisch anhand einer Situation aus dem Alltag.

Die Programmoberfläche von LogicTraffic (Abbildung 2) besteht im Wesentlichen aus der Darstellung einer Strassenkreuzungen, einer Wahrheitstabelle, der Formel zur Wahrheitstabelle, einem Formeleditor und Steuerelementen.

3.2 QueueTraffic – Warteschlangen bei Strassenkreuzungen

QueueTraffic führt am Beispiel der Verkehrssteuerung wichtige Grundbegriffe der Warteschlangentheorie wie Ankunftsrate, Durchsatz oder Verteilungen ein. Parameter wie das Verkehrsaufkommen oder die Länge von Grünphasen können verändert werden und die Auswirkungen lassen sich direkt in der Simulation visuell und statistisch beobachten und analysieren.

Die Programmoberfläche von QueueTraffic (Abbildung 3) besteht im Wesentlichen aus der Darstellung einer Strassenkreuzung, einer Anzeige für Daten und Statistiken, sowie je einem Bereich zur Kontrolle von Verkehr, Verkehrsaufkommen und Simulation.

Abbildung 3: Lernumgebung QueueTraffic.

3.3 DynaTraffic – Dynamische Systeme und Verkehrsaufkommen

DynaTraffic simuliert das Verkehrsaufkommen in einem Strassensystem mit mehreren Kreuzungen. Im Zentrum stehen Themen wie stationärer Zustand, Adjazenzmatrix, Markovkette oder Kantengraph.

DynaTraffic ist noch in Entwicklung, Abbildung 4 zeigt einen Entwurf der geplanten Programmoberfläche, die fünf Bereiche beinhaltet: Darstellung eines Strassensystems, Markovmodell der Situation als Graph und als Übergangsmatrix, Visualisierung der aktuellen Verkehrsverteilung und Simulationskontrolle.

4 Pragmatische Empfehlungen bei der Entwicklung interaktiver Lernumgebungen

Gute interaktive Lernumgebungen bzw. E-Learning Systeme ganz allgemein erfüllen im Idealfall drei übergeordnete Kriterien: (a) Die Lernumgebungen bringen einen didaktischen Mehrwert; die Schüler lernen mehr und besser. (b) Die Lernumgebungen schaffen einen organisatorischen Mehrwert, der Unterricht wird für den Lehrer und die Schüler einfacher. (c) Die Lernumgebungen rechnen sich auch ökonomisch, der Unterricht wird für die Bildungsinstitution „billiger“. Es wäre vermessen zu glauben, alle drei Ziele würden sich gleichzeitig erreichen lassen. Wir sind aber überzeugt, dass die nachfolgenden pragmatischen Empfehlungen zumindest ein Schritt in die richtige Richtung sind. Jede der Empfehlung kommentieren wir kurz aus Sicht der Lernumgebung InfoTraffic.

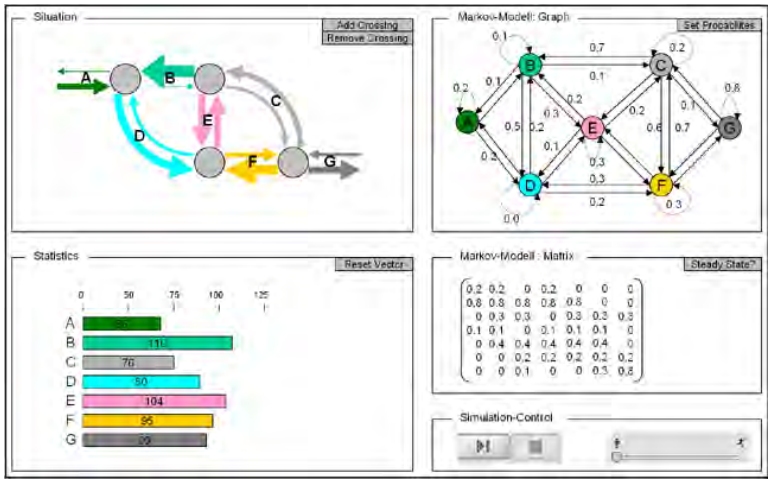


Abbildung 4: Mock-up GUI der Lernumgebung DynaTraffic.

1. Braucht es dazu überhaupt den Computer?

Es macht 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. Reine „Drill & Practice“ Übungen können oft einfacher auf Papier durchgeführt werden.

In der Schule ist der Logikunterricht meist unnötig abstrakt. Der intuitive und alltagsnahe Zugang über die Steuerung von Verkehrskreuzungen in LogicTraffic stellt einen didaktischen Mehrwert dar.

2. Ist das Unterrichtsthema auch in 10 Jahren noch relevant?

Der Entwicklungsaufwand für interaktive Lernsoftware ist im Allgemeinen gross und rechtfertigt sich nur für längerfristig relevante Themen.

Logik begleitet uns im Alltag und wird auch in Zukunft in vielen Wissenschaftsgebieten von zentraler Bedeutung sein.

3. Use-Cases: Ist Interaktivität möglich?

Ein hoher Grad an Interaktivität (zum Beispiel gemäss der Taxonomie von Schulmeister [Sc03]) is ausschlaggebend für die Qualität einer computergestützten Lernumgebung. Das Lesen von Bildschirmtexten oder das Betrachten einer Animation allein löst noch keinen Lernprozess aus.

Nicht jedes Thema eignet sich für eine computergestützte Mensch-Maschinen-Interaktion. Es empfiehlt sich deshalb bereits in der Spezifikationsphase des Projektes Musteraufgaben zusammenzutragen, welche mithilfe der geplanten Lernumgebung bearbeitet werden sollen. In der Sprache der Software-Entwicklung stellen diese Aufgaben Use-Cases für

das Programm dar. Aufgrund der Musteraufgaben kann abgeschätzt werden, ob die Lernumgebung zu einer wirklichen Interaktion mit den Lernenden führt und ob verschiedene kognitive Stufen angesprochen werden.

„Analysieren Sie die vorliegende Verkehrssteuerung. Mit welchen Massnahmen kann die globale Wartezeit minimiert werden?“ lautete eine Musteraufgabe bei QueueTraffic. Solche Aufgabenbeispiele legen fest, welche Funktionen die Lernumgebung anbieten muss.

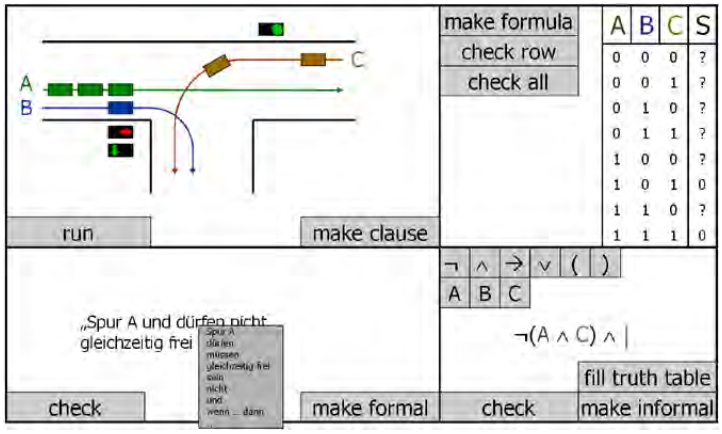


Abbildung 5: Mock-up GUI Entwurf der Lernumgebung LogicTraffic.

4. Paper Based Prototyping

Aus der Softwareentwicklung ist bekannt, dass ein früher Einbezug von Aspekten der Benutzerschnittstelle entscheidend Kosten spart. Bei Lernumgebungen kommt der Benutzerschnittstelle eine besonders hohe Bedeutung zu. Es lohnt sich, so lange als möglich nur auf Papier zu arbeiten. Neben Papier kommen in dieser Phase auch Klebband, Post-It, Schere, Flipchart und andere einfache Werkzeuge zum Einsatz. Für das schnelle Erstellen und Variieren von Programmoberflächen eignen sich auch Präsentationsgrafik-Werkzeuge.

Bei der Entwicklung von InfoTraffic haben wir lange konsequent nur mit Papierentwürfen gearbeitet. An einem Prototypen auf Papier lassen sich mögliche Aufgaben und Einsatzszenarien sehr gut analysieren, durchspielen und diskutieren. Einer der Hauptvorteile von Paper Prototyping: In Diskussionen trauen sich die Leute eher, kritische Bemerkungen anzubringen oder Teile eines Projektes ganz in Frage zu stellen. Liegt der Prototyp bereits als mit grossem Aufwand erstelltes Programm vor, ist die Hemmschwelle für grundlegende Kritik viel höher. Eine gute Einführung in die Methode „Paper Prototyping“ findet sich in [Sn03].

Abbildung 5 zeigt einen frühen PowerPoint-Prototypen von LogicTraffic. In diesem Mockup war auch noch ein Fenster für die natürlichsprachliche Formulierung aussagenlogischer Formeln vorgesehen. Diese Option erwies sich beim Lösen von Aufgaben anhand des Papier-Prototypen als schwierig formalisierbar und wurde fallen gelassen.

5. Rapid Prototyping

Nachdem die Use-Cases und die Benutzerschnittstelle auf Papier festgelegt worden sind, wird ein erster Prototyp implementiert. Dieser soll sobald als möglich einzelnen ausgewählten Testpersonen – Schülerinnen und Lehrern – gezeigt und experimentell erprobt werden.

Abbildung 6 zeigt die erste öffentlich vorgestellte Version von LogicTraffic. Die Unterschiede zur aktuellen Version (Abbildung 2) sind offensichtlich. Auch die interne Programmstruktur wurde überarbeitet und Verkehrskreuzungen werden beispielsweise nicht mehr durch Bilddateien erzeugt, sondern automatisch aus einer xml-Datei generiert.

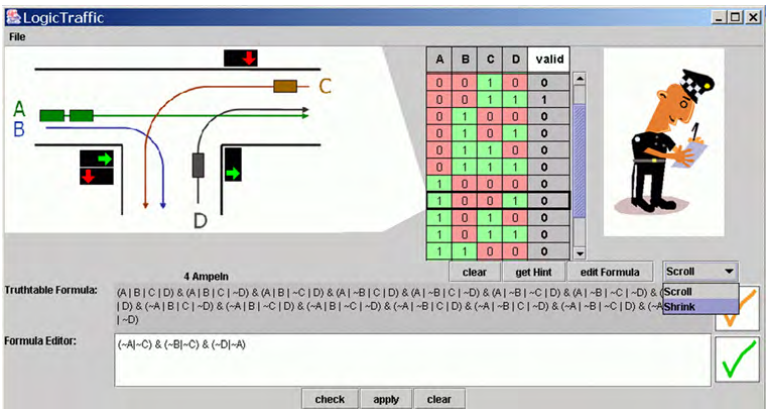


Abbildung 6: Ältere Version der Lernumgebung LogicTraffic

6. Technische Anforderungen: so einfach wie nur möglich

Der grosse Entwicklungsaufwand für eine Lernumgebung rechtfertigt sich nur, wenn die Software einfach auf verschiedenen Plattformen eingesetzt werden kann und wenn auch die Wartung längerfristig sichergestellt ist. Einfachheit und Verzicht auf unnötige Funktionalitäten sind deshalb gefragt.

InfoTraffic wurde in Java programmiert und benötigt nur eine Java Runtime Umgebung (JRE); diese ist heute verbreitet auf Schulrechnern zu finden.

7. Frühzeitige Erprobung

Sobald als möglich sollen erste Tests mit ganzen Schulklassen durchgeführt werden. 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!

Bei allen Teilen von InfoTraffic haben die Erprobungen beispielsweise ergeben, dass die Schüler sich auch ohne ausführliche Bezeichnungen von einzelnen Buttons etc. zurechtfinden und mit englischsprachigen Bezeichnungen keine Probleme haben.

8. Sparsamkeit bei der Benutzerschnittstelle

Die Benutzeroberfläche einer Lernumgebung soll im Idealfall selbsterklärend sein. Nach ersten Erprobungen ist es deshalb wichtig, das GUI der Lernumgebung einem Review-Prozess zu unterwerfen. Bei jedem Button, jeder Beschriftung usw. ist kritisch zu hinterfragen, ob dieses Element wirklich benötigt wird und ob es an der richtigen Stelle platziert ist.

Unsere Erfahrung zeigt, dass im Laufe dieser kritischen Analyse bis zur Hälfte aller ursprünglich vorhandenen Elemente wegfallen. Erstellt man professionell Lernumgebungen, sollte man sich auch mit wissenschaftlichen Erkenntnissen aus dem Bereich Multimedia Learning auseinandersetzen (vgl. zum Beispiel [Ma01]).

9. Verbreitung der Lernumgebung

Steht die Lernumgebung in einer ersten stabilen Version zur Verfügung, ist die Arbeit nicht abgeschlossen. Um überhaupt eine Wirkung der Lernumgebung im Unterricht zu erreichen, muss die Lernumgebung Schülern, Lehrern und weiteren Benutzern zur Verfügung gestellt werden, beispielsweise über eine bekannte Website. Die Verteilkanäle sind schon vorgängig genau abzuklären, es reicht heute längst nicht mehr, einfach eine Domain – zum Beispiel in unserem Fall www.infotraffic.ch – aufzuschalten.

Unsere Erfahrung zeigt, dass das Angebot von Lernumgebungen mit frei verfügbaren didaktischen Begleitmaterialien (z.B. Einführungsvortrag, Anleitung zur Bedienung, Aufgabenblätter, didaktische Hintergrundsinformationen) begleitet werden muss.

10. Sicherstellung von Unterhalt und Kontinuität

Nach der Veröffentlichung, also gewissermassen nach dem Übergang in den Einsatz, muss (wie bei Informatikprojekten üblich) der Unterhalt sichergestellt werden. Dies beinhaltet insbesondere das Beheben etwaiger Fehler und die Umsetzung allfälliger Erweiterungen und Anpassungen an neue Software-Versionen. In dieser Phase gilt es auch, Vertrauen zu schaffen. Fehlende Zeit ist eines der Hauptprobleme im Berufsalltag der meisten Lehrpersonen. Eine Lehrerin überlegt sich deshalb gut, ob sie den Aufwand für die didaktische Aufbereitung eines Themas basierend auf einer neuen Lernumgebung leisten soll oder nicht. Sie wird abwägen, ob die Software in ein paar Jahren noch zur Verfügung stehen wird. Viele interaktive Lernumgebungen entstehen im Hochschulumfeld im Rahmen studentischer Arbeiten und die geforderte Kontinuität ist deshalb nicht sichergestellt. Hier gilt es, rechtzeitig Massnahmen zu ergreifen.

Üblicherweise werden auch Wünsche nach Erweiterungen der Software an ein Entwicklerteam herangetragen. Hier heisst es vorsichtig zu sein: Die Qualität einer Lernumgebung misst sich nicht an der Fülle der angebotenen „Features“. Häufige Überarbeitungen und Erweiterungen verunsichern die Benutzer und führen dazu, dass bereitgestellte Unterrichtsmaterialien aktualisiert werden müssen. Und jede Erweiterung erhöht auch die Komplexität des Programmcodes und erschwert die Wartung. „Weniger ist mehr“ lautet deshalb das Motto.

5 Zusammenfassung und Fazit

Die Geschichte zeigt, dass die Entwicklung qualitativ hoch stehender interaktiver Lernumgebungen ein aufwändiges und schwieriges Unterfangen ist. Entscheidend für den Erfolg einer Lernumgebung sind unserer Überzeugung nach weniger die zur Verfügung stehenden Ressourcen, sondern das Zusammenspiel von erfahrenen Lehrpersonen, versierten Softwareentwicklern und Fachleuten aus der Bildungsforschung, welche bereit sind, sich auf eine nach ingenieurwissenschaftlichen Prinzipien funktionierende Forschungsarbeit einzulassen.
Der Einsatz computergestützter Lernumgebungen ist komplex und lässt sich nicht mittels einer den strengen wissenschaftlichen Kriterien der Bildungsforschung gerecht werdenden Methodik evaluieren. Gefragt ist von allen Beteiligten die Bereitschaft zu interdisziplinärer Zusammenarbeit und ein pragmatisches Vorgehen. Ganz speziell braucht es auch den Mut, sich der fast sicheren Kritik aus der Scientific Community der Lehr- und Lernforschung auf der einen Seite und der Informatik auf der anderen Seite auszusetzen. „Try everything out and follow your nose“ ist in akademischen Kreisen verpönt, aber ein erfolgsversprechender Ingenieuransatz, wenn es um Innovation im Bereich E-Learning geht.

Literaturverzeichnis

[AH07] R. Arnold und W. Hartmann. LogicTraffic - Logik in der Allgemeinbildung. Informatik-Spektrum, Volume 30, Number 1, Seiten 19–26, 2007.

[ALH07] R. Arnold, M. Langheinrich und W. Hartmann. InfoTraffic - Teaching Important Concepts of Computer Science and Math through Real-World Examples. In Proceedings of the ACM SIGCSE Technical Symposium 2007, Covington, Kentucky, USA, March 2007.

[BN04] M. Braendle und J. Nievergelt. Tackling Complexity: A Case Study on Educational Software. World Conference on E-Learning in Corp., Govt., Health., and Higher Ed. (ELEARN), Volume 2004, Issue 1, Seiten 1794–1799, 2004.

[La93] B. Laurel. Computers as Theatre. Addison-Wesley Longman, 1993.

[Ma01] R. Mayer. Multimedia Learning. Cambridge University Press, 2001.

[Ni75] J. Nievergelt. Interactive Systems for Education: The New Look of CAI. Proc. IFIP Conf. on Computers in Education, 53, No. 4:465–472, 1975.

[Re06] G. Reinmann. Nur “Forschung danach”? Vom faktischen und potentiellen Beitrag der Forschung zu alltagstauglichen Innovationen beim E-Learning. Arbeitsbericht Universität Augsburg, Nr. 14, 2006.

[RNH04] R. Reichert, J. Nievergelt und W. Hartmann. Programmieren mit Kara. Ein spielerischer Zugang zur Informatik, (ergänzte Neuauflage). Springer, Berlin, Dezember 2004.

[Sc03] R. Schulmeister. Taxonomy of Multimedia Component Interactivity. A Contribution to the Current Metadata Debate. Studies in Communication Sciences. Studi di scienze della communicazione., 3(1):61–80, 2003.

[Sn03] C. Snyder. Paper Prototyping - The Fast and Easy Way to Design and Refine User Interfaces. Elsevier Science, 2003.

[Sw07] SwissEduc. Bildungsserver SwissEduc, Sammlung von Unterrichtsmaterialien, Unterbereich Informatik. http://www.swisseduc.ch/informatik/, Januar 2007.

[Wü06] P. Wülser. Unterricht fürs Ohr - Podcasting in der Schule. Unveröffentlichtes Manuskript, 2006.

Von vernetzten fundamentalen Ideen zum Verstehen von Informatiksystemen – Eine Unterrichtserprobung in der Sekundarstufe II

Peer Stechert

Didaktik der Informatik und E-Learning
Universität Siegen
Hölderlinstraße 3
57068 Siegen
stechert@die.informatik.uni-siegen.de

Abstract: Die Förderung des Informatiksystemverständnisses ist eine vorrangige Aufgabe des Informatikunterrichts. Die vorliegende Arbeit stellt die exemplarische Umsetzung des Unterrichtsmodells für das Verstehen von Informatiksystemen mit Lernenden in der Sekundarstufe II vor und beschreibt den kognitiven Zugang zu vernetzten fundamentalen Ideen in Informatiksystemen. Es werden objektorientierte Entwurfsmuster als Wissensrepräsentation vernetzter fundamentaler Ideen der Informatik verwendet.

1 Motivation und Stand der Forschung

Die Förderung des Informatiksystemverständnisses ist eine vorrangige Aufgabe des Informatikunterrichts. Ein entsprechendes Unterrichtsmodell des Autors wurde auf nationaler und internationaler Ebene diskutiert. Ziel dieses Artikels ist es, Erkenntnisse aus der praktischen Erprobung des Unterrichtsmodells für das Verstehen von Informatiksystemen[1] vorzustellen und Schlussfolgerungen für das Unterrichtsmodell aufzuzeigen. Der Ansatz des Unterrichtsmodells [St06a], d. h. eines theoretischen Rahmens für unterschiedliche Unterrichtskonzepte und -reihen in der Sekundarstufe II, basiert auf der Vernetzung von fundamentalen Ideen der Informatik [Sc93] in ausgewählten Entwurfsmustern [Ga95]. Fundamentale Ideen der Informatik sind zurzeit Inhalt und Ausgangspunkt mehrerer fachdidaktischer Publikationen. Schneider begann in Anlehnung an das Fach Mathematik eine Gliederung der Informatik nach konzeptuellen Leitideen. Dafür ordnete er der Leitidee „funktionale Modellierung“ fundamentale Ideen zu, die in dem Themenbereich zu erlernen sind [Sc06].

Der vorliegende Artikel beschreibt, wie vernetzte fundamentale Ideen der Informatik für das Verstehen von Informatiksystemen im Informatikunterricht der Sekundarstufe II mit Vorkenntnissen in der objektorientierten Modellierung erlernt werden können. Dafür

[1] Es wird die Definition des Informatiksystems aus [CS03, S. 301] verwendet.

werden nach außen sichtbares Verhalten, innere Struktur und konkrete Realisierung von Informatiksystemen systematisch untersucht. Der Lehr-Lern-Prozess wird insbesondere durch Einsatz dedizierter Aufgaben, Lernsoftware als Unterrichtsmittel und die Durchführung von Informatikexperimenten strukturiert. Das Unterrichtsprojekt wurde im Schuljahr 2006/2007 an einem nordrhein-westfälischen Gymnasium durchgeführt.

2 Forschungsmodell

2.1 Forschungsmethodik

Die Forschungsmethodik ist inspiriert von praxisorientierter Fachdidaktikforschung wie sie von Hubwieser auf der INFOS 2005 vorgestellt wurde [Hu05]. Für solch eine intervenierende Fachdidaktik ist es kennzeichnend, dass neue Konzepte der Fachdidaktik von den Forschenden in Feldstudien umgesetzt und evaluiert werden. Die Rückkopplung mit der Praxis erlaubt eine kritische Reflexion der theoretischen Ergebnisse. Derartige Interventionsstudien sind stichprobenbasiert bei kleiner Stichprobengröße. Dies ist der Heterogenität des Informatikunterrichts geschuldet, die aufgrund breit angelegter und somit ebenso weit interpretierten Rahmenpläne vorherrscht.

Als zweite Säule der Forschungsmethodik und Voraussetzung der Intervention tritt das didaktische System [BS02]. Es ist eine Formalisierung des Lehr-Lern-Prozesses und besteht aus Wissensstrukturen, Aufgabenklassen und Lernsoftware. Stärke dieses Ansatzes ist, dass das didaktische System in unserer Forschergruppe seit 2001 auf die Bereiche Objektorientierung, Internetworking und Verstehen von Informatiksystemen angewendet und kontinuierlich weiterentwickelt wird (vgl. auch [FS06]).

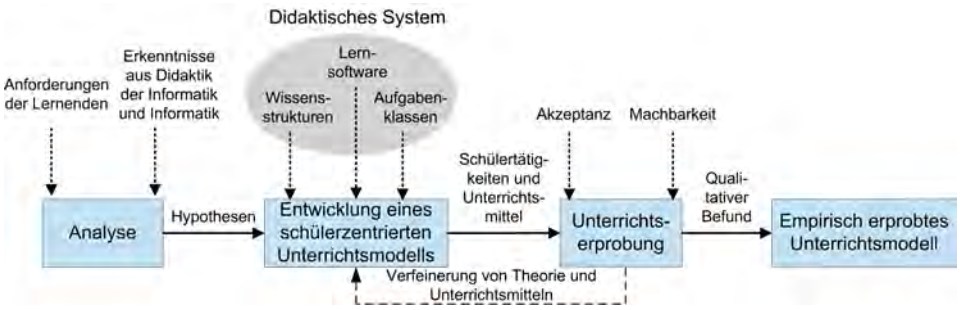


Abbildung 1: Entwurfs-, Interventions- und Evaluationszyklus der Unterrichtsentwicklung.

Eine Übersicht der Forschungsmethodik zeigt Abbildung 1. Während der Analyse des Forschungsfelds wurde der Bedarf der Lernenden bezüglich des Verstehens von Informatiksystemen unter Bezugnahme auf nationale und internationale Curricula und Empfehlungen untersucht. Die Kombination von Erkenntnissen sowohl aus der Fachwissenschaft und aus der Fachdidaktik erlaubt die Formulierung von Forschungshypothesen für den Lehr-Lern-Prozess.

In [St06b] wurde begründet, dass Entwurfsmuster informatikspezifische, extern darstellbare Wissensrepräsentationen von vernetzten fundamentalen Ideen sind. Einige nach fachdidaktischen Kriterien ausgewählte Entwurfsmuster sind gleichzeitig Lernmittel und Lerngegenstand. Zur Erstellung von Unterrichtsmitteln wurde vom Autor eine studentische Projektgruppe betreut, welche die Lernsoftware „Pattern Park“ entwickelte [PP07]. Damit können attraktive Aufgaben gestellt werden, die fundamentale Ideen und Entwurfsmuster in verschiedenen Sichten verknüpfen. Nächster Schritt in der Forschungsvorgehensweise war die Entwicklung eines Unterrichtsprojekts. Dafür mussten weitere Unterrichtsmittel und Schülertätigkeiten erstellt bzw. definiert werden. Auch die Formulierung von Lernzielen nach der überarbeiteten Bloom'schen Lernzieltaxonomie [St06c] wurde vorgenommen.

Die besondere Situation, Forschungs- und Lehrperson gleichzeitig zu sein, wird in der quantitativen Forschung kritisch gesehen. Deshalb und aufgrund der geringen Stichprobengröße ist die vorliegende Untersuchung qualitativ als Hinweis auf Machbarkeit, Durchführbarkeit und Akzeptanz bei den Lernenden zu interpretieren. Dafür wird davon ausgegangen, „dass die Teilnahme im Feld Empathie und Identifikation mit den Untersuchungspersonen voraussetzt, da erst so die Interpretationsprozesse der Untersuchungspersonen erfasst und verstanden werden können“ [At06, S. 94]. Merkmal solcher quasiexperimenteller Felduntersuchung im Sinne von Bassey – „previously developed theory is used as a template with which to compare the empirical results of the case study” [Ba99, S. 31] – ist, dass sie in natürlichen, d. h. nicht randomisierten, im Zuge des Forschungsprozesses kaum veränderten Umgebungen stattfinden. Jedoch lassen Ergebnisse von quasiexperimentellen Untersuchungen „mehr Erklärungsvarianten zu als die Ergebnisse reiner experimenteller Untersuchungen, d. h. sie haben eine geringere interne Validität“ [BD02, S. 527]. Die qualitativ-formativen Evaluationsaktivitäten, d. h., die durch den Einsatz von Interview und Akzeptanzbefragung um eine Überprüfung des Lehr-Lern-Szenarios angereicherten Ergebnisse, führen zu einem in empirischen, explorativen Phasen erprobten Unterrichtsmodell [At06, S. 31].

2.2 Strukturierung des Lerngegenstands

Das Unterrichtsprojekt hat drei Lernphasen S_i, im Sinne von Basiskompetenzen, die im Fokus des Unterrichtsmodells für das Verstehen von Informatiksystemen stehen (vgl. [CS03, S. 658]):

- S_1: Das Verstehen wesentlicher Aspekte des nach außen sichtbaren Verhaltens.
- S_2: Das Verstehen von Aspekten der inneren Struktur, die auf fundamentalen Ideen der Informatik basieren.
- S_3: Das Verstehen von Implementierungsdetails zur Entwicklung einer konkreten Realisierung.

Zur Vorbereitung wurde mit dem Informatiklehrer, in dessen Klasse das Unterrichtsprojekt durchgeführt werden sollte, vier Monate vor der Durchführung ein erster Entwurf der Unterrichtskonzeption diskutiert, um Umsetzungsprobleme frühzeitig zu erkennen.

Mit Blick auf das Schulcurriculum wurden die Lernziele zusammengestellt. Die Ergebnisse dieser Diskussionen flossen auch in [St06a] ein. Darüber hinaus hat der Autor in dem Kurs zwei Wochen vor der Durchführung hospitiert, um an vorangegangenen Unterricht anzuknüpfen. Dabei erwies es sich als unkompliziert, den bisherigen Unterricht nun nahtlos in einem Unterrichtsprojekt weiterzuführen, das dem Unterrichtsmodell für das Verstehen von Informatiksystemen folgte. Dies ist neben der Tatsache, dass konkrete Programme Unterrichtsgegenstand waren, auch auf die objektorientierten Entwurfsmuster zurückzuführen. Eingeführt in das Unterrichtsmodell, um vernetzte fundamentale Ideen der Informatik zu repräsentieren, sind sie als Strukturierungsmuster und Heuristiken aus der Informatikpraxis häufig mit wichtigen Informatikkonzepten verbunden, die wiederum Unterrichtsthemen sind. Somit finden sich für das Unterrichtsmodell sehr viele Anknüpfungspunkte an den traditionellen (hier objektorientierten) Informatikunterricht für eine Unterrichtsreihe zum Thema Verstehen von Informatiksystemen. Dennoch ist Verstehen von Informatiksystemen nicht zwangsläufig mit der objektorientierten Modellierung der Sekundarstufe II verknüpft. Vielmehr können im Sinne eines Spiralcurriculums besonders Themen und Fragestellungen aus S_1 bereits früher im Informatikunterricht behandelt werden.

In der durchgeführten Unterrichtserprobung lag der Schwerpunkt auf dem Zusammenspiel zwischen dem nach außen sichtbaren Verhalten und der inneren Struktur eines zum Teil selbst erstellten Programms (S_1 und S_2). Für die einzelnen unten angegebenen Schwerpunkte wurden Teillernziele durch Operatoren aus den Anforderungsbereichen I (z. B. Wiedergabe von Kenntnissen), II (z. B. Anwenden von Kenntnissen) und III (z. B. Problemlösen und Werten) der Einheitlichen Prüfungsanforderungen Informatik [EPA04] sowie nach Blooms überarbeiteter Taxonomie eingesetzt [AK01]. Exemplarische Schwerpunkte beim Verstehen des nach außen sichtbaren Verhaltens waren:

S_{11}: Die Datenstruktur Schlange und der Schlangentraversierung verstehen sowie weitere Anforderungen an das Entwurfsmuster Iterator erklären können.

S_{12}: Zugriffskontrolle anhand des Entwurfsmusters Proxy verstehen.

Die konkrete Umsetzung wird in Abschnitt 3.3 beschrieben. Exemplarische Schwerpunkte beim Verstehen der inneren Struktur waren:

S_{21}: Das Konzept der Schnittstelle anhand der Kombination von Schlange und Iteratorentwurfsmuster verstehen bei phänomenologischer Vorwegnahme der Zugriffskontrolle.

S_{22}: Zugriffskontrolle anhand des Entwurfsmusters Proxy sowohl statisch, d. h. via Klassendiagramm, als auch dynamisch mittels Sequenzdiagramm beschreiben können.

In der dritten Lernphase wurden folgende Schwerpunkte angestrebt:

S_{31}: Das Verstehen unterschiedlicher Traversierungsalgorithmen durch Modifikation des Iteratormusters.

S_{32}: Das Verstehen einer auf Vererbung beruhenden Zuweisung von Zugriffsrechten durch Modifikation von Implementierungsdetails im Proxyentwurfsmuster.

Die Phase S_3 war nicht Beobachtungsziel des Projekts. Nur vereinzelt wurde existierender Quellcode von den Schülern[2] erweitert, um an vorhandenes Vorwissen anzuknüpfen.

3 Unterrichtserfahrungen

3.1 Lerngruppe und zeitlicher Rahmen

Die Erprobung des Unterrichtsmodells fand im Herbst 2006 im Rahmen eines Grundkurses Informatik in der Jahrgangsstufe 12 eines Siegener Gymnasiums statt. Der Kurs setzte sich zusammen aus vier Schülerinnen und 19 Schülern. Dadurch, dass es sich hierbei um einen regulären Grundkurs handelte, sind valide qualitative Auswertungen möglich, da der Kurs nicht nur aus hoch motivierten Schülern besteht, wie es im Rahmen von Wahlfächern oft der Fall ist. Um die Ergebnisse der Fallstudie weitergehend verallgemeinern zu können, sollten die Schüler Vorkenntnisse in objektorientierter Modellierung besitzen und möglichst keine Spezialisierung im technisch-naturwissenschaftlichen Bereich vorweisen. Die Schüler nutzten bereits seit etwa einem halben Schuljahr die objektorientierte Modellierung u. a. mit der Klassenbibliothek „Stifte und Mäuse“ [LL07]. Vererbung, erstellen von Klassen und erzeugen von Objekten sowie ereignisgesteuerte Programmierung waren bekannt ebenso wie grundlegende Programmierkonzepte, z. B. Schleifen und Variablen. Es zeigte sich während der Unterrichtsfolge, dass Vererbung eine Fehlerquelle darstellte (siehe Abschnitt 3.3).

Die Erprobung umfasste 12 Unterrichtsstunden zu drei Stunden pro Woche. Zusätzlich wurden ein Abschlusstest und eine Akzeptanzbefragung mit den Schülern durchgeführt. Der Unterricht wurde von der Forschungsperson gehalten und vom Informatiklehrer sowie einem weiteren Mitarbeiter des Instituts für Didaktik der Informatik und E-Learning der Universität Siegen gezielt beobachtet, um erste Erkenntnisse über typische Bearbeitungsstrategien und Fehler zu gewinnen. Es wurde angenommen, dass die Schüler keine Vorkenntnisse zur systematischen Erkundung des nach außen sichtbaren Verhaltens eines Informatiksystems aufweisen.

Die Unterrichtsstunden wurden sowohl mit dem Lehrer als auch mit weiteren Mitarbeitern des Instituts für Didaktik der Informatik und E-Learning vor- und nachbereitet. Der Informatiklehrer stand abschließend für ein Leitfaden-Interview zur Verfügung. Die Erprobung wurde unterbrochen durch eine Woche, in der der Informatiklehrer (aufgrund einer Vortragsreise des Autors) unabhängig vom Unterrichtsprojekt unterrichtete. Darüber hinaus gab es eine Verzögerung durch einen beweglichen Ferientag direkt im Anschluss, so dass insgesamt eine anderthalbwöchige Pause entstand.

[2] Es werden wenn möglich geschlechtsneutrale Formulierungen verwendet. Wenn die männliche Form gewählt wird, sind damit, im Sinne eines generischen Maskulinums, immer Frauen und Männer gleichermaßen gemeint. Frauen mögen sich nicht ausgeschlossen fühlen.

3.2 Unterrichtsmethodik und technischer Rahmen

Bei der Gestaltung des Unterrichtsprojekts wurde besonders darauf geachtet, dass die Schülertätigkeiten im Mittelpunkt standen. Somit wurde möglichst wenig in Lehrervorträgen umgesetzt, stattdessen war das Unterrichtsprojekt schülerzentriert angelegt. Lehrervorträge nahmen nur wenige Minuten pro Unterrichtsstunde ein. Partnerarbeit war die vorherrschende Arbeitsform, sowohl an den Rechnern als auch bei schriftlichen Arbeitsaufträgen.

Die räumliche Gestaltung des Informatiklabors unterstützte die Trennung von Theorie und Informatikexperimenten mit Rechnern durch den zentralen Kommunikationsbereich und die dezentral an den Seiten aufgestellten Rechner. Das Schulintranet besaß eine Verbindung zum Internet und eine von allen gemeinsam nutzbare Speicherpartition des Dateiservers ermöglichte den problemlosen Austausch größerer Dateien. Der vordere Demonstrationsbereich des Lehrers war mit Whiteboard, Rechner und Projektor zur Präsentation ausgestattet.

3.3 Lernphasen und Problemstellen im Unterrichtsprojekt

Ziel der Erprobung war es, den Zusammenhang zwischen dem nach außen sichtbaren Verhalten und der inneren Struktur von Informatiksystemen zu verdeutlichen. Um Fehlvorstellungen analysieren zu können, wurden sämtliche in den Unterrichtsstunden handschriftlich oder digital erstellten Schülerlösungen eingesammelt. Nach Abschluss des Unterrichtsprojekts wurde ein Test in Form einer schriftlichen Übung durchgeführt. Um dessen Wichtigkeit für die Schüler zu unterstreichen, sollten die Ergebnisse in die Gesamtnote einfließen. Aus diesem Grund wurde die schriftliche Übung nicht anonymisiert durchgeführt.

Roter Faden des Unterrichtsprojekts war das Wechselspiel von Erkundung des nach außen sichtbaren Verhaltens eines Programms und der Betrachtung und Erstellung von Modellen der inneren Struktur. Startpunkt war ein Programm für eine Arztpraxis mit einer Warteschlange, das bereits im vorhergehenden Informatikunterricht von den Schülern mittels Klassendiagramm modelliert und in der Programmiersprache Objekt-Pascal mit Delphi 6 geschrieben worden war. Es wurde erweitert um das Iteratormuster (S_{11}).

Erst für das Thema Zugriffskontrolle (S_{12}) wurde ein neues Programm untersucht, das die Vergabe von Zugriffsrechten realisierte. Dies geschah jedoch immer mit Rückbezügen zum „Arztinformationssystem“. Spiralförmig wurden somit die Lernphasen wiederholt und auf die Themen Datenstruktur Schlange, Iteration und Zugriffskontrolle angewendet.

Die Hospitation in der Klasse kurz vor Beginn des Unterrichtsprojekts machte deutlich, dass Klassendiagramme als wichtiges Strukturbeschreibungsmittel eines Informatiksystems ohne weitere Vorbereitung eingesetzt werden konnten. Allgemein fiel während der Hospitation die Schwierigkeit der Schüler auf, zwischen Daten und Darstellung zu unterscheiden – ein Abgleich zwischen Datenhaltung und Darstellungsebene fand in den Schülerlösungen dementsprechend oft nicht statt. Dies sollte ein erster Ansatzpunkt für

die Erkundung des nach außen sichtbaren Verhaltens des Programms im Unterrichtsprojekt sein.

Datenstruktur Schlange und Iterator: Als erster Schwerpunkt zu Beginn des Unterrichtsprojekts wurde das nach außen sichtbare Verhalten des „Arztinformationssystem“ systematisch erkundet und anschließend schlossen die Schüler auf zugrunde liegende informatische Konzepte. Hierfür wurde ein Experimentiervorgang beschrieben, der aus sechs Schritten besteht und die sechs kognitiven Prozesse umfasst [St06c], die in der überarbeiteten Lernzieltaxonomie nach Bloom angegeben sind [AK01]. Er bestand aus dem Aufstellen von Hypothesen über das Systemverhalten, dem Beschreiben und Dokumentieren des Experimentierablaufs, den Rückschlüssen aus dem tatsächlichen Verhalten auf informatische Konzepte, dem Identifizieren von Sonderfällen wie der leeren Schlange, dem Überprüfen der Umsetzung der Sonderfälle im Informatiksystem sowie Auswirkungen des Einsatzes des Informatiksystems. Insbesondere das Erkennen von möglichen Konzepten fiel anfangs bei Schlange und Iterator schwer (S_{11}), obwohl das Listenkonzept bekannt war. Bei der Auswertung der ersten Durchführung des Informatikexperimentes wurde nur bei etwas mehr als der Hälfte der abgegebenen Lösungen das Konzept der Schlange wieder erkannt. Der Sonderfall, dass die Schlange leer, bzw. die Frage, ob die Schlange voll sein kann, wurde im Plenum diskutiert.

Die Demonstration der Informatikexperimente, sei es durch Lehrperson oder Schüler, unterstützt das Unterrichtsprojekt zum Verstehen von Informatiksystemen stark, da die neuen systematischen Vorgehensweisen zur Analyse erläutert werden konnten. Als zweiter Schwerpunkt galt es, die innere Struktur des Programms anhand des Quellcodes und eines unvollständigen Klassendiagramms zu untersuchen (S_{21}). Die Umsetzung der identifizierten informatischen Konzepte in eine Klassenstruktur war hier Schwerpunkt. Um dynamische Abläufe in der inneren Struktur beschreiben zu können, wurden während der ersten Woche in einer Partnerarbeit Objektdiagramme zu verschiedenen Zeitpunkten erstellt. Relativ überraschend war für die Schüler die Einsicht, dass der Iterator eine Variante der Zugriffskontrolle ist (S_{21}). Das Verstehen des Systemverhaltens führte dazu, in der inneren Struktur Fehler schneller finden und beheben zu können (S_{31}).

Zugriffskontrolle: Bei der Erkundung der Zugriffskontrolle (S_{12}) war den Schülern das Vorgehen zur systematischen Erkundung des Systemverhaltens bekannt. Überraschend war für die Schüler bei der inneren Struktur der auf Vererbung basierende Ansatz zur Überprüfung der Zugriffsrechte (Abbildung 2). Da das Verstehen von dynamischen Aspekten essenziell für Informatiksystemverständnis ist, wurde zur Beschreibung der Abläufe bei der Zugriffskontrolle (S_{22}) das Sequenzdiagramm genutzt. Einstiegsaufgabe war die Erstellung eines so genannten Interaktionsdiagramms, anschließend wurden einfache Sequenzdiagramme mit Elementen der Lernsoftware „Pattern Park“ erstellt.

Die Einführung der Zugriffskontrolle mittels Proxy ermöglichte den Schülern, das Prinzip der Zugriffrechte zu verstehen und selbst für bestimmte Situationen zu gestalten. So konnten neben Administrator, Benutzer und Gast weitere Anwender (-gruppen) mit bestimmten Rechten versehen werden (S_{32}).

Rolle der Lernsoftware: Durch Modularisierung bei der Entwicklung der Lernsoftware „Pattern Park“ konnten für die unterrichtliche Erprobung schon vorab Animationen zu Iteration (S_{11}) und Zugriffskontrolle (S_{12}) sowie Übungsmodule mit Sequenzdiagrammen (S_{22}) eingesetzt werden. Letzteres geschah bei der Nutzung eines Proxys zur Realisierung von Zugriffsschutz. Insbesondere das Übungsmodul griff eine Lebensweltsituation handlungsorientiert auf und schuf so den Brückenschlag zwischen dem nach außen sichtbaren Verhalten und der inneren Struktur eines Informatiksystems einerseits und die Verbindung der statischen und dynamischen Strukturbeschreibung andererseits. Die Beobachtung der Bearbeitungsweisen hat ergeben, dass gerade diese Aufgaben, die das nach außen sichtbare Verhalten von Informatiksystemen und die Analyse der inneren Struktur verknüpfen, Schlüssel zum Verstehen waren, denn hier wurden unterschiedliche Sichten auf das Informatiksystem kombiniert.

Außerdem zeigte sich während des Unterrichtsprojekts, dass das frühzeitige Vorführen der Animation den Schülern nicht nur die Problemstellung klarer werden ließ, sondern gleichzeitig durch den Bezug zum Entwurfsmuster, einem „Lösungsmuster“, eine konkretere Vorstellung von Lösungsansätzen unterstützte. Abbildung 2 zeigt das Klassendiagramm des Proxy zur Realisierung der Zugriffskontrolle. Anhand des Klassendiagramms mussten die Schüler ein korrektes Sequenzdiagramms eines typischen Ablaufs in dem vorgegebenen Szenario kreieren.

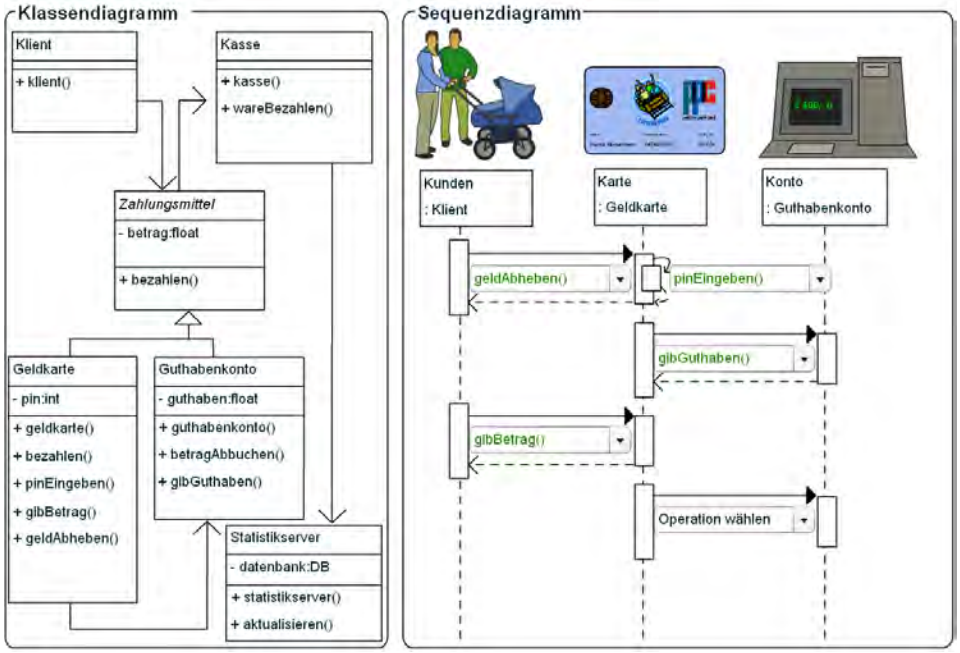


Abbildung 2. Klassen- und Sequenzdiagramm zum Zugriffsschutz aus der Lernsoftware „Pattern Park“

3.4 Auswertung des Abschlusstests

Für die schriftliche Übung wurden Multiple-Choice-Aufgaben und Lückentexte gewählt. Als Stimuli für die Testitems dienten u. a. Klassendiagramme. Inhaltlich waren die Schwerpunkte der schriftlichen Übung:

- nach außen sichtbares Verhalten von Informatiksystemen,
- innere Struktur von Informatiksystemen,
- Zusammenhang zwischen nach außen sichtbarem Verhalten und innerer Struktur,
- Zugriffskontrolle mittels Entwurfsmuster Proxy,
- Iteration mittels Entwurfsmuster Iterator.

Für die schriftliche Übung wurden zwei Varianten A und B eingesetzt, so dass ein Thema bei Bedarf in der einen Variante mit Lückentext, in der anderen mit Multiple-Choice-Aufgabe geprüft werden konnte. Die Bearbeitungszeit betrug 30 Minuten. Insgesamt wurden 21 Tests bearbeitet (Variante A: 11; Variante B: 10). Die Ergebnisse sind überdurchschnittlich gut ausgefallen (gerundete Werte): Minimalwert: 53%, Maximalwert 100%, Mittelwert: 75%, Median: 77%. Die häufigsten Werte (Modi) waren 63%, 67% und 77% mit je drei Ausprägungen. Der Mittelwert bei den Mädchen lag mit 72% leicht unter dem Gesamtdurchschnitt.

Bei den Fragen zum nach außen sichtbaren Verhalten hatten die Schüler Schwierigkeiten in Bezug auf eine systematische Erkundung. Hieraus könnte der Schluss gezogen werden, dass es kaum Vorkenntnisse bezüglich eines solchen Vorgehens gab. Im scheinbaren Widerspruch dazu steht die Aussage fast aller Schüler, dass das Erkunden der inneren Struktur schwieriger als das Erkunden des nach außen sichtbaren Verhaltens sei. Es haben alle Schüler angegeben, dass zum Verstehen des nach außen sichtbaren Verhaltens eines Informatiksystems ein systematisches Experimentieren notwendig sei und dass unerwartetes Verhalten eines Informatiksystems Rückschlüsse auf dessen innere Struktur liefere. Mit der Vorgehensweise zum Erkunden der inneren Struktur hingegen waren im Test weniger Schwierigkeiten verbunden. Dies mag an den Vorkenntnissen zur Erstellung von Klassendiagrammen und auch zu Delphi-Projekten liegen. Es zeigte sich eine teilweise Überforderung der Lernenden bei der Verbindung des nach außen sichtbaren Verhaltens mit der inneren Struktur. Angebracht gewesen wäre hier eine Aufteilung in kleinere Teilaufgaben.

Die Hauptschwierigkeiten lagen bei der Beschreibung der inneren Struktur der Datenstruktur Schlange und des Iterator, da oftmals falsche Assoziationen angegeben wurden. Des Weiteren ist einigen Schülern die Unterscheidung zwischen Schlange und Iterator nicht gelungen. Außerdem wurden bei dem Transfer des Proxy auf eine neue Situation Fehler gemacht, obwohl die zum Unterricht ähnliche Aufgabe zur Zugriffskontrolle gelöst wurde. In weiteren Unterrichtsprojekten sollten deshalb mehr Transferaufgaben hinsichtlich ähnlicher Situationen und Strukturen zu bearbeiten sein.

Die Ergebnisse aus der Analyse der Explizierung und Reflexion der jeweiligen Schüler-Vorgehensweisen in der Lerngruppe lassen sich jedoch nicht eindeutig Fehlvorstellungen beim Verstehen von Informatiksystemen zuordnen. Insbesondere ist zu vermuten, dass typische Fehlvorstellungen der Objektorientierung sehr dominant sind und Schülervorstellungen von Informatiksystemen überlagern. Zur Realisierung der Zugriffskontrolle wurden beispielsweise oft falsche Vererbungsbeziehungen verwendet.

3.5 Auswertung der Akzeptanzbefragung und des Interviews mit dem Informatiklehrer

Im Nachgang des Unterrichtsprojekts wurden die Lernenden des Kurses schriftlich befragt. Die ausgefüllten Fragebögen wurden von den Lernenden anonymisiert. Die Fragebögen enthielten jeweils ca. 35 Aussagen aus den vier Bereichen „Informatikunterricht allgemein“, „Schwierigkeitsgrad und Lernstoff“, „Befragung zum Unterrichtsthema“ und „Einschätzung des Lernfortschritts in den einzelnen Lernbereichen“, zu denen die Lernenden durch Ankreuzen ihre Zustimmung oder Ablehnung bei (vorwiegend) vier vorgegebenen Skalenwerten äußern sollten zuzüglich der Möglichkeit sich zu enthalten. Ein Ziel war es, die Motivation der Lernenden zu analysieren. Gefragt wurde deshalb auch nach außerschulischer Beschäftigung mit dem Thema, weitergehendem Interesse sowie Einschätzung des persönlichen Nutzens. Weitere ausgewählte Einflussgrößen bzw. Einstellungen der Lernenden waren beispielsweise Geschlecht und Anzahl versäumter Unterrichtstermine, um die Antworten interpretieren zu können. In der Akzeptanzbefragung ergab sich, dass von 23 Schülern 16 (70%) den Schwierigkeitsgrad des Unterrichtsprojekts als angemessen und nur vier (17%) ihn als hoch empfanden. Der Stoffumfang wurde von 16 (70%) als angemessen und von sechs (26%) als viel wahrgenommen. Dieser Aspekt wird auch durch zehn schriftliche Anmerkungen gestützt, welche die geringe zur Verfügung stehende Zeit für das Bearbeiten einiger Aufgaben beanstanden. Zu dem eigenen Lernfortschritt bei den inhaltlichen Themen der Analyse, der inneren Abläufe und des Aufbaus von Informatiksystemen geben jeweils 16 (70%) oder mehr Schüler an, einiges bzw. viel dazugelernt zu haben. Die Konzentration auf die Aufgaben fiel leicht. Der Aussage, dass Hilfsmittel wie Arbeitsblätter und (Lern-) Software ausreichend und in guter Qualität vorhanden waren, wurde von 17 (74%) Schülern etwas, bzw. voll zugestimmt. Dies ist sicherlich auch auf den Einsatz ausgewählter Animationen und Übungen aus der Lernsoftware „Pattern Park“ zurückzuführen. Durchaus mit den auf den Basiskompetenzen liegenden Zielen des Unterrichtsmodell im Einklang steht, dass fünf (22%) Schüler anzeigten nichts und elf (47%) nicht viel zur Delphi-Programmierung dazugelernt zu haben. Dass elf (48%) Schüler angaben eher keinen bzw. drei (13%) bestätigten keinen Spaß daran zu haben, ihr Verständnis für das Thema zu vertiefen, offenbart ein Motivationsproblem, dem in folgenden Erprobungen durch stärkeren Bezug zu Alltagsproblemen und anderen Fächern begegnet werden muss. Sequenzdiagramme als Darstellungsform von Abläufen wurden von den Schülern positiv aufgenommen. Abschließend ist zu erwähnen, dass 16 (70%) Schüler angaben, für sich etwas dazugelernt zu haben, und dass die Mehrheit von 19 (82%) Schülern der Aussage, dass die Inhalte des Unterrichtsprojekts für den Umgang mit Informatiksystemen sehr nützlich sind, etwas bzw. voll zustimmen. In dem Interview, für das der Informatiklehrer abschließend zur Verfügung stand, wurde wiederum bestätigt, dass der Schwierigkeits-

grad angemessen sei. Auch die Konzeption mit dem Wechsel der unterschiedlichen Untersuchungsbereiche der Informatiksysteme wurde befürwortet.

4 Schlussfolgerungen und Ausblick

Die Problemstellen im Unterricht haben eine Überarbeitung des Unterrichtsmodells eingeleitet, das im WS 2007/2008 mit Lehramtsstudierenden erneut erprobt werden soll. Neben Zugriffskontrolle wird das Unterrichtsprojekt erweitert um Themen wie Zustände, Kompatibilität und Zuverlässigkeit von Informatiksystemen. Als Entwurfsmuster bietet sich neben dem Zustandsmuster insbesondere der Adapter an, dessen Potential auf der Vernetzung von Systemkomponenten liegt. Da sich das Identifizieren der informatischen Konzepte allein anhand des nach außen sichtbaren Verhaltens eines Informatiksystems anfangs als kognitive Hürde erwies, soll als weitere Vorgehensweise systematisches Testen verwendet werden: D. h. Aufgaben, die sowohl das Verändern des Systemverhaltens durch unterschiedliche Eingaben als auch durch Modifikation der inneren Struktur erfordern. Damit werden unterschiedliche Sichten auf ein Informatiksystem genutzt, die ein umfassenderes Verstehen ermöglichen. Die Lernsoftware „Pattern Park“ kann den Lehr-Lern-Prozess diesbezüglich unterstützen. Darüber hinaus soll bei Aufgaben im Unterrichtsprojekt eine Stärkung des Beobachtens vorgenommen werden.

Die gesellschaftlichen Auswirkungen des Einsatzes der Informatiksystemen sind mit Blick auf die aktuelle Diskussion um Bildungsstandards für die Informatik in das Unterrichtsmodell zu integrieren. Eine Schwierigkeit bei der Auswertung der Schülerlösungen war, dass gerade die aus der objektorientierten Modellierung bekannten Fehlvorstellungen, wie Unterscheidung zwischen Klasse und Objekt, auftraten und möglicherweise Fehlvorstellungen bezüglich Informatiksystemen überlagerten. Um diese dennoch identifizieren zu können, wird deshalb die kognitionspsychologische Methode des Laut-Denkens in einer Vorstudie von Lehramtsstudierenden an drei typischen Aufgaben zum Verstehen von Informatiksystemen evaluiert.

Literaturverzeichnis

[AK01] Anderson, L. W.; Krathwohl, D. R. (Hrsg.): A taxonomy for learning, teaching and assessing: A revision of Bloom's Taxonomy of educational objectives. Addison Wesley Longman, New York, 2001.

[At06] Atteslander, P.: Methoden der empirischen Sozialforschung. 11. Auflage, Erich Schmidt Verlag, Berlin, 2006.

[Ba99] Bassey, M.: Case study research in educational settings. Open University Press, UK, 1999.

[BD02] Bortz, J; Döring, N.: Forschungsmethoden und Evaluation für Sozialwissenschaftler. 3. Auflage. Springer. Berlin, 2002.

[BS02] Brinda, T; Schubert, S.: Didactic System for Object-oriented Modelling. In: Watson, D.; Andersen, J. (eds.): Networking the Learner. Computers in Education. Kluwer Academic Publisher, Boston, 2002; S. 473-482.

[CS03] Claus, V.; Schwill, A.: Duden Informatik. Ein Fachlexikon für Studium und Praxis. Duden Verlag, Mannheim, 3. Auflage, 2003.

[EPA04] KMK – Ständige Konferenz der Kultusminister der Länder in der Bundesrepublik Deutschland. Einheitliche Prüfungsanforderungen in der Abiturprüfung Informatik. Url: http://www.kmk.org/doc/beschl/EPA-Informatik.pdf (27.04.2007), 2004.

[FS06] Freischlad, S.; Schubert, S.: Media Upheaval and Standards of Informatics. In: IFIP TC3 / WG 3.1, WG 3.3 & WG 3.5 Joint Conference "Imagining the future for ICT and Education". Alesund, Norway, 2006.

[Ga95] Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.: Design Patterns. Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, MA, 1995.

[Hu05] Hubwieser, P.: Von der Funktion zum Objekt – Informatik für die Sekundarstufe I. 11. GI-Fachtagung Informatik und Schule INFOS, 2005; S. 27-41.

[LL07] Bildungsserver Nordrhein-Westfalen: Objektorientierte Programmierung. URL: http://www.learnline.de/angebote/oop/ (28.01.2007), 2007.

[PP07] Franke, D.; Freischlad, S.; Friedrich, L.; Haug, F.; Klein, B.; Koslowski, R.; Stechert, P.; Ufer, J.: Projektgruppe Pattern Park, Universität Siegen, URL: http://www.die.informatik.uni-siegen.de/pgpatternpark/ (24.04.2007), 2007.

[Sc06] Schneider M.: Functional modelling, fundamental ideas and threads in the subject informatics; In Proceedings of Second International Conference on "Informatics in Secondary Schools. Evolution and Perspectives – ISSEP", Vilnius, Lithuania, 2006; S. 413-423.

[Sc93] Schwill, A.: Fundamentale Ideen der Informatik. Zentralblatt für Didaktik der Mathematik 1, 1993; S. 20-31.

[St06a] Stechert, P.: Unterrichtsmodellentwicklung zur Förderung des Informatiksystemverständnisses mit Entwurfsmustern. In Lecture Notes in Informatics: Schwill, A.; Schulte, C.; Thomas, M. (Hrsg.), 3. Workshop der GI-Fachgruppe Didaktik der Informatik, 2006; S. 89-98.

[St06b] Stechert, P.: Informatics System Comprehension – A learner-centred cognitive approach to networked thinking. In: Education and Information Technologies 11 (2006) 3, Springer Netherlands, 2006.

[St06c] Stechert, P.: Understanding of Informatics Systems – A theoretical framework implying levels of competence. In Proceedings of the 6th Baltic Sea Conference on Computing Education Research - Koli Calling, Koli, Finnland, November 9-12, 2006; S. 128-131.

Anwenden und Verstehen des Internets – eine Erprobung im Informatikunterricht

Stefan Freischlad

Didaktik der Informatik
Universität Siegen
Hölderlinstraße 3
57076 Siegen
freischlad@die.informatik.uni-siegen.de

Abstract: In diesem Beitrag geht es um die Erprobung eines Unterrichtskonzepts zum Thema „Strukturen des Internets“, die Teil eines größeren Forschungsprojekts ist. Das Ziel ist die Entwicklung eines Didaktischen Systems „Internetworking“. In diesem Kontext wurde eine vierwöchige Untersuchung im Informatikunterricht in der Sekundarstufe II durchgeführt. In diesem Artikel werden die Lernziele mit Wissensstrukturen begründet. Die Auswertung der Konzeption erfolgt auf der Grundlage der Beobachtungen im Unterricht und der Ergebnisse eines Abschlusstests. Es werden die Schwierigkeiten beschrieben und daraus Rückschlüsse zur Überarbeitung der Konzeption entwickelt.

1 Motivation

Das Ziel des Forschungsprojektes[1], zu dem dieser Artikel einen Beitrag leistet, ist es, das Didaktische System „Internetworking“ zu entwickeln, wobei die theoretischen Ansätze durch den Einsatz im Informatikunterricht erprobt werden. Dabei geht es um drei Komponenten des Didaktischen Systems: Wissensstrukturen, Aufgabenklassen und lernförderliche Unterrichtsmittel. Dazu haben wir Kompetenzen, die im Umgang mit dem Internet im Alltag benötigt werden, bestimmt. Eigenschaften von Informatiksystemen wurden im Zusammenhang von Internetanwendungen untersucht [Fr06b]. Ebenfalls notwendige Vorkenntnisse zu dem Thema Rechnernetze und zur Informationssicherheit wurden spezifiziert. Aus dieser Analyse haben wir einen theoretischen Ansatz für die Wissensstrukturen entwickelt. Zur Erprobung wurden Unterrichtsmaterialien, d.h. Aufgaben, Software für den Einsatz im Informatikunterricht und Unterrichtsentwürfe konzipiert und entwickelt. Die Evaluation der theoretischen Ansätze erfolgt unmittelbar durch die Materialien im Rahmen der Unterrichtserprobung. Dazu werden Beobachtungen während des Unterrichts, eine abschließende Lernerfolgskontrolle, eine Akzeptanzbefra-

[1] Das Forschungsprojekt „Informatikunterricht und E-Learning zur aktiven Mitwirkung am digitalen Medienumbruch“ wird im Zeitraum 7/2005-6/2009 von der Deutschen Forschungsgemeinschaft (DFG) gefördert.

gung und Selbsteinschätzung der Lernenden[2] und ein Interview des Lehrers genutzt. Die Ergebnisse fließen in die Entwicklung des didaktischen Systems und in die Konzeption für eine weitere Erprobung ein. In diesem Artikel geht es um die zweite Unterrichtserprobung als Teil des Projekts [FS06].

2 Theoretisches Konzept

2.1 Forschungsmethodik

Das Ziel der Erprobung im Unterricht ist es, eine Untersuchung zur Theoriebildung durchzuführen. Unsere Erwartungen an die Ergebnisse sind Aufschlüsse darüber, wie die Lernenden das Wissen verknüpfen bzw. wie sich die Lernphasen strukturieren lassen, welche Vorkenntnisse notwendig sind, um Abläufe im Internet zu verstehen, und welche Aufgabenklassen geeignet sind, zur Erarbeitung im Unterricht eingesetzt zu werden. Im ersten Unterrichtsprojekt nahm der Forscher eine beobachtende Teilnehmerrolle ein. In Kooperation mit dem Hochschullehrer und dem Lehrer betreute er Lehramtsstudierende bei der Durchführung des fachdidaktischen Praktikums. In der zweiten Erprobung nahm der Forscher die Rolle des Lehrenden ein. Diese intervenierende Unterrichtsforschung erfordert eine kritische Reflexion. Atteslander beschreibt dazu vier Problemkreise, die zur qualitativen teilnehmenden Forschung zu klären sind: „zum einen müssen die Teilnehmerrollen so offen und flexibel zu handhaben sein, dass der Forscher im Feld agieren und reagieren kann, zum Zweiten müssen die Rollen dem Feld entsprechen bzw. in diesem bereits angelegt sein, damit das Feld durch die Forschung nicht verändert wird, drittens muss überlegt werden, ob die Forscherrolle offen gelegt wird oder teilweise bzw. ganz verdeckt bleibt und viertens muss das Verhältnis zwischen Forscher- und Teilnehmerrolle (Distanz und Teilnahme) geklärt werden“ [At03, S. 109f]. Die Teilnehmerrolle als Lehrender hat demzufolge den Vorteil, dass der Forscher flexibel agieren und reagieren kann. Das Problem, dass die Rolle bereits angelegt sein muss, spricht für die aktive Teilnahme als Lehrender, wie sie in der zweiten Erprobung praktiziert wurde. Allerdings nimmt dann der Lehrer und in unserem Fall ein hospitierender Forscher eine nicht angelegte Rolle ein. Dies kann sich in einer aktiveren Teilnahme durch Schüler auswirken, die sich der besonderen Beobachtungssituation bewusst sind. Diese Situation trat in beiden Erprobungen ebenso wie die Offenlegung der Forscherrolle auf.

Im Gegensatz zur quantitativen Untersuchung wird die Teilnahme durch den Forscher an der qualitativen Studie mit Vorteilen verbunden, obwohl Kritik bezüglich der Repräsentativität und Wissenschaftlichkeit von Daten, die auf diese Weise gewonnen werden, geübt wird. Atteslander sagt dazu: „Eine solche Kritik verkennt aber die genuinen Vorzüge dieser Methode, denn qualitativ-teilnehmende Beobachtungen zeichnen sich gegenüber anderen Methoden ja gerade durch die Authentizität der gewonnen Daten aus“ [At03, S. 113]. Sowohl zur Unterrichtsvorbereitung wie auch zur Nachbereitung fanden

[2] Im Text werden weitgehend geschlechtsneutrale Formulierungen gewählt. Wenn die männliche Form verwendet wird, so sind damit immer Frauen und Männer gleichermaßen gemeint, sofern nicht explizit das Gegenteil behauptet wird.

zudem Gespräche mit den Lehrern und der Fachgruppe Didaktik der Informatik und E-Learning der Universität Siegen statt.

2.2 Begründung der Lernziele

Wichtiger Bestandteil des Didaktischen Systems sind die Wissensstrukturen. Brinda beschreibt drei didaktische Funktionen [Br04]. Die Wissensstrukturen veranschaulichen die fachlichen Zusammenhänge. Damit sind sie zum einen ein Gestaltungsmittel für Lehr-Lern-Prozesse und können zugleich auch für Lernende Orientierung im Lehr-Lern-Prozess bieten. Die Möglichkeit der fachdidaktischen Kommunikation und Diskussion ist die dritte Funktion. In Abbildung 1 wird der Ansatz für die Wissensstruktur, die Grundlage zur zweiten Erprobung war, dargestellt. In Tabelle 1 werden die in den Wissensstrukturen angeordneten Grobziele durch Feinlernziele präzisiert.

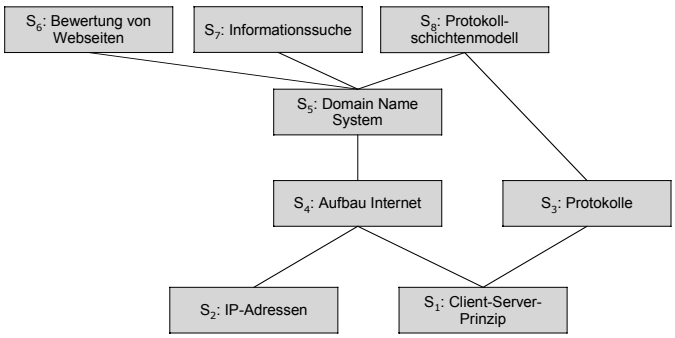


Abbildung 1: Wissensstruktur zu „Strukturen des Internets“

Als Vorwissen aus dem Bereich Rechnernetze wird das Client-Server-Prinzip am Beispiel des Schulrechnernetzes aufgegriffen. Neben den Funktionen wie zentraler Druckerzugriff und Dateizugriff geht es um die Verbindung zur Datenübertragung zwischen zwei Rechnern. Zunächst müssen die Lernenden erkennen, dass ein Server nicht nur einen besonders leistungsfähigen Rechner bezeichnet, sondern auch ein Programm oder einen Prozess bezeichnet, das bzw. der auf eingehende Anfragen von einem Client reagiert. Als Grundlage für die Interaktion der Prozesse wird eine logische Verbindung zwischen den Programmen benötigt. Anknüpfend an die Rollen von Rechnern im Rechnernetz wird deren Adressierung mit IP-Adressen thematisiert. Dazu müssen sie verstehen, dass beim Datenaustausch eine global eindeutige Adresse benötigt wird. Durch die Verbindung von IP-Adresse und der physischen Struktur des Internets wird das verdeutlicht. Nur mit einer IP-Adresse können Daten einen Rechner im Internet erreichen. Um den Zusammenhang zwischen IP-Adresse, Rechner- und Netz-ID beschreiben zu können, wird die Netzmaske benötigt. Die Berechnung der einzelnen Komponenten vertieft das Verständnis für den Aufbau der Adressen. Der Zusammenschluss mehrerer Netze zum Internet verdeutlicht den Charakter des Netzes der Netze. Dabei werden auch der Zusammenhang zu den IP-Adressen und die Unterscheidung von weltweit eindeutiger und privater IP-Adresse aufgezeigt. Die Möglichkeit zur Verwendung privater IP-Adres-

sen wird gerade in einem Heimnetzwerk mit einem Internetzugang durch einen Router häufig genutzt. Die Unterscheidung ist für ein Verständnis dafür notwendig, dass auch bei der Verwendung einer selbst vergebenen IP-Adresse eine weltweit eindeutige IP-Adresse zum Datenaustausch im Internet notwendig ist. Anknüpfend an das Client-Server-Prinzip geht es darum, ein Protokoll als Spezifikation von Vorschriften zum Datenaustausch zu verstehen. Das Thema soll am Beispiel des Hypertext Transfer Protocol (HTTP) betrachtet werden. Zur Formalisierung der Abläufe, die durch ein Protokoll beschrieben werden, wird das Interaktionsdiagramm verwendet. Das Domain Name System (DNS) wird nach der Adressierung im Internet durch IP-Adressen betrachtet. Die Verbindung zur IP-Adressierung und das Verständnis dafür, dass durch den Namensraum eine logische Strukturierung des Internets vorgenommen wird, sind wichtige Teillernziele. Außerdem wird so deutlich, dass auf Grundlage des Domainnamens keine geografische, aber eine organisatorische Zuordnung getroffen werden kann. Am Protokollschichtenmodell werden Funktionen der verschiedenen Protokolle besprochen. Dazu wird die Verbindung zur Funktionsweise des World Wide Web und zur Verbindung von Programmen in einem Rechnernetz hergestellt. Das Protokollschichtenmodell ermöglicht eine strukturierte Sicht auf das Internet. Neupert und Friedrich begründen die Thematisierung des Schichtenmodells mit dem sicheren Umgang in Fehlersituationen: „der Schüler kann nur durch Kenntnisse über Netzwerkstruktur und deren Schichtenmodell das Entstehen des Fehlers erklären und richtige Handlungen daraus für sich ableiten“ [NF97, S. 19f]. Dafür müssen die Lernenden den einzelnen Schichten Funktionen zuordnen können. Aufbauend auf ein Verständnis der Funktionsweise des World Wide Web werden Kriterien für die Bewertung von Webseiten erarbeitet und benutzt. Dabei können die Kenntnisse zur Hierarchie des DNS-Namensraumes mit dem URL und zu den Verzeichnisdiensten als Kriterien angewendet werden. Außerdem werden die Funktionsweise von Suchmaschinen und verschiedene Suchstrategien besprochen. Zu einem Verständnis sind das Client-Server-Prinzip und das HTTP für die Funktionsweise des Web-Roboters und der WWW-Benutzungsschnittstelle erforderlich.

Auf der Grundlage dieser Strukturierung der Lernphasen und Lernziele wurde der Unterricht geplant. Neben den informatischen Konzepten wie Protokolle, Verzeichnisdienste und Adressierung, wurde daran anknüpfend die Anwendung WWW zur Recherche im Internet aufgegriffen. Dabei ist es insbesondere für die abstrakten Konzepte notwendig, angemessene Zugänge zu nutzen. „Die Methode der Abstraktion ist oft nützlich, sie macht aber einem Neuling das Leben schwer. […] Je älter wir werden, umso zugänglicher sind wir für Erklärungen, die bildlich oder lediglich in Textform vorliegen“ [Ha05, S. 54]. Hartmann unterscheidet dazu nach Bruner die drei Repräsentationsebenen enaktiv, ikonisch und symbolisch. Enaktive Repräsentationen eignen sich besonders für den Einstieg in ein neues Thema. Obwohl die Erprobung in Jahrgangsstufe 13 durchgeführt wurde, haben wir zu mehreren Themen einen enaktiven Einstieg benutzt. Zum Thema HTTP haben wir beispielsweise eine Webseite mit Netcat, einem Programm zum Aufbau einer TCP/IP-Verbindung, eine Verbindung zu einem Webserver aufgebaut, eine Webseite abgerufen und mit dem Webbrowser angesehen. In diesem Fall sprechen Hartmann, Näf und Reichert von einer semi-enaktiven Repräsentation [HNR06, S. 117], weil es in Form einer Demonstration durch den Lehrer stattfand. Zu anderen Themen, wie beispielsweise zum Aufbau des Internets, haben wir eine ikonische Repräsentation als Zu-

gang mit einer Grafik, in der mehrere miteinander verbundene Rechnernetze dargestellt werden, gewählt.

Grobziele	**Die Schüler**
S_1: Client-Server-Prinzip	- können die Begriffe Client und Server erklären. - können das Client-Server-Prinzip zur Rechner-Rechner-Interaktion an Beispiel der Übertragung einer Datei beschreiben. - kennen die Möglichkeit des Datenaustauschs mit einer TCP/IP-Verbindung.
S_2: IP-Adressen	- können den Aufbau einer IP-Adresse aus Netz-Identifkationsnummer (Netz-ID) und Rechner-Identifikationsnummer (Rechner-ID) erklären. - können an einem Beispiel beschreiben, wie die Netzmaske zur Aufteilung einer IP-Adresse benutzt wird. - können Netz-ID und Rechner-ID mit der Netzmaske berechnen.
S_3: Protokolle	- können den Datenaustausch zwischen Webserver und Webclient mit Hilfe des Hypertext Transfer Protocol beschreiben. - können Protokolle als standardisierte Menge von Operationen für den Datenaustausch und für sonstige Aktivitäten in Rechnernetzen erklären. - können die Rechner-Rechner-Interaktion mit Interaktionsdiagrammen beschreiben.
S_4: Aufbau Internet	- können den Aufbau des Internets aus mehreren Rechnernetzen, die mit Vermittlungsrechnern verbunden sind, beschreiben. - können erklären, welche Information zum Standort eines Rechners in einer IP-Adresse steckt. - können private und öffentliche IP-Adresse unterscheiden. - können Network Address Translation (NAT) erklären.
S_5: Domain Name System	- können den Ablauf zum Auflösen eines Domainnamens in eine IP-Adresse beschreiben. - können den Namensraum des DNS als logische Struktur des Internets erklären. - können einen Domainnamen einer Person zuordnen.
S_6: Bewertung von Webseiten	- können die Teile eines Uniform Resource Locator (URL) erklären. - können mindestens 5 Kriterien zur Bewertung der Glaubwürdigkeit von Webseiten anwenden. - können mehrere Methoden zur Bestimmung von Daten über eine Webseite anwenden.
S_7: Informationssuche	- können wenigstens zwei Strategien zur Informationssuche im WWW beschreiben und anwenden. - können die Funktion der Komponenten einer Suchmaschine – Web-Roboter, Index und Benutzungsschnittstelle – erklären. - können den Ablauf zur Verarbeitung einer Anfrage an eine Suchmaschine beschreiben.
S_8: Protokollschichtenmodell	- können das Konzept des Schichtenmodells mit einem Beispiel erklären. - können die Schichten des Internet-Referenzmodells benennen. - können jeder Schicht zumindest eine Funktion zuordnen. - verstehen die Bedeutung der Vermittlungsschicht für den Austausch von Daten im Internet.

Tabelle 1: Grobziele und Feinlernziele zu „Strukturen des Internets“

3 Praxiserfahrungen

3.1 Lerngruppe und Rahmenbedingungen

Die erste Erprobung im Rahmen des Forschungsprojekts wurde durch Lehramtsstudierende für das Fach Informatik am Gymnasium im Rahmen des fachdidaktischen Praktikums im Sommersemester 2006 durchgeführt [Fr06a]. Dieses Unterrichtsprojekt fand in der Jahrgangsstufe 11 statt. Es umfasste sieben Doppelstunden. Im Anschluss wurden ein Abschlusstest und eine Akzeptanzbefragung unter den Schülern durchgeführt. Der betreuende Lehrer wurde nach der Erprobung in einem Interview zu seiner Einschätzung befragt. Inhaltliche Schwerpunkte in diesem Unterrichtsprojekt waren Authentifizierung und Vertraulichkeit im Kontext von E-Mail, grundlegende Konzepte wie das Client-Server-Prinzip und Protokolle sowie das Thema Schutz der Privatsphäre im Zusammenhang von Cookies. Im zweiten Halbjahr 2006 haben wir das zweite Unterrichtsprojekt in einem anderen Gymnasium durchgeführt, in der das Thema „Strukturen des Internets“ den Schwerpunkt gebildet hat. Hier konnten bereits Erfahrungen aus dem ersten Unterrichtsprojekt einfließen. Im Weiteren geht es um diese zweite Erprobung.

Das Unterrichtsprojekt im Wintersemester 2006/07 fand in einem Grundkurs Informatik in der Jahrgangsstufe 13 statt, der sich aus 4 Schülerinnen und 15 Schülern zusammensetzte. Vorwissen und Erfahrungen der Kursteilnehmer waren sehr unterschiedlich. Einer der teilnehmenden Schüler betreute das Schulrechnernetz. Ein weiterer Teilnehmer hat immer wieder technische Details beispielsweise zum Aufbau der Header, wie sie durch Protokolle spezifiziert werden, in den Unterricht eingebracht. Aus dem Informatikunterricht waren in dem Kurs keine Vorkenntnisse im Bereich Internet, Rechnernetze und Informationssicherheit vorhanden. Grundlegende Begriffe wie beispielsweise Client und Server mussten im Rahmen der Erprobung im Unterricht eingeführt werden. Vor dem Unterrichtsprojekt wurde in dem Kurs eine Software in Projektarbeit entwickelt. Insgesamt haben wir 12 Unterrichtsstunden durchgeführt und eine weitere Unterrichtsstunde für einen Abschlusstest genutzt. Der dreistündige Kurs setzte sich aus einer Einzelstunde und einer Doppelstunde zusammen. Das Projekt wurde durch einen Feiertag und eine Vortragsreise des Autors für eineinhalb Wochen unterbrochen.

3.2 Auswertung der Lernphasen

In diesem Unterrichtsprojekt standen die Schülertätigkeiten im Vordergrund. Lehrervorträge wurden selten und kurz gehalten. Überwiegend haben die Schüler in Einzel- und Partnerarbeit an Arbeitsaufträgen gearbeitet. In Unterrichtsgesprächen wurde die Verbindung zu Alltagserfahrungen hergestellt und sowohl die Ergebnisse der Schülertätigkeiten besprochen, wie auch weitere Gesichtspunkte diskutiert. Auf einen programmiersprachlichen Zugang haben wir aus zwei Gründen verzichtet. Der erste Grund ist, dass es uns in erster Linie darum geht, ausgehend von Internetanwendungen ein Verstehen der zugrunde liegenden Konzepte und nicht Gestaltungskompetenzen für Informatiksysteme zu fördern. Der zweite Grund ist, dass die Umsetzung der Konzepte weitestgehend unabhängig vom programmiersprachlichen Vorwissen der Lernenden sein soll. In der ersten Erprobung haben wir einen Arbeitsauftrag gegeben, in dem es darum ging, den

Quelltext eines SMTP-Clients geringfügig zu manipulieren. Trotz Programmiererfahrungen mit der Programmbibliothek „Stifte und Mäuse“ [OOP] fiel es den Lernenden schwer, sich in der Klassenstruktur zurechtzufinden. Im Folgenden werden die einzelnen Lernphasen und die aufgetretenen Schwierigkeiten ausgewertet, wie sie im Unterricht umgesetzt wurden bzw. auftraten.

Client-Server-Prinzip

Am Beispiel des Schulservers wurde der Nutzen von vernetzten Rechnern durch den Zugriff auf zentral bereitgestellte Ressourcen dargestellt. Als zentrales Prinzip wurde die Datenübertragung zwischen Programmen über die Grenzen eines Rechners untersucht. Mit Netcat, einem einfachen Werkzeug zum Aufbau einer TCP/IP-Verbindung, wurde in Partnerarbeit eine solche Verbindung zwischen Programmen hergestellt. Der Umgang mit Netcat hat sich als schwierig erwiesen, weil für die meisten Schüler die Eingabeaufforderung unter Windows nicht bekannt war. Anhand der Unterscheidung zwischen einem Programm, das auf eingehende Verbindungen wartet und zuerst gestartet wird, und dem Programm, das eine Verbindung zum wartenden Programm aufbaut, konnten die Begriffe Client und Server im Unterrichtsgespräch dennoch näher bestimmt werden. Von den Schülern wurde aus den Eigenschaften konkreter Beispiele für Client und Server eine verallgemeinerte Begriffsbeschreibung selbstständig formuliert.

Adressen und physikalische Struktur des Internets

Anknüpfend an Erfahrungen im Umgang mit dem WWW wurde das Thema der Adressierung von Rechnern im Internet, auf denen ein Server läuft, thematisiert. In Einzelarbeit haben die Schüler an Hand eines kurzen Textes den Aufbau einer IP-Adresse und die Bedeutung von Rechner- und Netz-Identifikationsnummer und Netzmaske in Stichpunkten zusammengefasst. Nach einem gemeinsam bearbeiteten Beispiel zur Berechnung der Teile einer IP-Adresse, haben die Schüler die Berechnung der Teile für die IP-Adresse des Rechners an ihrem Arbeitsplatz durchgeführt. Große Schwierigkeiten bereitete die Umrechnung vom Dezimal- zum Binärsystem. Das Vorwissen der Schüler, das sie im Kontext der Implementierung eines Algorithmus zur Umrechnung erworben hatten, konnten nur einzelne anwenden. Abschließend wurde eine Aufgabe bearbeitet, in der eine Grafik zur Darstellung mehrerer Rechnernetze mit End- und Zwischensystemen mit fehlenden IP-Adressen ergänzt wurde. Dazu war es notwendig, die Netz-Identifikationsnummern zu berechnen.

Anwendungsprotokolle

Am Beispiel HTTP wurde die Interaktion zwischen Client und Server mit Protokollen untersucht. Nach der oben beschriebenen Demonstration zum Abruf einer Webseite über eine TCP/IP-Verbindung haben die Schüler ein Interaktionsdiagramm erstellt. Die Repräsentation mit einem Interaktionsdiagramm hat keine Schwierigkeiten bereitet. Die Antwort auf die Frage nach einer Beschreibung des Begriffs Protokoll machte jedoch deutlich, dass eine klare Abgrenzung zu der Begriffsbedeutung im Sinne eines Stundenprotokolls notwendig ist.

Domain Name System

Zur vierten Unterrichtseinheit – einer Einzelstunde – war geplant, das Thema DNS zu behandeln. Anschließend sollten die Lernenden Kriterien zur Bewertung von Webseiten kennen lernen. Tatsächlich konnte das Thema DNS nicht abgeschlossen werden. Deshalb wurde für dieses Thema auch noch die erste Stunde der darauf folgenden Doppelstunde verwendet. Zum Thema Namens- und Verzeichnisdienste, wobei das Domain Name System (DNS) im Vordergrund stand, haben die Schüler über ein Web-Portal Information zu Domainnamen (DNS) und IP-Adressen (Whois) gesammelt. Am Ablauf einer Domainnamensauflösung wurde die logische Struktur des Namensraumes beschrieben. In einer Aufgabe wurde dazu ein Baum erstellt, in dem die logische Struktur direkt abgebildet wurde. Die Schüler konnten in der Baumdarstellung die hierarchische Struktur nicht erkennen. Ein typischer Fehler war, dass unterhalb des Wurzelknotens für jeden zu ergänzenden Domainnamen der Top-Level-Domain „de“ ein neuer Knoten verwendet wurde. Am Beispiel der Top-Level-Domain „vu“ haben sich die Schüler darüber informiert, welche Aussage über den Standort eines Rechners im Domainnamen steckt. Außerdem wurde besprochen, warum eine IP-Adresse deshalb besser zur Datenübertragung geeignet ist. Dazu wurde eine Grafik genutzt, in der mehrere Rechnernetze durch Router miteinander verbunden sind. In diesem Zusammenhang wurde die Frage gestellt, was diese Grafik mit dem Internet zu tun habe. Die Charakteristik des Internets als Zusammenschluss mehrerer Rechnernetze hatten demzufolge noch nicht alle Schüler verstanden. Nach dieser Lernphase haben die Lernenden die Zusammensetzung eines Uniform Resource Locators (URL) verstanden und konnten diese erklären.

Recherche im WWW

Aufbauend auf das Verständnis der Funktionsweise des WWW wurde im Vergleich zur Literaturrecherche in der Schulbibliothek im Unterrichtsgespräch die besondere Bedeutung davon herausgestellt, wie wichtig eine Bewertung von Webseiten ist. Anhand einer selbst gewählten Webseite zu einem vorgegebenen Thema haben die Schüler Kriterien zur Bewertung der Webseite formuliert. Im Unterrichtsgespräch wurden die Kriterien gesammelt und systematisiert. Ein Anknüpfungspunkt an die vorhergehenden Themen war die Bedeutung des Domainnamens in einem URL. Außerdem bekamen die Schüler den Arbeitsauftrag, zu offenen und geschlossenen Fragen zu recherchieren und ihr Vorgehen zu dokumentieren. Mit der „didaktischen Suchmaschine“ Soekia [Dr06] wurde der Ablauf zur Erstellung eines Suchindexes und zur Bearbeitung einer Suchanfrage durch eine Suchmaschine analysiert. Eine intensivere Auseinandersetzung mit diesem Thema durch die Schüler war wegen der zeitlichen Rahmenbedingungen nicht möglich. Ein Aktivitätsdiagramm zu dem Ablauf vom Webdokument zum Suchergebnis konnte nicht mit den Lernenden erarbeitet werden sondern wurde lediglich im Unterrichtsgespräch besprochen.

Protokollschichtenmodell

Zuletzt ging es darum, das Internet-Referenzmodell kennen zu lernen und den Schichten verschiedene Funktionen zuzuweisen. Am Beispiel Netcat wurde deutlich, dass Vorgänge in unteren Schichten wie beispielsweise der Verbindungsaufbau zwischen zwei Programmen nicht beobachtbar sind. Mit einer Analogie zur Kommunikation zwischen zwei Personen über den Postversand, wurde im Unterrichtsgespräch das Konzept der Schich-

ten erarbeitet. An einem Applet zum Protokollschichtenmodell zur Veranschaulichung von Übertragungsfehlern [DIE03] haben die Schüler die Funktionen der verschiedenen Schichten analysiert.

Die Lernziele zur Unterscheidung von öffentlichen und privaten IP-Adressen und zu Network Address Translation (NAT) (S_4) konnten ebenso wie das Lernziel zur Bedeutung der Vermittlungsschicht für die Datenübertragung im Internet (S_8) wegen der zeitlichen Rahmenbedingungen nicht umgesetzt werden.

4 Erfahrungen für die weitere Forschungsarbeit aus dem Abschlusstest

4.1 Testgestaltung und Testdurchführung

Der Abschlusstest wurde mit zwei Varianten durchgeführt. Jede Variante bestand aus sechs Aufgaben. Die Aufgaben in den zwei Varianten bezogen sich mit einer Ausnahme auf die gleiche Thematik. Die Varianten unterschieden sich im Aufgabentyp und der Aufgabenreihenfolge. Der Test bestand aus Aufgaben zu folgenden Inhalten:

1. die Fachbegriffe Client, Server und Protokoll,
2. der Uniform Resource Locator,
3. Interaktion zwischen Client und Server am Beispiel WWW,
4. der Verzeichnisdienst DNS (Namensräume und Ablauf einer Anfrage),
5. Rechner-ID und Netz-ID zu einer IP-Adresse und der Aufbau des Internets,
6. die logische und die physikalische Struktur des Internets und
7. Verzeichnisdienste DNS und Whois.

Der Test wurde zu Beginn einer Doppelstunde durchgeführt. Die Ergebnisse der Lernerfolgskontrolle flossen in die Gesamtnote für den Informatikkurs ein. Die Schüler haben konzentriert an den Aufgaben gearbeitet. Es haben vier Schülerinnen und 14 Schüler am Test teilgenommen. Zur Bearbeitung hatten die Schüler 30 Minuten Zeit. Zwei Schüler beendeten die Bearbeitung wenige Minuten bevor die zur Verfügung stehende Zeit abgelaufen war.

4.2 Erkenntnisse für die weitere Unterrichtsgestaltung

Für das nächste Unterrichtsprojekt haben wir zum Protokollschichtenmodell mehr Zeit eingeplant. Das Wissen darüber soll mit dem Wissen über das Client-Server-Modell verknüpft werden. Die Aufgabe im Abschlusstest zur Interaktion zwischen Client und

Server am Beispiel des WWW (Aufgabe 3) bestand in einer Variante daraus, dass ein Interaktionsdiagramm für den Ablauf zur Abfrage einer Webseite erstellt werden musste. Dabei trat der Fehler auf, dass die Auflösung des Domainnamens nach einer Verbindungsanfrage vom Webserver ausgeht. Ein weiterer Fehler bestand darin, dass die Anfrage an den DNS-Server auch die Anfrage nach dem Port umfasste. Diese Fehler sind damit verbunden, dass die Bedeutung bzw. die verschiedenen Funktionen von IP-Adresse, Domainname und Port nicht verstanden wurden. Eine nähere Betrachtung des Protokollschichtenmodells im Zusammenhang einer Internetanwendung kann die Unterscheidung der Begriffe zur Adressierung verdeutlichen, indem die IP-Adresse der Vermittlungsschicht, der Port der Transportschicht und der Domainname der Anwendungsschicht zugeordnet werden.

Auch zum Thema DNS haben wir in der nächsten Erprobung eine zusätzliche Unterrichtsstunde eingeplant. DNS muss als Internetdienst verstanden werden, der unabhängig vom WWW ist. In Aufgabe 3 trat der Fehler auf, dass die Auflösung des Domainnamens als Interaktion zwischen Webclient und Webserver dargestellt wurde. Bezüglich der Darstellung der Domainhierarchie mit einem Baum muss zudem deutlicher werden, dass ein Knoten eine durch einen Server verwaltete Domain des DNS-Namensraumes ist. In Aufgabe 4 zum DNS ging es darum, Domainnamen in einer Baumhierarchie anzuordnen und den Ablauf der Auflösung eines Domainnamens an einem Beispiel zu beschreiben. Dabei trat der typische Fehler auf, dass gemeinsame Teilbäume nicht erkannt wurden. Außerdem ist eine Unterscheidung von DNS-Einträgen danach vorzunehmen, ob die IP-Adresse eines DNS-Servers oder die gesuchte IP-Adresse zurückgeschickt wurde. In Aufgabe 4 trat mehrfach der Fehler auf, dass der letzte Schritt zur Auflösung des Domainnamens durch zwei Abfragen beschrieben wurde, d. h., dass zunächst die IP-Adresse eines DNS-Servers für die den vollständigen Domainnamen zur Bezeichnung des Webservers zurückgegeben und danach erst von diesem DNS-Server die IP-Adresse des eigentlichen Webservers abgefragt wurde.

Ein typischer Fehler in Aufgabe 5 war eine fehlerhafte Berechnung der Teile einer IP-Adresse. Hierbei wurden die Schwierigkeiten mit der Binärdarstellung und der Verknüpfung von Netzmaske und IP-Adresse deutlich. In der Aufgabe zur IP-Adressierung mussten die Schüler die Rechner-ID und die Netz-ID zu einer IP-Adresse berechnen und die IP-Adresse einem von zwei Rechnernetzen zuordnen. Zudem musste das dazu notwendige Vorgehen beschrieben werden. Sechs von neun Teilnehmern konnten das Vorgehen zur Bestimmung des richtigen Rechnernetzes durch die Berechnung der Netz-ID nicht beschreiben. In der Aufgabe zu diesem Thema in der zweiten Variante wurde deutlich, dass vor allem die Berechnung für viele ein Hindernis darstellt. Es ist daher notwendig, weitere Aufgabenklassen zu diesem Thema einzusetzen, die nicht davon abhängig sind, dass die Umrechnung von Dezimal- zur Binärdarstellung einer IP-Adresse beherrscht wird.

Auch für das Thema Vermittlung von Daten im Internet müssen weitere Aufgabenklassen im Unterricht benutzt werden. Dieses Thema wurde ausschließlich im Zusammenhang mit dem Zusammenschluss mehrerer Rechnernetze zum Internet thematisiert, was nicht zu einem angemessenen Verständnis geführt hat. Die Unterscheidung von logischer und physischer Struktur des Internets musste in der Beantwortung der Frage da-

nach, warum zum Verbindungsaufbau zwischen zwei Rechnern im Internet nicht der Domainname sondern die IP-Adresse verwendet wird, angewendet werden. Nur ein Schüler konnte eine vollständige Begründung geben, in der mit Entscheidungen zur Datenvermittlung argumentiert wurde.

Die Möglichkeit über die Verzeichnisdienste beispielsweise über ein Webportal Information über einen Webserver bzw. über einen Domainnamen zu erhalten, muss im Unterricht mit weiteren Aufgabenklassen durchgeführt werden. In Aufgabe 7 mussten die Schüler beschreiben, wie man zu Daten zum Standort eines Webservers gelangen kann. Zwei Teilnehmer konnten diese Frage mit der Information einer IP-Adresse in Verbindung bringen. In keiner Antwort wurden die Verzeichnisdienste Whois und DNS angeführt.

5 Schlussfolgerungen

Durch die beschriebenen Beobachtungen während des Unterrichtsprojekts und die Auswertung des Abschlusstests, konnten verschiedene Schwierigkeiten im Lehr-Lern-Prozess aufgezeigt werden. Zur Strukturierung des Wissens über die Adressierung im Internet und für ein tieferes Verständnis der Interaktion zwischen Client und Server muss das Protokollschichtenmodell mit weiteren Aufgabenklassen integriert werden. Für ein Verständnis dafür, was ein Anwendungsprotokoll ist, haben sich in der ersten Erprobung die E-Mail-Protokolle Simple Mail Transfer Protocol (SMTP) und Post Office Protocol (POP3) – im Vergleich zu HTTP komplexere Protokolle – bewährt, deren Betrachtung auch die Verwendung eines Zustandsdiagramms als weitere ikonische Repräsentation zuließ. Im Kontext der Verzeichnisdienste muss die hierarchische Gliederung des DNS-Namensraumes mit einer Baumdarstellung und die Anwendung von Whois durch weitere Aufgabenklassen unterstützt werden.

Für das erste Halbjahr 2007 ist deshalb eine weitere Erprobung mit Lehramtsstudierenden in einem Kurs der Jahrgangsstufe 11 geplant. Die Ergebnisse aus den vorangegangenen Erprobungen werden in acht Wochen mit jeweils drei Unterrichtsstunden aufbauend auf den bisherigen Materialien umgesetzt. In der ersten Woche geht es um die Frage, wie Client und Server zunächst im Schulrechnernetz und schließlich im Internet Daten austauschen. Dazu wird HTTP als Beispiel betrachtet. In der zweiten Woche geht es um den Aufbau des Internets aus mehreren Rechnernetzen. Zentrale Fragestellung ist, ob es möglich ist, sich im Internet anonym zu bewegen. Dabei wird zum einen die Hierarchie der Internet Service Provider und zum anderen der Aufbau der IP-Adressen untersucht. In der dritten Woche geht es darum, wie es möglich ist, unter einem bekannten Domainnamen eine falsche Webseite zu erreichen. Dazu wird zunächst die logische Struktur durch den DNS-Namensraum und schließlich der Ablauf zur Auflösung eines Domainnamens untersucht. Internetrecherche wird im Anschluss daran mit den Schwerpunkten Funktionsweise einer Suchmaschine, Suchstrategien und Bewertung von Webseiten thematisiert. Zum Protokollschichtenmodell werden in der fünften Woche Verbindungen zum Wissen aus den vorangegangenen Stunden hergestellt. Unter der Fragestellung, wie es möglich ist, sich vor Spoofing-Mails zu schützen, wird am Beispiel E-Mail das Thema Authentifzierung sowie der Verzeichnisdienst Whois untersucht. Dazu werden die

Protokolle SMTP und POP3 und der Übertragungsweg einer E-Mail analysiert. Zudem wird die Bedrohung der Vertraulichkeit und Authentizität in zwei aufeinander folgenden Wochen umgesetzt. In der letzten Woche wird im Zusammenhang zum Thema Network Address Translation das Thema Firewall behandelt.

Um einen Zugang auch auf der enaktiven Repräsentationsebene zu Konzepten wie Aufbau des Internets, die für den Benutzer verborgen bleiben, zu schaffen, kann eine Lernsoftware eingesetzt werden. Eine solche Software wird im Rahmen einer studentischen Projektgruppe seit dem Sommersemester 2006 entwickelt. Sie ermöglicht es, ein virtuelles Rechnernetz mit ausgewählten Internetanwendungen aufzubauen, zu manipulieren und zu simulieren. Mehrere auf diese Weise erstellte Rechnernetze können über Rechnergrenzen hinweg verbunden werden. Sie stellt damit ein Mittel dar, den Informatikunterricht mit Rechnerexperimenten zum Thema Internetworking zu ergänzen.

Literaturverzeichnis

[At03] Atteslander, P.: Methoden der empirischen Sozialforschung. Walter de Gruyter, Berlin, 2003.

[Br04] Brinda, T. (2004) Didaktisches System für objektorientiertes Modellieren im Informatikunterricht der Sekundarstufe II. Dissertation, Universität Siegen, 2004.

[DIE03] Applet zum Protokollschichtenmodell.
URL: http://rvs.die.informatik.uni-siegen.de/Wilus/ProtokolleDienste/ContentContainer/sl_tc_jar_cong (Januar 2007)

[Dr06] Dreier, M.: Soekia – eine didaktische Suchmaschine.
URL: http://www.swisseduc.ch/informatik/soekia/ (Januar 2007)

[Fr06a] Freischlad, S.: Beitrag des Informatikunterrichts zur Entwicklung von Medienkompetenzen. In: Schwill, A.; Schulte, C.; Thomas, M. (Hrsg.), Didaktik der Informatik. Gesellschaft für Informatik, 2006, S. 29-38.

[Fr06b] Freischlad, S.: Learning Media Competences in Informatics. In: Proceedings of Second International Conference on "Informatics in Secondary Schools. Evolution and Perspectives - ISSEP", November 7-11, 2006 Vilnius, Lithuania, S. 591-599.

[FS06] Freischlad, S.; Schubert, S.: Media Upheaval and Standards of Informatics. In: Proceedings of IFIP Conference "Imagining the future for ICT and Education", June 26-30, 2006, Alesund, Norway.

[HNR06] Hartmann, W.; Näf, M.; Reichert, R.: Informatikunterricht planen und durchführen. Springer, Berlin, 2006.

[Ha05] Hartmann, W.: Informatik – EIN/AUS – Bildung. In: Friedrich, S. (Hrsg.), Unterrichtskonzepte für informatische Bildung. Gesellschaft für Informatik, 2005, S. 43-56.

[NF97] Neupert, H.; Friedrich, S.: Lernen mit Netzen – Lernen über Netze. LOG IN 17, Heft 6, S. 18-23, 1997.

[OOP] Objektorientierte Programmierung.
URL: http://www.learn-line.nrw.de/angebote/oop/ (Januar 2007)

Wiki und die fundamentalen Ideen der Informatik

Beat Döbeli Honegger

Institut für Medien und Schule
Pädagogische Hochschule Zentralschweiz Schwyz
Zaystrasse 42
CH-6410 Goldau
beat.doebeli@phz.ch

Abstract: Wiki eignet sich nicht nur als Werkzeug und Medium in der Schule. Anhand von Wiki lassen sich auch einige fundamentale Ideen der Informatik aufzeigen. InformatiklehrerInnen, die Wiki in ihrer Schule propagieren, fördern damit nicht nur den ICT-Einsatz im Unterricht, sondern erhalten auch motivierende Anknüpfungspunkte, um fundamentale Ideen der Informatik zu vermitteln. Dies mindert die Gefahr eines praxisfernen Informatikunterrichts und zeigt exemplarisch allgemeinbildende Aspekte der Informatik.

1 Informatik als Thema und Werkzeug/Medium in der Schule

Informatik spielt in der Schule seit langem eine Doppelrolle. Einerseits ist Informatik ein Thema des Unterrichts, andererseits sind Informatikmittel Werkzeuge und Medien des Unterrichts. Im ersten Fall steht also Informatik als Unterrichtsgegenstand im Vordergrund, der thematisiert werden soll. Im zweiten Fall der Nutzung von Informatikmitteln als Werkzeuge oder Medien im Unterricht sollten diese zugunsten eines anderen Unterrichtsthemas möglichst in den Hintergrund treten.

Aus historischer Sicht hat eine Verschiebung von Informatik als Thema zu Informatik als Werkzeug und Medium stattgefunden. Dies lässt sich unter anderem an den Bezeichnungen der von Forneck identifizierten Phasen der Informatikdidaktik im deutschsprachigen Raum ablesen [Fo90]: Hardwareorientierter Ansatz, Algorithmenorientierter Ansatz, Anwendungsorientierter Ansatz, Benutzerorientierter Ansatz.

Die Doppelrolle der Informatik in der Schule war und ist auch heute noch konfliktträchtig. Mit den Kapitelüberschriften „Informatikunterricht hat Informatik als Gegenstand“, „Informatiklehrer unterrichten Informatik“ und „Informatiklehrer sind keine ICT-Supporter“ zeigen Hartmann et al. in ihrem aktuellen Buch mögliche Reibungsflächen dieser nicht überall geklärten Doppelrolle in Schulen, die über einen Informatiklehrer verfügen [HNR06]. Oft werden Informatiklehrpersonen für Expertinnen oder Experten in strategischen oder gar operativen Fragen des Informatikmitteleinsatzes in Schulen gehalten. Umgekehrt wird in erschreckend vielen Schulen Informatik von Lehrpersonen mit keinerlei oder nur ungenügender Informatikbildung unterrichtet.

Diese Vermischung von „Informatik als Werkzeug und Medium“ und „Informatik als Thema“ geschieht oft aus Unkenntnis der Fachdisziplin Informatik und ihrer allgemeinbildenden Bedeutung durch Schulbehörden und Lehrerinnen und Lehrer anderer Fächer. Wollen Informatiklehrerinnen und Informatiklehrer diesem Missstand abhelfen, so sind Abgrenzung und Rückzug auf die eigene Fachdisziplin nicht die optimale Strategie. Stattdessen sollten Chancen genutzt werden, Lehrerinnen und Lehrern anderer Fächer die allgemeinbildenden Aspekte der Informatik praktisch aufzuzeigen. Wiki bietet dazu zahlreiche Gelegenheiten. Die einfache Nutzung und vor allem die geringe Einstiegshürde von Wikis erleichtern es, Lehrerinnen und Lehrer anderer Fächer für die Nutzung von Wiki als Kooperations- und Publikationsplattform im Unterricht zu gewinnen. Danach bieten sich zahlreiche Möglichkeiten, sowohl den Schülerinnen und Schülern als auch den Kolleginnen und Kollegen einige fundamentale Ideen der Informatik aufzuzeigen.

2 Wiki als Werkzeug in der Schule

Ein Wiki ist ein Webserver mit Versionsverwaltung, bei dem alle ohne zusätzliche Werkzeuge neben dem Webbrowser und ohne HTML-Kenntnisse Webseiten erstellen, verändern und zu einem Hypertext verknüpfen können. Wikis gehören derzeit zu den einfachsten Möglichkeiten, gemeinsam Webseiten zu erstellen, zu überarbeiten und zu publizieren. Aus diesem Grund werden Wikis seit vielen Jahren weltweit und auch im deutschsprachigen Raum auf allen Schulstufen eingesetzt. An dieser Stelle soll jedoch nicht weiter auf die didaktischen Möglichkeiten von Wikis eingegangen werden, stattdessen sei auf die vielfältige Literatur zu diesen Thema verwiesen. Zu den frühen englischsprachigen Publikationen in diesem Bereich gehören u. a. [Gu00] und [GRK01]. Im deutschsprachigen Raum gehören u. a. [HZ03], [Dö05], [Jo05], [Kl05] zu den ersten Publikationen. Eine aktuelle Aufstellung der didaktischen Potenziale von Wikis liefert z.B. [Dö07].

Im Vergleich zu früher in der Schule verwendeten Webeditoren zur Publikation auf dem Web erfordert die Nutzung von Wiki weniger Vorkenntnisse und ist damit auf den ersten Blick auch kein Thema für den Informatikunterricht. Verschiedene Erfahrungen des Wikieinsatzes in Schule und Hochschule haben aber gezeigt, dass auch zur Nutzung von Wiki gewisse Grundkonzepte der Informatik bekannt sein müssen. Fehlen diese Kenntnisse und Fertigkeiten, dann ist auch mit Wiki ein effizientes Arbeiten nicht möglich. Diese Grundkonzepte wiederum sind jedoch nicht wiki-spezifisch, sondern auch in anderen Umgebungen und Situationen anwendbar. Es handelt sich um fundamentale Ideen der Informatik.

3 Wiki als Thema in der Schule

3.1 Fundamentale Ideen der Informatik

Um langlebige Konzepte in der scheinbar schnelllebigen Informatik zu identifizieren, konkretisierte Schwill [Sc93] die Überlegungen zu fundamentalen Ideen von Bruner

[Br60] und adaptierte sie für die Informatik. Schwill definierte vier Kriterien, die eine fundamentale Idee erfüllen muss:

- Horizontalkriterium: Ein Sachverhalt ist in verschiedenen Bereichen vielfältig anwendbar oder erkennbar.
- Vertikalkriterium: Ein Sachverhalt kann auf jedem intellektuellen Niveau aufgezeigt und vermittelt werden.
- Zeitkriterium: Ein Sachverhalt ist in der historischen Entwicklung deutlich wahrnehmbar und bleibt längerfristig relevant.
- Sinnkriterium: Ein Sachverhalt besitzt einen Bezug zur Sprache und zum Denken des Alltags und der Lebenswelt.

Von Hartmann et al. [HNS99], [HNR06] stammt ein fünftes Kriterium:

- Repräsentationskriterium: Ein Sachverhalt lässt sich auf verschiedenen kognitiven Repräsentationsstufen (enaktiv, ikonisch, symbolisch) darstellen.

Zwar ist das Repräsentationskriterium zur Identifikation fundamentaler Ideen nicht zwingend erforderlich, da gewisse Überschneidungen mit dem Vertikalkriterium bestehen. Bei der Vermittlung fundamentaler Ideen hingegen leistet das Repräsentationskriterium gute Dienste, da es die Anschaulichkeit von Erklärungen durch drei unterschiedliche Repräsentationsformen fördert.

Im Folgenden sollen nun wesentliche fundamentale Ideen der Informatik in Wiki identifiziert werden.

3.2 Fundamentale Idee in Wiki: Hyperlinks

Das Verständnis von Hyperlinks ist für die Wiki-Nutzung zentral. Das Surfen auf traditionellen Webseiten ist eine passive Nutzung von vorhandenen Hyperlinks in einer meist eher hierarchischen und wohlgeordneten Struktur. Im Gegensatz dazu erfordert die Wiki-Nutzung auch das aktive Setzen von Hyperlinks und das Zurechtfinden in einem kollektiv aufgebauten Hypertext, dem unter Umständen zeitweise ein allgemein akzeptiertes Strukturierungsprinzip fehlt. Insbesondere, um die in Wiki wesentliche Strukturierungsmöglichkeit nutzen zu können, ist ein fundiertes Verständnis von Hyperlinks notwendig.

Hyperlinks sind ein zentrales Element des gesamten World Wide Web und auch vieler Informationssammlungen ausserhalb des Webs, z.B. in Lexika auf elektronischen Datenträgern oder in Buchform. Um diese Informationssammlungen effizient nutzen zu können, muss ein gewisses Grundverständnis von Hyperlinks vorhanden sein, womit das das Sinnkriterium von Hyperlinks als fundamentale Idee der Informatik erfüllt ist.

Querverweise in Büchern als Vorläufer von Hyperlinks zeigen, dass das Konzept Hyperlink bereits vor der Erfindung von Computern genutzt wurde. Damit ist auch das Zeitkriterium erfüllt. Da sich das Konzept von Hyperlinks jüngeren Lernenden zum Beispiel

mit Wegweisern oder roten Wollfäden als Verbindungen zwischen und in Dokumenten erklären lässt, ist auch das Vertikalkriterium erfüllt.

Solche Erklärungen können statt symbolisch auch ikonisch durch entsprechende Bilder oder enaktiv durch eigene Aktivitäten der Kinder, z.B. erwandern oder aufzeichnen einer bestehenden Hypertextgeschichte erfolgen (siehe Abbildung 1), womit das Repräsentationskriterium ebenfalls erfüllt ist. Mit der grossen Verbreitung von Hyperlinks im und ausserhalb des Internets ist schliesslich auch das Horizontalkriterium erfüllt, so dass das Konzept Hyperlink alle fünf Kriterien einer fundamentalen Idee erfüllt.

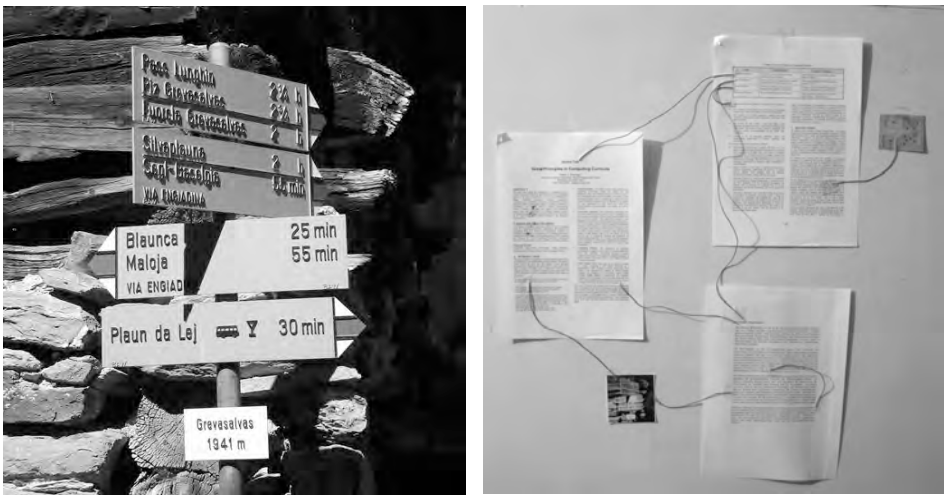


Abbildung 1: Wegweiser und Wollfäden als ikonische oder enaktive Repräsentationen des Hyperlink-Konzepts

Wiki bietet sich als Werkzeug an, um Lernende durch eigenes Handeln das Konzept von Hyperlinks erfahren zu lassen. So ist ein Wiki geradezu prädestiniert, um alleine oder in Gruppen Hypertextgeschichten zu schreiben. Laut Désilets und Paquet ist dies bereits mit Kindern in der 4. Klasse möglich [DP05].

Gewisse Wikiengines bieten sogar die Möglichkeit, die Struktur des erstellten Hypertexts grafisch darzustellen (siehe Abbildung 2). Eine solche Visualisierung wirkt für die Lernenden motivierend und bietet eine ikonische Darstellung des von den Lernenden virtuell-enaktiv erarbeiteten symbolischen Hypertextes.

3.3 Fundamentale Idee in Wiki: Namensräume

Auf grösseren Wikis wie z.B. bei der offenen Internet-Enzyklopädie Wikipedia, aber auch in Wikis von Schulen und Hochschulen werden meist mehrere, voneinander getrennte Bereiche gebildet. Diese Aufteilung erleichtert die Übersicht und ermöglicht bereichsspezifische Suchanfragen, Seitenauflistungen und Änderungsbenachrichtigungen. Solche Bereiche bilden meist auch getrennte Namensräume, was die Gefahr von

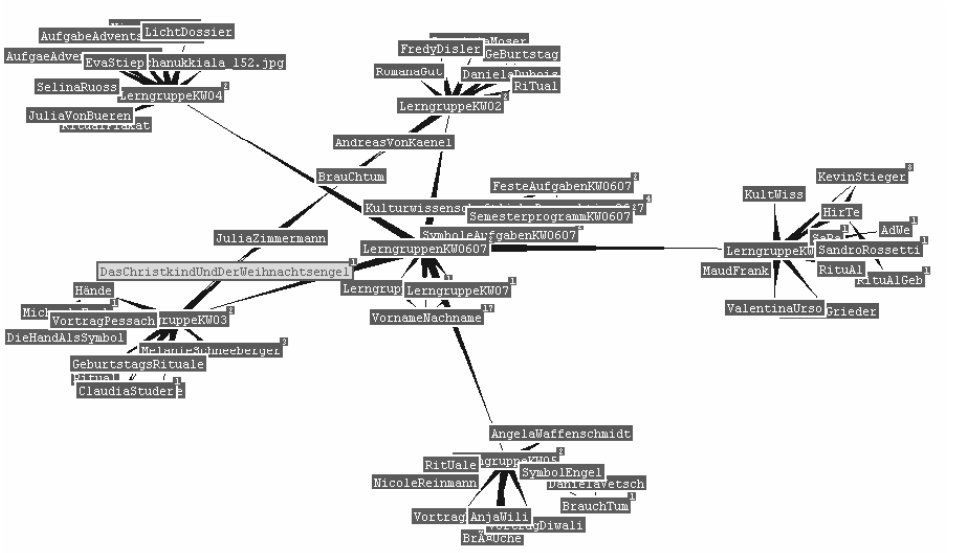


Abbildung 2: Dynamische Visualisierung des Hypernetzwerks eines Wikiprojekts mit Hilfe eines Java-Applets (TouchGraph)

Namenskonflikten bei beliebten Seitennamen (z.B. LiteraturListe, ArbeitsGruppen, StundenPlan) verringert.

Für Wiki-Neulinge ist diese Bereichsaufteilung grosser Wikis anfänglich eine Erleichterung, da sich dadurch die Komplexität des zuerst wahrgenommenen Systems scheinbar reduziert. Erst nach einer gewissen Einarbeitungszeit und zunehmender Wikierfahrung zeigt sich die Kehrseite dieser Modularisierung. In der Praxis zeigt sich dieses Problem beispielsweise, wenn jemand in mehreren Namensräumen arbeitet, ohne sich dessen bewusst zu sein. Der Versuch, einen Hyperlink von einer Wikiseite auf eine zweite Wikiseite in einem anderen Namensraum zu setzen, misslingt und führt zu einer Irritation. Verweise auf Seiten in anderen Namensräumen erfordern den Namen des Zielnamenraums als Präfix (Sport.UnterrichtsBeispiele).

Das Konzept Namensräume lässt sich Studierenden gut am Beispiel der internationalen, nationalen und servicespezifischen Vorwahlen bei Telefonnummern erklären. Als weniger technisches Beispiel bieten sich auch Ortschaften als Namensräume für Strassennamen an: Innerhalb einer Ortschaft wird ein Strassenname nur einmalig verwendet, aber es wird in verschiedenen Ortschaften eine Bahnhofstrasse geben. Jüngeren Kindern kann das Konzept Namensraum mit gewissen Einschränkungen anhand von Vornamen und Nachnamen anschaulich gemacht werden. Innerhalb einer Familie (mit gleichem Nachnamen) ist es empfehlenswert, nicht mehrfach den gleichen Vornamen zu verwenden. Bei der Internetnutzung ist man ständig mit Namensräumen konfrontiert, beispielsweise mit den hierarchischen Namensräumen des Domain Name Systems oder den durch unterschiedliche Protokolle in URLs gebildeten Namensräumen.

Mit diesen vielfältigen Anwendungsbeispielen erfüllt das Konzept Namensräume sowohl das Horizontalkriterium (erkennbar in vielen Bereichen), das Sinnkriterium (Bezug zum

Alltagsdenken und zur Lebenswelt) als auch das Vertikalkriterium (vermittelbar auf allen intellektuellen Niveaus) und das Zeitkriterium (längerfristig relevant). Abbildung 3 zeigt, dass sich das Konzept nicht nur wie oben geschehen symbolisch, sondern auch ikonisch und enaktiv repräsentieren lässt, womit auch das Repräsentationskriterium erfüllt ist.

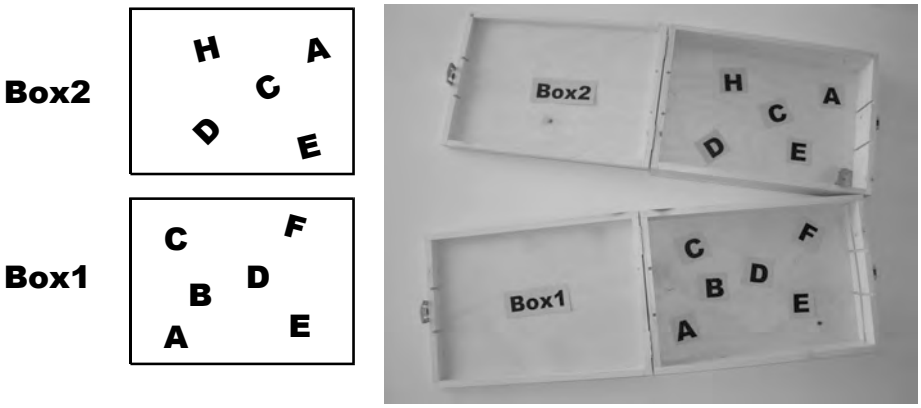


Abbildung 3: Ikonische und enaktive Repräsentation von Namensräumen

Das Konzept von Namensräumen ist wiederum - nicht nur bei Wiki – Voraussetzung für das Verständnis von relativen versus absoluten Verweisen. Nützlich ist dieses Verständnis nicht nur bei jeglichen Hypertexten, sondern auch im Umgang mit Dateisystemen oder in Tabellen einer Tabellenkalkulatioon.

3.4 Fundamentale Idee in Wiki: Codierung von Rastergrafiken

Bereits nach kurzer Einarbeitungszeit sind Wiki-Neulinge nicht mehr zufrieden mit dem Erstellen reiner Textseiten. Vom Internet heruntergeladene, selbst fotografierte oder eingescannte Bilder sollen die erstellten Seiten verschönern. Während das Einbinden solcher Bild in Wikiseiten mit Hilfe der Anleitung eines Wikis mehrheitlich gelingt, beklagen sich gewisse Nutzerinnen und Nutzer über lange Wartezeiten beim Anzeigen der Bilder oder gar Zeitüberschreitungen beim Versuch, Bilder aufs Wiki zu laden. Solche Bilder stammen meist aus der neulich erworbenen Zehn-Megapixel-Digitalkamera oder stellen eine mit maximaler Auflösung und Farbtiefe gescannte Schwarz-Weiss-Skizze dar. Hier fehlt das entsprechende Konzeptwissen, wie Bilder digital codiert werden. Dass die Unterscheidung in Raster- und Vektorgrafiken eine fundamentale Idee der Informatik darstellt, wird bereits von Hartmann et al. in [HNR06] gezeigt und soll hier nicht wiederholt werden. Stattdessen konzentriert sich die nachfolgende Betrachtung auf die Konzepte Farbtiefe und Auflösung von Rastergrafiken.

- Horizontalkriterium: Farbtiefe und Auflösung von Rastergrafiken sind zwei Parameter, welche den Detaillierungsgrad der Bildbeschreibung festlegen. Solche Parameter sind nicht nur bei Rastergrafiken anzutreffen. Jegliche Digitalisierung erfordert aus Effizienzgründen die Festlegung gewisser Grenzen der Detaillierung,

mit der die Messwerte codiert werden. Alltagsrelevant ist dies nicht nur bei Einzelbildern, sondern auch bei Audiodaten (Musik-CDs, MP3-Daten) und bei bewegten Bildern. Je nach Transport- oder Trägermedium ist ein anderer Detaillierungsgrad möglich. Auch ohne Digitalisierung ist bei wissenschaftlichen Beobachtungen oder Experimenten eine Beschränkung der Datenerfassung unumgänglich. Auch hier muss vor Beginn der Datenerfassung der Verwendungszweck bedacht und die Datenerfassung entsprechend ausgerichtet werden.

- Zeitkriterium: In der Drucktechnik werden Raster in unterschiedlicher Auflösung und Farbtiefe bereits seit längerem eingesetzt. Akzeptiert man Mosaike als spezielle Form von Rastergraphiken, so ist das Prinzip bereits seit dem Altertum gebräuchlich.

- Sinnkriterium: Das Problem der Codierung von Rasterbildern beschränkt sich nicht auf die Wiki-Nutzung. Aus diesem Grund ist im Handbuch eines Wikis meist auch nichts zu diesem Thema zu finden. Mit der zunehmenden Verbreitung der Übertragung und Speicherung von digitalen Bildern müssen Anwenderinnen und Anwender aber in der Lage sein, geeignete Datenformate und Codierungsparameter auszuwählen und Grössenabschätzungen des benötigten Speicherbedarfs vornehmen zu können.

- Vertikalkriterium und Repräsentationskriterium: Auflösung und Farbtiefe eines Rasterbildes kann mit Papier und Farbstiften auch jüngeren Kindern enaktiv erklärt werden. Die Kinder erhalten dabei verschieden eng karierte Blätter und eine unterschiedliche Anzahl an Farbstiften. Sie werden nun aufgefordert, ein vorhandenes Bild als Rastergraphik abzuzeichnen (d.h. jedes Karo darf nur eine einzige Farbe annehmen). Dabei zeigt sich die Abhängigkeit der resultierenden Bildqualität und der benötigte Arbeitsaufwand von der Anzahl Farbstifte und der auszumalenden Karos: Mehr Farbstifte oder mehr Karos ergeben zwar ein besseres Bild, benötigen aber dafür auch mehr Aufwand. Dürfen die Kinder aus einer grossen Farbstiftschachtel eine gewisse Anzahl auswählen, bevor sie mit dem Abzeichnen beginnen, so wird auch das Konzept der Farbpalette enaktiv erfahrbar (siehe Abbildung 4)

Gelegentlich bietet die Integration von Bildern auf Wikiseiten die Chance, eine weitere fundamentale Idee der Informatik aufzuzeigen. Manchmal kommen Schülerinnen und Schüler auf die Idee, dem Problem der zu grossen Bilder durch Verwendung von Größenangaben in HTML-Code zu begegnen. Die Bilder werden denn auch tatsächlich kleiner angezeigt, aber das Herunterladen der Bilder dauert weiterhin lange. Ohne Verständnis des Client-Server-Prinzips ist dieses Phänomen schwierig zu verstehen.

3.5 Weitere fundamentale Ideen in Wiki

Neben den eben detailliert dargestellten Konzepten lassen sich anhand eines Wikis weitere fundamentale Ideen der Informatik im Unterricht thematisieren. Aus Platzgründen wird im Folgenden darauf verzichtet, die Erfüllung der Kriterien als fundamentale Ideen

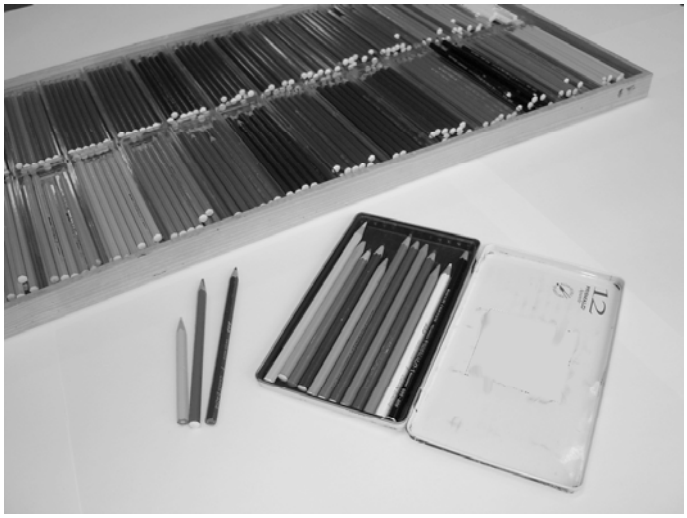


Abbildung 4: Farbtiefe und Farbpaletten enaktiv

zu belegen. Stattdessen wird in der nicht abschliessenden Liste gezeigt, warum die Kenntnis dieser Konzepte für eine effiziente Arbeit mit Wiki notwendig ist.

- Trennung von Inhalt, Layout und Struktur: Zur gemeinsamen Gestaltung übersichtlicher und lesbarer Seiten in Wiki ist es wichtig, Layoutmarkierungen von Strukturmarkierungen unterscheiden zu können. Umgekehrt lässt sich in verschiedenen Wikis sehr anschaulich die Wirkung unterschiedlicher Formatvorlagen auf selbst geschriebene Wikiseiten zeigen.

- Kollisionsproblematik: Beim gemeinsamen Bearbeiten von Text müssen Wikis das Problem paralleler Schreibzugriffe lösen. Dies wird von verschiedenen Wikis unterschiedlich gelöst, gewisse Implementationen verzichten ganz auf eine Behandlung der Kollisionsproblematik. In jedem Fall kann aber eine Kollision zum Problem werden, entweder in Form einer Warn- oder Fehlermeldung oder aber beim unabsichtlichen Überschreiben fremder Texte durch eine nicht abgefangene Kollision.

- Versionsverwaltung: Sei es, um ein unabsichtliches oder absichtliches sequentielles Überschreiben von Inhalten wieder rückgängig zu machen, oder um den Entstehungsprozess einer Wikiseite aufzuzeigen: Die Versionsverwaltung von Wikis ist ein nützliches Werkzeug. Die Versionsverwaltung von Wikis kann aber auch zum Anlass genommen werden, über das Konzept und die Funktionsweise von Versionsverwaltungen in anderen Produkten (Textverarbeitung, Betriebssystemen) nachzudenken.

- Client-Server-Prinzip: Bereits in Abschnitt 3.4 wurde ein Beispiel erwähnt, bei welchem das Verständnis, welche Aktivitäten auf dem Server, und welche auf dem Client ablaufen, wesentlich ist.

- Gestaltgesetze, Gestaltung von Benutzerschnittstellen, Navigation in Hypertext: Bei der Erstellung von Wikiseiten werden Schülerinnen und Schüler zu Produzenten von digitalem Informationsmaterial, das meist von Kolleginnen und Kollegen genutzt werden soll. Sie schlüpfen somit sowohl in die Rolle als Produzenten als auch als Konsumenten und erleben intuitiv gut und schlecht gestaltete Informationsangebote. Die in aktiven Wikis oft auftretenden Orientierungsprobleme können als Anlass genommen werden, benutzerfreundliche Navigationsstrukturen und die dahinter steckenden Gestaltgesetze zu thematisieren.
- Bring-Prinzip versus Hol-Prinzip: In einem aktiven Wiki kann es unter Umständen schwierig sein, auf dem Laufenden zu bleiben. Viele Wikis bieten vielfältige Varianten an, wie sich Nutzerinnen über Änderungen im Wiki informieren können (RSS-Feed, Regelmässige E-Mail-Benachrichtigungen, Auflistung der letzten Änderungen auf einer Spezialseite). Eine Diskussion über Vor- und Nachteile dieser Informationskanäle führt bald zur Unterscheidung von Bring-Prinzip und Hol-Prinzip.

Die aus Unkenntnis eines solchen Konzepts sich ergebenden Probleme in der Praxis können als Aufhänger für dessen Thematisierung im (Informatik-)Unterricht dienen.

4 Schlussfolgerungen

Die oben beschriebenen Nutzungsprobleme stammen alle aus dem praktischen Einsatz von Wiki im Unterricht an Schulen und Hochschulen. Sie bieten einen willkommen Anlass, die dahinter stehenden fundamentalen Ideen der Informatik mit den Lernenden genauer anzuschauen. Dies verhindert, dass der Informatikunterricht als langweilige und lebensferne Theorie wahrgenommen wird. Stattdessen kann am Beispiel gezeigt werden, dass die Kenntnis grundlegender Informatikkonzepte zu einer effizienten Arbeitsweise beitragen kann. Diese Wahrnehmung ist nicht nur wesentlich für Lernende, sondern mindestens ebenso sehr auch für Lehrerinnen und Lehrer anderer Fächer. Will Informatik als Teil der Allgemeinbildung verstanden werden, so muss der Informatikunterricht auch das Sinnkriterium erfüllen, d.h. praktisch zeigen, dass er einen Bezug zur Sprache und zum Denken des Arbeits- und Lebenswelt besitzt. Wiki bietet hier gute Gelegenheiten zur Verbindung von Informatikmittel als Werkzeug und Medium und Informatik als Thema.

Informatiklehrerinnen und -lehrer kennen die fundamentalen Ideen der Informatik, die in Wiki stecken. Wenn sie den Einsatz von Wiki in anderen Fächern propagieren und ihre Kolleginnen und Kollegen unterstützen, fördern sie nicht nur den ICT-Einsatz in der Schule und das Verständnis fundamentaler Ideen der Informatik sondern auch die Wahrnehmung der Informatik als notwendigen Teil der Allgemeinbildung.

Literaturverzeichnis

[Br60] Bruner, J. S.: The Process of Education. Harvard University Press, 1960.

[Dö05] Döbeli Honegger, B.: Wiki und die starken Lehrerinnen. In: Friedrich, S.: Unterrichtskonzepte für informatische Bildung, Proceedings der 11. GI-Fachtagung Informatik und Schule, GI-Edition- Lecture Notes in Informatics (LNI), P-60, 2005.

[Dö07] Döbeli Honegger, B.: Wiki und die starken Potenziale - Unterrichten mit Wikis als virtuellen Wandtafeln. In: Web 2.0 und Schule, Zeitschrift Computer und Unterricht Nr 66, S. 39-41, Friedrich Verlag, 2007.

[DP05] Désilets, A.; Paquet, S.: Wiki as a Tool for Web-based Collaborative Story Telling in Primary School. In: EdMedia 2005, World Conference on Educational Multimedia, Hypermedia & Telecommunications, Montréal, Québec, Canada.

[Fo90] Forneck, H.: Entwicklungstendenzen und Problemlinien der Didaktik der Informatik. In Beiträge zur Didaktik der Informatik, S. 18-54, Verlag Sauerländer.

[Gu00] Guzdial, M.; Kehoe, C.; Realff, M.; Morley, T.: A Catalog of CoWeb Uses, ftp://ftp.cc.gatech.edu/pub/gvu/tr/2000/00-19.pdf

[GRK01] Guzdial, M.; Rick, J.; Kehoe, C.: Beyond Adoption to Invention, Teacher-Created Collaborative Activities in Higher Education. In: Journal of the Learning Sciences, Volume 10 Number 3 July 2001

[HNR06] Hartmann, W; Näf, M.; Reichert, R.: Informatikunterricht planen und durchführen, Springer Verlag, 2006

[HNS99] Hartmann, W; Näf, M.; Schäuble, Peter: Effiziente und effektive Informationsbeschaffung im Internet – wie soll man das unterrichten? In: Schwill, A.: Informatik und Schule, Fachspezifische und fachübergreifende didaktische Konzepte, Springer Verlag.

[HZ03] Hennicken, D.; Zahiri, C.: Arbeiten im Netz, Erste Erfahrungen mit der Kooperations- & Austauschplattform Wiki in der Architekten & Planer-Ausbildung, http://www.uni-kassel.de/notebook/publikationen/ap_arbeiten_im_netz.pdf

[Jo05] Jonietz, D.: Ein Wiki als Lernumgebung? Überlegungen und Erfahrungen aus schulischer Sicht. In: Haake, J.; Lucke, U.; Tavangarian, D.: DeLFI 2005: 3. Deutsche e-Learning Fachtagung Informatik, GI-Edition-Lecture Notes in Informatics (LNI), P-66

[Kl05] Klampfer, A.: Wikis in der Schule, Eine Analyse im Lehr-/Lernprozess, http://teaching.eduhi.at/alfredklampfer/bachelor-wikis-schule.pdf

[Sc93] Schwill, A.: Fundamentale Ideen der Informatik. Zentralblatt für Didaktik der Mathematik, 25(1):20-31, 1993

Ada – dieser Zug hat Verspätung

Daniel Boettcher, Astrid Grabowsky, Ludger Humbert,
Oliver Poth, Constanze Pumplün, Jörg Schulte
Fachseminargruppen Informatik an den Studienseminaren Hamm und Arnsberg

Abstract: Ein objektorientierter Einstieg in die Informatik muss sich an der Berücksichtigung der Lebenswirklichkeit der Schülerinnen und Schüler messen lassen. Wir liefern mit dieser Arbeit Ideen für einen schülerorientierten Anfangsunterricht. Es werden sowohl Konzeptelemente für die Mittel- als auch für die Oberstufe einschließlich invarianter Aspekte und ihrer Umsetzung vorgestellt. Darüber hinaus dokumentieren wir einen Test, der ausgewählte Kompetenzen der Schülerinnen und Schüler – vor allem im Hinblick auf die in der Diskussion befindlichen Standards der informatischen Bildung – überprüft.

Die folgenden Konzeptelemente wurden von Referendarinnen und Referendaren vor Beginn des zweiten Ausbildungshalbjahres entwickelt und entstanden aus dem Planungsprozess für den bedarfsdeckenden Unterricht[1]. Ziel ist es, für die Mittelstufe und für die Oberstufe jeweils einen objektorientierten Ansatz zu gestalten. Dabei unterscheiden sich die jeweiligen Unterrichtsreihen in der Zielorientierung und in der Ausformung deutlich. Dargestellt werden im Folgenden die Gestaltung und die Umsetzung der Konzepte, und es wird abschließend aus den gewonnenen Erfahrungen ein Resumee gezogen.

1 Unterrichtseinstieg – Informatik in der Mittelstufe

Die Unterrichtsreihe erlaubt einen objektorientierten Einstieg in die Informatik in der Mittelstufe unter besonderer Berücksichtigung des Bezugs zur Lebenswelt der Schülerinnen und Schüler. Die grundlegenden Begriffe der Objektorientierung wie Objekt, Klasse, Objektkarte, Klassenkarte, Objektdiagramm, Klassendiagramm sollen für die Schülerinnen und Schüler handhabbar gemacht werden. Um eine zu große Theorielastigkeit zu vermeiden, soll jeweils die Verbindung des Erarbeiteten zur Anwendung mit den Informatiksystemen herangezogen werden.
Konkret sollen die Schülerinnen und Schüler in der Unterrichtsreihe entsprechend den Zielen des Informatikunterrichts die folgenden Kompetenzen erwerben: Sie sollen Objekte identifizieren, klassifizieren und die zugehörige Schreibweise kennen. Sie sollen mit verschiedenen vektororientierten Zeichenprogrammen Methoden ausführen können. Weiter sollen sie für Textdokumente die Enthält-Beziehung beschreiben und in einem Dia-

[1] In diesem Ausbildungselement unterrichten die Referendarinnen und Referendare eigenverantwortlich. Sie haben damit die Verantwortung, eine Lerngruppe als voll verantwortliche Lehrperson über einen längeren Zeitraum (typischerweise ein bis zwei Halbjahre) zu begleiten.

Aus einem Tag der Schülerin Ada

Montagmorgen 7:00 Uhr: Ada reibt sich die Sandmännchen aus den Augen, nachdem sie eine sanfte Melodie aus ihrem Handy geweckt hat. Es ist ihr neuer Lieblingssong, den sie sich erst gestern vom Internet herunter geladen hat. Aber wie jeden Montagmorgen kommt die arme Ada kaum aus dem Bett. Gestern Abend hatte sie lange mit ihrer besten Freundin über den Grundriss ihrer neuen Zimmereinrichtung gechattet, den sie zuvor mit einem Grafikprogramm gezeichnet hatte – das hat sie nun davon ... Und dann hat sie die Datei mit der Zeichnung auch nicht mehr in ihren Verzeichnissen wiedergefunden, um sie ihrer Freundin zu mailen. Sie muss heute unbedingt noch einmal danach suchen!

Abbildung 1: Szenario – Ada

gramm erkennen können. Für bekannte Gegenstände sollen sie eigenständig ein Objektmodell erstellen können.[2] Insgesamt sollen die Schülerinnen und Schüler Begriffs- und Handlungskompetenzen erwerben. Unter Begriffskompetenzen fallen unter anderem die Begriffe Objekt, Objektkarte, Attribut, Attributwert, Klasse und Klassenkarte. Diese Kompetenzen unterscheiden sich qualitativ von den darüber hinausreichenden Kompetenzen, die in der gymnasialen Oberstufe im Schulfach Informatik erworben werden (müssen).
Wichtig ist uns, gerade in den ersten Stunden des Informatikunterrichts in der Mittelstufe[3] für die Schülerinnen und Schüler einen Lebensweltbezug herzustellen. Es soll um Informatik im Alltag gehen, da dies für die Schülerinnen und Schüler vorstellbar und realistisch ist. Hierzu wählen wir die Geschichte von Ada[4] als Ausgangspunkt, von der wir in der Abbildung 1 einen Ausschnitt wiedergeben.[5]
Viele Wege stehen von hier aus offen, da die Geschichte so angepasst werden kann, dass Bezüge zu allen im Informatikunterricht vorkommenden Bereichen vorhanden sind.

Konzepte und Ideen – Mittelstufe

Um diesen Bezug zur Lebenswelt der Schülerinnen und Schüler herzustellen, wird von den Schülerinnen und Schülern ein Rollenspiel erarbeitet, das mit dem Text aus Abbildung 1 motiviert wird. Anschließend wird der Objektbegriff anhand von Objekten in einem Grafikprogramm[6] eingeführt. Dabei werden von den Schülerinnen und Schülern zunächst Zimmereinrichtungen auf Papier gezeichnet, die dann mit Hilfe von Informatiksystemen

[2] In diesem Kontext sind Standards für eine informatische Bildung und die dort auszuweisenden konkreten Kompetenzen zu berücksichtigen. Wir gehen davon aus, dass die hier genannten Kompetenzen in ähnlicher Form Bestandteil der Standards sein werden.
[3] Durchgeführt wird die Unterrichtsreihe in Stufe 9 – die Planung kann auf andere Jahrgangsstufen übertragen werden.
[4] Die Idee dieser Geschichte wurde [BDS06] entnommen:
`http://campus.ph.fhnw.ch/Kompass/LebensweltArgument`
[5] Das komplette Aufgabenblatt befindet sich unter
`http://www.ham.nw.schule.de/pub/bscw.cgi/491272/Aufgabenblatt_Ada.pdf`
[6] Hier wird mit OpenOffice Draw gearbeitet – die Umsetzung kann allerdings mit jedem Vektorgrafikprogramm erfolgen.

Eine Kursfahrt von Dortmund nach Dresden

Der Kurs unternimmt eine Fahrt mit dem Zug nach Dresden, die Schülerinnen und Schüler müssen in Hamm umsteigen. Schon beim Einsteigen in Dortmund müssen die Schülerinnen und Schüler feststellen, dass sie mit einer Verspätung von zehn Minuten in Hamm ankommen werden.
Können sie den Zug noch erreichen, oder müssen sie einen Regionalzug nehmen . . . ?

Abbildung 2: Szenario – Kursfahrt

dargestellt werden (vgl. [Br04, Seite 23]). Die Schülerinnen und Schüler beschreiben anschließend ihre gezeichneten Möbel wobei Attribute und Attributwerte, wie Breite und Höhe mit ihren Werten auftreten. Alle Beschreibungen werden an der Tafel gesammelt und Gemeinsamkeiten und Unterschiede heraus- gearbeitet. Auf diese Weise wird der Objektbegriff veranschaulicht und die Begriffe Attribut, Attributwert und Bezeichner werden am konkreten Beispiel deutlich als Fachbegriffe herausgestellt. Im Anschluss daran werden Objektkarten zu grafischen und auch anderen Objekten erstellt und Objekte werden anhand von vorgegebenen Objektkarten rekonstruiert.
Jetzt ergibt sich die Einführung des Klassenbegriffs, indem die Schülerinnen und Schüler Grafikobjekte nach Ähnlichkeiten ordnen. Es werden beispielsweise die Objekte von Betten zur grafischen Darstellung zu der Klasse Rechteck zusammengefasst. Der nächste Schritt ist die Übertragung des Objektbegriffs auf Objekte in Textdokumenten. Dazu werden Adjektive in einem Text durch die Änderung der entsprechenden Attributwerte der zugehörigen Substantive visuell dargestellt. Durch die Visualisierung wird den Schülerinnen und Schülern das Auffinden von Objekten in Texten und deren Attributen erleichtert[7]. Zum Schluss werden Objektdiagramme und Klassendiagramme zu Textdokumenten erstellt und diese neuen Begriffe auch auf andere – nicht-informatische – Bereiche übertragen.
Anschließend können verschiedene Wege eingeschlagen werden. Es bietet sich beispielsweise an, das Gelernte auf Multimedia-Dokumente oder eine andere Art der Dokumentenerstellung (wie LaTeX oder HTML) zu übertragen. An dieser Stelle ist auch ein Einschnitt und Wechsel zu einem anderen Konzept wie dem funktionalen der Tabellenkalkulation möglich. So kann die Reihe an viele Schulcurricula angepasst werden.

2 Unterrichtseinstieg – Informatik in der Oberstufe

Die Unterrichtsreihe realisiert einen schülerorientierten Einstieg in die objektorientierte Modellierung. Ausgehend von einer Situation mit Bezug zur Lebenswelt der Schülerinnen und Schüler werden verschiedene Standardmethoden zur Modellierung und Implementierung erarbeitet und angewendet.
Die angestrebten Kompetenzen zielen nicht nur auf Begriffe wie für die Mittelstufe beschrieben, sondern zu einem großen Teil auf Methoden. Diese veränderte Zielorientierung

[7] Ein Aufgabenblatt hierzu ist zu finden unter `http://www.ham.nw.schule.de/pub/bscw.cgi/491272/Aufgabenblatt_Attribute_Zeichen.pdf`

ergibt sich aus der Anforderung, dass die Schülerinnen und Schüler erweiterten Modellierungsanforderungen gerecht werden müssen: Die Implementierung in einer Programmiersprache ist erklärter Inhalt des Informatikunterrichts in der gymnasialen Oberstufe. Diese Notwendigkeit setzt eine Reihe von Elementen voraus, die als konkrete Kompetenzen erworben und ausgewiesen werden. Zur Verdeutlichung dieser Kompetenzen werden in Abbildung 3 die im Zusammenhang mit der Modellierung herangezogenen informatischen Methoden in Beziehung gesetzt.

Konzepte und Ideen – Oberstufe

Die folgenden Inhalte in der Reihe sind für die Modellierung unabdingbar, im Anschluss an diese Lerngegenstände kann die Gestaltung des Unterrichts auf verschiedenen Wegen erfolgen. Am Beginn der Unterrichtsreihe steht die umgangssprachliche Beschreibung eines Szenarios aus der Lebenswelt der Schülerinnen und Schüler. Nach der Erläuterung des Objektbegriffs werden die Objekte mittels der Methode von Abbott (vgl. [Ab83]) identifiziert. Je nach Vorgabe kann dieser Teil stärker geleitet oder offen gestaltet werden (vgl. Abbildungen 2, 4 und Storyboard[8]).
Es werden Objektdiagramme und Sequenzdiagramme benötigt, um die Beziehungen bzw. dynamischen Abläufe zwischen den Objekten darzustellen. Das so dargestellte Modell wird implementiert. Diese Implementierung ist mit verschiedenen Programmiersprachen und -umgebungen[9] möglich. Damit wird eine Festlegung und Einschränkung vermieden. Die jeweilige Programmiersprache dient der Umsetzung der Konzepte und ist kein Selbstzweck. Dennoch müssen die für die spezifische Umsetzung notwendigen Kenntnisse in der Programmiersprache erworben werden.
Als Ergebnis werden in der Implementierung die modellierten Klassen und Objekte realisiert. Die implementierten Methoden geben Texte wie „Methode A wurde aufgerufen“ auf den Bildschirm aus und erlauben damit den direkten Vergleich zwischen Rollenspiel und Sequenzdiagramm. Die Ergebnisse der Implementierung, insbesondere das erarbeitete und umgesetzte Fachkonzept, können in späteren Phasen des Unterrichts wieder aufgenommen werden, indem zum Beispiel Züge grafisch auf dem Bildschirm dargestellt werden.

3 Invarianten der dargestellten Ansätze

Die Darstellung der beiden Ansätze für die Mittel- und für die Oberstufe verdeutlicht den zentralen Stellenwert der Modellierung in einer objektorientierten Ausprägung. Damit werden aktuelle Entwicklungslinien der Fachdidaktik für die konkreten Unterrichtseinsatzszenarien und für die verschiedenen Stufen verfügbar. Wir kommen nicht umhin, für Schülerinnen und Schüler ohne Vorkenntnisse aus der Informatik auch heute noch in der Oberstufe einen Anfangsunterricht zu gestalten, der bei der informatischen Modellierung von Null ausgeht.

---

[8] vgl. `http://www.ham.nw.schule.de/pub/bscw.cgi/491272/Storyboard_Zugszenario.pdf`
[9] Das Konzept wird mit den Programmiersprachen Java (Entwicklungsumgebung BlueJ), ObjectPascal (Delphi) und Python umgesetzt. Dabei ist zu berücksichtigen, dass für Java und ObjectPascal vor der Implementierung noch weitere Themen wie das Variablenkonzept erarbeitet werden müssen.

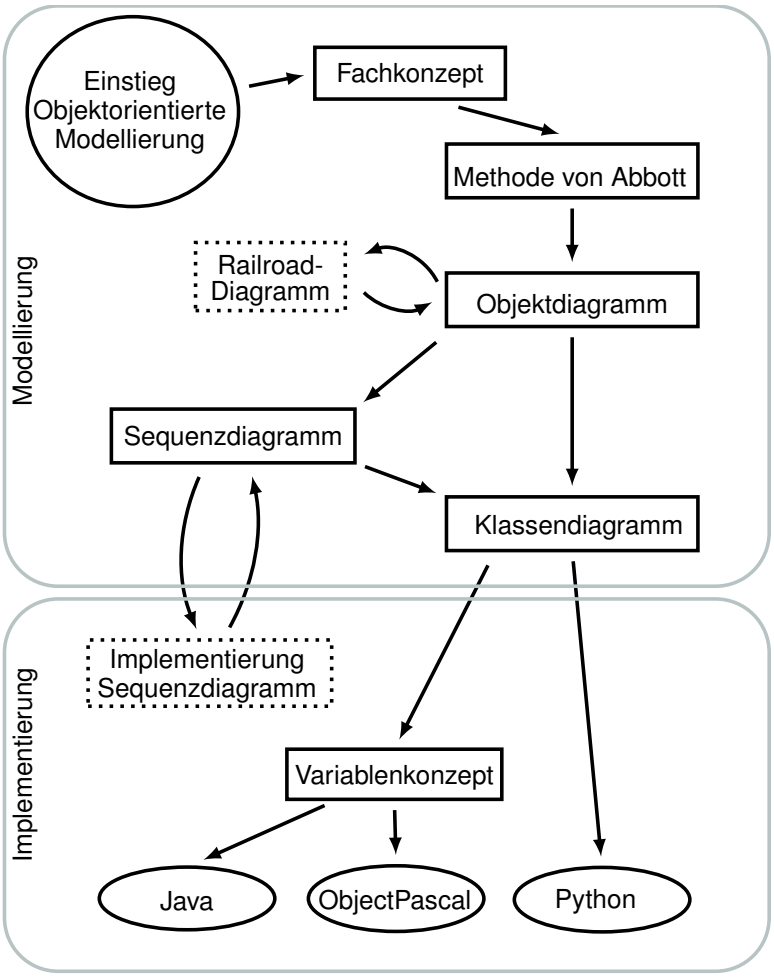


Abbildung 3: Oberstufeneinstieg – Informatische Methoden des Konzepts

Damit haben die Schülerinnen und Schüler einen Vorteil, die bereits in der Mittelstufe im Schulfach Informatik unterrichtet wurden. Dies kann und sollte nicht dadurch verhindert werden, dass Schülerinnen und Schüler vor der Oberstufeninformatik ausschließlich mit Inhalten arbeiten, die in der Oberstufe nicht thematisiert werden.
Der Begriffsapparat und die Form der grafischen Darstellung von Objekten und Klassen und ihrer Beziehungen sind unabdingbar, um einfache Sachverhalte verständlich darzustellen. Ohne diese Darstellung ist die Analyse und damit die notwendige informatische Modellierung nicht sinnvoll durchzuführen. Also sind diese Elemente in beiden dargestellten Konzeptansätzen vorhanden. Sie werden allerdings auf kognitiv unterschiedlichen Anspruchsniveaus umgesetzt.

So stellt die objektorientierte Analyse in der Mittelstufe ein Konzept zum Verstehen bereit, in der Oberstufe muss es als Konzept so verankert sein, dass anschließend eine erfolgreiche Implementierung in einer Programmiersprache möglich wird.

4 Umsetzung Mittelstufe

Unabhängig von der jeweiligen Jahrgangsstufe ist die Unterrichtsreihe als Einstieg in die Informatik in der Mittelstufe einsetzbar. Konkret wurde die Reihe in mehreren Wahlkursen Informatik des Jahrgangs 9 durchgeführt.[10]
Als Einstieg dient die Geschichte aus dem Alltag einer fiktiven Schülerin (vgl. Abbildung 1). Diesem Text können verschiedene alltägliche Informatikbegriffe entnommen werden, um so die Relevanz der Informatik im Alltag für die Schülerinnen und Schüler zu verdeutlichen. In einer anschließenden Gruppenarbeit wird die Geschichte fortgesetzt und in Form eines Rollenspiels den anderen Schülerinnen und Schülern vorgestellt.
Einen Punkt der Geschichte aufgreifend, werden danach – zunächst mit Papier und Stift, später dann mit den Informatiksystemen – Grundrisse von Zimmereinrichtungen angefertigt. Auf Basis dieser Grundrisse wird der Objektbegriff eingeführt. Die Schülerinnen und Schüler identifizieren die in den Grundrissen vorkommenden Objekte mit ihren Attributen und Methoden und lernen, die zugehörigen Objektkarten zu erstellen. Um zu verdeutlichen, welche Attribute auf einer Objektkarte wichtig oder unwichtig sind und wie exakt die Objektkarte sein muss, hat sich bewährt, die Schülerinnen und Schüler in Einzelarbeit Objektkarten erstellen zu lassen und diese dann mit einem Partner zu tauschen, um das unbekannte Objekt dann möglichst exakt zu rekonstruieren.
Hier ist es unbedingt nötig, auch zu anderen Objekten wie beispielsweise Stiften aus den Etuis der Schülerinnen und Schüler, Spielzeugautos oder weiteren Alltagsgegenständen die zugehörigen Objektkarten zu erstellen, um den Objektbegriff zu verdeutlichen und vernetzen. Auch ein vertiefender Umgang mit dem Grafikprogramm durch eine andere Aufgabenstellung ist hier sinnvoll. Anregungen hierzu können den Materialien (vgl. [Bo07]) entnommen werden.
Anschließend wird durch Ordnen der Grafikobjekte nach „Ähnlichkeit“ die Einführung von Klassen eingeleitet. Die Darstellungsform der Klassenkarte wird thematisiert. An dieser Stelle der Unterrichtsreihe kann in einer separaten Lerneinheit auf die verschiedenen Grafikformate eingegangen werden.
Um den Objektbegriff auf andere Bereiche zu übertragen, werden nun Texte untersucht. Es hat sich hier gezeigt, dass von den Schülerinnen und Schülern der Begriff „Objekt“ mit „Grafikobjekt“ assoziiert wird.
Die Darstellung von Adjektiven wie zum Beispiel „groß“, „tief“, „gelb“ oder „breit“ durch entsprechendes Auswählen der Attributwerte der zugehörigen Substantive ist ein geeignetes Mittel, um die Möglichkeiten der Veränderung der Attributwerte der Klasse „Zeichen“ auszuloten (vgl. [Br04, Seite 38]). Die Ergebnisse dieser Partnerarbeit können dann im Plenum vorgestellt werden.

[10] Der Informatikunterricht wurde in den Schulformen Gymnasium und Gesamtschule erteilt.

Eine Fahrt mit dem Zug

Du möchtest mit dem Zug von Dortmund nach Dresden fahren. Du musst in Eisenach umsteigen, doch der Zug hat eine Verspätung...

1. Führe die oben skizzierte Geschichte ausführlicher aus und achte darauf, möglichst alle *unverzichtbaren* an dem Geschehen beteiligten Personen und Gegenstände und ihre Kommunikation/Handlungen in die Geschichte aufzunehmen.
2. Notiere deine Idee für den Fortlauf der Handlung!
3. Stelle kurz dar, warum du dich genau für diese Gegenstände und Personen entschieden hast.

Abbildung 4: Szenario – Kursfahrt – offener Arbeitsauftrag

Da Texte eine Gliederung in Zeichen und Absätze erlauben, kann nun auch die Darstellungsform des Objektdiagramms eingeführt werden. Weiter bietet sich darauf aufbauend die Einführung von Klassendiagrammen (beschränkt auf „Enthält"-Beziehungen) an.[11]
Auch hier ist es wichtig, die Begriffe an anderen Objekten und Klassen als den in Textverarbeitungsprogrammen vorkommenden zu vertiefen, möglichst mit Alltagsdingen (wie beispielsweise dem Klassenraum, einem Regal).
Abschließend ist zu bemerken, dass man gerade für Doppelstunden immer eine Arbeitsphase mit den Informatiksystemen einplanen sollte, da für die Schülerinnen und Schüler der Mittelstufe hierdurch die Motivation deutlich erhöht wird und eine Verbindung zwischen den theoriegeleiteten Konzepten und der praktischen Umsetzungsmöglichkeit geboten wird.

5 Umsetzung Oberstufe

Die Unterrichtsreihe startet zu Beginn des Schuljahres in Informatikgrundkursen der gymnasialen Oberstufe[12], so dass bei der Strukturierung der Stundenfolge zu berücksichtigen ist, dass einige Schülerinnen und Schüler kein Vorwissen aus dem Schulfach Informatik besitzen.
Die Modellierungsmethoden, deren Vermittlung den Schwerpunkt dieser Unterrichtsreihe bildet, werden in folgender Reihenfolge erarbeitet:

- Methode von Abbott[13]

[11] Vergleiche vertiefend zu diesem Ansatz [Vo06].
[12] Der Unterricht wurde in den Schulformen Gymnasium und Gesamtschule durchgeführt.
[13] Das Verfahren von Russell J. Abbott nach [Ab83] in einer vereinfachten Form:
Zunächst werden die Hauptwörter (Substantive) herausgefiltert. Die Hauptwörter sind mögliche Objekte. Meist nicht beachtet werden allerdings Mengen- und Grössenangaben , Sammelnamen, Materialbezeichnungen und abstrakte Begriffe. Zeitwörter, die als Hauptwörter benutzt werden, werden behandelt wie die zugehörigen Zeit-

- Railroaddiagramme (fakultativ)
- Objektdiagramme[14]
- Storyboard[15]
- Sequenzdiagramme (Objektrollenspiel, Nachrichtenkonzept)
- Klassendiagramme[16]

Am Ende dieser Reihe steht der Übergang in die Implementierung der erstellten Modelle, die in den verschiedenen Kursen in den Programmiersprachen ObjectPascal (Delphi-IDE), Python und Java (BlueJ) umgesetzt worden sind. Ziel der Implementierung ist ein Programm, in dem die verschiedenen modellierten Objekte mit ihren Attributen und Methoden zur Verfügung stehen. In den Methoden werden entweder weitere Methoden augerufen oder aber einfache Textausgaben der Form „Methode xy wurde aufgerufen“ ausgeführt. Ergebnis eines Programmdurchlaufes sind dann die Textausgaben der verschiedenen Methoden, die mit den zuvor erstellten Sequenzdiagrammen verglichen werden können.
Zum Einstieg in die Reihe werden in den verschiedenen Kursen zwei verschiedene Varianten ausgewählt. Während in der ersten Variante den Schülerinnen und Schüler die Geschichte bereits vorgegeben worden ist (vgl. [Bo07]), auf die dann die Methode von Abbott angewendet werden sollte, so wird bei der zweiten Variante eine offenere Aufgabenstellung gewählt, bei der die Schülerinnen und Schüler ohne Steuerung durch die Lehrperson die Geschichte zu dem Thema entwickeln.
In beiden Varianten wird den Schülerinnen und Schülern an dieser Stelle der Objektbegriff anhand einer Definition und einfachen Beispielen erläutert. So werden zum Beispiel ein Fahrrad und ein Auto modelliert. An dieser Stelle werden die Grenzen der Modellierung sichtbar. Die Lehrperson weist die Schülerinnen und Schüler darauf hin, dass man sich für eine praktikable Modellierung auf die für eine Problemstellung wichtigen Eigenschaften eines Objekts beschränken kann und muss.
Die Ergebnisse der Anwendung der Methode von Abbott stellen Objekte sowie deren Attribute und Methoden dar. Da hier syntaktisch[17] korrekte Bezeichner eingesetzt werden sollen, bietet sich an dieser Stelle eine optionale Lerneinheit zum Thema Syntax und deren Notation mit Hilfe von Railroad-Diagrammen an. Alternativ kann die Beschreibung an dieser Stelle auch umgangssprachlich erfolgen.
Die Objekte werden nun im nächsten Unterrichtsschritt in Objektdiagrammen (Zustand) modelliert – hier können dann bereits grundlegende Objektbeziehungen (enthält und kennt) eingeführt werden. Damit ist es den Schülerinnen und Schüler an dieser Stelle möglich,

---

wörter. Gattungsnamen sind ebenfalls meist keine Objekte. Im nächsten Schritt werden die Zeitwörter (Verben) herausgefiltert. Verben bezeichnen häufig die Aktionen von Objekten. Es ist festzustellen, welchem Objekt die Aktion zugeordnet werden kann. Schließlich weden die Adjektive herausgefiltert. Adjektive bezeichnen häufig die Attribute von Objekten. Auch hier ist festzustellen, welchem Objekt die Aktion zugeordnet werden kann.

[14] Die eingesetzten Diagramme orientieren sich an dem Standard UML 2.0.

[15] `http://www.ham.nw.schule.de/pub/bscw.cgi/491272/Storyboard_Zugszenario.pdf`

[16] Ein Beispiel für ein Informationsblatt zu Klassen ist verfügbar:
`http://www.ham.nw.schule.de/pub/bscw.cgi/491272/Information_zu_Klassen.pdf`

[17] Die Konventionen sind in [Bo07] beschrieben.

aus einer umgangsprachlichen Beschreibung ein statisches, abstraktes Modell zu entwickeln. Zur Sicherung und Verallgemeinerung der erarbeiteten Lerninhalte werden von den Schülerinnen und Schülern verschiedene andere Sachverhalte modelliert.
Das statische Ojektmodell wird nun um die Darstellung der dynamischen Abläufe in einem Sequenzdiagramm ergänzt. Hier werden im Unterricht zur Veranschaulichung des dynamischen Charakters Objektrollenspiele verwendet. Aus der dynamischen Interaktion der verschiedenen Objekte wird das zentrale Nachrichtenkonzept entwickelt. Die erstellten Sequenzdiagramme können durch die Schülerinnen und Schüler in einfache Programme übertragen werden, insofern die Klassen/Objekte durch den Lehrer vorbereitet worden sind.
Bevor das Modell von den Schülerinnen und Schülern nun in die jeweilige Implementierung umgesetzt wird, sind der Klassenbegriff und die dazugehörigen Klassendiagramme zu thematisieren. Dabei ist besonders auf eine sehr exakte und sorgfältige Trennung zwischen Objekten und Klassen zu achten.

6 Vergleichende Auswertung

6.1 Mittelstufe

Für die Mittelstufe erweist sich das Konzept im Wesentlichen als tragfähig, aber die Schülerinnen und Schüler haben Schwierigkeiten, nach den Objekten in Grafiken dann Objekte in Texten zu identifizieren: Sie versuchen, die Buchstaben durch grafische Objekte wie Kreis oder Linie zu beschreiben und erkennen als Objekte der Klasse Zeichen zunächst nur Satzzeichen. Es ist zu überlegen, ob eine Änderung der Reihenfolge (zunächst Textdokumente und danach Grafikdokumente) für die Schülerinnen und Schüler die Schwierigkeiten vermindern könnte. Ebenso sollte ein noch intensiveres Einüben des Identifizierens und Klassifizierens von Objekten an alltäglichen Beispielen erwogen werden. Auch bleibt zu prüfen, wie die Schülerinnen und Schüler, die nun in der Mittelstufe den objektorientierten Ansatz kennen gelernt haben, in der Oberstufe damit umgehen werden.

6.2 Oberstufe

Aufgrund der Komplexität des Szenarios bedarf der Einstieg einer genauen didaktischen Gestaltung. Da die Schülerinnen und Schüler in der Regel noch keine Erfahrung im Modellieren haben, muss die Lehrperson hin und wieder eingreifen, um die Modellierung praktikabel zu halten. Dies geschieht allerdings niemals, indem die Schülervorschläge als falsch bezeichnet werden. Vielmehr erkennen die Schülerinnen und Schüler, dass es zwar keine Musterlösung gibt, aber dennoch einige Modelle für eine bestimmte Problemstellung praktischer sind als andere.
Um diesen Einstieg etwas einfacher und gelenkter zu gestalten, kann die Lehrperson ein Storyboard[18] vorgeben. Dieses Vorgehen ist weniger offen, bietet den Schülerinnen und

[18] Das komplette Storyboard ist zugänglich unter
`http://www.ham.nw.schule.de/pub/bscw.cgi/491272/Storyboard_Zugszenario.pdf`

Schülern aber eine Orientierung. Auf diese Weise können sie im Unterricht selbständiger und unter weniger Aufsicht der Lehrperson eine Modellierung erarbeiten.
Die Komplexität der inhaltlichen Anforderungen in der Oberstufe übersteigt das, was man Schülerinnen und Schülern üblicherweise zumuten kann. Dies liegt darin begründet, dass die Schülerinnen und Schüler keine Vorkenntnisse aus dem Bereich der Objektorientierung aus der Mittelstufe mitbringen und so zunächst die Grundlagen der Objektorientierung eingeführt werden müssen, bevor dann der eigentliche oberstufengerechte Stoff erarbeitet werden kann. So ist das zu bewältigende Pensum wahrlich eine Herausforderung für die Schülerinnen und Schüler.

6.3 Vergleich

Insgesamt zeigt sich in weiten Teilen die Tragfähigkeit der vorgestellten Konzeptelemente sowohl für die Mittel- als auch für die Oberstufe. Auf Basis der entwickelten Grundstrukturen der Unterrichtsreihen kann der konkrete Unterricht gut auf die Anforderungen der jeweiligen Lerngruppen angepasst werden.
Im Vergleich der Ansätze für Mittel- und Oberstufe sind die verschiedenen kognitiven Anforderungen an die Schülerinnen und Schüler gut zu erkennen: In der Mittelstufe wird im Informatikunterricht ein Objektbegriff vermittelt, der sich stark an konkreten Anwendungen orientiert, die Schülerinnen und Schüler erwerben Begriffskompetenzen. Die praktische Umsetzung in verschiedenen Lerngruppen hat gezeigt, dass die zu vermittelnden Lerninhalte sehr gut auf bereits vorhandenem Wissen aus anderen Fächern aufbauen und daher keine zu großen Lernwiderstände zu überwinden sind. Da für die Vermittlung der Lerninhalte Anwendungsgebiete wie zum Beispiel die Textverarbeitung oder ein vektororientiertes Zeichenprogramm ausgewählt worden sind, die an vielen Schulen im Informatikuntericht der Mittelstufe eingesetzt werden, können die vorgestellten Unterrichtsideen in bestehende Schulcurricula meist gut integriert werden.
In der Oberstufe ist der Schwerpunkt dagegen im Erwerb der Methodenkompetenzen anzusiedeln. Da in der Oberstufe auch Schülerinnen und Schüler die Informatikkurse belegen, die in der Mittelstufe keinen Informatikunterricht besucht haben, kann der Unterricht, wie in der Mittelstufe, auf keinen Vorkenntnisse aus dem Bereich der Informatik aufbauen und es müssen Überschneidungen der Lerninhalte vermieden werden. Sowohl in der Ober- als auch der Mittelstufe ist es wichtig, das Identifizieren und Klassifizieren von Objekten ausreichend einzuüben. Da der Schwerpunkt in der Mittelstufe auf den Erwerb von Begriffskompetenz gelegt wird, in der Oberstufe jedoch beim Erwerb von Methodenkompetenzen anzusiedeln ist, gibt es über die grundlegende Begriffsbildung hinaus keine signifikanten Überschneidungen der Unterrichtsinhalte, so dass auch für Schülerinnen und Schüler, die in der Mittelstufe bereits das Konzept der Objektorientierung kennen gelernt haben, keine wesentlichen Redundanzen im Informatikunterricht der Oberstufe zu erwarten sind.
Um das Ziel, die objektorientierte Modellierung und Implementierung, zu erreichen, müssen von den Schülerinnen und Schülern Kenntnisse über eine Vielzahl von Modellierungsmethoden erworben werden (vgl. Abbildung 3), die aufeinander aufbauen. Im Gegensatz zur Mittelstufe sind hier von den Schülerinnen und Schülern zum Teil große Lernwiderstände zu überwinden. In der ersten Phase der Unterrichtsreihe steht nicht die direkte Anwendung des Gelernten an einem Informatiksystem im Vordergrund, sondern die Model-

lierung. Die Implementierung des erstellten Modells in ein lauffähiges Programm, welche für die Schülerinnen und Schüler eine starke Motivation darstellt, kann erst nach der umfassenden Modellierung erfolgen. Daher muss in der Oberstufe im Gegensatz zur Mittelstufe, wo sich die Erfolgserlebnisse für die Schülerinnen und Schüler umgehend durch den Einsatz der Anwendungsprogramme einstellen, stärker auf Erfolgskontrollen und -erlebnisse geachtet werden.
Die vorgestellten Konzeptelemente können ebenso wie auch die der Mittelstufe gut in bestehende Schulcurricula eingebunden werden, da sie unabhängig von der eingesetzen Programmiersprache sind und sich an den Vorgaben für die Prüfungsanforderungen in der Abiturprüfung orientieren (vgl. [KM04]).

7 „Standard“–orientierte Auswertung

Ausgehend von der Fragestellung, ob Schülerinnen und Schüler der Mittel- und Oberstufe eine Vergleichsaufgabe im Bereich der Informatik – unabhängig vom Curriculum – erfolgreich bearbeiten können, führten wir in verschiedenen Schulen und Jahrgangsstufen[19] einen Test[20] mit insgesamt 178 Schülerinnen und Schülern durch. Ziel war es, durch die vielen verschiedenen Kurse einen Überblick zu gewinnen, inwiefern die Schülerinnen und Schüler im Informatikunterricht Informatikkompetenz erwerben.
In der Auswertung wurde deutlich, dass die Schülerinnen und Schüler, die bisher noch keinen Informatikunterricht hatten, zu einem guten Teil nicht einmal die Kompetenzstufe des Wissens erreichten. Die Vertrautheit mit Inhalten des Informatikunterrichts ermöglichte hier einen kompetenteren Umgang mit der Testaufgabe.
Im Übergang zu den Transferleistungen schlug sich die Erfahrung mit Objektorientierung positiv in den Ergebnissen nieder. Auch zeigte sich hier eine größere Kompetenz der Schülerinnen und Schüler des Gymnasiums gegenüber denen der Gesamtschule.
Die höchste Kompetenzstufe der Analysefähigkeit wurde erwartungsgemäß nur von sehr wenigen Schülerinnen und Schülern erreicht, wobei sich auch hier die Leistungen der Gymnasiastinnen und Gymnasiasten von denen der Gesamtschülerinnen und Gesamtschüler abhoben.
Für diejenigen Schülerinnen und Schüler der Mittelstufe, denen der Gegenstand des Tests durch den Unterricht bereits bekannt ist, war die Aufgabenstellung trotz der ungewohnten grafischen Darstellung offensichtlich leichter zu bearbeiten als für die Schülerinnen und Schüler der Mittel- und Oberstufe, denen der Gegenstand nicht unmittelbar vertraut ist. Die Beschäftigung mit dem Gegenstand im Unterricht führte zu mehr Bereitwilligkeit, sich auf die Aufgaben einzulassen. Es ist also auch wichtig, welche Inhalte im Informatikunterricht vermittelt werden; Methodenkompetenzen alleine reichen zur Informatischen Bildung im Sinne der Standards nicht aus.

---

[19] Der Test wurde in drei Mittelstufenkursen (Mathematikkurs Stufe 8, Informatikkurs Stufe 9 mit objektorientiertem Unterricht, Informatikkurs Stufe 9 ohne objektorientierten Unterricht) und einem Oberstufenkurs (Informatikkurs Stufe 11) eines Gymnasiums und zwei Mittelstufenkursen (Informatikkurs Stufe 9 zu Beginn des Kurses und Informatikkurs Stufe 9 nach einem Halbjahr mit Informatikunterricht) und einem Oberstufenkurs (Informatikkurs Stufe 11) einer Gesamtschule durchgeführt.

[20] Die Aufgabenstellung aus [Hu06, Seite 70] ist zu finden unter
`http://www.ham.nw.schule.de/pub/bscw.cgi/491272/Standardorientierte-Aufgabe-HTML.pdf`

8 Ausblick

Die hier vorgestellten Konzeptideen für die Mittelstufe orientieren sich an den Zielen, die auch bei der Erstellung der Standards eine wichtige Rolle spielen. Die Konzeptideen für die Oberstufe orientieren sich an der Einheitlichen Prüfungsordnung Abitur für Informatik (vgl. [KM04]).
Der Aufbau eines fundierten Objektverständnisses bei Schülerinnen und Schülern ist abhängig von der kognitiven Leistungsfähigkeit. In der Gestaltung wurde dies grundsätzlich berücksichtigt. Dennoch zeigen sich Probleme in der Realisierung. Die beschriebenen Schwierigkeiten der Schülerinnen und Schüler bei der Übertragung des Objektbegriffs von einem Grafikprogramm auf eine Textverarbeitung deuten die zu erwartenden Mißverständnisse an. Sowohl in der Mittelstufe als auch in der Oberstufe ist zur Motivation eine möglichst weit ins Konkrete führende Implementierung der abstrakten Modellierung wünschenswert.
Wir bedanken uns bei den anonymen Gutachterinnen/Gutachtern für ihre wertvollen Hinweise.

Literaturverzeichnis

[Ab83] Abbott, R. J.: Program Design by Informal English Descriptions. Comm. ACM. 26(11):882–894. November 1983.

[BDS06] Bruelhart, S., Döbeli Honegger, B., und Schwab, S. ICT-Kompass. 2006. http://www.ict-kompass.ch/ – geprüft: 10. Januar 2007.

[Br04] Brichzin, P., Freiberger, U., Reinold, K., und Wiedemann, A.: Ikarus – Natur und Technik. Schwerpunkt: Informatik 6/7. Oldenbourg. München, Düsseldorf, Stuttgart. 1. Aufl. 2004.

[Bo07] Boettcher, D., Grabowsky, A., Humbert, L., Poth, O., Pumplün, C., und Schulte, J. Ada – dieser Zug hat Verspätung. Material für die Hand der Lehrkraft. Januar 2007. http://www.ham.nw.schule.de/pub/bscw.cgi/491272/ – geprüft: 23. Januar 2007.

[CS06] Claus, V. und Schwill, A.: Duden Informatik A–Z. Fachlexikon für Studium und Praxis. Bibliographisches Institut. Mannheim, Leipzig, Wien, Zürich. 4., überarb. u. aktualis. Aufl. Februar 2006.

[Hu06] Humbert, L.: Didaktik der Informatik – mit praxiserprobtem Unterrichtsmaterial. Leitfäden der Informatik. B.G. Teubner Verlag. Wiesbaden. 2., überarbeitete und erweiterte Aufl. August 2006. http://humbert.in.hagen.de/ddi/ – geprüft: 23. Januar 2007.

[KM04] KMK (Hrsg.): Einheitliche Prüfungsanforderungen in der Abiturprüfung »Informatik«. KMK. Bonn. 2004. KMK – Ständige Konferenz der Kultusminister der Länder in der Bundesrepublik Deutschland http://www.kmk.org/doc/beschl/EPA-Informatik.pdf – geprüft: 17. April 2007.

[Vo06] Voß, S.: Modellierung von Standardsoftwaresystemen aus didaktischer Sicht. Dissertation. Technische Universität – Institut für Informatik. München. Juni 2006.

Datenbanken – (etwas) anders gesehen

Peter K. Antonitsch

Institut für Informatiksysteme/Informatik Fachdidaktik
Alpen-Adria Universität Klagenfurt
Universitätsstraße 65-67
A 9020 Klagenfurt
Peter.Antonitsch@uni-klu.ac.at

Abstract: Im Informatikunterricht werden Datenbanken entweder vom Ansatz des Anwendens oder des Modellierens behandelt. Diese beiden Ansätze sind als „Verwenden von Strukturen“ bzw. „Erzeugen von Strukturen“ zueinander komplementär, vernachlässigen aber häufig die Rückkoppelung des durch die Datenbank repräsentierten Modells mit dem modellierten Realweltausschnitt. Dieser Artikel beschreibt erste Erfahrungen mit einem alternativen Ansatz, der ausgehend von Datenbankabfragen die beiden oben genannten Strukturaspekte zu vereinen sucht und auch dem Erlernen der bei relationalen Datenbanken verwendeten Strukturierungsmethoden sowie dem Hinterfragen der Strukturen Raum gibt. Zentraler Gedanke ist dabei das Erforschen vorgegebener Datenbankstrukturen durch die Lernenden.

1 Datenbanken im Informatikunterricht

Relationale Datenbanken werden traditionell als Kernstoff im Informatikunterricht angesehen (vgl. [BM04] zur Situation in Österreich). Zwar wird die didaktische Diskussion über Methoden zur schülergerechten Aufbereitung des Themas „Datenbanken“ bei weitem nicht so intensiv geführt wie dies z.B. bei der Einführung in das Programmieren der Fall ist, dennoch lassen sich in der fachdidaktischen Literatur zumindest zwei gängige Vermittlungskonzepte ausmachen. Diese unterscheiden sich voneinander dadurch, dass entweder der „Anwendungsaspekt“ oder der „Modellierungsaspekt“ von Datenbanksoftware besonders betont wird:
Motiviert durch die zunehmende Popularität von Zertifikaten als Nachweis von „Computerwissen“ findet das Konzept der Anwenderschulung auch in den Schulen Beachtung. Dies gilt (zumindest in Österreich) weitgehend für den Informatikunterricht der Sekundarstufe 1, wo grundlegende Fertigkeiten des Arbeitens mit dem Computer vermittelt werden [Ro06]. Dies setzt sich aber auch in der Sekundarstufe II bei Datenbanken als „anwendungsnahe“ Software fort [Ne04] [Ha05a]. Charakteristisch für das Arbeiten auf dieser „Anwendungsebene“ ist die Konzentration auf den Werkzeugaspekt der jeweiligen Datenbanksoftware, während der konzeptionelle Hintergrund bzw. das Design von Datenbanken allenfalls als Weiterführung bzw. Vertiefung angesprochen werden [Ko04] [Ha05b].

Nahezu komplementär zur Anwenderschulung betont das zweite gängige Vermittlungskonzept das Modellieren. Mit der häufig anzutreffenden Themenfolge „konzeptueller Entwurf – logischer Entwurf – Normalisierung – Abfragen“ (z.B. [Sc05] oder [Ja03]) orientiert sich dieser Ansatz offenbar an universitären Datenbankkursen. Der Schwerpunkt der Schülerinnen- und Schülertätigkeit liegt dabei auf dem Entwurf von Datenbanken im Sinne von regelbasiertem Strukturieren eines vorgegebenen Realweltausschnittes. Das Benutzen der Datenbank als Quelle von Informationen tritt auf dieser „Modellierungsebene“ etwas in den Hintergrund.

2 Ein alternativer Ansatz...

Die Lernenden werden bei der Auseinandersetzung mit Datenbanksoftware nach obigen Vermittlungskonzepten auf der Anwendungs- oder der Modellierungsebene tätig. Beide Ebenen sind über das Datenmodell der relationalen Tabellenstruktur miteinander gekoppelt und überschneiden sich im Bereich der Datenbankabfragen (vgl. Abb. 1).

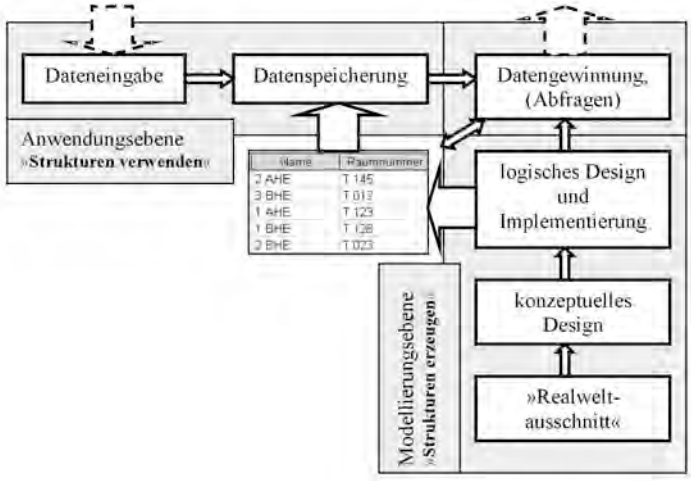


Abbildung 1: Tätigkeitsorientierte Sicht auf Anwendungs- und Modellierungsebene von Datenbanksoftware

Die Lernenden verwenden dabei die zugrunde liegende Struktur in unterschiedlicher Weise: Auf der Anwendungsebene meint „verwenden“ tatsächlich das Benutzen bereits vorhandener Strukturen, auf der Modellierungsebene das Erzeugen der Strukturen, die für die Darstellung des Realweltausschnitts passend sind. Beides lässt sich grob in „Strukturieren“ zusammenfassen.
Strukturieren kennzeichnet sowohl den informatischen als auch den allgemeinbildenden Kern des Unterrichtsthemas „Datenbanken“. Einerseits kennt der fachdidaktische Diskurs das fundamentale informatische Prinzip der strukturierten Zerlegung ([Sc93], S. 23), das im Informatikunterricht vermittelt werden soll. Die „Beherrschung von Komplexität durch Strukturierung“ wird als Kompetenz zur Bewältigung von informatischen

Alltagsanforderungen angesehen ([SS04], S. 40). Diese Kompetenz gilt es, im Rahmen der informatischen Bildung zu entwickeln. Dies weist darauf hin, dass Strukturieren andererseits grundlegende Bedeutung über die Informatik hinaus hat: Durch das Strukturieren von Erfahrungen erschließt sich der Mensch die Welt, Lernen selbst kann als Wahrnehmen, Erkennen und Anwenden von Strukturen verstanden werden [An06].
Als basales Ziel von Allgemeinbildung, das durch Datenbankunterricht erreicht werden kann, ist allerdings das Strukturieren noch zu präzisieren – eine Beschränkung bloß auf das Erzeugen und Benutzen von Strukturen greift zu kurz. „Vollständiges Strukturieren" ist in diesem Zusammenhang vielmehr als Prozess mit den Phasen „Erlernen von Regeln zur Strukturbildung" – „Anwenden von Regeln zur Strukturbildung" – „Datengewinnung aus erzeugten Strukturen" – „Hinterfragen erzeugter Strukturen" zu verstehen (vgl. Abb. 2).

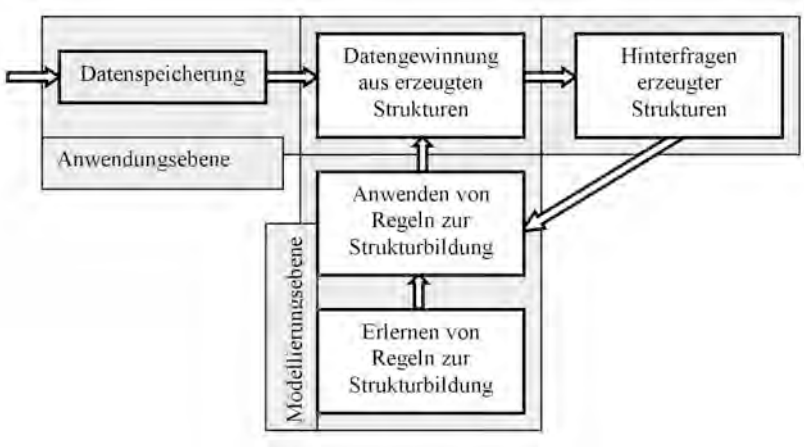


Abbildung 2: Phasentruktur von „vollständigem Strukturieren", integriert in die Anwendungs- und Modellierungsebene nach Abb. 1

2.1 Konzept „Erforschen und Anwenden"

Abb. 1 zeigt das Abfragen von Daten als zentrale Tätigkeit bei der Auseinandersetzung mit Datenbanken. Dies motiviert ein Vermittlungskonzept, das ausgehend von Datenbankabfragen alle vier oben identifizierten Strukturierungsaspekte integriert und so nicht nur die innerinformatische Bedeutung von Datenbanken betont, sondern auch dem allgemeinbildenden Aspekt des Strukturierens genügen kann:
Der typische Anwendungsfall für Datenbanken ist das Verknüpfen von Daten, die auf verschiedene Tabellen verteilt sind, mit dem Ziel, dadurch subjektiv neue Informationen zu gewinnen. Das Verknüpfen von Tabellen erfordert allerdings die Kenntnis der relationalen Datenstruktur. Dadurch ergibt sich die Möglichkeit, elementare Bausteine bzw. Regeln des Strukturierens/Modellierens wie „Entität", „Attribut", „Schlüsselattribut" oder (Abbilden einer) „Beziehung" mit Hilfe von Beispieldatenbanken lernbar zu machen, indem die Struktur von den Lernenden anhand geeignet formulierter Lernaufgaben

erforscht und erschlossen wird. Ausgangspunkt in dieser Phase des Erforschens ist dabei die allgemeinen Fragestellung: „Welche Tabellen müssen verknüpft werden, um eine bestimmte Information zu erhalten, und welcher Mechanismus ermöglicht das Verknüpfen dieser Tabellen?“ Dieser Zugang zum „Erlernen von Regeln zur Strukturbildung“ folgt den Ergebnissen neurobiologischer Forschung, nach denen Regeln von unserem Hirn selbst produziert werden, wenn es (viele) strukturell ähnliche Beispiele verarbeitet ([Sp02], S. 68ff).
Die Erfahrungen der Lernenden aus der ersten Phase führen auf drei Anwendungskontexte: Die Verknüpfungen, die zum Extrahieren gewünschter Daten benötigt und beim Erforschen gefunden wurden, können direkt als Datenbankabfragen formuliert werden („klassischer“ Anwendungskontext). Die durch die Anfragen gewonnenen Informationen bilden die Grundlage für die Rückkoppelung des Modells mit der Realität, für das „Hinterfragen erzeugter Strukturen“ („reflexiver“ Anwendungskontext[1]). Die erschlossenen Strukturierungsregeln schließlich erlauben das eigenständige Erstellen weiterer Datenbankmodelle von Realweltausschnitten durch die Lernenden („gestaltender“ Anwendungskontext).
Kern des Ansatzes ist also die didaktische Transformation von Anwendungs- und Modellierungsebene unter der Leitidee des vollständigen Strukturierens und dem Ziel, die aktive Beteiligung der Lernenden am Lernprozess zu erhöhen. Diese didaktische Transformation äußert sich auf Inhaltsebene durch die geänderte Akzentuierung und Abfolge von Komponenten gängiger Datenbank-Vermittlungskonzepte (vgl. Abb. 3).

2.2 Zum Problem des „guten“ Einstiegsbeispiels

Das „gute“ Einstiegsbeispiel ist der Angelpunkt des skizzierten Vermittlungskonzepts. Primär muss es Abfragen ermöglichen, die zur Auseinandersetzung mit der vorbereiteten Datenbank und damit zum Erforschen der vorhandenen Strukturen motivieren. Interessante Abfragen erfordern aber, dass die Lernenden mit dem modellierten Realweltausschnitt vertraut sind. Dies ist auch wichtig, damit auf Basis der Vertrautheit mit dem realen System die Abstraktionen des relationalen Modells erfasst werden können und auch eine ernsthafte Rückkoppelung des Modells mit der Realität möglich wird. Letztlich muss das Einstiegsbeispiel einerseits groß genug sein, damit die wesentlichen Strukturmerkmale des relationalen Modells repräsentiert sind, andererseits sollte es aber nicht zu komplex sein, damit die wesentlichen Strukturmerkmale erkennbar bleiben.
Zur Erläuterung des letzten Aspekts dient eine datenbankmäßige Abbildung von „Schule“ (vgl. Abb. 4), die die genannten Kriterien erfüllt und auch für die erste Erprobung des beschriebenen Vermittlungskonzepts im schulischen Informatikunterricht als Einstiegsbeispiel herangezogen wurde (s. Abschnitt 3).
An dem aus Sicht der Lernenden tatsächlich „großen“ Beispiel wurden zur Reduktion der Komplexität folgende strukturellen Vereinfachungen vorgenommen:

- Ausklammern von Aggregationen und Generalisationen als „strukturelle Besonderheiten“ (die auch anders modelliert werden können);

[1] Dieser reflexive Anwendungskontext kann (und sollte) auch zum Thematisieren von Aspekten des Datenschutzes und des Rechts auf informationelle Selbstbestimmung genutzt werden.

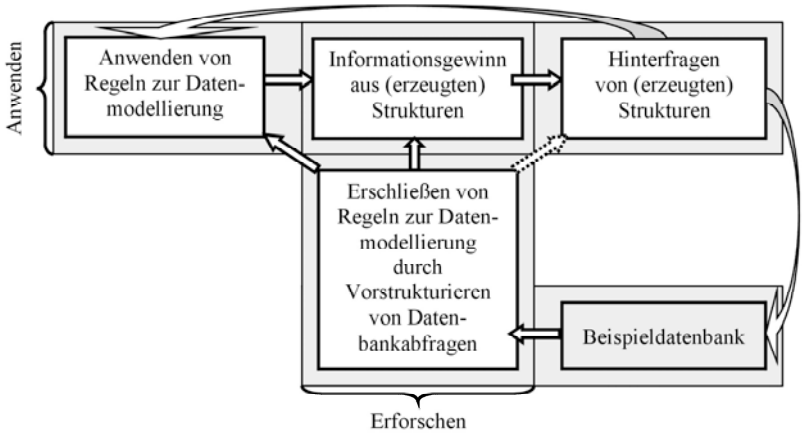


Abbildung 3: Didaktische Transformation der Phasentruktur von „vollständigem Strukturieren“ zur Vermittlung im Datenbankunterricht

- Darstellen jeder Assoziation als eigene Tabelle, unabhängig von den Kardinalitäten der beteiligten Entitäten, und damit im Zusammenhang
- Unterscheiden der Tabellen, die Entitäten darstellen, von denen, die Assoziationen darstellen, z.B. durch unterschiedliche Benennung;
- ausschließliches Modellieren von Entitäten, deren Primärschlüssel aus einem einzigen Schlüsselattribut besteht
- Visualisieren der Datenbankstruktur.

Diese Vereinfachungen und auch die in Abb. 4 dargestellte Beispieldatenbank weisen darauf hin, dass mit dem Einstiegsbeispiel keine „perfekte“ Datenbanklösung angestrebt werden soll. Dies scheint auch unter dem Aspekt des Hinterfragen-Könnens wenig sinnvoll. Wesentlich ist jedoch, dass das Einstiegsbeispiel der Erfahrungswelt der Lernenden entstammt (s.o.) und das strukturelle Grundkonzept des relationalen Datenmodells korrekt repräsentiert wird. „Fehler“ im Modell sind daher durchaus didaktisch intendiert und sollen als Möglichkeit für neue Lernerfahrungen genutzt werden, ganz im Sinne einer konstruktivistischen Fehlerkultur (siehe z.B. [Le01]).

3 ...und erste Erfahrungen

Das oben skizzierte Konzept wurde im Sommersemester des Schuljahres 2005/06 mit Schülern[2] im Alter von 16 bis 17 Jahren erstmals erprobt, als DBMS wurde Microsoft

[2] Konkret handelte es sich um eine Gruppe von 13 Schülern und einer Schülerin. Als Tribut an die Lesbarkeit wird im Artikel bei Pluralformen jedoch nur das männliche grammatikalische Geschlecht verwendet, das biologische weibliche ist aber selbstverständlich stets mitgemeint.

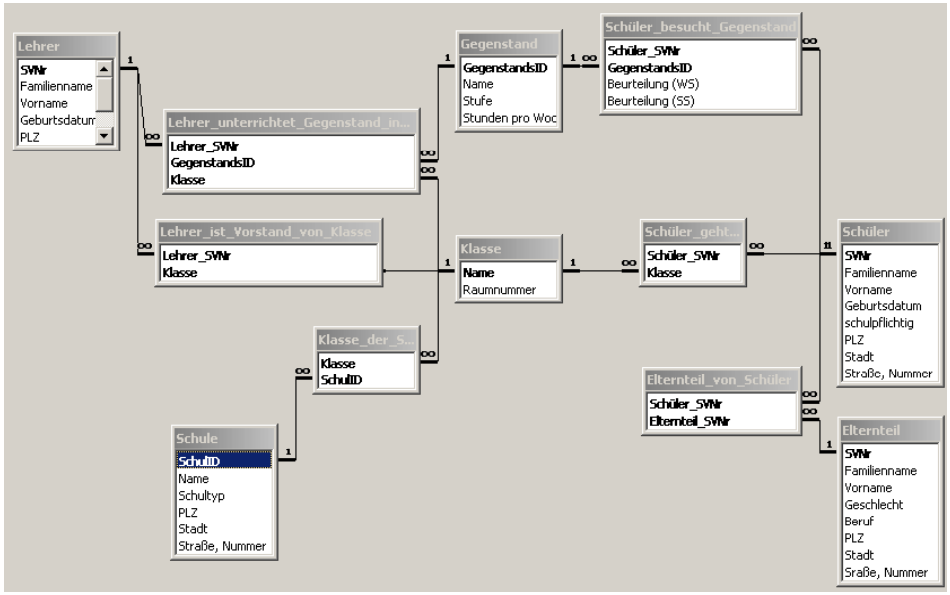


Abbildung 4: Beziehungsdiagramm eines Einstiegsbeispiels zum Erforschen und Hinterfragen der gegebenen Datenbankstruktur gemäß dem beschriebenen Ansatz

Access verwendet[3]. Geplant war die eine Einführung in das Thema „Datenbanken“ im Ausmaß von 12 Doppeleinheiten (à 100 Minuten), infolge von nicht vorhergesehenen Stundenausfällen stand jedoch tatsächlich nur etwas mehr als die Hälfte dieser Zeit zur Verfügung. Die informatischen Vorkenntnisse der Lernenden bestanden im Wesentlichen aus Programmierkenntnissen in C, wobei vor dem Datenbankkapitel der Verbund-Datentyp zum Verwalten von Datensätzen behandelt wurde.

3.1 Erforschen: Semantik des Beziehungsdiagramms

Die erste Phase (ca. eine Einheit) diente der Klärung grundlegender Begrifflichkeiten. Die Lehrperson erläuterte den Sinn einer Datenbank als „Beschreibung einer statischen Situation mit dem Ziel, zusammengehörige Daten zu verknüpfen“, und führte „Attribut“ als Synonym für „Komponente“ eines Verbunddatentyps ein. Mit diesem sprachlichen Grundgerüst ausgestattet erhielten die Lernenden die Aufgabe, anhand des Beziehungsdiagramms (vgl. Abb. 4) die Bedeutung der dargestellten „Rechtecke“ und „Linien“, insbesondere aber der unterschiedlich ausgezeichneten Attribute („normal“ bzw. „fett“) zu klären. Dazu erhielten sie einen zusätzlichen Impuls der Form: „Im Beziehungsdiagramm sind bestimmte Rechtecke durch Linien verbunden. Sie möchten wissen, in wel-

[3] Die Verwendung von Microsoft Access erklärt sich aus dem Umstand, dass diese Datenbanksoftware an österreichischen Schulen am verbreitetsten ist und auch die für die benötigten Visualisierungen der Datenbankstruktru geeignete Werkzeuge bereitstellt. Selbstverständlich ist aber jedes andere DBMS mit vergleichbarer Funktionalität ebenso geeignet.

chem Raum die Klasse eines bestimmten Schülers (z.B. des Schülers ‚Huber') ist. Helfen Ihnen die Verbindungslinien beim Lösen dieser Aufgabe?"
Die Schüler äußerten[4] sich zunächst derart, dass die Rechtecke das darstellen, „was es gibt, und auch anderes" und führten als Unterscheidungskriterium die unterschiedlich gebauten „Namen" der Rechtecke an. Nach der Analyse der „Linien" wurde die Vermutung geäußert, dass dieses „Andere" beschreibt, wie „das, was es gibt, zusammenhängt."
Hinsichtlich der fett ausgezeichneten Attribute fanden die Schüler daraufhin selbst, dass diese „Anknüpfungspunkte" zwischen den Rechtecksverbindungen darstellen, einige wenige wandten auch ein, dass diese die „Rechtecke" identifizieren, weil ja z.B. die Sozialversicherungsnummer eindeutig sein müsse.
Die gegensätzliche Interpretation dieser Attribute konnte erst durch die Erläuterung der Lehrperson, dass es sich dabei sowohl um identifizierende als auch um verbindende Attribute handelt, aufgelöst werden. „Identifizierendes Attribut" bzw. „verbindendes Attribut" wurde auch in weiterer Folge anstelle des abstrakteren „Schlüsselattribut" verwendet. Diese Begriffsbildung führte bei den Schülern zu der weiteren Beobachtung, dass diejenigen Rechtecke, die „Dinge" bezeichnen, nur ein identifizierendes Attribut besitzen, die anderen aber, die Verbindungen darstellen, so viele, „wie durch sie Rechtecke verknüpft werden". Schließlich fragten die Schüler auch noch nach den „Ziffern" 1 bzw. ∞, die bei den Linien dabeistehen, was durch die Lehrperson damit beantwortet wurde, dass jedes Ding (1) beliebig oft (∞) an den entsprechenden Verbindungen teilnehmen könne[5].

3.2 Erforschen: Makro- und Mikrostruktur der Datenbank

In der zweiten Phase (ca. eine Einheit) wurden die „Rechtecke" des Beziehungsdiagramms als Tabellen identifiziert. Dabei wurden Tabellenentwurf und Datenblattansicht unter dem Aspekt der Datenverknüpfung einander gegenübergestellt (vgl. Abb. 5). Die Schüler erhielten sodann die Aufgabe, den Aufbau und den Zusammenhang der Tabellen „Lehrer", „Lehrer_ist_Vorstand_von_Klasse" und „Klasse" (ohne Verwendung des Computers) ebenfalls in dieser Form darzustellen.
Mit der neuen Sichtweise „alles ist eine Tabelle" wurden die Schüler durch Aufgaben ähnlich der nachfolgenden angeleitet, im Beziehungsdiagramm „Verknüpfungspfade" über mehrere Tabellen zu identifizieren: „Die Schülerin Huber möchte eine Liste aller Lehrer haben, die sie unterrichten. Welche Tabellen werden benötigt, um diese Information aus der Datenbank zu erhalten? Wie werden die Verknüpfungen zwischen den einzelnen Tabellen hergestellt?" Dieser Wechsel von der Mikrostruktur der Tabellen zurück zur Makrostruktur der Beziehungen diente einerseits dem Zusammenführen dieser beiden Sichtweisen, andererseits aber auch zur Vorbereitung auf die Formulierung von Abfragen.

[4] Bei der Darstellung von Schüleräußerungen wird teilweise der Originalwortlaut verwendet, weil sich in diesem auch die Annäherung der Schülervorstellungen an die Begrifflichkeiten der Datenbankwelt dokumentiert.
[5] Bei dieser Frage verstellt die Modellierung jeder Assoziation als eigene Tabelle natürlich den Blick auf die im konzeptuellen Entwurf auftretenden Kardinalitäten/Konnektivitäten. Offenbar war aber die Erklärung der Lehrperson zu diesem Zeitpunkt mit den Vorstellungen der Schüler vereinbar und wurde akzeptiert.

Angesichts der aus Sicht der Schüler teilweise recht hohen Zahl von bis zu fünf zu verknüpfenden Tabellen wurde die Frage aufgeworfen, weshalb nicht die gesamte Information in einer einzigen Tabelle abgelegt werden könne. Klarerweise konnte dies entsprechend dem Kenntnisstand der Lernenden an dieser Stelle nicht mit „Anomalie“ und „Redundanz“ erklärt werden. Die Lehrperson machte aber anhand eines einfachen Beispiels (Zusammenführen der Tabellen „Schüler“, „Schüler_besucht_Gegenstand“ und „Gegenstand“) bewusst, dass die Speicherung in einer Tabelle zunächst einen Mehraufwand bei der Eingabe der Daten bedeutet, weil in diesem Fall gleiche Daten mehrfach in die Tabelle eingetragen werden müssen. Dabei konnte der Begriff „Redundanz“ eingeführt (!) werden.

3.3 Anwenden: Erstellen von Abfragen mit QBE und SQL

In der dritten Phase (2 Einheiten) wurden zunächst mit Hilfe von „Query by Example“ erste Datenbankabfragen formuliert. Die Wahl von QBE zum Einstieg in Datenbankabfragen erwies sich in mehrfacher Hinsicht von Vorteil:

- Erstens konnten die zuvor gefundenen „Verknüpfungspfade“ eins zu eins in QBE übertragen werden, was schnelle Erfolgserlebnisse bei den Lernenden ermöglicht.
- Die Lernenden hatten – entsprechend dem erworbenen „mentalen Modell“ der Datenbank – individuelle Erwartungshaltungen, wie das Ergebnis einer Abfrage aussehen sollte. Die Möglichkeit des schnellen Wechsels von der Entwurfsansicht zur Datenblattansicht bot die Gelegenheit, das erwartete mit dem tatsächlichen Ergebnis zu vergleichen, und so Fehler in der Formulierung der Abfrage oder im eigenen Verständnis von der Struktur der Datenbank (!) zu entdecken.
- Abfragen können in QBE vergleichsweise schnell „codiert“ werden. Dadurch erwarben die Lernenden bald einen „Erfahrungsschatz“, vor allem aber gelangten sie von selbst zur Erkenntnis, dass das Ergebnis von Abfragen auf Tabellen stets wieder eine Tabelle ist, und so Grundlage für eine weitere Abfrage sein kann.

Der Übergang von QBE zu SQL erfolgte durch den Wechsel zum „im Hintergrund“ erzeugten SQL-Code, der die wesentliche Struktur einfacher Abfragen (SELECT – FROM – WHERE) zeigt. Für zusammengesetzte Abfragen (d.h. Queries mit Subqueries) wurde der Aspekt wieder aufgegriffen, dass Abfragen stets wieder Tabellen liefern, die Grundlage für weitere Abfragen sein können. Dies sei am Beispiel der folgenden Abfrage kurz erläutert: „Gesucht sind die Beurteilungen aller Schüler in allen Gegenständen. Die Ergebnistabelle soll auch die Namen der Schüler und die Namen der Unterrichtsgegenstände enthalten.“ Der „Verknüpfungspfad“ besteht aus den Tabellen „Schüler“, „Gegenstand“ und „Schüler_besucht_Gegenstand“ (vgl. Abb. 4). Als Zwischenergebnis wurde zunächst eine Tabelle „Schüler_und_Beurteilungen“ mit den relevanten Attributen erstellt:

```
SELECT ... FROM Schüler INNER JOIN Schüler_besucht_Gegenstand
ON Schüler.SVNr = Schüler_besucht_Gegenstand.Schüler_SVNr;
```

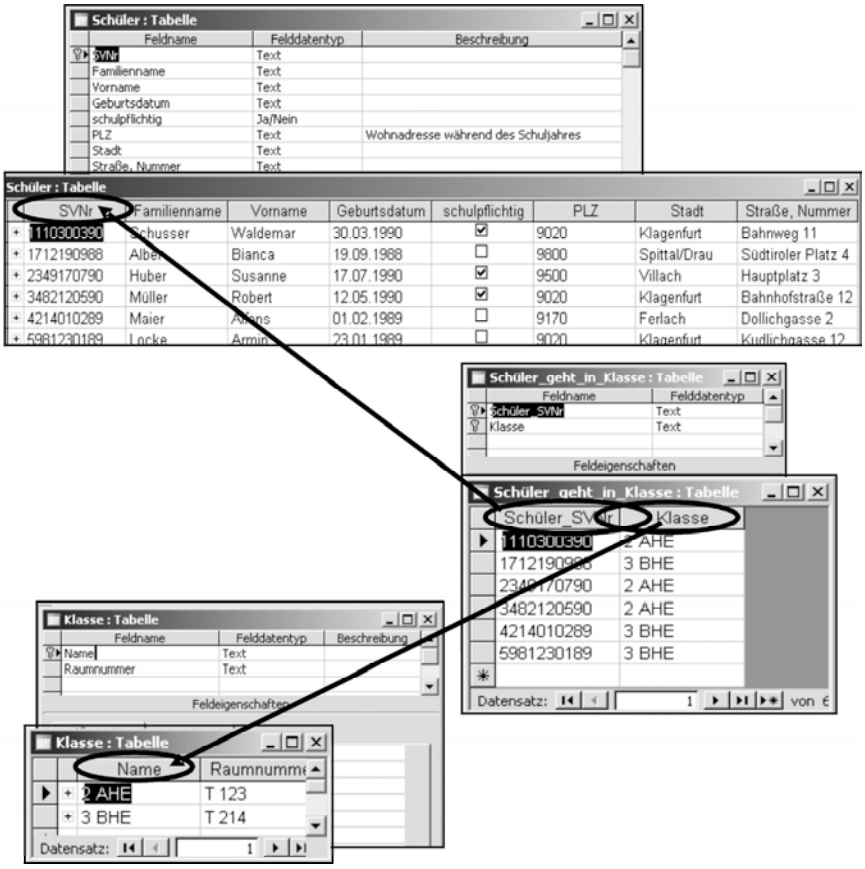


Abbildung 5: Darstellung der „Rechtecke“ aus der Beziehungssicht als Tabellen unter Betonung der „verbindenden Attribute“ – deren Funktion wird durch die einander überlappenden Ellipsen in der Tabelle „Schüler_geht_in_Klasse“ hervorgehoben

Auf Basis der Tabelle zu dieser Subquery lässt sich dann die eigentliche Abfrage recht einfach formulieren:

```
SELECT ... FROM Gegenstand INNER JOIN Schüler_und_Beurteilungen
ON Gegenstand.GegenstandsID = Schüler_und_Beurteilungen.GegenstandsID,
```

verglichen mit:

```
SELECT ... FROM Gegenstand INNER JOIN
        (Schüler INNER JOIN Schüler_besucht_Gegenstand
        ON Schüler.SVNr=Schüler_besucht_Gegenstand.Schüler_SVNr)
ON Gegenstand.GegenstandsID=Schüler_besucht_Gegenstand.GegenstandsID
```

Zusätzlich wurde dieser Prozess der Modularisierung mit Hilfe der QBE-Entwurfsansicht visualisiert:

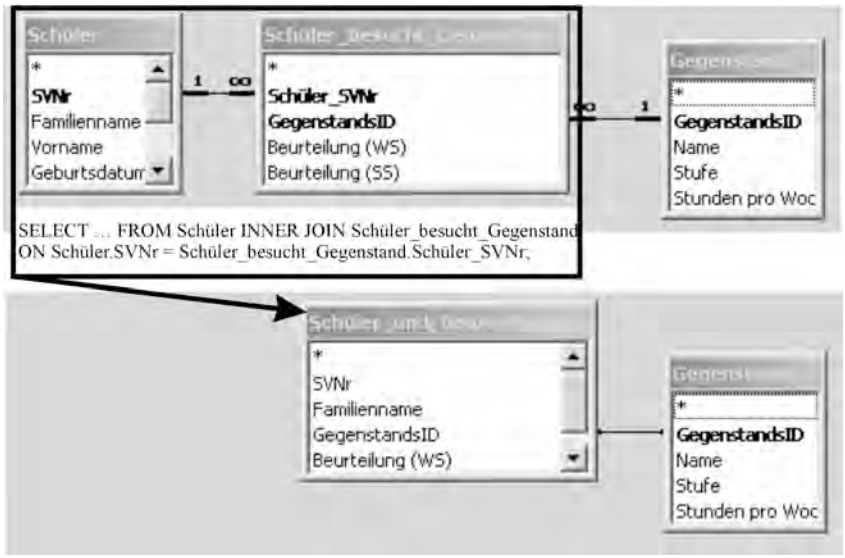


Abbildung 6: Visualisierung von Subqueries mit Hilfe der Ansicht „Abfrageentwurf“

Die Schüler erwarben durch derartige Sequenzen ein „Vorgehensmodell“ zum Formulieren zusammengesetzter Abfragen.

3.4 Anwenden: Hinterfragen und Modellieren

In der letzten Phase (etwa 4 Einheiten) der Unterrichtssequenz wurden die Lernenden angeleitet, die vorgegebene Datenbankstruktur zu hinterfragen. Zusätzlich sollte auch ein einfaches „Tabellenschema“ (entsprechend dem bisher diskutierten) für einen anderen Realweltausschnitt eigenständig entworfen werden.
Zum Einüben des Formulierens von Abfragen wurden den Schülern zunächst weitere, „interessante“ Aufgaben vorgelegt, wie z.B.: „Erzielen Mädchen in Schule bessere Ergebnisse als Burschen?“ (vgl. [Mo92] für weitere Beispiele von Abfragen, die auch „sensiblere“ Daten erschließen). Abgesehen davon, dass derartige Fragestellungen vielfältige Assoziationen der Lernenden zur eigenen Schulsituation hervorriefen (als „Rückkoppelung“ zur „Realität“), ist speziell diese Frage in der gegebenen Beispieldatenbank schlechterdings nicht zu beantworten, da in der Tabelle „Schüler“ das Geschlecht nicht „gemerkt“ wird.
Auf der Suche nach weiteren „Schwächen“ der Datenbank bemerkten die Schüler, dass ja jede Klasse nur einen Klassenvorstand haben kann. Diese Eindeutigkeit kann aber durch die Assoziationstabelle „Lehrer_ist_Vorstand_von_Klasse“ nicht abgebildet werden. Die Schüler äußerten aber recht schnell eigenständig (!) die Vermutung, dass dieses Problem doch durch „Verschieben“ des identifizierenden Attributs von „Lehrer“ zu „Klasse“ gelöst werden könne. Es gäbe offenbar verschiedene Formen von Assoziatio-

nen („eindeutige und nicht-eindeutige“), die im Datenbankmodell auch unterschiedlich abgebildet werden müssten. Besonders bemerkenswert ist die auf dieser Erkenntnis aufbauende Schlussfolgerung eines Schülers, der meinte, es sei dann wohl nicht optimal, beim „Entwerfen“ der Datenbank von dem Tabellenschema auszugehen. Sinnvollerweise sollte es ein „vorgelagertes Schema“ geben.
Mit diesem Impuls und dem bereits vorhandenen Strukturwissen der Schüler konnte die konzeptuelle Modellierung (außer Plan!) rasch besprochen werden. Dabei wurde die UML-Notation verwendet, da diese der Darstellung im Access-Beziehungsdiagramm sehr ähnlich ist. Auch beim daraufhin eigenständig zu erstellenden Datenbankentwurf verwendeten die Schüler diese Form der Notation. Die Überführung in ein konsistentes Tabellenschema (inklusive „besserer“ Auflösung von (1:n)-Beziehungen) bereitete aufgrund der Vorerfahrung mit Datenbank-Tabellen keine Schwierigkeiten.

4 Resümee und Ausblick

Informatikunterricht erhebt den Anspruch, allgemeinbildend zu sein. Damit sind bei der Auswahl und der Aufbereitung der zu unterrichtenden Themen unter anderem die Fragen zu beantworten, inwieweit das jeweilige Thema ein über sich hinausweisendes Merkmal aufweist, und ob dieses Allgemeine von den Lernenden in der arrangierten Lernsituation auch tatsächlich erfasst werden kann (nach [Hu00], S.57).
Strukturieren kann als ein allgemein bildendes Merkmal der Beschäftigung mit Datenbanken angesehen werden. Die Beobachtung, dass bei gängigen Vermittlungskonzepten zu Datenbanken das Strukturieren aber häufig auf das bloße Anwenden mehr oder weniger verstandener Regeln durch die Lernenden reduziert wird, motivierte den alternativen Ansatz, nach dem die Lernenden durch Anwenden von Datenbanken zunächst Verständnis für die zugrunde liegenden Strukturen gewinnen. Dabei ist das vorgestellte Konzept nicht das einzige, das mit diesem Ziel den von Anwenderschulungen her bekannten Ansatz des Arbeitens mit „fertigen“ Beispieltabellen aufgreift. Diese Idee findet sich auch in [FKN06] (S. 5 – 78), jedoch werden dort vergleichsweise „kleine“ Datenbanken verwendet und die Strukturen schrittweise vom Lehrer/von den Autoren erklärt. Zum Unterschied davon betont das oben erläuterte Konzept das „angeleitete Erforschen und Hinterfragen“ der Strukturen einer „großen Datenbank durch die Lernenden selbst.
Die beschriebenen ersten Erfahrungen weisen darauf hin, dass der vorgeschlagene Ansatz – auch bei knapper Zeitressource – zum intendierten Lernerfolg führen kann. Für den Einsatz des Konzepts im Informatikunterricht scheint aber noch zweierlei notwendig. Zum einen sind weitere geeignete Beispiele zu finden. Dies meint Einstiegsbeispiele, die der Erfahrungswelt der Schülerinnen und Schüler entstammen (vorstellbar wäre z.B. das Modell eines Pizza-Service oder einer Online-Tauschbörse), aber auch Beispiele, an denen das erworbene Strukturverständnis angewandt werden kann. Insbesondere bei der zweiten Gruppe kann auf vorhandene und kommentierte gute Beispieldatenbanken, wie z.B. „Flotter Flitzer“ von J. Penon/S. Spolwig zurückgegriffen werden[6]. Zum anderen erscheint die Erweiterung des Ansatzes in Richtung „Erforschen von Redundanz

[6] Die angeführte Beispieldatenbank findet sich – neben anderen – unter http://www.be.schule.de/bics/inf2/datenbanken/unterrichtsbeispiele.html. (24. 04. 2007)

und Anomalien“ ebenso sinnvoll und notwendig, wie eine Vertiefung des reflexiven Anwendungskontextes und das Entwickeln von Konzepten zur Überprüfung der von den Lernenden mit diesem Ansatz erworbenen Kompetenzen.

Literaturverzeichnis

[An06] Antonitsch P.K.: Databases as a Tool of General Education. In (Mittermeir R.T. Ed.): Informatics Education – The Bridge between Using and Understanding Computers. Proc. 2nd Int. Conf. On Informatics in Secondary Schools – Evolution and Perspectives (ISSEP), Vilnius 2006. Springer, Berlin Heidelberg, 2006; S. 59-70

[BM04] Bundesministerium für Bildung, Wissenschaft und Kultur: Lehrplan der AHS-Oberstufe für den Pflichtgegenstand Informatik (Version 2004); verfügbar unter: http://www.bmukk.gv.at/medienpool/11876/lp_neu_ahs_14. (24.04.2007) bzw. Lehrplan für den Wahlpflichtgegenstand Informatik (Version 2004); verfügbar unter http://www.bmukk.gv.at/medienpool/11876/lp_neu_ahs_21. (24.04.2007)

[FKN06] Fischer H., Knapp T., Neupert H.: Grundlagen der Informatik II. Oldenbourg, München et al. 2006

[Ha05a] Habenicht K.: Grundlagen der Datenverarbeitung mit MS Access. ADIM Schriftenreihe Wissen in Beispielen, Band 205, Wien 2005; s.a.: http://www.adim.at/wib/ (17.01.2007)

[Ha05b] Habenicht K.: Fortgeschrittene Datenverarbeitung mit MS Access. ADIM Schriftenreihe Wissen in Beispielen, Band 206, Wien 2005; s.a.: http://www.adim.at/wib/ (17.01.2007)

[Hu00] Hubwieser P.: Didaktik der Informatik. Springer, Berlin – Heidelberg 2000

[Ja03] Jaros H.: Grundkurs Datenbankentwurf, 3. Aufl. Vieweg, Wiesbaden 2003

[Ko04] Konopasek K.: Access 2003 Advanced ECDL Modul AM 5, 3. Aufl. ikon, Brunn am Gebirge 2004

[Le01] Leuders T.: Qualität im Mathematikunterricht. Cornelsen, Berlin 2001

[Mo92] Modrow, W.: Zur Didaktik des Informatik-Unterrichts, Band2. Dümmler, Bonn 1992

[Ne04] Neubauer G.: Access 2003 Grundlagen ECDL Modul 5, 5. Aufl.. ikon, Brunn am Gebirge 2004

[Ro06] Rohrer M.: Evaluation des Informatikunterrichts in den 1./2. Klassen der AHS in Kärnten. Veröffentlichung im Rahmen des IMST^3-Projektes, Villach 2006; verfügbar unter: http://imst.uni-klu.ac.at/materialien/2006/1145_337_Langfassung_Rohrer.pdf (17.01.07)

[Sc05] Schöndorfer C (Betr.): Materialien zu Datenbanken im Rahmen des Lehrganges „Angewandte Informatik“ der Initiative „e-teaching austria“. Onlineversion 2005; verfügbar unter: http://www.e-teaching-austria.at/AINF/ (17.01.2007)

[Sc93] Schwill A.: Fundamentale Ideen der Informatik. In: Zentralblatt für Didaktik der Mathematik 1, 1993; S. 20-31

[Sp02] Spitzer M.: Lernen. Gehirnforschung und die Schule des Lebens. Spektrum, Heidelberg – Berlin 2002

[SS04] Schubert S., Schwill A.: Didaktik der Informatik. Spektrum, Heidelberg – Berlin 2004

ELTIS :: Technische Informatik – Fernstudium für Schüler

Johanna Bucur, Werner Grass, Rudolf Kammerl, Franz Weitl

Universität Passau, 94030 Passau
{Johanna.Bucur, Werner.Grass, Rudolf.Kammerl, Franz.Weitl}@uni-passau.de

Abstract: Dieser Beitrag stellt das innovative Projekt „ELTIS - Schülerfernstudium durch die Verwendung von Elearning-Kursen in Technischer Informatik“[1] vor und berichtet über erste Ergebnisse. ELTIS verfolgt den Ansatz des Blended-Learning. Im Rahmen eines internetgestützten Fernstudiums bekommen Schülerinnen und Schüler die Möglichkeit, schon vor dem Abitur Studienleistungen im Bereich der Technischen Informatik zu erbringen, auch wenn sie in keiner Universitätsstadt zur Schule gehen. Das Angebot sieht hier sowohl die Teilnahme im Rahmen von Unterricht wie auch im Selbststudium vor. Der Beitrag beschreibt schwerpunktmäßig das didaktisch-methodische Konzept des Elearning-Angebots. Anhand zweier Fallbeispiele werden die Perspektiven von Lehrenden und Lernenden durchleuchtet. Fragestellungen, Methoden und erste Ergebnisse der Begleitforschung werden erläutert. Gefördert wird ELTIS von der Deutschen Telekom Stiftung und vom Europäischen Sozialfonds.

1 Einführung: Warum ein Fernstudienangebot für Schülerinnen und Schüler?

1.1 Motivation für den Einsatz von Elearning

ELTIS ist ein von der Deutschen Telekom Stiftung und vom Europäischen Sozialfonds gefördertes Elearning-Projekt der Universität Passau mit dem Ziel, Lehrern und Schülern ein ergänzendes Lernangebot im Bereich der Technischen Informatik anzubieten und einen Bezug zum späteren Studium und zur Praxis herzustellen.

Das neue Hochschulgesetz ermöglicht Lernenden an Gymnasien Studienleistungen zu erbringen, die bei einem späteren Studium angerechnet werden. Damit werden mehrere Ziele verfolgt: Zum einen soll Schülern eine ihren Fähigkeiten entsprechende Förderung zu Teil werden. Zum andern wird durch Anrechnung der von Schülern erbrachten Leistungen in einem späteren Studium eine Reduzierung der Studiendauer angestrebt.

[1] Schülerfernstudium mit Hilfe von eLearning Kursen http://elearning.fmi.uni-passau.de/elearning/index.htm

Die bisherigen Angebote von verschiedenen Universitäten erfordern allerdings eine Präsenz der Schüler in den entsprechenden Lehrveranstaltungen am Hochschulort. Daher können solche Angebote nur von Schülerinnen und Schülern wahrgenommen werden, die in der Nähe einer Universitätsstadt wohnen. Schüler/innen, die in strukturschwachen Regionen leben, erfahren dadurch eine indirekte Benachteiligung, da es für sie ungleich aufwändiger ist, ein entsprechendes Angebot wahrzunehmen. Gleichzeitig stehen aber die Technologien zur Verfügung, mit denen interessierte Schüler prinzipiell auf weiterführende Bildungsangebote zugreifen könnten. Nach den repräsentativen Daten des Medienpädagogischen Forschungsverbundes Südwest nutzen über 80 % der Gymnasiasten Computer und Internet regelmäßig zu Hause (JIM 2005).

Im Rahmen von ELTIS wird ein orts- und zeitunabhängiger Zugang zu multimedial aufbereiteten und tutoriell begleiteten Online-Kursen über geeignete Themen der Technischen Informatik geschaffen. Die Möglichkeit, selbstständig, eigenverantwortlich und ohne „Lernzwang“ Lösungen für praxisrelevante Probleme zu entwickeln, weckt die Begeisterung für ein technisch ausgerichtetes Studium. Hochbegabte Schüler/innen werden durch anspruchsvolle Zusatzaufgaben herausgefordert und gefördert. Die Möglichkeit, im Rahmen von ELTIS bereits als Schüler später anrechenbare Studienleistungen zu erbringen, steigert zusätzlich die Leistungsbereitschaft, und verkürzt die Studiendauer.

Die Betreuung der Schüler erfolgt in enger Zusammenarbeit mit den teilnehmenden Schulen. Hierbei ist eine Vielzahl von organisatorischen, methodisch-didaktischen und technischen Fragestellungen zu lösen. Dieser Beitrag thematisiert folgende Kernherausforderungen:

- Die Entwicklung multimedialer Online-Lernangebote ist zeit- und kostenintensiv. Wie können Lernangebote gestaltet werden, so dass sie sich gleichermaßen zur Unterstützung der Präsenzhochschullehre wie zur Ergänzung des Schulunterrichts im Rahmen eines Fernstudiums eignen?
- Die anfängliche Begeisterung für ein neues Lernangebot nimmt oft schnell ab. Wie können die für Online- und Fernstudiumsangebote üblichen hohen Drop-Out Quoten vermieden werden?
- Isoliertes Lernen am Rechner ist selten Ziel führend. Wie kann eine Lehr-/ Lerngemeinschaft etabliert und die Kommunikation zwischen Universität, Lehrkräften und Schüler/innen effektiv und nachhaltig gestaltet werden?

Der Beitrag ist wie folgt strukturiert: In Kapitel 2 stellen wir die Struktur, die Inhalte und den methodischen Ansatz des Projekts vor. In Kapitel 3 beschreiben wir die Fallbeispiele, berichten über erste Evaluationsergebnisse und diskutieren, inwiefern die Zielsetzungen des Projekts bereits in der Pilotphase verwirklicht werden konnten. Der Artikel endet mit einer Zusammenfassung der wichtigsten Erkenntnisse und einem Ausblick auf die nächsten Schritte.

2 Umfang, Ziele und Methodischer Ansatz des Projekts

2.1 Umfang und Status des Projekts

Die Förderung von ELTIS ist in der ersten Ausbaustufe für das Schuljahr 2006/2007 terminiert und auf Themen der Technischen Informatik beschränkt. Das Projekt gliedert sich in eine Vorbereitungsphase, Akquisephase, Pilotphase und eine Durchführungsphase. Begleitet wird das ELTIS durch ein assoziiertes Forschungsprojekt am Lehrstuhl für Allgemeine Pädagogik der Universität Passau. So wird eine systematische und methodisch abgesicherte Erhebung relevanter Erfolgsfaktoren gewährleistet mit dem Ziel, die Übertragbarkeit der Ergebnisse des Projekts für andere Maßnahmen sicherzustellen und bereits während der Projektdurchführung schnell Verbesserungsmöglichkeiten zu realisieren.

An der Pilotphase nehmen zwei Schulen mit je einem betreuenden Lehrer und insgesamt 25 Schülern teil. Die Teilnehmer werden auf der Seite der Universität Passau von 2 Professoren und 3 wissenschaftlichen Mitarbeitern betreut. Für die Durchführungsphase konnten 13 weitere Schulen mit insgesamt 100 Schülern akquiriert werden. Ein weiterer Ausbau der Nutzerbasis und des Lehrangebots nach Abschluss der ersten Ausbaustufe ist geplant.

Zum Zeitpunkt dieses Beitrags befindet sich das Projekt im Übergang zwischen Pilot- und Durchführungsphase. Über erste Erkenntnisse und Evaluationsergebnisse aus der Pilotphase berichten wir in Kapitel 3.

2.2 Fachliche und Pädagogisch-Didaktische Ausrichtung

Das Kursmaterial führt in die Konzeption digitaler Schaltungen ein, die die Bausteine eines Prozessors bilden. Die Themen sind so gewählt, dass sowohl ein hoher Praxisbezug als auch ein guter Einblick über grundlegende Methoden und Konzepte der (Technischen) Informatik sichergestellt ist. Beispiele für behandelte Themen sind: Funktion Boolescher Komponenten und von elementaren Speichern, Verhaltens- und Strukturbeschreibung von Schaltnetzen und Schaltwerken, Analyse und Synthese, Zahlendarstellung und Rechenschaltungen, Modellierung des Zeitverhaltens auf verschiedenen Abstraktionsebenen. Als formale Konzepte werden Boolesche Terme, Termersetzung, Termdarstellung grafischer Strukturen, Hierarchisierung, Zahlendarstellung, zeitbehaftete Boolesche Terme, endliche Automaten eingeführt.

Ziel ist nicht die vollständige Abdeckung des behandelten Gebietes. Vielmehr wird exemplarisch der Nutzen mathematischer Methoden und Modelle für die Lösung praxisrelevanter Probleme beim Aufbau digitaler Schaltungen vermittelt. Bei den Lernzielen stehen die Analyse von Problemen, die Synthese bekannter Konzepte zur Lösung neuer Aufgabenstellungen und die Beurteilung der Bedeutung der dargestellten Konzepte und Methoden im Vordergrund. Für hochbegabte Schüler werden zusätzlich spezifische Lernpfade mit anspruchsvollen Aufgaben angeboten. Damit wird eine leistungsorientier-

te Unterrichtsdifferenzierung ermöglicht. Einzelheiten über inhaltliche Abstufung sowie Lernziele und –pfade sind auf dem ELTIS Portal[2] verfügbar.

Für das Kursmaterial wurde eine deutlich formalere Darstellung als an den meisten Universitäten gewählt. Sie orientiert sich unter anderem an der Einführung in die Informatik von M. Broy [Br95]. Mathematische Funktionen bilden die formale Basis. Damit kann weitgehend auf die Voraussetzung einer technischen Intuition bei den Schüler/innen verzichtet werden. Speziell für didaktische Erläuterungen wurde ein Schaltungssimulator verwendet [DKW03, DKW04]. Dieser ermöglicht das interaktive Experimentieren mit den vermittelten Konzepten und das Lösen von praxisrelevanten Aufgabenstellungen mit unmittelbarem Feedback bzgl. möglicher Fehler (Abbildung 1).

Durch die Implementierung des Schüler-Fernstudiums wird auch die Medienkompetenz der Teilnehmer/innen gefördert. Den Lernenden wird das Internet als ein potentieller Bildungskanal nahe gebracht. Da gerade Jugendliche das Medium Internet in der Regel eher freizeitorientiert (Spiele, Chats) nutzen, kann das Aufzeigen dieser Nutzungsalternative zur Erweiterung der Mediennutzung beitragen.

2.3 Methodisches Konzept und Lehr-/Lernszenarien

Folgende Nutzungsszenarien werden durch ELTIS unterstützt:

- Einbettung des Elearningkurses im Unterricht:
  - als Unterrichtseinheit für alle Schüler/ Schülerinnen: Lehrer entscheiden, ob und in wie weit das Elearningangebot für die Schüler und Schülerinnen im Rahmen des Fachunterrichts, integriert wird.
  - als Differenzierungsmaßnahme/Angebot für leistungsstarke Schülerinnen/ Schüler: Lehrer entscheiden sich dafür, dass insbesondere leistungsstarke Schülerinnen und Schüler das Elearningangebot als individuelles Angebot bearbeiten. Während für die schwächere Schülergruppe z. B. Wiederholungen angeboten werden, kann die stärkere Gruppe an diesem weiterführenden Angebot teilnehmen.
- Selbstständiges Studium: Unabhängig vom Unterrichtsangebot entscheiden sich einzelne Schülerinnen und Schüler für die Teilnahme am Elearningangebot. Sie werden vor Ort nicht von einem Lehrer betreut und bearbeiten das Material selbständig.

Im Rahmen des begleitenden Forschungsprojekts wurden Maßnahmen zur Gestaltung der tutoriellen Begleitung (3.1 & 3.2), zur Unterstützung und Moderation der Peer-to-Peer-Kooperation und zur Begleitung der Lehrkräfte erarbeitet. Spezielle Lehrerworkshops, Infoveranstaltungen und Online-Foren werden sich in der Durchführungsphase als noch wichtiger erweisen, wenn die Kernaufgabe der Universität darin bestehen wird,

[2] http://elearning.fmi.uni-passau.de/elearning/pages/selbststudium/selbststudium.htm

viele Teilnehmer/innen und deren betreuenden Lehrkräfte von verschiedenen Gymnasien zu betreuen.

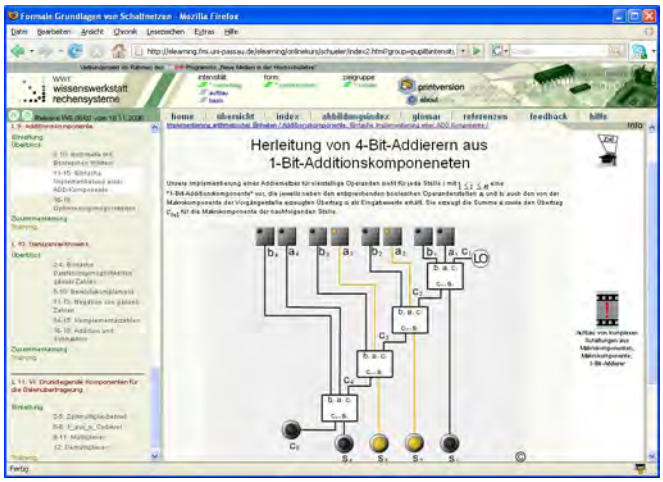


Abbildung 1: interaktiv manipulierbarer Schaltungssimulator

2.4 Gestaltung der Lernmedien

Die in ELTIS eingesetzten Lernmedien basieren auf den im Rahmen des Projekts „WWR - Wissenswerkstatt Rechensysteme“ [LT04, Ko04] entwickelten multimedialen Kurseinheiten für die Hochschulausbildung in Technischer Informatik. Die Materialien sind optimiert für einen multimedialen Unterricht. Die hohe Anschaulichkeit der zu vermittelnden Inhalte durch Diagramme, Schemata, Schaltbilder, Animationen zur Darstellung der Funktionsweise, Simulationen und sowie der Verzicht auf unstrukturierte lange Texte begünstigt die interaktive Nutzung.

Allerdings sind nur wenige der WWR-Kurse für Anfänger geeignet. Selbst der hier verwendete Grundkurs musste für den Einsatz im Rahmen von ELTIS überarbeitet und Zielgruppen-spezifisch ergänzt werden.
Lehrende sowie Lernende können auf ein klassisches Skript, Folienkopien der entsprechenden Präsenzveranstaltung und Aufgabenblätter zugreifen, die Lehrenden zusätzlich auf Musterlösungen. Letztere können den Schülern auch zur Verfügung gestellt werden, jedoch nicht eigenverantwortlich, vielmehr entscheidet der betreuende Lehrer über Umfang und Zeitpunkt der Verfügbarkeit.

Der Transfer in ein neues Lehr-/Lernszenario gelingt durch die Modularität und insbesondere durch die Skalierbarkeit der WWR-Kurseinheiten, die durch eine speziell für das WWR-Projekt entwickelte XML-Auszeichnungssprache ermöglicht wird.[We02] Durch die Modularität können Teile eines Kurses leicht neu kombiniert und an neue Lehr-/Lernszenarien angepasst werden. Die Skalierbarkeit der WWR-Module hinsichtlich des

Umfangs und Schwierigkeitsgrads (Basis, Aufbau und Vertiefung) ermöglicht die Unterstützung verschiedener Leistungsniveaus. WWR-Module erlauben darüber hinaus die Integration von Zielgruppen-spezifischen Inhalten.

Die Ergänzungen der Lernmaterialien für ELTIS kommen auch den Studierenden an der Universität Passau zugute. Anschauliche Beispiele und vereinfachte Aufgabenstellungen werden gerne von Studierenden wahrgenommen und können eingesetzt werden, um heterogene Vorkenntnisse im Vorfeld einer Lehrveranstaltung auszugleichen.

Die technische Realisierung der Lernmedien ermöglicht, dass alle Nutzungsszenarien innerhalb von ELTIS und der Hochschullehre aus einer einzigen Medienquelle bedient werden können. Der Überarbeitungs- und Pflegeaufwand wird dadurch drastisch reduziert, die Qualität gesteigert, die Anpassbarkeit an neue Lehr-/Lernszenarien erhöht und die Nachhaltigkeit der Bemühungen sichergestellt.

2.5 Technisches Konzept, Web-Portal, Diskussionsforen

Das Kursmaterial wurde in eine Plattform eingebunden, welche Kommunikationskanäle wie Diskussionsforen sowie Hintergrund- und Ergänzungsmaterial anbietet.

Es werden verschiedene Versionen des Kursmaterials angeboten: für Selbstlerner (Passwort-geschützt), für Lehrkräfte (Passwort-geschützt) sowie für Schüler, die in einem Klassenverband durch Lehrkräfte betreut werden.

3 Evaluation und vorläufige Ergebnisse

Der Elearning-Kurs wurde in der Pilotphase an zwei Schulen durchgeführt jeweils in einem unterschiedlichen Szenario: Am Goethe Gymnasium Regensburg wird das Modell des Virtuellen Klassenzimmers eingesetzt, und am Gymnasium Vilshofen das synchrone, Unterricht flankierende Modell. Die folgenden Fallbeispiele illustrieren charakteristische Ergebnisse und Erkenntnisse aus der Pilotphase.

3.1 Fallbeispiel 1: Synchrones, den Präsenzunterricht flankierendes Schülerstudium, 1:1 Betreuung (GY Vilshofen) Erfahrungsbericht

Die Schülerinnen und Schüler konnten sich bei diesem Konzept entscheiden, ob sie „regulär“ am Unterricht teilnehmen oder Elearning praktizieren. Die Elearning-Kandidaten und Kandidatinnen werden per Email über die zu bearbeitenden Themen informiert und mit Übungen versorgt. Sie müssen zwei mündliche Prüfungen und natürlich die Klausur ablegen.

Rahmenbedingungen

Das Vorwissen der Schüler war sehr unterschiedlich. Ein Großteil der Schüler/innen besaß ein „sehr rudimentäres informatorisches Wissen“. Daneben gab es auch Teilneh-

mer/innen mit weit überdurchschnittlichem Wissen, das sich diese selbstständig über das von der Universität Passau angebotene Schülerstudium angeeignet hatten.

Um den Ansprüchen des leistungsdifferenzierten Unterrichts gerecht zu werden, musste dieser Grundkurs „zweigleisig" gestaltet werden.

- Schüler/innen mit wenig Erfahrung besuchten wöchentlich den Grundkurs; sie nahmen regelmäßig an der Präsenzveranstaltung teil. Die Präsenzveranstaltung lief in der Regel nach folgendem Schema ab:
    1. Teil (ca. 45 Min): Durcharbeiten des Onlineskripts unter Vorgabe der behandelnden Abschnitte
    2. Teil (ca. 45 Min): Besprechung und Nachbereitung des erarbeiteten Stoffes und Behandlung von Übungsaufgaben zur Vertiefung
- Schüler/innen mit ausreichender Erfahrung konnten den Kurs als reinen Elearning-Kurs absolvieren. Die Schüler wurden über E-Mail über den momentanen Stand in der Präsenzveranstaltung informiert. Eine Anwesenheitspflicht in den Grundkursstunden bestand nicht, eine Teilnahme war aber jederzeit möglich. Termine für die notwendigen mündlichen Leistungserhebungen wurden mit diesen Schülern individuell abgesprochen und durchgeführt. Für die Klausur sowie die Veranstaltungen an der Universität Passau bestand Anwesenheitspflicht.

Die Schüler konnten sich zu Schuljahresbeginn für eine der beiden Schienen entscheiden. Dabei wurden sie auf Wunsch auch individuell beraten.

Vor der eigentlichen Bearbeitung des Elearning-Kurses wurden – in Anlehnung an die entsprechende Präsenzvorlesung – noch die grundlegenden Themengebiete Codierung und Fehleranalyse bei der Nachrichtenübermittlung behandelt. Dabei wurde auf das in der Plattform zur Verfügung gestellte Material[3] zurückgegriffen.

Didaktische Strategie im Grundkurs

Für den Grundkurs wurde eine Webseite (http://www.gym-vilshofen.de/hp-gymvof/index.php?main=10) eingerichtet, auf der der aktuelle Stand der Schul-Präsenzveranstaltung zeitnah dokumentiert wurde. Auf dieser Seite waren auch die Übungsblätter mit Lösungen sowie aktuelle Hinweise, beispielsweise zu Veranstaltungen an der Universität Passau, hinterlegt.

Um dem E-Learnen einen Rahmen zu geben, wurde diesen wöchentlich eine Email mit den derzeit in der Schul-Präsenzveranstaltung behandelten Abschnitte und dem Übungsblatt zugesendet.

Tutorielle Unterstützung

Der Großteil der tutoriellen Betreuungsarbeit wurde wie intendiert an den Schulen selbst geleistet. Die Kommunikation zwischen Betreuungslehrern und Schülern fand über E-

[3] http://elearning.fmi.uni-passau.de/elearning

Projektleiter Gymnasium **Projektleiter Uni Passau**	**Hr. OStR Stefan Winter** **Prof. Dr. Werner Grass**		
Teilnehmer/innen	**14** Schüler	**1** Schülerin	**Σ 15**
Hintergrund	▪ Mathematisch-naturwissenschaftlich ▪ Neusprachlich		
Dauer / Termine	▪ Wintersemester 2006/ 2007		
Alter	▪ 17 und 18 Jahre		
Selektionskriterien	▪ Keine ▪ Wahlmöglichkeit des **Grundkurses** im Rahmen des Zusatzangebotes der Kollegstufe ▪ Kein **Vorwissen** vorausgesetzt, Vorwissen jedoch vorhanden		
Ziele für die Lernenden	▪ Neues Lernen, Einblick in Theorie und Praxis ▪ Anwendungen der Informatik selbst ausprobieren ▪ Zusammenarbeit mit der Universität ▪ Ansprüche der Universität kennen lernen ▪ Formale Inhalte begreifen und die abstrakten Zusammenhänge „veranschaulichen“ ▪ Klausuren an der Uni mitschreiben ▪ ECTS-Punkte für das Studium „sammeln“ ▪ Anwendungen der Informatik selbst ausprobieren und Neues dazulernen ▪ Erfahren was es heißt, ein Problem im Team zu lösen ▪ Eigene Informatik-Studierfähigkeit testen ▪ Einen Einblick in das Berufsfeld von Diplom-Informatikern bekommen		
Ziele für den Lehrer	▪ Einstieg in die Technische Informatik ▪ Vermittlung einer großen Bandbreite an Informationen; Überblick vermitteln und keine zu große Vertiefung ▪ Kurs hat einen sehr hohen Motivationswert ▪ Klare abgeschlossene Einheiten in einer 45-minütigen Phase im Team erarbeiten mit anschließender Besprechung und einer Transferaufgabe ▪ Messbaren Wissenszuwachs: Leistungskontrollen, Notenvergabe ▪ Begabtenförderung ▪ Praktikum an der Uni Passau im März 2007		
Beobachtungen	▪ Ungleichgewicht in der Förderung der Guten und sehr Begabten gegenüber den Schwachen ▪ Für durchschnittliche Schüler/innen ist der Kurs eine „Herausforderung“. Langfristig gesehen – ein Gewinn. ▪ Geschlechterproblematik ändert sich - G8 Schülerinnen sind die besseren Zuhörerinnen und erbringen im Durchschnitt die besseren Ergebnisse		

Abbildung 2: Grundkurs Informatik: Pilotprojekt „Informatik studieren - bereits während der Schulzeit“

mails oder persönliche Gespräche, hauptsächlich aber in den Präsenzveranstaltungen bzw. Unterrichtseinheiten an den Gymnasien statt.

- Die elektronischen Kommunikationsangebote der Uni Passau, im Speziellen das Forum und der Chatbereich auf der Elearning-Homepage, wurden von den Teilnehmern bisher wenig genutzt, da in der Pilotphase der Kontakt zwischen den Schulen und der Uni sehr eng und die Teilnehmerzahl bei zwei Gymnasien überschaubar war.
- Die beiden Praxis-Veranstaltungen an der Uni Passau wurden von den Lehrern als sehr motivationsfördernd eingeschätzt.
- Die Schüler bekamen einen echten Eindruck vom Uni-Alltag mit Lehrveranstaltungen im Unterschied zum Schulalltag. Auf diesem Weg konnten auch viele Fragen zwischen Betreuungslehrern und dem Informatik-Lehrstuhl persönlich im Dialog geklärt werden.

Auch während der normalen Kursphase fand der Austausch zwischen den Schulen und der Uni meist in persönlichen Gesprächen per Telefon oder E-Mail statt.

In der bevorstehenden Projektphase wird aufgrund der großen Teilnehmerzahl und der räumlichen Distanz zwischen Uni und den teilnehmenden Schulen angestrebt, den Großteil der Kommunikation auf der elektronischen Ebene, also im Forum und Chatbereich der Elearning-Homepage, abzuwickeln.

3.2 Fallbeispiel 2: Asynchrones Schülerstudium, Elearning (Goethe-Gymnasium - Regensburg) Erfahrungsbericht

Rahmenbedingungen

Das Goethe-Gymnasium Regensburg ist die zweite der Pilotschulen, die im Wintersemester 2006/2007 am Schülerfernstudium teilgenommen haben. Der Kurs wurde in einem kleinen Rahmen (5 Schüler) angeboten. Aufgrund der Heterogenität der Arbeitsgruppe im GK Informatik (Klasse, Alter, Vorwissen) hat sich der Lehrer für das Szenario des Virtuellen Klassenzimmers entschieden. Dieses Szenario unterstützte optimal eine leistungsorientierte Unterrichtsdifferenzierung und Begabtenförderung. Seine Vorgehensweise ist mit der am Gymnasium Vilshofen (3.1.1) vergleichbar.

Didaktische Strategie

Zwei formelle Treffen in der Schule wurden ursprünglich geplant, die auch plangemäß stattgefunden haben. Diese dienten in erster Linie dazu, sich über beim Studium aufgetretene Probleme auszutauschen und zu lösen. Zur Halbzeit konnten die Teilnehmer/innen an einem Multiple-Choice-Test teilnehmen, was sich als guter „Motivationsschub“ erwies.

Tutorielle Unterstützung

Die tutorielle Betreuung dieser Gruppe war von einer sehr großen Flexibilität seitens des Lehrers geprägt. Der Lehrer stand seinen Schülern jederzeit zur Verfügung: „Kleine Fragen sofort, größere Fragen in einer Freistunde“ – so der Lehrer.

Trotz der virtuellen Schiene, haben die Schüler aufgrund der geringen TN-Zahl die elektronischen Online-Kommunikationsmöglichkeiten (Forum, Chat) kaum in Anspruch nehmen müssen. „Schulintern hat uns das persönliche Treffen und Gespräch als Kommunikationsplattform gereicht.“ (Aussage eines Schülers)

3.3 Erste Rückmeldungen und Befunde aus beiden Pilotgruppen

An der Pilotphase haben insgesamt 25 Schüler/innen teilgenommen, wobei niemand den Kurs vorzeitig abgebrochen hat. Was bedeutet die Beteiligung an ELTIS für Schüler und Schülerinnen, was bedeutet sie für Lehrer/innen, und was bedeutet sie für Schulen? An einer freiwilligen Zwischenevaluation haben 12 Schüler/innen teilgenommen. Den Evaluationskommentaren zufolge bietet ein Studium zu Schulzeiten Schülern und Schülerinnen diverse Vorteile:

- Es macht Spaß, weil der Stoff viele Beispiele und Animationen aus der realen Welt beinhaltet.
- Es ist interessant, auch mal über den Tellerrand der Schule zu schauen und ins Studium rein zu schnuppern.

- Schüler/innen können eine Prüfung machen, die später im „normalen“ Studium angerechnet wird.
- Ab dem Schuljahr 2007/2008 soll es möglich sein, Leistungen, die in Hochschulveranstaltungen erzielt worden sind, auf Antrag in der Jahresfortgangsnote im entsprechenden Fach zu berücksichtigen. Das führt zu einer sehr erfreulichen „Win-Win-Situation“.

Schülerfernstudium aus Sicht der Lehrenden

Laut Aussagen der bisher beteiligten Schulen bietet der Online-Kurs auch für Lehrer entscheidende Vorteile:

- Für Informatik begabte Schüler und Schülerinnen können ohne größeren Aufwand auf Seiten der Lehrer motiviert und gefördert werden.
- Dem Lehrer bietet sich die Möglichkeit, sich unkompliziert weiter zu bilden.
- Der Einsatz des Kurses ist evtl. in dem ab 2009 in der Kollegstufe anzubietenden Seminarfachs möglich.
- Begabtenförderung
- Elearning als Möglichkeit der Stoffvermittlung und Kooperation mit der Universität Passau ist „äußerst interessant und gewinnbringend“.
- Beide Lehrer der Pilot-Gruppen haben den Wunsch geäußert, dass weitere Projekte (sowohl Aufbau- als auch thematisch neue Projekte) initiiert werden.

Schulen

Auch für die Schulen eröffnen sich durch die Elearning Plattform neue Perspektiven:

- Das Angebotsspektrum der Schulen wird vergrößert.
- Ein „lebendiger“ Kontakt zu den Hochschulen fördert die Öffnung der Schule nach außen. Es entstehen Kontakte zur Universität Passau mit Besuchsmöglichkeiten, Informationsaustausch, Gewinnung von Referenten für Studienvorstellungen, etc.
- Die Schule erhält auf einfache Weise die Möglichkeit der Begabtenförderung.

Kursbegleitende Veranstaltungen an der Universität Passau

Die Veranstaltungen an der Universität (18.10. 2006 und 21.03.2007) waren sehr motivationsfördernd. Am zweiten Termin konnten die Teilnehmer den Lernstoff an einem umfangreichen praktischen Beispiel erproben.

Darüber hinaus ergaben sich weitere Vorteile:

- Einblick in die Universität in Hinblick auf „Größe“ und „Flair“.

- Kontakt mit Professoren und Assistenten und der damit verbundene Abbau von Berührungsängsten.
- Einblick in die Lehrveranstaltungen, die von einem Professor bzw. einer/m Assistenten gehalten werden, die sich in sehr vielen Aspekten vom gewohnten Schulalltag unterscheiden.

4 Zusammenfassung und Ausblick

ELTIS ist ein innovativer Ansatz, das Interesse von Schülerinnen und Schülern an technischen Berufen zu wecken und frühzeitig an Studienangeboten von Universitäten heranzuführen. In Form von Blended-Learning können auch Schüler, die abseits von Universitätsstädten wohnen, das Bildungsangebot wahrnehmen und bereits vor dem Abitur Studienleistungen erbringen. Speziell aufbereitete multimediale Lernmaterialien ermöglichen eine leistungsbezogene Differenzierung des Angebots und damit die Förderung von Schülerinnen und Schülern mit unterschiedlichen Begabungen und Interessen. ELTIS ist eng verzahnt mit der grundständigen Hochschulausbildung und pädagogischer Forschung, wodurch eine hohe inhaltlich-methodische Qualität, ein begrenzter Aufwand für Materialerstellung und tutorielle Begleitung sowie die Nachhaltigkeit der Ergebnisse sichergestellt ist. Die Lehrerinnen und Lehrer profitieren vom Zugang zu hochwertigen Unterrichtsmaterialien, vom gegenseitigen Austausch mit Projektpartnern und von der Möglichkeit der individuellen Weiterbildung. Die Schulen profitieren durch die Erweiterung ihres Bildungsangebots und die Universität durch den frühzeitigen Kontakt mit zukünftigen Studenten. Erste Evaluationsergebnisse bestätigen die Gesamtkonzeption von ELTIS und ermutigen zu weitergehenden Schritten.

Für die bevorstehende Durchführungsphase des Projekts sind folgende Erweiterungen in Planung bzw. bereits realisiert:

- Im Sommersemester können die Pioniere (Spontananmelder, die nicht im Klassenverband arbeiten werden) in die Materie einsteigen.
- Es werden zusätzliche Übungsblätter erstellt (2 pro Kurs), die von Mitarbeiter/innen der Universität korrigiert und bewertet werden
- Vernetzung der neu akquirierten Schulen, die territorial weit von einander entfernt liegen, durch Intensivierung der Synchronisationstreffen an der Universität Passau und elektronischer Austauschplattformen.

Danksagung

Wir bedanken uns bei der Deutschen Telekom-Stiftung und dem Europäischen Sozialfonds- für die Förderung des Projektes. Herrn StR Meiringer vom Goethe-Gymnasium Regensburg und Herrn OstR Winter vom Gymnasium Vilshofen danken wir für die Durchführung des Pilotprojektes und die Anregungen zur Verbesserung des Ablaufs.

Literaturverzeichnis

[Br95] M. Broy: „Informatik: eine grundlegende Einführung Band I und II“, Berlin: Springer Verlag, 1995

[DKW03] M. Damm, B. Klauer, K. Waldschmidt: „LogiFlash – A Flash-based Logic-Simulator for educational Purposes”, World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED-MEDIA 2003), Honolulu, Hawaii / USA, 23.-28. Juni 2003.

[DKW04] M. Damm, B. Klauer, K. Waldschmidt: „LogiFlash: Ein didaktischer Logiksimulator zur Integration in Lernobjekten“, Workshop Structured eLearning – Wissenswerkstatt Rechensysteme, Rostock, 30. März 2004.

[Ko04] L. Kornelsen, U. Lucke, D. Tavangarian, D. Voigt, M. Waldhauer: „Inhalte und Ergebnisse des Verbundprojekts Wissenswerkstatt Rechensysteme (WWR)“, GI Softwaretechnik-Trends 24/01, Februar 2004.

[LT04] U. Lucke, D. Tavangarian (Hrsg.): „Structured eLearning: Wissenswerkstatt Rechensysteme“, Tagungsband des Workshops Structured eLearning – Wissenswerkstatt Rechensysteme, Rostock, 2004. - ISBN 3-86009-267-7.

[We02] F. Weitl, C. Süß, R. Kammerl, B. Freitag: „Presenting Complex e-Learning Content on the Web: A Didactical Reference Model” In: G. Richards (Ed.), Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2002/ (pp. 1018-1025). Chesapeake, VA: AACE.

[We04] F. Weitl, B. Freitag, W. Grass, B. Sick, R. Kammerl, M. Göstl, A. Wiesner: „Mediendidaktische Aufbereitung von Vorlesungsinhalten für das Online-Lernen“, Workshop Structured eLearning – Wissenswerkstatt Rechensysteme, Rostock, 30. März 2004.

Einstieg Informatik – Aktivitäten und Erfahrungen

Wolfgang Pohl, Katharina Kranzdorf, Hans-Werner Hein

Bundeswettbewerb Informatik, Ahrstr. 45, 53175 Bonn
pohl@bwinf.de

Abstract: Aufgabe des Projekts „Einstieg Informatik“[1] war, im Informatikjahr 2006 Kinder und Jugendliche für Informatik zu interessieren und für eine aktive Beschäftigung mit Informatik zu gewinnen. Dazu führte das Projekt eine öffentliche Kampagne durch, unterstützt durch das Webportal einstieg-informatik.de, und realisierte ein Online-Spiel und einen Online-Wettbewerb. Für den Einsatz in der Kampagne wurden zahlreiche Spiel- und Lernangebote zur Informatik entwickelt, die vielfach ohne Computereinsatz auskommen. Entsprechendes Material wurde auch Dritten zur Verfügung gestellt. Die gute Resonanz auf die Angebote des Projekts untermauert insbesondere, dass Informatikgrundlagen attraktiv (auch ohne Technikeinsatz) einem breiten Publikum vermittelt und Informatikkonzepte in geeigneter Weise mit Kindern ab dem Grundschulalter behandelt werden können.

1 Motivation

Die Verarbeitung von Informationen mit Informatik-Systemen ist für den überwiegenden Teil der Bevölkerung der so genannten entwickelten Staaten Lebensbestandteil. Von der Geburtsanzeige bis zur Sterbemeldung werden persönliche Daten in Informatik-Systemen erfasst und verarbeitet; im Arbeitsleben und zunehmend auch im privaten Bereich gehört der eigenständige Umgang mit Informatik-Systemen zum Alltag.

Voraussetzung für ein selbstbestimmtes und -verantwortliches Leben in einer derart von informatischen Phänomenen geprägten Gesellschaft ist eine Einsicht in diese Phänomene; nach [Br00] trägt ein Verständnis der Wirkprinzipien von Informatik-Systemen zu deren Entmystifizierung bei. Ein solches Verständnis kann aber nur durch zumindest grundlegende Kenntnisse und Kompetenzen in der zu Grunde liegenden Wissenschaft erreicht werden: der Informatik. Erforderlich ist also eine „informatische Allgemeinbildung“ – durchaus auch in der von Koubek [Ko05] geprägten Bedeutung des Begriffs (vgl. hierzu auch [Sc06, S. VII] und [Hr06, S. 339-346]). Diese ist wiederum Grundbaustein für eine vertiefte informatische Bildung, die insbesondere die Fähigkeit zur konstruktiven Nutzung von Informatik-Systemen und damit auch einer ausreichenden Zahl junger Menschen die

[1] „Einstieg Informatik“ wurde vom 1. November 2005 bis 31. Januar 2007 als Projekt des „Wissenschaftsjahr 2006 – Informatikjahr“ von der Geschäftsstelle des Bundeswettbewerbs Informatik durchgeführt. Das Projekt wurde unter dem Kennzeichen 01WJE05 vollständig vom Bundesministerium für Bildung und Forschung (BMBF) gefördert, die administrative Trägerschaft hatte die Gesellschaft für Informatik e.V. (GI).

nötige Begeisterung vermitteln sollte, sich in Ausbildung und Beruf der Informatik zuzuwenden und zu qualifizierten und verantwortungsvollen Experten zu reifen.

Das Projekt „Einstieg Informatik“ (im weiteren auch kurz: EI), das im Informatikjahr 2006 durchgeführt wurde, sollte in beiden Bereichen wirken: zum einen ganz vorne in der informatischen Bildungskette ansetzen und Wege zu frühen Informatik-Erfahrungen sammeln und probieren, zum anderen Interesse an Informatik bei Jugendlichen schaffen und stärken. Die Angebote des Projekts waren also auf Kinder und Jugendliche ausgerichtet, bei vielen der öffentlichen Aktionen konnte Informatik aber auch Eltern und anderen Erwachsenen näher gebracht werden. Eine erste Beschreibung der Projektansätze findet sich in [PKH06].

Im folgenden Abschnitt wollen wir die grundlegende Arbeitsweise des Projekts näher beschreiben. Anschließend, im Kern dieses Beitrags, stellen wir anhand einiger Beispiele vor, welche Inhalte vermittelt wurden, auf welche Art dies versucht wurde und welche Erfahrungen dabei gemacht wurden. Abschließend wird dargelegt, welche Schlussfolgerungen wir aus der Arbeit im Informatikjahr ziehen.

2 Einstieg Informatik

Zur Erreichung seiner Ziele hat sich das Projekt auf vier Wegen an Kinder, Jugendliche und Lehrkräfte (speziell des Faches Informatik; letztere als Multiplikatoren) gewandt:

Kampagne: Bei öffentlichen Veranstaltungen, die sich zumindest teilweise an Kinder und Jugendliche wandten, war Einstieg Informatik mit Informationsständen, Lernspielen, Mitmach-Aktionen, Bühnenshows und Workshops präsent. Darunter waren Veranstaltungen des Wissenschaftsmarketings wie die Regionalveranstaltungen der GI und der Fraunhofer IuK-Gruppe, die Dresdener Woche der Informatik oder die Science Days im Europapark Rust; Berufs- und Ausbildungsmessen wie die Veranstaltungen der Reihe „Einstieg Abi“; fachbezogene Schülercamps und -kongresse; und auch andere Messen mit jugendlichem Publikum wie die Games Convention und die Jugendmesse YOU. Außerdem präsentierte EI seine Angebote auch Lehrkräften, speziell der Informatik, u.a. auf mehreren Tagungen von Lehrerfachgruppen der GI mit Ständen, Vorträgen und Workshops. Insgesamt waren Mitarbeiter von Einstieg Informatik in 12 Monaten auf 27 Veranstaltungen präsent, dazu kamen fünf Veranstaltungen, wo EI durch Dritte präsentiert wurde.

Webportal: Das Webportal einstieg-informatik.de sollte Jugendlichen, die durch die Kampagne von EI erfahren hatten, eine Möglichkeit zur Vertiefung ihres Interesses an Informatik-Inhalten bieten. „Live-Berichte“ von aktiven Informatikern, Interviews mit vorbildlichen Informatik-Persönlichkeiten (berühmte und weniger berühmte), aber auch Tipps und Hinweise auf Bücher, Software und Ausbildungsmöglichkeiten sollten insbesondere die Älteren ansprechen. Außerdem wurden über das Webportal Material zu Projektaktivitäten (s.u.) veröffentlicht. Vom Start am 1.4.2006 bis zum 25.1.2007 verzeichnete das Portal 108.175 Besuche (Visits). Trotz dieses guten Wer-

tes lässt sich die Wirkung der Angebote des Webportals nur schwer beurteilen. Es gab nur wenige konkrete Rückmeldungen, diese allerdings positiver Natur. Interaktive Möglichkeiten wie die Kommentarfunktion des Portals wurden kaum benutzt.

Service: Eine zentrale Aufgabe des Projekts war, Material zu sammeln oder selbst zu entwickeln, das zur Arbeit mit Kindern und Jugendlichen verwendet werden konnte. So wurden u.a. einige Kapitel des Buches „Computer Science Unplugged“ [BFW98] mit Genehmigung der Autoren übersetzt und zur Nutzung in Workshops oder Mitmach-Aktionen aufbereitet. Insbesondere aber wurden eigene Ideen umgesetzt. Im nächsten Abschnitt werden die wichtigsten dieser Spiel- und Lernangebote beschrieben. Auf Anfrage wurde Arbeitsmaterial zusammen mit Werbematerial verschickt; insgesamt wurden 160 Materialpakete versandt.

Spiel: Das Online-Spiel EI:SPIEL für Jugendliche von 10 bis 16 Jahren vermittelt im Sinne einer informatischen Allgemeinbildung Kenntnisse zur Informatik, Denkanstöße zum Leben in der Informationsgesellschaft und gleichzeitig Anregungen zum aktiven Umgang mit Informatik. An diesem Spiel nahmen seit seinem Start am 1.6.2006 558 Kinder und Jugendliche teil. Ergänzend zu diesem auf eine dauerhafte Beteiligung angelegten Angebot wurde unter dem Titel EI:SPIEL blitz! eine Sonderrunde durchgeführt, bei der die Teilnehmer unter Zeitdruck einen Multiple-Choice-Test zu absolvieren hatten. An diesem Online-Wettbewerb nahmen 2.126 Jugendliche teil.

Im folgenden Abschnitt wollen wir einige der von Einstieg Informatik entwickelten und genutzten Spiel- und Lernangebote näher beschreiben, außerdem wird auf EI:SPIEL und insbesondere EI:SPIEL blitz! näher eingegangen.

3 Beispiele

3.1 EI:CODE

Der EI:CODE ist ein einfaches Lernzeug zur Textverschlüsselung mit einem Schiebecode – auch Caesar-Code genannt. Es wird als DIN-A5-Blatt verteilt, aus dem der fertige EI:CODE durch eigene Bastelarbeit erstellt werden kann. Für den Einsatz des EI:CODE im Rahmen der Kampagne wurden in verschiedenen Schwierigkeitsstufen „Botschaften“ erstellt: verschlüsselte Wörter oder Sätze, die mit Hilfe des EI:CODES entschlüsselt werden können. Zur Lösung dieser Rätsel musste die Funktionsweise des EI:CODE verstanden werden; auf die Beachtung von Buchstabenhäufigkeiten wurde hingewiesen. Diese Art von Denksport war generell sehr beliebt, und zwar in allen Altersgruppen (also auch bei Erwachsenen), was mit dem Sudoku-Phänomen verglichen werden kann: Es ist populär, sich durchaus schwierigen, aber auf strukturierte Art und Weise zu lösenden Aufgaben zu stellen. Für die Vermittlung eines positiven Bildes der Informatik kann dies noch viel stärker genutzt werden.

Schon Grundschülerinnen und Grundschüler konnten die einfachen Botschaften entschlüsseln und dabei intuitiv das Verstehen und Nachvollziehen eines Algorithmus üben. In einer

4. Klasse wurde an einem suggestiven Beispiel (Verschlüsselung des Wortes „Erdbeere“) erkannt, dass die Häufigkeit bzw. Wahrscheinlichkeit des Auftretens von Buchstaben bei der Entschlüsselung eine Rolle spielen kann.

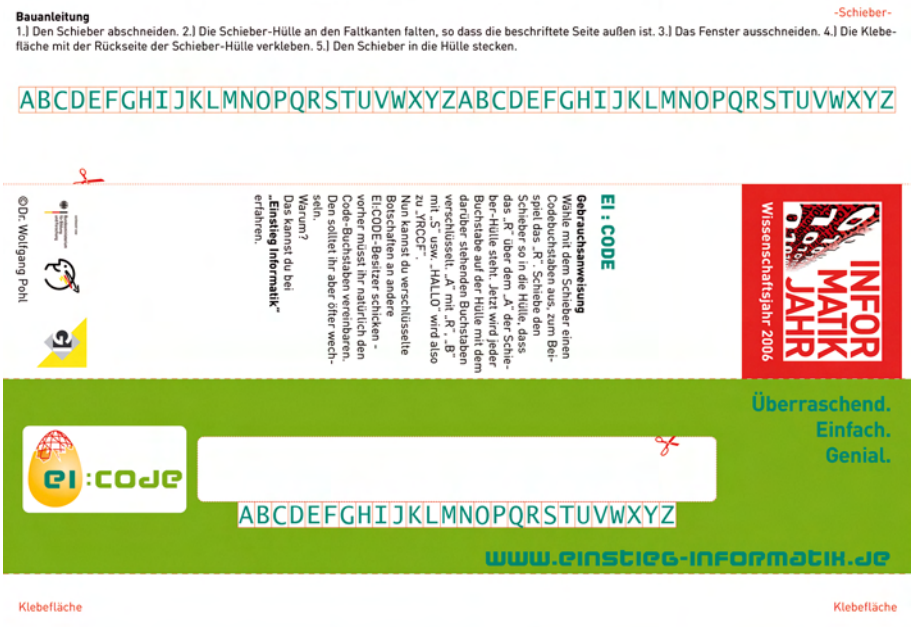


Abbildung 1: Verschlüsselungslernzeug EI:CODE

Mit dem EI:CODE werden Kinder und Jugendliche auf ein wichtiges Thema im Zusammenhang mit Informatik-Systemen aufmerksam gemacht, nämlich Sicherheit und Privatheit von Informationen. Durch die teilweise leichte Entschlüsselbarkeit der Botschaften wird die Bedeutung der Qualität kryptographischer Verfahren unmittelbar vermittelt. Abbildung 1 zeigt einen EI:CODE der ersten Auflage.

Vom EI:CODE wurden im Rahmen der Projektaktivitäten oder auf Anfrage knapp 20.000 Exemplare gezielt ausgegeben, an Besucher von Messeständen, aber in großer Zahl auch an Lehrkräfte und andere Erzieher. Außerdem wurden 55.000 Exemplare als Beilage der Zeitschrift bildung+science sowie 6.500 Exemplare mit der Aussendung der Erstrundenaufgaben des 25. Bundeswettbewerbs Informatik verteilt.

3.2 EI:DOIT

Der EI:CODE vermittelt Phänomene der Informatik, ohne dabei Informatiksysteme zu nutzen – ganz im Sinne des Prinzips „Computer Science Unplugged“ [BFW98]. Die Aktion EI:DOIT (sprich wie englisch „I do it“) geht noch weiter und benötigt gar kein Material; Tafel und Kreide können allerdings hilfreich sein. Bei diesem „Tu-was-Spiel“, wie es

zuerst genannt wurde, können die Spieler sich gegenseitig mit Bewegungsbefehlen „programmieren“, wobei sich zuerst die Spielleitung als „Roboter“ zur Verfügung stellen sollte.

Ausgangspunkt ist eine möglichst kleine Menge von Bewegungsbefehlen wie „schritt“ (mache einen Schritt vorwärts) und „rechts“ (drehe dich um 90 Grad nach rechts). Mit diesen Befehlen kann ein „Programmierer“ den gespielten Roboter steuern. Schnell treten dabei Wünsche nach Erweiterung des Befehlssatzes auf, anhand derer unterschiedliche Konzepte der Programmierung behandelt werden können:

Zählschleifen: Dreimaliges Ausführen des Befehls „rechts“ bewirkt eine Linksdrehung; es kann vereinbart werden, dass die Programmiererin statt „rechts rechts rechts“ auch „dreimal rechts“ sagen darf.

Prozeduren: Um dem Programmierer (aber nicht dem Roboter!) das Leben zu erleichtern, kann vereinbart werden, dass dreimaliges Rechtsdrehen auch durch den Befehl „links“ abgerufen werden kann.

Hardware-Implementation: Am Beispiel des neuen Befehls „links“ ist leicht zu demonstrieren, wie vorteilhaft es sein kann, häufig benötigte Befehle in der Hardware des Systems zu implementieren. Dann ist dem Roboter erlaubt, sich beim Befehl „links“ direkt nach links zu drehen.

Parameter: Die Programmierer haben großen Spaß daran, den Roboter gegen Hindernisse zu steuern. Zusammenstöße werden oftmals durch die feste Schrittweite des Roboters mit verursacht. Wenn der Ruf nach einem „kurzen Schritt“ oder „langen Schritt“ laut wird, kann vereinbart werden, dass Varianten des Befehls „schritt“ erlaubt sind, wie z.B. „schritt:kurz“ und „schritt:lang“.

EI:DOIT wurde bevorzugt mit Kindern gespielt. In der vierten Klasse einer Grundschule konnten alle oben genannten Konzepte ohne Schwierigkeiten erarbeitet werden. In einer Variante, bei denen die schriftliche Notation der Befehle durch eine grafische ersetzt wurde, konnte EI:DOIT auch mit Kindern ab fünf Jahren gespielt werden. Auch als Bühnenaktion mit Laufpublikum (in einer großen und belebten Einkaufspassage) wurde EI:DOIT verwendet. Zur Unterstützung von EI:DOIT wurde ein Java-Applet entwickelt, mit dem man ein EI:DOIT „Programm“ notieren und interpretieren lassen kann: die Abfolge der auszuführenden Basisbefehle wird visualisiert (s. Abbildung 2; der obere Kasten enthält die Grundbefehle, darunter Prozeduren und dann das Hauptprogramm. Im unteren Kasten zeigt der „Interpreter“ den aktuell auszuführenden Befehl an).

3.3 Schwanensee

Für ältere Jugendliche wurde der von der BWINF-Geschäftsstelle entwickelte Programmier-Workshop „Zufriedenheit am Schwanensee“ weiter ausgearbeitet. Grundlage ist die Programmierumgebung „StarLogo“ (http://education.mit.edu/starlogo/) in der Version 2.2. Dies ist eine Logo-Variante, bei der das programmierte Verhalten simultan agierender Akteure („turtles“) simuliert und in einer eingebauten grafischen Benutzeroberfläche visualisiert wird.

Abbildung 2: EI:DOIT „Programmierumgebung“

Dieser Workshop soll den Teilnehmenden eine erste motivierende und ermutigende Begegnung mit Programmierung bieten und wurde unter anderem erfolgreich mit ausschließlich weiblicher Beteiligung durchgeführt. Hierfür sind nach unseren Erfahrungen die folgenden Elemente von Bedeutung:

1. Die Programmierumgebung und die darin verwendete Programmiersprache sind nicht allgemein bekannt, so dass bei keinem Lernenden Vorkenntnisse vermutet werden; alle können mit gleichen Chancen an den Start gehen.

2. Die Programmierumgebung vermittelt eher einen spielerischen als einen technischen Eindruck. In StarLogo kann ein Benutzer auch ohne Programmierung erste Handlungen durchführen, wie Platzierung der turtles auf dem „Spielfeld“ oder Wahl des Erscheinungsbilds der turtles. StarLogo-Anweisungen können auf die so gestaltete Umgebung einzeln angewandt werden, so dass Erfolgserlebnisse leicht eintreten. In den verschiedenen Durchführungen des Workshops haben insbesondere Mädchen diese spielerischen Möglichkeiten gerne zum „Aufwärmen“ genutzt.

3. Die Aufgabenstellung hat eine positive Zielsetzung. In „Zufriedenheit am Schwanensee“ sollen die turtles (für die eine Visualisierung als „Schwan“ gewählt wurde)

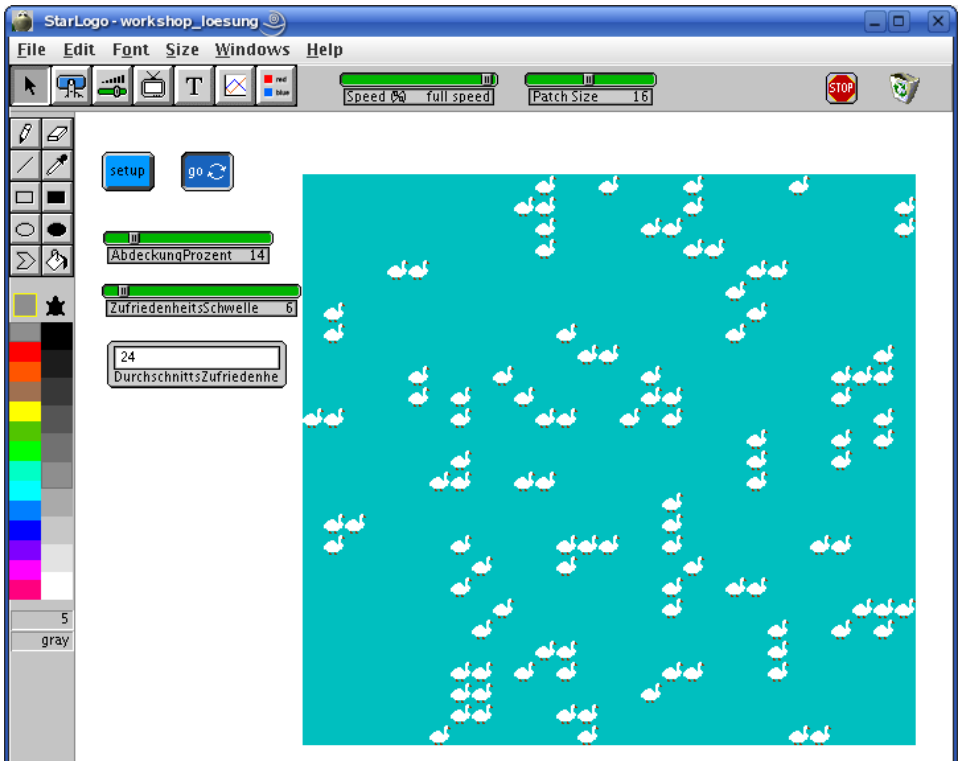


Abbildung 3: Zufriedene Schwäne: stabile Konstellation der turtles im StarLogo-Workshop von Einstieg Informatik

so programmiert werden, dass sie aus einer zufälligen Bewegung in eine stabile, als „zufriedenstellend“ bezeichnete Gruppenkonstellation übergehen (vgl. Abb. 3).

Für diesen Workshop wurde ein Ansatz gewählt, der als „Lücken füllen“ bezeichnet werden kann. Der überwiegende Teil des Quellcodes ist vorhanden; alle benötigten Sprachkonstrukte sind (zum Abschauen) darin enthalten. An einigen markierten Stellen sind vereinfachte Anweisungen enthalten, die zum Erreichen der gewünschten Funktionalität angepasst und erweitert werden müssen. Diese Methode wurde auf die Verwendung mit Novizen und in einem geringen Zeitrahmen zugeschnitten. Die Teilnehmenden kamen gut damit zurecht.

3.4 EI:SPIEL

Das EI:SPIEL (www.ei-spiel.de) ist ein Internet-Teamspiel für Kinder und Jugendliche bis zur 9. Klasse. In verschiedenen Räumen kann die Welt der Informatik erforscht werden. Die Teilnehmerinnen und Teilnehmer müssen sich in Gruppen von mindestens drei zusam-

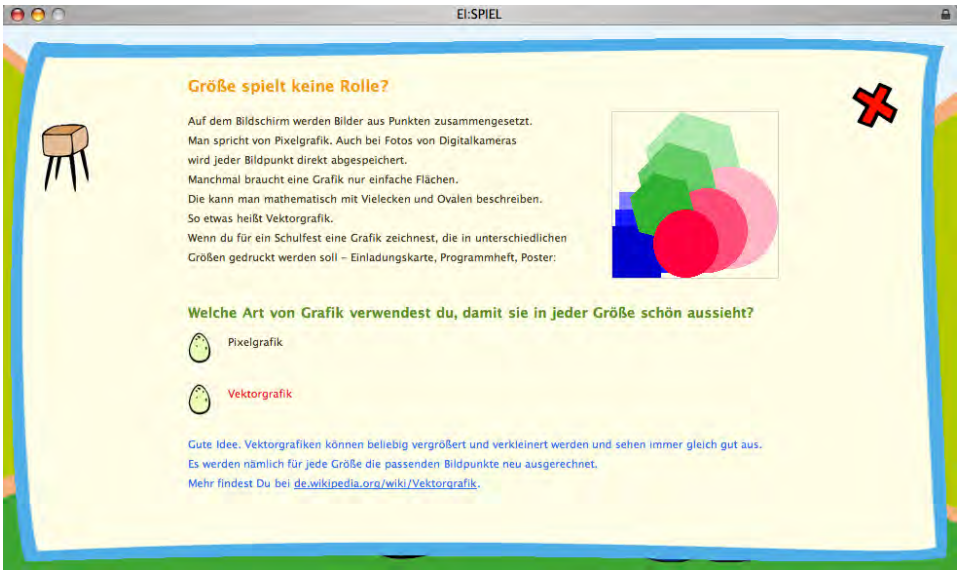


Abbildung 4: Dual-Choice-Frage im EI:SPIEL

menfinden, um mit der Bearbeitung von Aufgaben virtuelles Geld erwerben zu können. So wird Einzelgängertum vermieden, und die Inhalte des Spiels werden in reale soziale Kontexte transportiert. Das erworbene Guthaben kann in regelmäßig stattfindenden Auktionen gegen Preise eingetauscht werden.

Als Aufgaben werden vor allem Dual-Choice-Fragen gestellt, die informatisches Grund- und Hintergrundwissen vermitteln, aber auch logisches Denken und Problemlösen fördern. Abbildung 4 zeigt eine Frage, die den Unterschied zwischen Vektor- und Pixelgrafik anspricht. Der Fragetext ist blau eingerahmt, außerhalb sind andeutungsweise noch Seitenwände, Decke und Boden eines Raumes zu erkennen, und links oben im Kasten wird der Gegenstand angezeigt, mit dem die Frage im Raum verbunden ist. Die Frage wurde gerade richtig beantwortet, und ein erläuternder Kommentar wird angezeigt. Auch bei einer falschen Antwort gibt es weiterführende Informationen (aus Fehlern lernen!), und die Teilnehmer haben die Gelegenheit, die Frage noch einmal zu beantworten.

Neben den Wissensfragen gibt es auch Aufgaben, bei denen die Teilnehmerinnen und Teilnehmer aktiv werden können: Decken, Wände und Böden der Räume können von den Teilnehmenden selbst gestaltet werden. Weitere Aufgabentypen sind noch in Arbeit, das Spiel wird ehrenamtlich weitergeführt und ausgebaut.

3.5 EI:SPIEL blitz!

Zum Abschluss des Informatikjahres führte Einstieg Informatik unter dem Titel „EI:SPIEL blitz!“ einen Online-Wettbewerb zur Informatik durch. Dieser war gleichzeitig die ers-

Kategorie	**Altersgruppen**			
	insgesamt	Kl. 5-8	Kl. 9/10	Kl. 11+
Logisches Denken	9	4	4	6
Informationsverständnis	3	2	2	1
Kombinatorik	6	2	3	2
Algorithmisches Problemlösen	7	4	4	3
Technische Kenntnisse	4	3	2	2
Allgemeines Wissen	1	0	0	1

Tabelle 1: Inhaltliche Zuordnung der Aufgaben im EI:SPIEL blitz!

te deutsche Beteiligung am internationalen „Beaver Contest“, eine in Litauen im Jahr 2004 ins Leben gerufene internationale Initiative, das Konzept des Wettbewerbs „Mathe-Känguru“ auf die Informatik zu übertragen [Da06]. Im Informatikjahr standen zum ersten Mal die Ressourcen zur Durchführung eines deutschen „Informatik-Biber“ zur Verfügung.

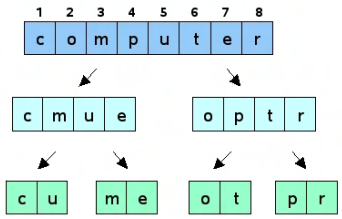
Auf einem internationalen Workshop im Juni 2006 in Litauen (mit Vertretern aus Deutschland, Estland, Lettland, Litauen, den Niederlanden, Österreich und Polen) wurden Aufgaben für den Beaver Contest 2006 entwickelt. Aus diesem Fundus stellte Einstieg Informatik gemeinsam mit dem Arbeitsbereich „Didaktik der Informatik“ der Universität Münster die Aufgaben für EI:SPIEL blitz! zusammen. Während der „Beaver Contest“ bislang mit Hilfe interaktiver PDF-Dokumente ausgetragen wurde, sprachen sich niederländische und deutsche Verantwortliche für eine Web-basierte Durchführung aus und gaben die Entwicklung eines entsprechenden Systems gemeinsam in Auftrag.

Der Wettbewerb wurde in drei Altersklassen durchgeführt: 5. bis 8. Klasse, 9./10. Klasse und ab 11. Klasse. In jeder Altersstufe wurden 15 Multiple-Choice-Fragen gestellt, die in 40 Minuten zu beantworten waren. Die 15 Aufgaben waren in drei Schwierigkeitsstufen eingeteilt, mit 5 Aufgaben in jeder Stufe. Für eine richtige Antwort gab es, je nach Schwierigkeitsstufe, 1, 2 oder 3 Punkte, bei einer falschen Antwort wurden 1/3, 2/3 bzw. 1 Punkt abgezogen. Durch ein Anfangsguthaben von 10 Punkten wurden negative Punktzahlen als Endergebnis ausgeschlossen.

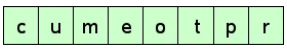
Insgesamt gab es 30 verschiedene Aufgaben, einige Aufgaben wurden in mehreren Altersstufen eingesetzt, dann aber in der Regel mit unterschiedlichem Schwierigkeitsgrad. Die Aufgaben sind sechs Kategorien zugeordnet, die nach dem durch die Aufgaben der Kategorie geprüften Kompetenzbereich benannt sind. Tabelle 1 zeigt die Anzahl der Aufgaben in den jeweiligen Kategorien. In den einzelnen Altersgruppen waren die Aufgaben relativ gleichmäßig über die Kategorien verteilt, mit Ausnahme des „allgemeinen Wissens“; logisches Denken (insbesondere in der höchsten Altersstufe) und algorithmisches Problemlösen waren etwas mehr gefragt als Informationsverständnis (damit ist die Fähigkeit zur korrekten Interpretation gegebener Informationen gemeint), Kombinatorik und technische Kenntnisse. Ein Erfolg im EI:SPIEL blitz! setzte wenig Vorkenntnisse voraus, nämlich nur in den Kategorien „technische Kenntnisse“ und „allgemeines Wissen“.

Biber-Code

Im Biber-Code wird jedes Wort in zwei Teile unterteilt. Im ersten Teil stehen die Buchstaben von den ungeraden Positionen des Ursprungswortes, im zweiten Teil die Buchstaben von den geraden Positionen. Dies wird solange wiederholt, bis die sich ergebenden Teilstrings aus höchstens zwei Buchstaben bestehen. Diese werden dann in der vorliegenden Reihenfolge wieder zu einem Wort zusammengefügt.
So verschlüsselt ist der Biber-Code von computer gerade cumeotpr und der Biber-Code von biber ist brbie.



ergibt



Wie lautet der Biber-Code von kaenguru?
A) kgeraunu B) kuhaugen C) kgeranuu D) kauerung

Abbildung 5: Beispielaufgabe aus dem EI:SPIEL blitz! (Richtige Lösung: A)

Abbildung 5 zeigt beispielhaft eine Aufgabe der Kategorie „logisches Denken“, die in der Altersstufe 5.-8. Klasse als Aufgabe hoher Schwierigkeit und in der Altersstufe 9./10. Klasse als Aufgabe mittlerer Schwierigkeit verwendet wurde.

Die Resonanz auf den Wettbewerb war erfreulich hoch, was u.a. auf das Engagement einiger GI-Lehrerfachgruppen zurückzuführen ist, die sich an der Ankündigung des Wettbewerbs stark beteiligten; über die Hälfte der Teilnehmenden stammte aus Mecklenburg-Vorpommern und Sachsen. Insgesamt nahmen 2.126 Schülerinnen und Schüler aus 50 Schulen am EI:SPIEL blitz! teil. Der Mädchenanteil lag insgesamt bei 32,83%, in der Altersstufe 5.-8. Klasse sogar bei 41%. Weitere Zahlen sind der Tabelle 2 auf der nächsten Seite zu entnehmen.

Die hohe Beteiligung auch in der jüngsten Altersstufe zeigt, dass mit dem EI:SPIEL blitz! ein Weg gefunden wurde, Informatik und die Teilnahme an einem Informatik-Wettbewerb breit zu platzieren. Auch Schülerinnen und Schüler aus Realschulen und berufsbildenden Schulen wurden mit diesem Format erreicht. Erwähnenswert ist auch eine Äußerung des Siegers der höchsten Altersgruppe, der ansonsten Fremdsprachen bevorzugt: „Es hat einfach Spaß gemacht, die Aufgaben zu lösen.“

Altersstufe	Teilnehmende	Mädchen		Jungen	
		Anzahl	*Anteil*	*Anzahl*	*Anteil*
insgesamt	2.126	698	32,83%	1.428	67,17%
5.-8. Klasse	959	394	41,08%	565	58,92%
9.+10. Klasse	479	133	27,77%	346	72,23%
11. Klasse+	688	171	24,85%	517	75,15%

Tabelle 2: Beteiligung am EI:SPIEL blitz!

108.175	Besuche des Webportals einstieg-informatik.de (1.4.2006-25.1.2007)
20.000	gezielt oder auf Anfrage ausgegebene EI:CODEs (ungefährer Wert)
2.126	Teilnehmende am Online-Wettbewerb EI:SPIEL blitz!
558	Teilnehmende am Online-Spiel EI:SPIEL
27	Veranstaltungen mit EI-Präsentation durch Projektmitarbeiter

Tabelle 3: Quantitative Bilanz von Einstieg Informatik

4 Resümee und Ausblick

Tabelle 3 fasst die wichtigsten Kennzahlen des Projekts zusammen. Doch auch einige qualitative Aussagen sollen abschließend versucht werden, auch wenn diese rein auf Erfahrungen und Eindrücken basieren.

Die Resonanz auf die Aktivitäten von Einstieg Informatik konnte eindrucksvoll belegen, dass bei Kindern und Jugendlichen ein großes Interesse an Themen, Ideen und Problemen der Informatik vorhanden ist. Dieses Interesse liegt nicht in der Attraktivität moderner Computersysteme begründet; Angebote von Einstieg Informatik ohne Einsatz von Computertechnik wurden mindestens genau so gut angenommen wie solche, bei denen Rechner benutzt wurden. Ein besonderer Beleg dafür ist der Erfolg des EI:CODE, der stets zu den beliebtesten Angeboten des Projekts gehörte. Selbst bei der Games Convention konnten Angebote wie der EI:CODE gegen die attraktive Bilderwelt der Computerspiele konkurrieren; der Stand des Projektes hatte sogar einen besonders großen Zuspruch.

Auch die gute Resonanz auf das EI:SPIEL blitz! widerspricht dieser These nicht; zwar mussten die Aufgaben dieses Online-Wettbewerbs am Computer gelöst werden, aber nicht *mit* dem Computer. Eine Durchführung mit Papier und Stift wäre auch möglich gewesen, hätte aber einen höheren Aufwand bei der Auswertung verursacht. Wie auch beim EI:CODE war zu beobachten, dass Aufgabenstellungen, zu deren Lösung informatische Kompetenzen erforderlich sind, auch für (noch) nicht Informatik-affine Jugendliche eine attraktive Herausforderung darstellen.

Einfache Informatik-Konzepte wie Grundelemente der Algorithmik (vgl. Abschnitt 3.2) konnten mit Kindern im Grundschulalter bearbeitet werden. Überraschend ist dies letztlich nicht; Einstieg Informatik war von Anfang an ermutigt worden durch Schwill, der nachweisen konnte, „dass eine Reihe wichtiger fundamentaler Ideen der Informatik bereits von

Kindern im Grundschulalter erfasst werden kann, vorausgesetzt, die Gegenstände werden altersgemäß aufbereitet [. . .]“ [Sc01]. Auch Gallenbacher ist der Überzeugung, dass die in seinem Buch „Abenteuer Informatik“ [Ga06] enthaltenen Informatik-Spiele zumindest bei Anleitung durch Erwachsene auch von Kindern im Grundschulalter nachvollzogen werden können. Nicht zuletzt sieht das K12-Curriculum „Computer Science“ der ACM einen ersten Einstieg in algorithmisches Denken für die Schuljahre 3 bis 5 vor [Tu03]. Beobachtet wurde durch EI auch, dass in diesem frühen Alter Mädchen keinerlei Hemmungen gegenüber Themen der Informatik oder dem Begriff „Informatik“ per se zeigen.

Danksagung

Wir danken Anke Werner und den ehrenamtlichen EI:SCOUTs, insbesondere Thomas Leineweber, für die Mitarbeit bei Einstieg Informatik, sowie Marco Thomas, Maria Pazzanese, Ries Kock und Eljakim Schrijvers für die Zusammenarbeit beim EI:SPIEL blitz!

Literaturverzeichnis

[Br00] Norbert Breier et al. Empfehlungen für ein Gesamtkonzept zur informatischen Bildung an allgemein bildenden Schulen. LOG IN, 20(2):Beilage, 2000.

[BFW98] Timothy Bell, Mike Fellows und Ian H. Witten. Computer Science Unplugged. Self-published, 1998.

[Da06] Valentina Dagiene. Information Technology Contests – Introduction to Computer Science in an Attractive Way.Informatics in Education, 5(1):37–46, 2006.

[Ga06] Jens Gallenbacher. Abenteuer Informatik. Elsevier Spektrum Akademischer Verlag, Heidelberg, 2006.

[Hr06] Juraj Hromkovič. Sieben Wunder der Informatik. B. G. Teubner Verlag, 2006.

[Ko05] Jochen Koubek. Informatische Allgemeinbildung. In Steffen Friedrich, Hrsg., Unterrichtskonzepte für informatische Bildung, Lecture Notes in Informatics, Seiten 57–66. Gesellschaft für Informatik, 2005.

[PKH06] Wolfgang Pohl, Katharina Kranzdorf und Hans-Werner Hein. First Steps into Computer Science – the German Project Einstieg Informatik. In Valentina Dagiene und Roland Mittermeir, Hrsg., Information Technologies at School: Selected Papers of the 2nd International Conference “Informatics in Secondary Schools: Evolution and Perspectives”, Seiten 62–70, Vilnius, 2006. Institute of Mathematics and Informatics.

[Sc01] Andreas Schwill. Ab wann kann man mit Kindern Informatik machen? Eine Studie über informatische Fähigkeiten von Kindern. In Reinhard Keil-Slawik und Johannes Magenheim, Hrsg., Informatikunterricht und Medienbildung, Lecture Notes in Informatics, Seiten 13–30. Gesellschaft für Informatik, 2001.

[Sc06] Uwe Schöning. Ideen der Informatik. Oldenbourg Verlag, 2. Auflage, 2006.

[Tu03] Allen Tucker et al. A Model Curriculum for K-12 Computer Science. Bericht, ACM, New York, 2003.

Gibt es einen mobilkommunikationszentrierten Ansatz für die Schulinformatik?

Gerrit Kalkbrenner

Institut für Informatik
Universität Potsdam
August Bebel Straße 89
14482 Potsdam
gerrit.kalkbrenner@udo.edu

Abstract: Die technische Entwicklung und die inzwischen gegebene Allgegenwart von Geräten der Mobilkommunikation machen auch vor Schulen nicht Halt. Insbesondere verändert sie auch das im Informatikunterricht diskutierte Themenspektrum. Dieser Artikel diskutiert, an welchen Stellen die Mobilkommunikation als ein innovativer Ansatz für den Informatikunterricht dienen kann.

1 Einleitung

Die aktuelle Diskussion um die Vermittlung von ECDL-Inhalten (European Computer Driving License) im Informatikunterricht [Mi05] demonstriert, dass der Informatikunterricht einen anderen Anspruch erhebt, als die Vermittlung von elementaren Grundfertigkeiten in der Bedienung von Informatiksystemen. Das zugrunde liegende Problem, die Reduktion der Fachwissenschaft Informatik auf Bedienerfertigkeiten wird oftmals als die „Krise des Informatikunterrichtes“ bezeichnet [Pe89]. Die Entwicklung Ende der 1980er und der 1990er Jahren waren durch die Erfindung des WEB und der Verbreitung von so genannter Standardsoftware in weiten Bereichen der Anwendung von Informatiksystemen geprägt. In diesem Zusammenhang wurden Stimmen lauter, welche vom Informatikunterricht die Vermittlung von Kenntnissen in der Bedienung der beschriebenen Systeme forderten. Verfolgt wurde fortan eine Doppelstrategie: zum einen wurden Kurse unter der Bezeichnung „Informations- und kommunikationstechnische Grundbildung“ mit dem Schwerpunkt Anwenderschulung in der Sekundarstufe I angeboten, auf der anderen Seite verlagerte sich der einstige Anspruch der Informatik in die Kurse der Oberstufe.

Erst in jüngerer Zeit wurde deutlich, dass aber gerade der Anspruch der Informatik wesentlich für eine längerfristige und allgemeinbildende Informatikausbildung essentiell ist. Es zeigte sich ebenso, dass auch die Medienbildung nicht ohne Kenntnisse der Informatik auskommt.

Basierend auf dieser Erkenntnis wurden verschiedene Konzepte für den allgemeinbildenden Informatikunterricht insbesondere für die Unterstufe erarbeitet. Insbesondere ist hier das bayerische Konzept von Hubwieser [Hu05] hervorzuheben. Dieses Konzept dreht in gewisser Weise die „Informations- und kommunikationstechnische Grundbildung“ um und stellt zentrale Konzepte der Informatik, insbesondere die der objektorientierten Programmierung in den Mittelpunkt. Die gewünschte Vermittlung von Bedienungskompetenzen ist in diesem Rahmen fast schon nebenbei exemplarisch sehr gut möglich (vergleiche insbesondere die empirischen Ergebnisse aus [Hu05]).

Neben der objektorientierten Programmierung werden auch andere Konzepte als zentral für die Informatik angesehen. Norbert Breier [Br05] stellt z.B. den Begriff der Information in den Mittelpunkt seines informatikdidaktischen Ansatzes. Basierend auf der Erkenntnis von Norbert Wiener: „Information ist eine dritte Grundgröße der Natur, wohl zu unterscheiden von den beiden anderen Größen Materie und Energie.“ [AU01] leitet er eine Didaktik her, deren Fundament das Wesen der Information und deren Verfahren zur Verarbeitung sind.

Weiterhin hat auch die Modellierung [Th02], [BS04] in der Schulinformatik einen wichtigen Stellenwert erhalten. Die Zunahme der Bedeutung von z.B. UML in der Fachwissenschaft macht auch vor der Schule nicht halt. Standen noch in den 1980er Jahren der Pascal-Kurs und die Kunst der Codierung im Mittelpunkt, so wird heute Programmcode nur noch anhand von vorgegebenen Beispielen studiert und z.B. in Lückentexten ergänzt. Einen größeren Stellenwert bekommt auch die Modellierung eines Problembereiches, d.h. die Analyse eines Anwendungsbereiches und anschließende Beschreibung mit ikonischen Modellen [Br04].

2 Mobilkommunikation

Seit der Etablierung dieses „neuen“ Informatikunterrichtes ist die technische Entwicklung jedoch weiter fortgeschritten. Der PC als Informatiksystem hat an Bedeutung verloren [Ma03a], [Ma03b], [Sc03], [Sa06]. Vielmehr gibt es inzwischen diverse elektronische Kleingeräte wie PDA (Personal Digital Assistant), Mobiltelefone, MP3-Player, Smartphone und mobile Spielekonsolen, die in der Alltagswelt zunehmend an Bedeutung gewinnen und den PC aus dem Fokus verdrängen (vergleiche Ubiquitous Computing, [We93a], [We93b]). Der Informatikunterricht an Schulen darf sich keinesfalls gegenüber diesen Neuerungen verschließen, sondern sollte vielmehr die damit verbundenen Themen aufgreifen und hoch motivierte Schüler mit ihrem Interesse an diesen Systemen „dort abholen, wo sie stehen“ [Mu93]. Der hiermit einhergehende Sichtwechsel (Ubiquitous Computing) würde es rechtfertigen, vom mobilkommunikationszentrierten Ansatz für die Schulinformatik zu sprechen. Auf der anderen Seite behalten aber auch die Konzepte der objektorientierten Programmierung und der informationszentrierten Sicht ihre unangetastet zentrale Bedeutung.

Es stellt sich nun die Frage, wie ein zugehöriger Informatikunterricht zu gestalten ist. In Parallelität zu der vorangegangenen Diskussion ist die Wiederholung von Fehlern und eine damit einhergehende zweite Krise der dieses mal mobilkommunikationszentrierten

Informatik unbedingt zu vermeiden. Bei der Gestaltung einer Unterrichtsreihe mit dem Schwerpunkt Mobilkommunikation wird es daher nicht darum gehen, Bedienerfertigkeiten von Handys und anderen Geräten zu vermitteln. Unterrichtsbeispiele wie der Handyführerschein [Ha07] sind sicherlich ein wichtiger Beitrag für die Allgemeinbildung in der Informationsgesellschaft, gehören jedoch eher in den Gemeinschaftskunde- oder Deutschunterricht (z.B. Interpretation von Vertragstexten der Anbieter von Klingelton- und Spiele-Abonnements). Weitere Themenbereiche wie etwa Funktechnik sind interdisziplinär und passen auch in den Physikunterricht.

Für den Informatikunterricht vermag das Thema Mobilkommunikation vielmehr einen Rahmen darstellen, anhand dessen auch weitergehende zentrale Informatikkompetenzen vermittelbar sind [Ka06]. So führt z.B. das Verfahren zur Zuordnung von Frequenzen auf Funkzellen (D- und E-Netz Mobiltelefonie) anschaulich zu Themen der Komplexitätstheorie. Weitere informatische Themen sind: objektorientierte Programmierung, Schnittstellen, Kommunikationsprotokolle, Datensicherheit, Datenschutz, Frequenz-, Raum-, Code- und Zeitmultiplexing, (GSM, Infrarot, Bluetooth, WLAN, DECT), Embedded Systems, multimediale Anwendungen, WAP (Wireless Application Protocol)/WML (Wireless Markup Language) sowie die gesellschaftliche Auswirkungen neuer Technologien und Kommunikationsmedien.

Wichtig an der Thematik Mobilkommunikation ist insbesondere die Nähe zum Alltagsleben der Schüler und Lehrer. Auf diese Weise lassen sich auch sehr abstrakte informatische Themen motivieren.

3 Thesen zum mobilkommunikationszentrierten Ansatz

Nachdem wir die Bedeutung der Mobilkommunikation im Informatikunterricht diskutiert haben, wollen wir einige Zielrichtungen für den Informatikunterricht in drei Thesen zusammenfassen:

These 1) Das Thema Mobilkommunikation selbst ist für die Schule aufzubereiten.

Das Handy oder andere mobile Kleingeräte werden bereits als Anschauungsmittel vielfach im Informatikunterricht eingesetzt. Für diesen Themenbereich sind weitere Beispiele zu erarbeiten, die unmittelbar im Unterricht eingesetzt werden können. Das Interesse der Schüler wird aufgegriffen und es wird intrinsisch motiviert der Einblick in ein aktuelles Informatik-Thema gegeben. Es sind Stellen zu identifizieren, an denen die Thematik in die aktuellen Lehrpläne (Zentralabitur) einzuordnen ist.

These 2) Das Thema Mobilkommunikation als Einstieg für Informatik-Themen nutzen.

Mobilkommunikation vermag auch als Einstieg zu bisherigen Informatik-Themen dienen. Die Mobilkommunikation mit seinen vielfältigen Realisierungen mit Informatik-Systemen vermag eine intrinsisch motivierte Überleitung zu nahezu allen Themen der Informatik bieten.

These 3) Entwicklung von Plattformen für Projekte im Bereich Mobilkommunikation.

Insbesondere Projekte und Modellierung werden oft als Alleinstellungsmerkmale des Informatikunterrichtes genannt. Um Informatik-Projekte im Bereich der Mobilkommunikation durchzuführen, bedarf es didaktisch geeigneter Implementierungsplattformen für Projekte. Mit diesen wird es möglich, sich auf die unmittelbare Umsetzung der Projektziele zu konzentrieren und dabei allgemeinbildende Informatik-Themen zu erlernen.

Bevor wir zu konkreten Unterrichtsvorschlägen kommen, wollen wir sehen, in wieweit das Thema Mobilkommunikation auch allgemeinbildende Anteile enthält.

4 Fundamentale Ideen der Informatik und Mobilkommunikation

Zur Überprüfung von Informatik-Themen bezüglich ihrer Eignung für den (allgemeinbildenden) Informatik-Unterricht an Schulen gibt es das Instrument der „Fundamentalen Ideen der Informatik“ nach Schwill [SS04]. Um für den Unterricht geeignet zu sein, müssen die Themen den Kriterien für fundamentale Idee entsprechen. Diese sind:

1. Horizontalkriterium (betrifft viele Anwendungen)
2. Vertikalkriterium (betrifft viele Niveaus)
3. Zielkriterium (Annäherung an ein ideelles Ziel)
4. Zeitkriterium (längerfristige Gültigkeit)
5. Sinnkriterium (entspringt dem Alltagsdenken)

Untersuchen wir einige der vorangehend dargelegten Schwerpunkte zur Mobilkommunikation mit dem Konzept der fundamentalen Ideen. Insbesondere verbleiben u.a. folgende Punkte, die sich als würdig für die Einbeziehung in den Informatikunterricht herausstellen:

Thema: Schnittstellen zwischen Komponenten mobiler Informatiksysteme

1. Horizontalkriterium: aktuelle Informatik-Systeme verwenden Schnittstellen, über die sie Information austauschen.
2. Vertikalkriterium: Schnittstellen können auf der einen Seite sehr simpel betrachtet werden z.B. eine Haustürklingel. In der Mobilkommunikation spielen komplexe Übertragungsprotokolle aber eine entscheidende Rolle und erlauben eine komplexe Betrachtung.
3. Zielkriterium: auf der Basis heterogener verteilter Systeme mit offenen Schnittstellen ist die Entwicklung der Mobilkommunikation möglich.

4. Zeitkriterium: Schnittstellen zwischen mobilen Systemen wird es auch in zwanzig Jahren noch geben, lediglich Übertragungsraten, Protokolle und Verschlüsselungsmechanismen werden sich voraussichtlich unterscheiden.
5. Sinnkriterium: die Bedeutung von Schnittstellen ist durch die Themenwahl unmittelbar einzusehen.

Thema: Protokolle

1. Horizontalkriterium: Schnittstellen (siehe oben) erfordern Übertragungsprotokolle.
2. Vertikalkriterium: Ein einfaches Protokoll z.B. ist der Austausch einer Einladungsliste. Alle Personen, die nicht auf dieser Liste stehen, werden z.B. nicht zugelassen. Anspruchsvollere Protokolle sind z.B. die Standards für WLAN zur Kollisionsvermeidung im gemeinsamen Funk-Medium.
3. Zielkriterium: Protokolle erlauben robuste, schnelle und komfortable Kommunikation.
4. Zeitkriterium: Übertragungsprotokolle wird es auch in Zukunft weiterhin geben (siehe oben).
5. Sinnkriterium: Die Wirkungsweise von Protokollen sind insbesondere bei Zugangskontrollen unmittelbar einsehbar.

Die vorangegangene Erörterung zeigt, dass ausgewählte Themen der Mobilkommunikation den Kriterien der fundamentalen Ideen entsprechen. Die Themen sind zwar nicht ausschließlich spezifisch für die Mobilkommunikation, spielen dort jedoch eine so wichtige Rolle, um eine genauere Betrachtung in diesem Umfeld zu rechtfertigen. Darüber hinaus gibt es auch weitere Aspekte der Mobilkommunikation wie z.B. Software-Entwicklung in Projekten, deren Bedeutung als Masteridee unumstritten ist.

5 Curriculum

Nachdem die Bedeutung der Mobilkommunikation anhand des Instrumentes der fundamentalen Ideen deutlich geworden ist, möchten wir uns mit einem konkreten Curriculum zum Thema befassen.

Wir gehen davon aus, dass ein Unterrichtshalbjahr in einem 12/13er Leistungskurs stattfindet und somit bereits Grundkenntnisse in Informatik vorhanden sind. Als Einstieg ist es denkbar ein relativ simples Spiel wie z.B. „Arkanoid“ für mehrere Handys (multiplayerfähig) zu entwickeln. Dieses erfordert als Ausgang eine Fassung, deren Quellcode zur Verfügung steht und erweitert wird. Während des Projektes werden folgende Themen zur Mobilkommunikation bearbeitet:

1. Vorstellung des Projektes; Bildung von Projektgruppen.

2. Vorstellung der Plattform, für die das Spiel entwickelt werden soll (z.B. ein Handy mit Java Sandbox und Python sowie IrDA-Schnittstelle oder für WAP).
3. Andere und zukünftige Schnittstellen von Mobilfunksystemen (z.B. IrDA, Bluetooth, WLAN, GPRS, UMTS).
4. Technische Sicht von den Schnittstellen (z.B. Frequenzen, Modulation und Übertragungsarten).
5. Übertragungsprotokolle verschiedener Schnittstellen (insbesondere Fehlerkorrektur).
6. Gesellschaftliche Relevanz von Mobilkommunikation und deren Grenzen.
7. Mögliche Gesundheitsrisiken/Auswirkungen auf den menschlichen Körper von „Handystrahlung".
8. Sicherheit von Mobilkommunikationsverfahren, Kryptographie.

Der Großteil der Zeit ist für die Planung, Durchführung und Implementierung des Projektes zu berücksichtigen. Werden Konzepte der objektorientierten Programmierung berücksichtigt, ist der Unterrichtsvorschlag auch weitgehend lehrplankonform.

6 Zusammenfassung

Es wurde gezeigt, dass mit dem Thema Mobilkommunikation ein hochaktuelles Thema gegeben ist, welches auf der einen Seite viele Themen der Informatik aufgreift, auf der anderen Seite aber auch der Motivationslage der Schüler sehr entgegenkommt.

Als Abgrenzung wurde erörtert, dass es bei der Gestaltung einer Unterrichtsreihe mit dem Schwerpunkt Mobilkommunikation nicht darum gehen kann, Bedienerfertigkeiten von Handys und anderen Geräten zu vermitteln. Für den Informatikunterricht vermag das Thema Mobilkommunikation vielmehr einen Rahmen darstellen, anhand dessen auch weitergehende zentrale Informatikkompetenzen vermittelbar sind. Die Bedeutung des Konzeptes des Ubiquitous Computing, welchem die Mobilkommunikation zuzurechnen ist, würden es erlauben, von dem mobilkommunikationszentrierten Ansatz für die Schulinformatik zu sprechen.

Das Instrument der fundamentalen Ideen von Andreas Schwill wurde für den Nachweis angewendet, dass es sich bei dem Thema Mobilkommunikation um Inhalte handelt, die schulgeeignet sind. Ein Curriculum wurde skizziert, welches einen möglichen Unterrichtsverlauf beispielhaft darstellt.

Die Realitätsnähe des Themas Mobilkommunikation zum Alltagsleben der Schüler und Lehrer ist ein wichtiger Vorteil, um auch weitergehende informatische Themen anhand der Mobilkommunikation zu vermitteln.

Literaturverzeichnis

[AU01] Aßmann, U.; Ungerer, TH.: „Informatik in der Schule“ In: Informatik-Spektrum, Band 24, Nummer 6, Dezember 2001

[Br05] Norbert Breier: „Informatik im Fächerkanon allgemeinbildender Schulen – Überlegungen zu einem informationsorientierten didaktischen Ansatz“, in Friedrich, Steffen (Hrsg.) „Unterrichtskonzepte für informatische Bildung“, GI-Edition, Bonn 2005, ISBN 3-88579-389-X

[Br04] Brinda, T.: „Didaktisches System für objektorientierte Modellierung im Informatikunterricht der Sek. II“, Fachbereich Elektrotechnik und Informatik, Universität Siegen, 2004

[BS04] Manfred Broy, Ralf Steinbrüggen: „Modellbildung in der Informatik“, Springer Verlag Berlin, 2004

[Ha07] „Handyführerschein für Kids, Ein Angebot der Verbraucherzentrale NRW für die 6. Klasse“, http://www.learnline.de/angebote/handy/modul/fuehrerschein.htm, geprüft 12.1.207

[Hu05] Peter Hubwieser: „Von der Funktion zum Objekt – Informatik in der Sekundarstufe 1“, in Friedrich, Steffen (Hrsg.) „Unterrichtskonzepte für informatische Bildung“, GI-Edition, Bonn 2005, ISBN 3-88579-389-X

[Ka06] Kalkbrenner, G; Schultebraucks, B.; Sawatzki, M.: „Mobilkommunikation im Informatikunterricht, Thesen und empirische Ergebnisse“, in Schwill, A (Hrsg.): „Didaktik der Informatik“, 3. Workshop der GI-Fachgruppe Didaktik der Informatik, ISBN 978-3-88579-193-5

[Ma03a] Friedmann Mattern (ETH Zürich): „Total vernetzt“, Springer Verlag, 2003

[Ma03b] Friedmann Mattern: „Allgenwärtiges Rechnen“, LOG IN Heft Nr. 125, 2003

[Mi05] Micheuz, Peter: „Auf dem Weg zu Standards“, LOG IN, Nr. 135, 2005

[Mu93] Muhlak, Renate: „Die Leute dort abholen, wo sie in ihrem Wissen und Können stehen“, In: Qualifizierung in Portionen. neue Konzepte beruflicher Weiterbildung im europäischen Vergleich. - (1993) , S. 45-50, ISSN 0935-3526

[Pe89] R. Peschke: „Die Krise des Informatikunterrichts in den neunziger Jahren“. In Stetter, F. und Brauer, (Hrsg.) Informatik und Schule 1989: „Zukunftsperspektiven der Informatik für Schule und Ausbildung.“ Nummer 220, Informatik-Fachberichte, Springer Verlag 1989

[Sa06] Martin Sauter: „Grundkurs Mobile Kommunikationssysteme. Von UMTS, GSM und GPRS zu Wireless LAN und Bluetooth Piconetzen“, Vieweg Verlag 2006

[Sc03] Jochen Schiller: „Mobilkommunikation“, Pearson Education, München, 2003, ISBN 3-8273-7060-4

[SS04] S. Schubert, A. Schwill: „Didaktik der Informatik“ Spektrum Akademischer Verlag 2004

[Th02] Marco Thomas: „Informatische Modellbildung, Modellierung von Modellen als ein zentrales Element der Informatik für den allgemeinbildenden Schulunterricht“, Dissertation, Universität Potsdam 2002

[We93a] Mark Weiser: „Some Computer Science Problems in Ubiquitous Computing”, Communications of the ACM, July 1993. (reprinted as „Ubiquitous Computing”. Nikkei Electronics; December 6, 1993; pp. 137-143.)

[We93b] Mark Weiser: „Hot Topics: Ubiquitous Computing”, IEEE Computer, October 1993

Informatikunterricht: anschaulich, nützlich – und fundiert

Martin Lehmann, Diana Jurjevic, Nando Stöcklin

Zentrum für Bildungsinformatik
Pädagogische Hochschule PHBern
Muesmattstraße 29
CH-3012 Bern
martin.lehmann@phbern.ch
diana.jurjevic@phbern.ch
nando.stoecklin@phbern.ch

Abstract: Nach einer Phase der einseitig auf Anwenderkompetenzen ausgerichteten informationstechnischen Grundbildung (ITG) ist der Informatikunterricht heute oft zu theorielastig. Der Transfer des erworbenen Wissens auf Anwendungen findet kaum statt. Für die anderen Schulfächer ist deshalb wenig Nutzen des Informatikunterrichtes sichtbar, was die Etablierung des Faches Informatik behindert. Die ICT-Ausbildung der Lehrkräfte auf der anderen Seite ist immer noch zu stark auf ICT-Fertigkeiten ausgerichtet. Im Artikel wird gezeigt, wie eine stärkere Gewichtung des Orientierungswissen in der ICT-Lehrerausbildung und ein anschaulicher, von Anwendungen ausgehender Informatikunterricht zur Aufwertung des Stellenwertes des Faches beitragen kann.

1 Informatische Bildung umfasst Anwenderkompetenz und Orientierungswissen

An den allgemein bildenden Schulen soll es ein Fach Informatik geben, das gleichberechtigt zu anderen Fächern ist, fordert die Gesellschaft für Informatik e.V. (GI) im Memorandum „Digitale Spaltung verhindern – Schulinformatik stärken!“ [Ge04]. Als Begründung führt sie unter anderem an:

> „Denn genau dieses Schulfach Informatik gibt jungen Menschen die notwendige Orientierung in einer Gesellschaft, die zunehmend von Informations- und Kommunikationssystemen geprägt ist und in der auf dem Arbeitsmarkt verstärkt fundierte informatische Kompetenzen erwartet werden. […] Ziel dieses Faches muss es sein, den Schülerinnen und Schülern auf altersgemäße Weise Erkenntnisse über die grundlegende Funktionsweise von Informatiksystemen zu vermitteln, die ihnen eine effiziente Nutzung, einen verantwortungsvollen Umgang sowie eine Abschätzung der prinzipiellen Chancen und Risiken moderner Informatiksysteme ermöglichen. Diese Fähigkeiten werden in unserer Informationsgesellschaft

> eben nicht mehr nur von ausgebildeten IT-Spezialisten verlangt, sondern zunehmend von jeder und jedem Einzelnen."

Wenn die Informatik in der Schule stärker verankert wäre, würde dies die allgemeine IT-Kompetenz und somit letztlich die Effizienz zahlreicher Arbeitsprozesse erheblich erhöhen, führt die Gesellschaft für Informatik weiter aus.

Um die allgemeine ICT-Kompetenz zu erhöhen, reicht es nicht, dass nur das Orientierungswissen gefördert wird, also nur Grundlagen wie Modellierung, Algorithmen und Datenstrukturen oder Berechnungsmodelle vermittelt werden. Ein Schulfach Informatik muss auch unmittelbar dazu beitragen, ICT-Werkzeuge effizient im Schulalltag und später im Arbeitsprozess zu nutzen. Es sollte deshalb neben dem Orientierungswissen die Anwenderkompetenz fördern.

Hier unterscheiden sich die Ziele des Informatikunterrichtes von den Zielen beispielsweise des Geschichts- oder Physikunterrichtes. In diesen Fächern wird in erster Linie Orientierungswissen vermittelt, das nicht primär der unmittelbaren Bewältigung des Alltags dient, sondern den Lernenden längerfristig hilft, die Welt zu verstehen und sich darin zu orientieren. Das Fach Geschichte hilft, staatspolitische Konflikte wie im Nahen Osten oder im Balkan, Umweltprobleme, Rassismus oder Terrorismus besser zu begreifen. Menschen können dank dem Geschichtsunterricht die komplexe Welt „lesen" und fühlen sich deshalb weniger ausgeliefert und fremd darin. Ähnlich verhält es sich mit dem Physik-Unterricht. Orientierungswissen wie Energie- und Impulserhaltung, Thermodynamik, Optik oder Elektromagnetismus hilft den Lernenden zu verstehen, wie ein Motor oder Kraftwerk funktioniert [HN02]. Unmittelbar verwendbares Anwenderwissen vermittelt die Physik ebenso wenig wie die Geschichte. Wobei man sich vom Physikunterricht auch Hilfestellungen bei Alltagsproblemen wie zum Beispiel nicht funktionierenden Fahrrad-Dynamos etc. erhoffen könnte.

Anders präsentiert sich der Deutsch- oder der Mathematikunterricht in der Unter- und Mittelstufe. Lesen und Schreiben sind unmittelbar nutzbare Fertigkeiten, ebenso wie das Einmaleins und das Prozentrechnen. Zusätzlich bedingt Allgemeinbildung in Deutsch oder Mathematik ein Verständnis für Orientierungswissen wie Grammatik und Logik. Genauso umfasst ein guter Informatikunterricht sowohl Anwenderkompetenz als auch Orientierungswissen und ist also fundiert und nützlich zugleich.

Diese Doppelrolle der Informatik führt zu Konflikten und Missverständnissen. Anstatt Synergien zu nutzen, bekämpfen sich die verschiedenen Standpunkte. In der informationstechnischen Grundbildung (ITG) etwa, also der in allen Fächern integrierten Informatik, stehen einseitig die Fertigkeiten im Vordergrund. Konzeptwissen oder Transferfähigkeiten bleiben auf der Strecke. Auf der anderen Seite stellen sich die Befürworter eines eigenständigen Schulfaches Informatik auf den Standpunkt, die langlebigen Grundlagen der Informatik müssten im Zentrum des Informatikunterrichts stehen. Dazu gehören Themen wie Berechnungsmodelle, grundlegende algorithmische Prinzipien oder Modellierung von Informatiksystemen. Der Bezug zu Anwenderfertigkeiten, der kompetente Umgang mit Standardsoftware, findet kaum Eingang bei diesen Überlegungen.

Besonders deutlich tritt die unterschiedliche Gewichtung von Anwenderkompetenz und Orientierungswissen zu Tage, wenn man die Inhalte einer allgemeinen Lehrerausbildung und die Inhalte eines Faches Informatik für Schüler vergleicht.

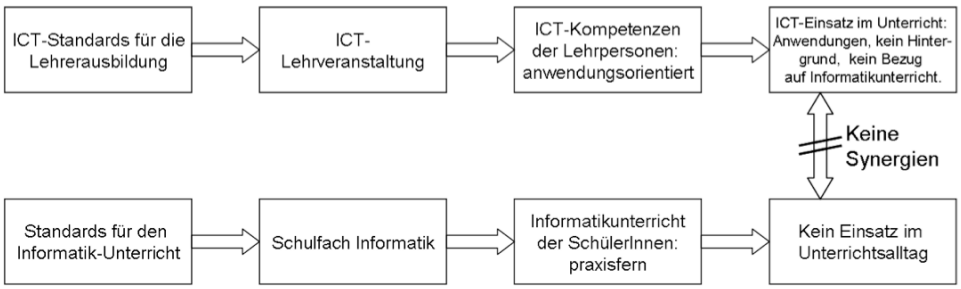


Abbildung 1: Keine Synergien zwischen dem Informatikunterricht und anderen Fächern.

2 Welche informatische Bildung brauchen Lehrkräfte heute?

2.1 Standards für die Inhalte einer ICT-Bildung für Lehrer

Die UNESCO veröffentlichte 2002 in ihrem Bericht „Information And Communication Technologies In Teacher Education“ [UN02a] Standards für die informatische Lehrerbildung. ICT sollte sich durch die gesamte Lehrerausbildung hindurch ziehen und den Studierenden immer im pädagogischen Kontext vermittelt werden.

Neue Entwicklungen im ICT-Bereich verbessern den Unterricht nicht, solange der Fokus auf den ICT-Werkzeugen bleibt. Entscheidend ist, was die Entwicklungen im Bereich ICT zur Verbesserung des Unterrichts leisten können.

Die UNESCO nennt vier Schlüsselqualifikationen, die den Rahmen für die ICT-Kompetenz der Lehrer liefern sollen. Diese werden unterlegt von den sechs Standards der International Society for Technology in Education (ISTE) [UN02b], die wir wie folgt zusammenfassen:

1. Technische Kompetenz: Lehrkräfte haben ein Verständnis von grundlegenden technischen Konzepten und können diese anwenden.
2. Vorbereitung und Gestaltung von Lernumgebungen: Lehrkräfte planen und gestalten effektiv ICT-gestützte Lernumgebungen.
3. Lehren, Lernen und Lehrplan: Lehrkräfte implementieren ICT in ihren Unterricht, um den Lernprozess der Schülerinnen und Schüler effizienter zu gestalten.
4. Bewertungssysteme und Evaluation: Lehrkräfte nutzen die Vielfalt an aussagekräftigen Bewertungssystemen und Evaluationsstrategien, welche die ICT-Technologien ermöglichen.

5. Arbeitsproduktivität und professionelles Handeln: Lehrkräfte nutzen die ICT-Werkzeuge zur Optimierung Ihrer Arbeitsproduktivität und Unterrichtspraxis.
6. Soziale, ethische und juristische Themen: Lehrkräfte reflektieren Themen, die mit der Expansion neuer Technologien auf die globale Kommunikation und Bildungsangebote einhergehen.

Alle sechs Standards machen deutlich, dass von den Lehrkräften sowohl Anwenderkompetenz als auch Orientierungswissen verlangt wird.

2.2 Heutiger Stand der Lehrer-Ausbildung in ICT

In der Schlussphase der Bildungsinitiative „Schulen im Netz“ wurde 2006 in der Schweiz eine Erhebung zum Stand der Ausbildung der Lehrkräfte im Bereich der Informations- und Kommunikationstechnologien durchgeführt [Sf06]. Im Unterschied zu einer Erhebung aus 2000 [Sf01] liegt nun der Ausbildungsschwerpunkt eindeutig auf der Methodik-Didaktik, Kurse für reine Anwenderkompetenz sind in der Minderheit. Der größte Teil der ICT-Ausbildungen findet immer noch in Form von Weiterbildungen für bereits tätige Lehrkräfte statt, die ersten Pädagogischen Hochschulen haben aber ICT/Medienpädagogik in die Grundausbildung der Lehrkräfte aufgenommen. Die jungen Lehrkräfte bringen also zunehmend ein Rüstzeug in Sachen ICT mit. Weiterbildungsaktivitäten für tätige Lehrkräfte werden kritisch beleuchtet, da sie häufig Teil von temporären Maßnahmen wie “Schulen im Netz“ und damit befristet sind.

Nimmt man die Angebote zu ICT/Medienpädagogik in der Grundausbildung der Lehrkräfte in der Schweiz etwas genauer unter die Lupe, stellt man unschwer fest, dass diese Angebote meistens noch bescheiden ausfallen. So taucht zum Beispiel in den Standards für die Ausbildung an der Pädagogischen Hochschule Zürich kein einziger ICT-bezogener Begriff auf, während „Umgang mit Heterogenität“ und ähnlichen Aspekten ein großer Stellenwert beigemessen wird. Andere Pädagogische Hochschulen bieten Freifachkurse an, die sich am Niveau der „European Computer Driving Licence“ orientieren und zu einem angemessenen Umgang mit ICT befähigen sollen. Werden verpflichtende Kurse angeboten, stehen oft medienpädagogische Aspekte im Vordergrund.

Die Situation bezüglich der Lehrerausbildung in ICT unterscheidet sich auch in Deutschland und Österreich nicht grundlegend von den Schweizer Verhältnissen. In Deutschland durchliefen im Rahmen von „Intel® Lehren für die Zukunft“ bis Ende 2006 über 300.000 Lehrkräfte aller Stufen eine Weiterbildung, deren Schwergewicht auf Anwenderkompetenzen liegt [In07].

Heutige Lehrkräfte verfügen also nicht über informatisches Orientierungswissen und damit über keine ausreichende informatische Allgemeinbildung. Im sprachlichen oder mathematischen Bereich ist eine solche selbstverständlich. Gerade diese Allgemeinbildung ist aber unabdingbare Voraussetzung für eine Integration von ICT im Unterricht. Bei einer Geographielehrerin begnügen wir uns auch nicht damit, dass sie nur den Taschenrechner bedienen kann. Wir erwarten ein Verständnis für grundlegende Konzepte

aus der Mathematik und Statistik. Und vom Mathematiklehrer erwarten wir mehr Sprachkompetenz, als zur Bewältigung des Alltags nötig ist.

2.3 ICT-Ausbildung der Lehrkräfte: integriert oder stand-alone?

Wie müsste eine ICT-Ausbilung für Lehrkräfte aussehen, die sowohl Anwenderkompetenz als auch Orientierungswissen fördert? Es gibt heute im Wesentlichen zwei Modelle zur Grundausbildung der Lehrer im ICT-Bereich. Das altbekannte Modell der stand-alone-Computerkurse stammt aus den achtziger Jahren. Ziel solcher Lehrveranstaltungen war es, die Computerkenntnisse der Lehramtsstudierenden zu verbessern. Die erworbenen Kenntnisse fanden jedoch selten ihren Weg bis in die Klassenzimmer. Die Lehrkräfte wussten nicht, wie sie die neuen Technologien in ihren Unterricht integrieren sollten. Zudem gab es auch keine Strategien, die neuen Technologien in die bestehenden Lehrpläne aufzunehmen. Vor allem in den Fachdidaktiken wurde der Ruf laut: Wie kann man die neuen Technologien effektiv zum Lernen und Lehren im Unterricht nutzen? Die Kritiker der stand-alone-Kurse forderten integrierte Kurse, unter anderem auch mit der Begründung, dass Studienanfänger bereits ausreichende Computerkenntnisse mitbringen würden.

Verschiedene Studien (z.B. [Wa06]) belegen aber, dass Studienanfänger in der Regel auch in neuester Zeit nicht über ausreichend Computerkenntnisse verfügen. Gemäß Anderson und Borthwick tragen stand-alone-Kurse nicht nur zur Verbesserung der ICT-Anwenderkompetenz bei, sondern fördern auch den Einsatz des Computers im Unterricht [AB02]. In [Wa06] wird davor gewarnt, die neuen Technologien direkt in den Fachdidaktiken zu integrieren. Zusätzliche stand-alone-Kurse würden zu einem einheitlichen Wissensstand führen, sowohl bezüglich Anwenderkompetenz als auch bezüglich Orientierungswissens. Pierson und Thompson schlagen vor, beide Methoden ergänzend in der ICT-Grundausbildung von Lehrkräften einzusetzen [PT05]. Am Anfang des Studiums stehen stand-alone-Kurse, die im Laufe des Studiums immer mehr von integrierten Kursen abgelöst werden.

Die stand-alone-Kurse vermitteln den Lehramtsstudierenden die Fähigkeiten, die sie später für die integrierten Kurse brauchen. Die integrierten Kurse liefern den Kontext zum Üben, Nachbereiten und Abstützen des Gelernten. Stand-alone und integrierter Unterricht ergänzen sich. Diese Überlegungen liegen dem Konzept der ICT-Lehrerausbildung an der Pädagogischen Hochschule Bern zugrunde.

3 Konzepte der ICT-Ausbildung an der PHBern

Seit 2005 sind in der Ausbildung für Gymnasiallehrer an der Pädagogischen Hochschule Bern stand-alone-Kurse in ICT für alle Lehrkräfte obligatorisch. Damit wird Informatik zu einem festen Bestandteil der Lehrerbildung.

Bei der Konzeption dieser stand-alone-Kurse ging man davon aus, dass alle zukünftigen Lehrkräfte bereits Anwenderwissen im persönlichen Umgang mit dem Computer mit-

bringen, dieses Wissen aber auf keinem stabilen Fundament aufbaut. Ein Schwerpunkt des Kurses liegt deshalb auf der Vermittlung von Orientierungswissen. Gleichzeitig gibt es Übungen am Computer, die im pädagogischen Kontext angesiedelt sind und die Anwenderkompetenz fördern (s. ISTE-Standard I und V: Technische Kompetenz; Arbeitsproduktivität und professionelles Handeln). In der Lehrveranstaltung bilden konkrete Unterrichtssituationen den Ausgangspunkt für die Behandlung von Themen wie Einsatz von Präsentationssoftware, Kommunikationsplattformen, Recherche und Plagiate im Internet, Umgang mit Audio und Video oder Gütekriterien für Lernsoftware (s. ISTE-Standard III und IV: Lehren, Lernen und Lehrplan; Bewertungssysteme und Evaluation). Auf einer kursbegleitenden Online-Plattform diskutieren die Studierenden über den Einfluss Neuer Medien auf das Bildungswesen (s. ISTE-Standard VI: Soziale, ethische und juristische Themen). Dabei werden immer auch die den ICT-Werkzeugen zugrunde liegenden informatischen Konzepte beleuchtet. Dieses Zusammenspiel von Werkzeugen und Konzepten ist für das Beispiel der Bildbearbeitung in der Matrix in Abb. 1 dargestellt.

	Konzepte	Umsetzung
Werkzeuge	Mit Ebenen arbeiten Filter anwenden Export und Import Schneiden, Rotieren,…	Werkzeuge für Bitmapbilder: Paint, GIMP, Photoshop,… Werkzeuge für Vektorbilder: PowerPoint, Illustrator,…
Objekte	Raster- vs. Vektorgrafiken Auflösung und Dateigröße Farbtiefe und Farbtabellen Kompressionsarten …	Bitmapformate: GIF, JPEG, PNG, BMP,… Vektorformate: EPS, PDF, AI

Abbildung 2: Anwenderkompetenz und Orientierungswissen am Beispiel Bildbearbeitung (nach [Ha06])

Parallel zur Lehrveranstaltung für die Studierenden werden den Dozierenden der Fachdidaktiken die gleichen Inhalte im Rahmen von Workshops vermittelt. Damit wird die Grundlage geschaffen, dass das erworbene Wissen der Studierenden später in der Fachdidaktik genutzt wird. Die folgenden vier Beispiele zeigen, wie in der Lehrveranstaltung sowohl Anwenderkompetenz als auch Orientierungswissen vermittelt wird.

Design-Richtlinien für Präsentationen

Die Vorteile der Trennung von Struktur, Inhalt und Layout kommen in Präsentationsprogrammen wie PowerPoint oder Impress sehr rasch zur Geltung. Nur wenige Studierende, aber auch wenige tätige Lehrkräfte nutzen Funktionen wie z.B. Formatvorlagen und Folienmaster. Am Beispiel von PowerPoint lässt sich das informatische Prinzip der Trennung von Struktur, Inhalt und Layout gut veranschaulichen. Nachdem die Studierenden in Übungen selbst erfahren haben wie schnell sich bei Verwendung von Folien-

mastern Präsentationen auf Neue anpassen lassen, fällt es ihnen leichter, dieses Prinzip auch beim Erstellen von Webseiten oder Programmen anzuwenden.

Verschiedene Bildformate

Heute ist es einfach Bilder zu erstellen, zu bearbeiten und auszutauschen. Mittels Digitalkamera, Scanner oder Bildersuche im Internet können sich Privatpersonen sehr schnell riesige Bildarchive zusammenstellen; Lehrkräfte können diese Möglichkeit für ihren Unterricht nutzen. Im Umgang mit digitalen Bildern gibt es beliebig viele Fallstricke: Ein Bild lässt sich nicht in Word einfügen, beim Vergrößern wird ein Bild unscharf, ein Bild lässt sich in PowerPoint nicht zuschneiden. In der Lehrveranstaltung wird im Detail auf die verschiedenen Bildformate eingegangen, besonders auf den Unterschied zwischen Raster- und Vektorgrafiken. Mit dem Wissen um Vor- und Nachteile verschiedener Bildformate setzen die Studierenden in den Übungen Bildformate effizient ein. Dadurch lassen sich bereits im Vorfeld viele Fehler im Umgang mit Bildern vermeiden.

	Konzepte	Umsetzung
Werkzeuge	Trennung von Inhalt, Layout und Struktur	PowerPoint: Masterfolie sowie zwei, drei Beispielfolien erstellen.
Objekte	Raster- vs. Vektorgrafiken	Kleine .jpg-Datei sowie kleine .svg-Datei in PowerPoint einfügen und anschliessend vergrössern.

Abbildung 3: Anwenderkompetenz und Orientierungswissen am Beispiel einer Übungsaufgabe

Effizient und effektiv recherchieren

Generell überschätzen NutzerInnen von Informationsdiensten ihre Fähigkeiten und vertrauen den Ergebnissen von Suchmaschinen blind (vgl. z. B. [Fa05]). In der Lehrveranstaltung wird anhand von einfachen Beispielen demonstriert, dass die Informationsbeschaffung im Internet anspruchsvoller ist als gemeinhin angenommen wird. Anschließend werden grundlegende Konzepte von Informationsdiensten gezeigt: Kategorisierung von offenen und geschlossenen Fragen, Unterschied zwischen vertikalen und horizontalen Dokumentenkollektionen, Ausbeute und Präzision bei einer Suche, Indexierung als ein Schlüsselelement in der maschinellen Durchforstung großer Datenmengen. Einige dieser Prinzipien werden mithilfe der speziell zu diesem Zweck entwickelten didaktischen Suchmaschine Soekia [So06], aufgezeigt. Mit dem so erworbenem Orientierungswissen bewältigen die Studierenden in den anschließenden praktischen Übungen anspruchsvolle Recherchieraufgaben. Neben einer verbesserten Anwenderkompetenz bei der Informationsbeschaffung erfahren die angehenden Lehrkräfte auch, dass Internetrecherche ein Thema im Unterricht sein muss.

Moderne Kommunikationsmittel

SMS, Instant Messenger, Email, MySpace, YouTube und wie sie heute alle heißen sind fester Bestandteil im Alltag heutiger SchülerInnen. Die Schule selbst macht sich diese Kommunikationsmittel jedoch kaum zu Nutze. In der Lehrveranstaltung wird zweierlei gezeigt: Erstens charakteristische Klassifikationsmerkmale wie synchron vs. asynchron und push vs. pull. Und zweitens wie sich die Kommunikationsmittel für den Unterricht gewinnbringend einsetzen lassen. Es versteht sich von selbst, dass auch in der Lehrveranstaltung eine Gruppenarbeitsplattform eingesetzt wird. Damit lernen die Studierenden durch praktische Anwendung die Möglichkeiten von modernen Kommunikationsmitteln im Unterricht kennen.

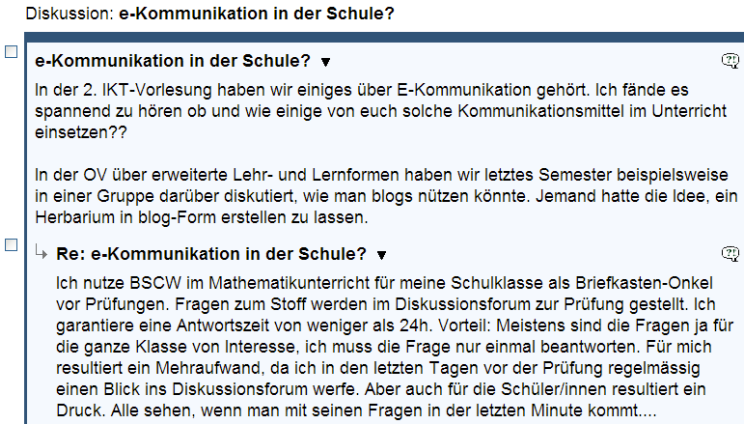


Abbildung 4: Diskussionsbeiträge von Studierenden

4 Guter Informatikunterricht muss Brücken bauen

Aus dem Bedürfnis heraus, nicht nur auf die bloßen Anwendungen der Informatik reduziert zu werden, sondern als vollwertiges wissenschaftliches Fach im Fächerkanon akzeptiert zu werden, ist das heutige Schulfach Informatik sehr stark auf Grundlagenvermittlung ausgerichtet. Darunter leidet die Förderung der Anwenderkompetenz bei den SchülerInnen. Ohne den Bezug zu ihrem Alltag bleibt das Fach Informatik für die SchülerInnen wenig fassbar; das Fach ist nur für Wenige attraktiv. Auch die Lehrkräfte anderer Fächer erkennen nur schwer den Nutzen eines Schulfaches Informatik. Eine positive Wahrnehmung des Faches Informatik seitens der SchülerInnen und Lehrkräfte ist aber wesentlich für die Etablierung eines eigenständigen Faches Informatik.

So wie in der Lehrerbildung nicht der Fehler gemacht werden darf, nur Anwenderfertigkeiten zu vermitteln, darf im Informatikunterricht nicht nur Orientierungswissen gelehrt werden. Wie in der ICT-Lehrerausbildung muss auch im Informatik-Unterricht sowohl der Anwenderkompetenz als auch dem Orientierungswissen der nötige Stellenwert beigemessen werden. So entstehen im ICT-Umfeld Synergien zwischen dem Informatik-Unterricht und anderen Fächern, beziehungsweise zwischen Lehrkräften und

Schülern. Konkret: Lernen SchülerInnen im Informatikunterricht grundlegende Prinzipien der Bildbearbeitung und Internetrecherche kennen und können sie diese Kenntnisse anwenden, können sie ihr Wissen auch für die anderen Fächer sichtbar nutzen. Ein solcher Informatikunterricht baut eine Brücke von der Theorie zur Anwendung.

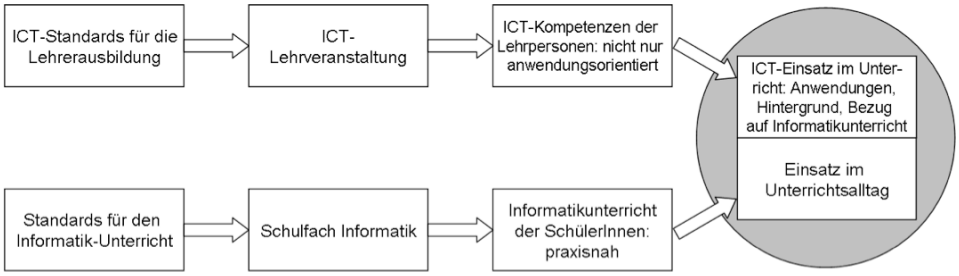


Abbildung 5: Synergien aus der ICT-Ausbildung von Lehrkräften und dem Informatikunterricht für Schülerinnen und Schüler

Wie ein Informatikunterricht aussehen könnte, der der Anwenderkompetenz einen angemessenen Stellenwert zuteilt, beschreibt W. Coy in LOG IN [Co05] anhand von vier Beispielen: 1) „Kopier doch mal“: Die Entwicklung großer digitaler Speicher; 2) „Das steht doch im Netz“: Wikipedia; 3) „Das Recht, in Ruhe gelassen zu werden“: über SPAM; 4) „Alles hängt mit allem zusammen“: RFID-Tags. Alle Themen haben einen starken Alltagsbezug und beruhen auf wichtigen informatischen Konzepten. Auch die vorgestellten Beispiele aus der ICT-Lehrerausbildung an der PHBern lassen sich im Informatikunterricht thematisieren. „Effizient und effektiv recherchieren“ zum Beispiel im Zusammenhang mit Themen wie Datenstrukturen, Index, Suchen, Sortieren oder Boole’sche Operationen. „Verschiedene Bildformate“ bietet sich unter anderem an bei den Themen Speichermedien, Kompressionsverfahren, Vektorgrafik.

Guter Informatikunterricht kann also wie guter Deutsch- oder Mathematikunterricht durchaus anschaulich und im Alltag nützlich sein. Dies bedeutet nicht, dass er an Qualität oder Tiefgang verlieren muss, ganz im Gegenteil: Dank dem alltagsbezogenen Zugang zu komplexen informatischen Themen werden diese konkreter und somit besser verständlich und anwendbar.

Literaturverzeichnis

[AB02] Anderson C., Borthwick A.: Results Of Seperate And Integrated Technology Instruction in Preservice training. Wissenschaftliche Veröffentlichung dargelegt an der National Educational Computing Conference, San Antonio, Texas, Juni 2002

[Co05] Coy, W.: Informatik... im Großen und Ganzen. In: LOG IN, 25. Jg. (2005), H. 136/137, S. 17-23

[Fa05] Fallows, D.: Search Engine Users. Internet searchers are confident, satisfied and trusting – but they are also unaware and naïve. PEW Internet & American Life Project, Washington, 2005

[Ge04] Gesellschaft für Informatik (Hrsg.): Digitale Spaltung verhindern – Schulinformatik stärken! Memorandum, Ulm, 2004.

[Ha06] Hartmann, W.; Näf, M.; Reichert, R.: Informatikunterricht planen und durchführen. Springer-Verlag 2006

[HN02] Hartmann, W.; Nievergelt, J.: Informatik und Bildung zwischen Wandel und Beständigkeit. In: Informatik Spektrum, Vol. 25, Nr. 6, Dezember 2002

[In07] Intel: Das Lehrerfortbildungsprojekt „Intel Lehren“. http://www.intel.com (Stand: 19.01.2007)

[PT05] Pierson M., Thompson M.: The Re-envisioned Educational Technology Course: If Addition Isn’t Possible” Computing in Teacher Education, Vol. 22, Nr 1, 2005, S. 31-36

[Sf01] SFIB (Hrsg.): Aktivitäten, Projekte, Konzepte zur Aus- und Weiterbildung der schweizerischen Lehrpersonen in ICT. Bern, 2001

[Sf06] SFIB (Hrsg.): Umfrage Aus- und Weiterbildung in ICT und Medienpädagogik. Bern, 2006

[So06] Soekia – Ein Blick hinter die Kulissen von Suchmaschinen. http://www.swisseduc.ch/informatik/soekia/ (Stand 04.01.2007)

[UN02a] UNESCO (Hrsg.): Information And Communication Technologies In Teacher Education. Paris 2002

[UN02b] UNESCO (Hrsg.): Information And Communication Technologies In Teacher Education. Paris 2002, S. 50-53

[Wa06] Wang Y.: Stand-alone computer Courses in Teachers’ IT Training. EDUCAUSE Quarterly Volume 29 Number 3 2006.

Auf dem Weg zu Bildungsstandards für Konzepte der Theoretischen Informatik in der Sekundarstufe

Kirsten Schlüter, Torsten Brinda

Didaktik der Informatik
Universität Erlangen-Nürnberg
Martensstraße 3
91058 Erlangen
{schlueter,brinda}@informatik.uni-erlangen.de

Abstract: Die Konzepte der Theoretischen Informatik (TI) sind ebenso fundamental wie abstrakt und anspruchsvoll in der Vermittlung. Zur Orientierung der Lehrenden in der Sekundarstufe bedarf es daher in besonderem Maße konkreter Bildungsstandards für die TI. In dieser Arbeit wird ein Ansatz zur Gewinnung von Kompetenzen und Kompetenzstufen als Bausteine eines Kompetenzmodells der TI entwickelt. Exemplarisch werden unterrichtsrelevante Fachinhalte strukturiert dargestellt, Kompetenzziele und Aufgaben formuliert und ein Konzept zur Unterscheidung von Niveaustufen der TI-Kompetenz vorgestellt.

1 Motivation

Die Theoretische Informatik (TI) ist das wissenschaftliche Fundament der Informatik. Sie hinterfragt die prinzipielle Lösbarkeit von Problemen und untersucht die Grenzen der Automatisierung. Sie stellt Modelle zur Verfügung, mit deren Hilfe Problemlösungen auf grundsätzliche Machbarkeit hin überprüft und miteinander verglichen werden können. Ein jeder Informatikunterricht muss auch die TI berühren. Im Hinblick auf den Eintritt in das Berufsleben, eventuell bereits nach der Sekundarstufe I, ist es zwar dienlich, die Möglichkeiten der Computernutzung kennen zu lernen. Zur Computerbeherrschung gehört darüber hinaus, deren Grenzen zu kennen, die erst die TI sichtbar macht. Im Hinblick auf ein Studium im Anschluss an die Sekundarstufe II gewinnen Schülerinnen und Schüler einen Einblick in Fragestellungen der Fachwissenschaft und werden in die Lage versetzt, eine nachhaltige Studienentscheidung zu treffen.
Als Unterrichtsinhalt befindet sich die TI im Spannungsfeld der fachwissenschaftlichen Forderung nach theoretischer Fundierung der Informatik von Anfang an und der Maßgabe der Schulung von Bedienerfertigkeiten, die das Fach Informatik für Schüler und Eltern auf den ersten Blick attraktiv macht [Hu03, S. 48f]. Um die abstrakte und formale TI dem Unterricht in der Sekundarstufe besser zugänglich zu machen, bedarf es einer sorgfältigen Auswahl der Inhalte, einer Abgrenzung der zu vermittelnden Kompetenzen und ihrer Strukturierung anhand eines Kompetenzmodells. Modellvorstellungen über den Kompetenzerwerb bieten Anhaltspunkte für eine Unterrichtspraxis, die an Lernpro-

zessen der Schüler orientiert ist und nicht allein an der Fachsystematik [Kl03, S. 71]. In diesem Sinne sind Bildungsstandards für die TI in besonderem Maße als Orientierung, mehr noch, als Gerüst für den Unterrichtsentwurf unverzichtbar.
Die Entwicklung von Kompetenzmodellen und Bildungsstandards der Informatik steht im Vergleich zu den Kernfächern noch ganz am Anfang. Erste theoretische Entwürfe, zum Beispiel von Friedrich [Fr03], Puhlmann [Pu03] oder Magenheim [Ma05] adressieren die informatische Bildung insgesamt. Weitere Ansätze betreffen spezifische Teilgebiete, jedoch nicht der TI zuzuordnende Kompetenzbereiche. Die informatikdidaktische Forschung zur TI im Unterricht konzentriert sich bisher auf die Entwicklung und Erprobung von Vermittlungskonzepten und Lernhilfen, nicht auf die Kompetenzmodellierung, die jedoch für die Alleinstellung des informatischen Kompetenzbeitrags, den die TI leistet, unabdingbar ist.
In der vorliegenden Arbeit werden zunächst in den Kapiteln 2 und 3 die Situation der TI in der Schule und Vermittlungsmethoden für ihre Inhalte betrachtet. In Kapitel 4 werden Anforderungen an ein Kompetenzmodell für die TI in der Sekundarstufe formuliert. Darauf abgestimmte Aspekte der Aufgabenklassifikation und -konstruktion werden in den Kapiteln 5 und 6 behandelt. Insbesondere wird im letzteren Kapitel ein Konzept zur Unterscheidung von Niveaustufen der Kompetenz exemplarisch implementiert. Schließlich werden in Kapitel 7 weiterführende Maßnahmen der Entwicklung eines Kompetenzmodells der TI skizziert.

2 Situation der Theoretischen Informatik in der Schule

Zur Situation der Theoretischen Informatik in der Schule sind die Didaktik-Forschung, die Lehrpläne und die unmittelbare Unterrichtspraxis zu betrachten.
In der Didaktik-Forschung herrscht Konsens, dass wesentliche Gegenstände der TI, zum Beispiel „Syntax und Semantik von Sprachen“, fundamental sind [Sc93]. Die TI ist integraler Bestandteil einer wissenschaftspropädeutischen Informatikausbildung in der Sekundarstufe II, deren fachliche Inhalte in den Einheitlichen Prüfungsanforderungen in der Abiturprüfung (EPA) Informatik [KK04, S. 8f] festgelegt sind.
Zumindest im Wahlbereich ist das Fach Informatik mittlerweile in den Stundentafeln der Sekundarschulen der meisten deutschen Bundesländer verankert. Die TI als Lehrplaninhalt ist Gegenstand der Diskussion, wie hoch der Theorieanteil des Informatikunterrichts zu wählen ist, um begründeten, aber divergierenden Forderungen fachwissenschaftlicher, politischer und pragmatischer Art zu genügen. In den aktuellen Informatik-Lehrplänen ist die TI durch die Themenbereiche Sprachen, Automaten und Grenzen der Berechenbarkeit vertreten. In Bayern und Thüringen etwa hat die TI bereits in der Sekundarstufe I ihren Platz, unterrichtsthematisch anknüpfend an Algorithmen und Automaten.
Die TI ist integraler Bestandteil einer grundständigen Ausbildung für Informatik-Lehrende, wie sie in nationalen (z. B. Berlin, Brandenburg, NRW) und internationalen Curricula (z. B. Israel [GH98]) angelegt ist beziehungsweise gefordert wird. Dass Kernideen der Theoretischen Informatik auf Sekundarstufenniveau gut vermittelbar sind, zeigen u. a. Fothe [Fo05] und Loidl, Mühlbacher und Schauer [LMS05]. Dennoch lassen Gespräche mit Informatik-Lehrenden oft vermuten, dass die Schwerpunktsetzung bei der Unterrichtsplanung pragmatisch nach Ausbildungshintergrund des Lehrenden und nach Verfügbarkeit geeigneter Unterrichtsmaterialien erfolgt und die TI dabei nachrangig

berücksichtigt wird. Es besteht Bedarf an Unterrichtskonzepten und Lernhilfen für die TI sowie an einer schulgerechten Strukturierung der Fachinhalte und Aufgaben zur Veranschaulichung, Übung und Überprüfung des TI-Wissens.

3 Vermittlungsmethoden für Inhalte der Theoretischen Informatik

Eine übliche Gliederung der Theoretischen Informatik umfasst die Teilgebiete Sprachen und Automaten, Berechenbarkeit und Komplexität [Sc01], die die Kriterien der Fundamentalität erfüllen [Sc93], indem sie (1) vielfältig anwendbar und erkennbar sind, (2) auf jedem intellektuellen Niveau aufgezeigt und vermittelt werden können, (3) langfristig relevant bleiben und (4) einen Bezug zur Lebenswelt besitzen. Besonderes Augenmerk ist auf Punkt (2) zu richten: die TI ist überwiegend in der Sekundarstufe II verankert und in der Sekundarstufe I nur schwach vertreten, möglicherweise weil Zweifel an einer altersgerechten Vermittelbarkeit bestehen. Schon jüngere Schülerinnen und Schüler, die Informatiksysteme oft genug geschickt, aber unreflektiert, einsetzen, sollten jedoch eine Vorstellung von den Grenzen prinzipieller und praktischer Berechenbarkeit entwickeln. Kritischer Umgang mit Informatiksystemen erfordert neben deren Handhabung auch das Einschätzungsvermögen der Angemessenheit und Leistungsfähigkeit. Eine dem Alter entsprechende Informatik-Mündigkeit setzt das Wissen voraus, dass man mit dem Computer nur bearbeiten kann, was man formalisieren kann. Sprachen und Automaten zur Modellierung von Programmiersprachen und zustandsgesteuerten Abläufen bieten sich an, weil sie in didaktisch reduzierter Darstellung Einblick in wesentliche Informatik-Konzepte gewähren.

Die Herangehensweise im Schulunterricht ist meist problemorientiert. Zu den für die Sekundarstufe relevanten Gebieten der TI existieren vielfältige Beispiele aus der Erfahrungswelt der Schüler, bei deren Bearbeitung und Lösung – selten am Computer, häufiger mit Papier und Bleistift – sich Konzepte der TI als schlüssig und hilfreich erweisen. Als spielerisches Beispiel sei die Lösung des bekannten Missionare-und-Kannibalen-Rätsels allein durch Modellierung der Zustandsübergänge genannt [Ni95]. Auch anhand der Modellierung technischer Geräte des täglichen Lebens mit einfachen Automaten kann der fundamentale Begriff des Zustands vermittelt werden. Ebenso können formale Sprachen in Gestalt einfacher Grammatiken anhand der Syntax von E-Mail-Adressen, Handynummern oder auch KFZ-Kennzeichen alltagsbezogen eingeführt werden. Zur Veranschaulichung der Problematik nicht polynomiell beherrschbarer Komplexität kann das Problem des Handlungsreisenden dienen. Die Problemstellung ist offensichtlich, die exponentielle Komplexität wird bereits an einfachen Zahlenbeispielen einsichtig und heuristische Lösungsverfahren sind durch die Schüler selber auffindbar. Als simples Rechnermodell ist der Know-how-Computer [Ba03a] geeignet, eine Registermaschine mit Streichhölzern als Bits und fünf Befehlen. Am Tageslichtprojektor oder in der Computersimulation ist beispielsweise Schritt für Schritt nachzuvollziehen, wie Addieren auf wiederholtes Inkrementieren zurückzuführen ist und Multiplizieren auf wiederholtes Addieren. So konnte die Autorin bereits in achten Realschulklassen handlungsorientiert eine intuitive Idee von Berechenbarkeit wecken.

Für das Gebiet der Automaten, besonders für Turingmaschinen, ist eine Reihe von Werkzeugen und Visualisierungen verfügbar, von Animationen des fleißigen Bibers bis zur Lernumgebung TuringKara [Re03] samt Unterrichtsbeispielen. Auf dem Gebiet der

formalen Sprachen gibt es neben den Werkzeugen aus dem Compilerbau, „lex“ und „yacc“ bzw. ihren Weiterentwicklungen „flex“ und „bison“, nur wenige sekundarstufengeeignete E-Learning-Materialien, wie die Lernumgebung AtoCC [HW06]. In der Zeitschrift LOG IN finden sich immer wieder Artikel, in denen TI-nahe Themen, beispielsweise Kryptographie [WS06], schülergerecht aufbereitet werden. Dewdney [De95] zeigt in populärwissenschaftlichem Stil, wie Kernideen der TI, etwa das Halteproblem oder nicht berechenbare Probleme, auf einem Niveau vermittelt werden können, das lediglich Interesse an mathematischen und logischen Zusammenhängen voraussetzt.
Zusammengefasst zeigt sich, dass Vermittlungsideen und unterrichtsgeeignete Materialien für die TI durchaus vorhanden sind, wenn auch selten in Form von didaktisch kommentierten Vermittlungskonzepten und noch seltener nach einem Curriculum geordnet. Um die vorhandenen Vermittlungskonzepte den Unterrichtenden besser verfügbar zu machen, ist eine systematische Ordnung der Ideen und Materialien, unter Einbeziehung übertragbarer Konzepte verwandter Disziplinen wie der Mathematik, notwendig.

4 Auf dem Weg zu einem Kompetenzmodell für die TI

Kompetenzmodelle erfüllen in Bezug auf Bildungsstandards zwei Zwecke [Kl03]. Sie beschreiben erstens das Gefüge der Anforderungen, deren Bewältigung von Schülerinnen und Schülern erwartet wird, in einem Komponentenmodell. Zweitens liefern sie wissenschaftlich begründete Vorstellungen, welche Abstufungen eine Kompetenz annehmen kann beziehungsweise welche Grade oder Niveaustufen sich bei den einzelnen Schülerinnen und Schülern feststellen lassen, in einem Stufenmodell. Aufgaben dienen im Kompetenzmodell der Implementierung des Modells. Sie werden zur Veranschaulichung und zur Überprüfung des Modells, zur Übung und Vertiefung erworbener Fertigkeiten sowie zur Leistungsbewertung eingesetzt.
Es gibt viele mögliche Gründe dafür, dass die Theoretische Informatik, vor allem in der Sekundarstufe I, nur eine Nebenrolle spielt: (1) Der Anwendungsorientierung wird der Vorzug vor der Wissenschaftsorientierung gegeben. (2) Es wird angenommen, dass die TI die Forderung nach Aktualität und Lebensnähe nicht erfüllt. (3) Das Abstraktionsvermögen der Lernenden wird als noch nicht ausreichend eingestuft. (4) Es sind noch nicht genug grundständig ausgebildete Informatiklehrpersonen verfügbar. In jedem Fall besteht Bedarf an einer systematischen Ordnung der Kompetenzziele als Wesenskern von Bildungsstandards und der Aufgaben zur Umsetzung, Überprüfung und Normierung von Bildungsstandards. In den ersten beiden Fällen gilt es, in den Kompetenzzielen zu verankern, dass der erwünschte Kompetenzzugewinn in der TI mit einem Kompetenzzugewinn bezüglich aktueller und anwendungsnaher Informatikinhalte, wie der Modellierung und Algorithmisierung, verbunden ist. Im dritten Fall ist aufzuzeigen, dass sich aus der TI Teilgebiete und Problemstellungen herauslösen lassen, die zufriedenstellend und bereichernd auf dem Niveau der Sekundarstufe I behandelt werden können. Im vierten Fall ist eine systematische Ordnung der Kompetenzziele und Aufgaben unabdingbar als Hilfestellung etwa für Lehrpersonen, die fachfremd unterrichten. Um Wahrnehmung und Wirksamkeit im Schulalltag zu erzielen, muss die Darstellung der Kompetenzziele und Aufgaben detailliert nachvollziehbar sein und den Ausführenden Bezugspunkte, idealerweise ein Gerüst liefern, um Unterricht zu planen und seine Qualität zu sichern.

Bestehende Regelwerke liefern Orientierungspunkte für den Entwurf eines Kompetenzmodells der TI. So werden in den EPA Informatik im Lern- und Prüfungsbereich „Grundlegende Modellierungstechniken“ die zustandsorientierte Modellierung, insbesondere Automaten (Zustände und Zustandsübergänge), Zustandsdiagramme ausgewiesen; in „Interaktion mit und von Informatiksystemen“ die Sprache als Werkzeug der Kommunikation, Aspekte formaler Sprachen, Syntax und Semantik; in „Möglichkeiten und Grenzen informatischer Verfahren“ die prinzipiellen und praktischen Grenzen der Berechenbarkeit. Einen Ausgangspunkt für die inhaltliche Strukturierung bildet die Gliederung, die der GI-Arbeitskreis „Bildungsstandards in der Informatik“ (vgl. [Pu05], [Br06]) dem ersten Entwurf zugrunde legt: fünf Inhaltsbereiche – Daten und Information; Aufbau und Funktionsweise von Informatiksystemen; Algorithmen; Sprachen und Automaten; Informatik, Mensch und Gesellschaft – werden mit fünf Prozesskompetenzen – Modellieren und Implementieren; Begründen und Bewerten; Kommunizieren und Kooperieren; Darstellen und Interpretieren; Zusammenhänge herstellen – verknüpft.
Bei der Entwicklung eines Kompetenzmodells der TI müssen die Fragen der Repräsentation, das heißt der lehrer- und schülergerechten Darstellung der Kompetenzziele, und des Aufgabenmaterials, das heißt der quantitativen und qualitativen Beurteilung der vorhandenen Aufgaben, bearbeitet werden. Innerhalb des bestehenden Orientierungsrahmens soll ein Grundgerüst entstehen, in dem sukzessive Wissen und Fertigkeiten abgegrenzt werden können, die sekundarstufenrelevante TI-Kompetenz bedeuten, und in dem die Überprüfung der TI-Kompetenz einschließlich der Unterscheidung von Niveaustufen empirisch untersucht werden kann.

5 Kompetenzen und Aufgaben

Aufgaben spielen im Kompetenzmodell eine tragende Rolle. Sie dienen der Operationalisierung der Kompetenzanforderungen und der Diagnose der Kompetenzerfüllung. Das Augenmerk dieser Arbeit liegt zunächst auf der Analyse des verfügbaren Aufgabenmaterials für die Theoretische Informatik in der Sekundarstufe. In Brinda [Br04] wird die Entwicklung einer strukturierten Sammlung von Aufgabenklassen für Objektorientiertes Modellieren (OOM) im Informatikunterricht der Sekundarstufe II beschrieben. Ausgewählte Aufgaben aus Fachbüchern werden zu so genannten Aufgabenklassen abstrahiert, indem sie von Erläuterungen, Kontexten und Bezeichnern getrennt werden. Diese Aufgabenklassen werden hinsichtlich dreier Dimensionen, Fachkern, Gegenstand und Aufgabentyp, analysiert und klassifiziert. Die identifizierten Aufgabentypen werden mit Blooms Lernzieltaxonomie verknüpft und es wird begründet, dass für jede kognitive Lernzielstufe Aufgaben gestaltet werden können. Die Gestaltung von Aufgaben mittels Aufgabenklassen erfolgt durch Einbettung in einen Kontext (vgl. auch [Sc05]).
Die hier vorzunehmende Analyse von Aufgaben der TI baut in wesentlichen Teilen auf diesem Konzept auf. Die Auswahl, Abstraktion und Klassifikation erfolgt analog, wenn auch mit TI-bezogenen Kriterien zur Aufgabenauswahl und TI-bezogenen Dimensionen zur Klassifizierung. Die Aufgabentypen werden allerdings nicht direkt mit den Lernzielstufen verknüpft. Vielmehr wird der besonderen Bedeutung, die der Unterscheidung von Niveaustufen der Kompetenz zukommt, durch eine Differenzierung nach dem Grad der benötigten Hilfestellung innerhalb der Aufgabe entsprochen (s. Kapitel 6).

Aufgaben der TI werden in Schulbüchern und Schülerwettbewerben im Stil von Textaufgaben sprachlich formuliert, um den Lebensweltbezug herzustellen und Schülerinnen und Schülern den Zugang zu erleichtern. Eine Klassifizierung der Aufgaben erfordert die Identifizierung des abstrakten Kerns einer jeden Aufgabe, damit sie eindeutig einem Fachinhalt zugeordnet werden kann. Zur Aufgabenauswahl kann nun die Relevanz des wesentlichen Fachinhaltes herangezogen werden, wissenschaftlich nach Fundamentalität oder pragmatisch nach Lehrplandeckung beurteilt. Im Weiteren sind Ordnungskriterien wie der Anforderungsbereich (z. B. wiedergeben (I), anwenden/konstruieren (II) und begründen/beweisen, in Anlehnung an die EPA Informatik [KK04, S. 14 – 16]), der Formalisierungsgrad (von mathematisch notiert bis sprachlich ausformuliert) und der Schwierigkeitsgrad auf ihre Eignung für die Aufgabenklassifizierung zu untersuchen. Zum Schwierigkeitsgrad als Ordnungskriterium ist zu bemerken, dass die Klassifizierung der Aufgaben einen Schritt auf dem Weg zu einem Kompetenzmodell darstellt, das wiederum eine Einordnung von Aufgaben bezüglich ihres Schwierigkeitsgrades ermöglichen soll. Um diese zyklische Abhängigkeit aufzulösen, können die Aufgaben im ersten Gang nicht unmittelbar nach Schwierigkeit geordnet werden, sondern etwa nach der adressierten Jahrgangs- bzw. Altersstufe des Schulbuches oder Schülerwettbewerbes.
Die erste Sichtung des Aufgabenmaterials lässt vermuten, dass eine strukturelle Klassifizierung von TI-Aufgaben zu einem Repertoire an Aufgabenschemata führt, das kombiniert mit altersgerechten und lebensweltbezogenen Gestaltungsvorschlägen den Grundstock zur Konstruktion von Aufgaben zu Übungs- und Testzwecken bildet. Es ist zu untersuchen, inwieweit die Aufgabenkonstruktion inklusive einem Niveaustufenkonzept, wie es im folgenden Kapitel entworfen wird, automatisiert werden kann.

6 Identifizierung von Kompetenzstufen

Die unterrichtsrelevanten Fachinhalte der Theoretischen Informatik werden zunächst strukturiert dargestellt. Für einen Ausschnitt, die regulären Sprachen, werden exemplarisch Kompetenzziele formuliert und mit typischen Aufgaben verknüpft. Anhand dieser Beispielaufgaben wird ein Ansatz zur Unterscheidung von Leistungsstufen entlang einer Lernzieltaxonomie vorgestellt.

Strukturierung der Fachinhalte: Vereinfachend werden diejenigen Kapitel der Fachwissenschaft TI als unterrichtsrelevant angenommen, die in mindestens einem Sekundarstufenlehrplan Berücksichtigung finden. Eine detaillierte Untersuchung der Unterrichtsrelevanz der TI-Fachinhalte ist noch zu leisten. Die relevanten Inhalte werden nach vier Teilgebieten der TI – Automaten; Sprachen; Berechenbarkeit; Komplexität – geordnet und tabellarisch dargestellt mit der Semantik, dass aufeinander aufbauende Inhalte eines Kapitels von oben nach unten aufeinander folgen und Pfeile assoziative Sequenzen des Wissenserwerbs markieren. Die TI-Inhalte der EPA Informatik (vgl. Kap. 4, [KK04]), in der Tabelle grau hinterlegt, werden nach Prüfungsbereichen – Modellierungstechniken; Interaktion; Möglichkeiten und Grenzen – geordnet und so in die Tabelle eingebettet, dass ihre vertikale Position zeigt, auf welche fachwissenschaftlichen Inhalte sie sich beziehen (vgl. Abb. 1). Weiterführend bietet es sich an, Hypertexte zur Repräsentation zu wählen, deren assoziative Struktur die Verflechtung der Inhalte wiedergibt.

<table>
<tr><th colspan="4">Fachwissenschaft: Teilgebiete der TI</th></tr>
<tr><th>Automaten</th><th>Sprachen</th><th>Berechenbarkeit</th><th>Komplexität</th></tr>
<tr><td>Endliche Automaten</td><td>↔ Reguläre Sprachen</td><td colspan="2">Turingmaschinen</td></tr>
<tr><td>Automaten (Zustände und Zustandsübergänge), Zustandsdiagramme</td><td>↔ Aspekte formaler Sprachen, Syntax und Semantik</td><td>Berechenbarkeits-theorie</td><td>Zeitkomplexität</td></tr>
<tr><td>Kellerautomaten</td><td>↔ Kontextfreie Sprachen</td><td>Unentscheidbare Probleme</td><td>NP-Vollständige Probleme</td></tr>
<tr><td>Linear beschränkte Automaten</td><td>↔ Kontextsensitive Sprachen</td><td rowspan="2">Prinzipielle Grenzen der Berechenbarkeit</td><td rowspan="2">↔ Praktische Grenzen der Berechenbarkeit</td></tr>
<tr><td></td><td>Chomsky-Hierarchie</td></tr>
<tr><td>Grundlegende Modellierungstechniken (Zustandsorientierte Modellierung)</td><td>Interaktion mit u. von Informatik-systemen (Sprache als Werkzeug der Kommunikation)</td><td colspan="2">Möglichkeiten und Grenzen informatischer Verfahren</td></tr>
<tr><th colspan="4">Schule (EPA Informatik): Lern- und Prüfungsbereiche</th></tr>
</table>

Abbildung 1: Verzahnung der Lerngebiete der EPA Informatik mit Fachinhalten der TI

Kompetenzziele: Exemplarisch werden Prozesskompetenzen (vgl. Kap. 4) für den Inhalt Reguläre Sprachen konkretisiert, differenziert nach Sekundarstufe I/II (vgl. Abb. 2).

Prozesskompetenz	**Sekundarstufe I**	**Sekundarstufe II**
Modellieren und Implementieren (M)	Einfache Sprachen mit Syntaxdiagrammen modellieren	Einfache Sprachen mit Grammatiken modellieren
Begründen und Bewerten (B)	Syntaktische Korrektheit von Beispielen überprüfen	Korrektheit eines Entwurfs beweisen (z. B. vollständige Induktion)
Kommunizieren und Kooperieren (K)	Eigene Entwürfe präsentieren, Verbesserungs-vorschläge diskutieren	Eigene Entwürfe präsentieren, Verbesserungsvorschläge diskutieren
Darstellen und Interpretieren (D)	Gegebene Syntaxdiagramme interpretieren und anpassen	Gegebene Grammatiken interpretieren und anpassen
Zusammenhänge herstellen (Z)	Natürliche und künstliche Sprachen vergleichen (z. B. Fehlertoleranz)	Reguläre Sprachen gegen nichtreguläre Sprachen abgrenzen

Abbildung 2: Prozesskompetenzen im Bereich der TI in der Sekundarstufe

Aufgaben: Die Kompetenzziele für die Sekundarstufe I werden durch Aufgaben veranschaulicht (vgl. Abb. 3). Für die Prozesskompetenz „Kommunizieren und Kooperieren “ wird hier keine Aufgabe formuliert, weil von einer schriftlichen Prüfungssituation zur Erfassung der Einzelleistung ausgegangen wird. Die Aufgaben werden nach der

Bloom'schen Lernzieltaxonomie [Bl76] den Kategorien Wissen (1), Verstehen (2), Anwendung (3), Analyse (4), Synthese (5) und Bewertung (6) zugeordnet.

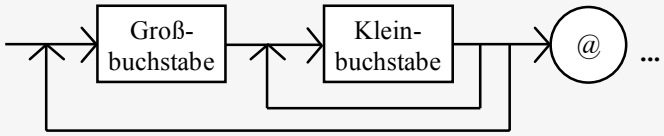
Für E-Mail-Adressen stehen die großen und die kleinen Buchstaben ohne Umlaute sowie die Ziffern und die Zeichen „–“, „.“ und „@“ zur Verfügung. Buchstaben und Ziffern dürfen in beliebigen Folgen kombiniert werden. Zeichen dürfen nur einzeln vorkommen, davor und danach müssen Buchstaben oder Ziffern stehen. Das Zeichen „@“ muss genau einmal auftreten. Betrachte der Einfachheit halber nur E-Mail-Adressen der Top-Level-Domäne „.de“. Beispiele gültiger E-Mail-Adressen sind `abc@muenchen.ag.de`, `Peter.Schilling@Mentzenhof.de`, `anna-marie@ursulinen.gym.de`.

(M) Erstelle ein Syntaxdiagramm für E-Mail-Adressen. (5)

(B) a) Finde geeignete Beispiele und überprüfe, ob dein Syntaxdiagramm E-Mail-Adressen korrekt beschreibt. (4)

b) Erkläre anhand einer gültigen und einer ungültigen Adresse, wie dein Diagramm aufgebaut ist. (3)

(D) In einer Schule werden nur E-Mail-Adressen vergeben, die der folgenden Syntax entsprechen:



a) Beschreibe die Syntax. Nenne zwei gültige Beispiele. (4)

b) Stelle dar, wie Namenskonflikte vermieden werden können, wenn zwei oder mehr Schüler den gleichen Namen tragen. Ergänze das Syntaxdiagramm entsprechend. (5)

(Z) Vergleiche künstliche Sprachen mit natürlichen Sprachen. Beziehe dich auf das E-Mail-Beispiel. (6)

Abbildung 3: Verknüpfung von Aufgaben und Prozesskompetenzen der TI

Erwartungsbild für die Lösungen: In (M) soll ein Syntaxdiagramm erstellt werden. Dazu sind die Buchstaben und Ziffern zu einer Menge beliebig kombinierbarer „Schriftzeichen“ zusammenzufassen, Trennstrich und Punkt zu einer Menge „Trennzeichen“. Das Diagramm soll die spezifizierte Abfolge von Schrift- und Trennzeichen modellieren. Mögliche Fehler bestehen darin, Restriktionen nicht zu berücksichtigen, z. B. können Trennzeichen nur zwischen Schriftzeichen vorkommen, syntaktische Varianten außer Acht zu lassen, z. B. können mehrere Sequenzen aus Schrift- und Trennzeichen aufeinander folgen, oder Sonderfälle nicht zu betrachten, z. B. E-Mail-Adressen ohne Trennzeichen im Namen. In (B) soll das zuvor erstellte Syntaxdiagramm anhand geeigneter Beispiele (vgl. Abb. 4) überprüft werden. Ein vorgegebenes Syntaxdiagramm soll in (D) sprachlich dargestellt und zwecks Vermeidung von Namenskonflikten (vgl. Abb. 4) angepasst werden. In (Z) sollen künstliche und natürliche Sprachen mit Hilfe geeigneter Kriterien, wie Fehlertoleranz, voneinander abgegrenzt und verglichen werden.

Kompetenzstufen: Zur Überprüfung der Kompetenzziele ist eine quantitative und qualitative Erfassung der Leistung erforderlich. Im „Programme for International Student Assessment (PISA)“ [Ba03b], dessen Ergebnisse die aktuelle Diskussion um Bildungsstandards in das öffentliche Interesse gerückt haben, dienen der quantitativen Leistungserhebung Testaufgaben. Die qualitative Interpretation der Testergebnisse erfolgt mithilfe eines Kompetenzstufenmodells, das auf Lösungshäufigkeiten beruht. In der Mathematik-Didaktik wird diskutiert, ob eine inhaltliche Interpretation der Lösungshäufigkeiten angesichts der Vielfalt von Lösungszugängen überhaupt möglich ist. Meyerhöfer [Me04] hat sechs veröffentlichte PISA-Aufgaben untersucht. Er argumentiert, dass die Aufgaben mehrere Lösungswege aufweisen, die unterschiedlichen Kompetenzstufen zuzuordnen sind, sodass eine Zuweisung der Aufgabe zu einer Kompetenzstufe fragwürdig ist.

Der hier verfolgte Ansatz geht auf die Idee des „Aufgabenlösens mit Hilfekarten“ aus dem BLK-Programm zur Steigerung der Effizienz des mathematisch-naturwissenschaftlichen Unterrichts (SINUS) zurück [BK02, S. 50]. Dort stehen bei der selbstständigen Bearbeitung einer Aufgabe den Schülerinnen und Schülern Hilfekarten mit Lösungshinweisen zur Verfügung, die eine Binnendifferenzierung in der Übungsphase ermöglichen. Für die Leistungserhebung wird nun vorgeschlagen, die Inanspruchnahme einer Hilfestellung als Indikator für die Kompetenzstufe, auf der die Lösung einer Aufgabe erfolgt, zu interpretieren. Dazu wird die Aufgabe in eine Lernzieltaxonomie, hier Bloom, eingeordnet, die orthogonal zu den überprüften Prozesskompetenzen sein muss. Ein Nachweis der Orthogonalität der Lernzielstufen und Prozesskompetenzen steht noch aus. Die Aufgabe wird ergänzt durch eine Hilfestellung (vgl. Abb. 4), auf die die Schülerinnen und Schüler zurückgreifen, die die Aufgabe ohne Hilfe nicht lösen können. Die ergänzte Aufgabe wird wiederum taxiert. Werden die Lernzielstufen als Niveaustufen der Kompetenz angenommen, ist somit eine abgestufte Kompetenzeinschätzung möglich, die das Niveau, auf dem die Bearbeitung der Aufgabe erfolgt, berücksichtigt.
In Aufgabe (M) liegt die Synthese-Leistung (5) in der Modellierung der syntaktischen Struktur von E-Mail-Adressen. Wird die Grobstruktur in der Hilfe i. vorgegeben, ist die Ergänzung zu einem vollständigen Syntaxdiagramm durch schematische Anwendung (3) des Wissens über die Notation zu leisten. Um das Vorhandensein des Notations-Wissens (1) zu testen, wird in der Hilfe ii. das vollständige Syntaxdiagramm vorgegeben.
In Aufgabe (B) a) besteht die geforderte Leistung in der Analyse (4) der syntaktischen Struktur von E-Mail-Adressen, um geeignete Prüfbeispiele zu finden. Werden als Hilfestellung die Beispiele vorgegeben, besteht die Überprüfung darin, die syntaktische Korrektheit der E-Mail-Adressen durch Anwendung (3) des Syntaxdiagramms zu prüfen und das Ergebnis, gültig oder nicht, mit der Angabe zu vergleichen. Die Hilfestellung hat keinen Einfluss auf den Aufgabenteil b).
Die in Aufgabe (D) a) verlangte sprachliche Beschreibung der E-Mail-Syntax erfordert die Analyse (4) des Syntaxdiagramms. Die Entscheidung, ob die in der Hilfestellung gegebenen Aussagen zutreffen, ist durch probeweise Anwendung (3) der Syntaxregeln zu treffen. In Aufgabenteil b) stellt die Problemlösung der Namenskonflikte mithilfe des erworbenen Wissens eine Synthese-Leistung (5) dar. Wird in der Hilfestellung eine Lösung des Problems vorgezeichnet, besteht die Leistung in der Analyse (4) des Lösungsvorschlags und der Umsetzung in die Syntaxdiagramm-Notation.

(M)	i.	Vervollständige das Syntaxdiagramm für E-Mail-Adressen (Syntaxdiagramm fragmentarisch vorgeben).	(3)
	ii.	Benenne die Komponenten des Syntaxdiagramms (Syntaxdiagramm vollständig vorgeben).	(1)
(B)	a)	Überprüfe dein Syntaxdiagramm anhand der Beispiele `p@chwork.de` (gültig) `info@gymnasium.schulzentrum` (nicht gültig) `1.2.3@zauber.schule.de` (gültig) `4.5.6.@hexen.kessel.de` (nicht gültig)	(3)
	b)	Erkläre anhand der letzten beiden Adressen, wie dein Diagramm aufgebaut ist.	(3)
(D)	a)	Kreuze die zutreffenden Aussagen an: □ Vor „@“ stehen genau zwei Namen. □ Vor „@“ stehen beliebig viele Namen. □ Jeder Name beginnt mit einem Großbuchstaben. □ Ein Name besteht aus großen und kleinen Buchstaben in beliebiger Reihenfolge. □ „moritz“ ist ein gültiger Name. □ „BeNeDict“ ist ein gültiger Name.	(2)
	b)	Mit einer Ziffer nach dem Namen lassen sich Namenskonflikte vermeiden: `CarolineBauer1@..., CarolineBauer2@...` Ergänze das Syntaxdiagramm entsprechend.	(4)
(Z)		Verwende die Stichwörter „Grammatik“, „Syntax“, „Semantik“, „Versprecher“, „Tippfehler“.	(5)

Abbildung 4: Hilfestellungen zur Identifikation von Kompetenzstufen

Der Vergleich künstlicher und natürlicher Sprachen unter Bezugnahme auf das E-Mail-Beispiel in Aufgabe (Z) ist mit der Verwendung sinnvoller Kriterien und der Beurteilung ihrer Erfüllung eine Bewertungs-Leistung (6). Werden in der Hilfestellung Stichwörter, die auf Gemeinsamkeiten und Unterschiede hinweisen, vorgegeben, ist die geforderte Leistung die Synthese (5) zu einer schlüssigen Argumentation.

7 Zusammenfassung und Ausblick

In dieser Arbeit wurde begründet, dass die Situation der Theoretischen Informatik in der Schule in besonderem Maße die Entwicklung von Bildungsstandards erfordert. Diese sind unverzichtbar zur Orientierung der Lehrenden, um im Spannungsfeld zwischen theoretischer Fundierung und Benutzerschulung eine balancierte Auswahl der Unterrichtsinhalte zu unterstützen sowie um vorhandene Vermittlungskonzepte und Unterrichtsmaterialien einzuordnen und besser in die Unterrichtspraxis einzubinden. Ein erster Ansatz zur Gewinnung von Kompetenzen und Kompetenzstufen als Wesenskern von Bildungsstandards wurde vorgestellt.

Als nächste Schritte der Entwicklung eines Kompetenzmodells der TI in der Sekundarstufe sind folgende Maßnahmen geplant: Eine Studie der Gestaltungsprinzipien erprobter und etablierter Kompetenzmodelle und ihrer Übertragbarkeit auf die TI ist zu erstellen. Einzubeziehen sind vor allem solche Modelle, die inhaltlich dem abstrakten Wesen der TI entsprechen und die die Sekundarstufe betreffen. Des Weiteren sind die Fachinhalte anhand geeigneter Kriterien auf ihre Schulrelevanz hin zu untersuchen und entsprechend auszuwählen. Geeignete Kriterien sind beispielsweise die Fundamentalitätskriterien [Sc93], insbesondere der Lebensweltbezug, die Eignung für die Vermittlung in der Sekundarstufe, mögliche Anknüpfungspunkte und Vernetzungsmöglichkeiten mit anderen Unterrichtsinhalten, das Vorhandensein von Vermittlungskonzepten, Lernhilfen und Unterrichtsmaterialien. Die ausgewählten Fachinhalte und die damit verbundenen Lernziele sind unter Berücksichtigung im Unterricht bewährter Vermittlungsreihenfolgen strukturiert darzustellen. Ein erster Ansatz hierzu wurde in Kapitel 6 beschrieben.
Die abstrakten und anspruchsvollen Inhalte der TI erfordern eine sensible Auswahl unterrichtsgeeigneter Aufgaben. Dazu werden in Schulbüchern und Schülerwettbewerben, wie dem Bundeswettbewerb Informatik, vorhandene Aufgaben auf ihren abstrakten Kern (vgl. [Br04]) zurückgeführt, sodass eine Auswahl, etwa nach der Relevanz der Fachinhalte, getroffen werden kann. Im Hinblick auf die Entwicklung eines Kompetenzmodells ist eine Klassifizierung der Aufgaben nach geeigneten Ordnungskriterien, zum Beispiel. Anforderungsbereich, Formalisierungs- oder Schwierigkeitsgrad, erforderlich.
Auf der Basis der klassifizierenden Merkmale, die die Schwierigkeit von Aufgaben charakterisieren, werden Hypothesen über die Dimensionen und Niveaustufen für den ersten Entwurf eines Kompetenzmodells formuliert. Wie in Kapitel 6 anschaulich dargestellt und diskutiert, werden Testaufgaben mit Hilfestellungen entlang der charakterisierenden Merkmale zur Operationalisierung des Modells konstruiert.
Zur empirischen Überprüfung werden in einer Feldstudie mit Schülerinnen und Schülern Testdaten erhoben, die die schriftlichen Bearbeitungsergebnisse der Aufgaben sowie die Informatik- und Mathematiknoten und eine kriterienorientierte Kompetenzeinschätzung der Probanden durch die Lehrperson umfassen. Die Testdaten werden mit statistischen Methoden ausgewertet und interpretiert. Im ersten Gang wird eine Faktorenanalyse durchgeführt, um die charakterisierenden Merkmale hypothesengemäß zu orthogonalen Faktoren, den postulierten Dimensionen, zu bündeln. Gegebenenfalls erfolgt eine Revision des Modells. Im zweiten Gang wird mit den orthogonalen Dimensionen aus der Faktorenanalyse eine Clusteranalyse der individuellen Kompetenzprofile durchgeführt, um Kategorien typischer Kompetenzprofile sichtbar zu machen. Zur Validierung der Kategorien wird die Korrelation mit der Kompetenzeinschätzung der Lehrperson und den Informatik- und Mathematiknoten herangezogen.

Literaturverzeichnis

[Ba03a] Back, W.: Der Know-how Computer. Ein Computer, der mit Streichhölzern arbeitet. http://www.wolfgang-back.com/knowhow_home.php, 2003; letzter Zugriff 23.01.2007.

[Ba03b] Baumert, J. et al.: PISA 2000 – Ein differenzierter Blick auf die Länder der Bundesrepublik Deutschland. Leske + Budrich, Opladen, 2003.

[BK02] Bayerisches Staatsministerium für Unterricht und Kultus (Hrsg.): Weiterentwicklung des mathematisch-naturwissenschaftlichen Unterrichts. Erfahrungsbericht zum BLK-Programm SINUS in Bayern. München, 2002.

[Br04] Brinda, T.: Didaktisches System für objektorientiertes Modellieren im Informatikunterricht der Sekundarstufe II. Dissertation, Fachbereich für Elektrotechnik und Informatik, Universität Siegen, 2004.

[Br06] Breier, N.; Brinda, T.; Fothe, M.; Friedrich, S.; Koerber, B.; Puhlmann, H.: Neuer Wein in alten Schläuchen? In: Computer und Unterricht 16 (2006) 63; S. 14 – 15.

[De95] Dewdney, A.: Der Turing Omnibus. Eine Reise durch die Informatik mit 66 Stationen. Springer, Berlin, 1995.

[Fo05] Fothe, M.: Rekursion. Ein Thema für den Informatikunterricht. In: LOG IN 25 (2005) 133; S. 46 – 54.

[Fr03] Friedrich, S.: Informatik und PISA – vom Wehe zum Wohl der Schulinformatik. In: Hubwieser, P. (Hrsg.): Informatische Fachkonzepte im Unterricht. Köllen, Bonn, 2003; S. 133 - 144.

[GH98] Gal-Ezer, J., Harel, D.: What (Else) Should CS Educators Know? In: Communications of the ACM 41 (1998) 9; pp. 77 – 84.

[Hu03] Hubwieser, P.: Didaktik der Informatik. Springer, Berlin, 2003.

[HW06] Hielscher, M.; Wagenknecht, C.: AtoCC: Learning environment for teaching theory of automata and formal languages. In: ACM SIGCSE Bulletin 38 (2006) 3, ACM Press, New York, 2006; p. 306.

[KK04] Sekretariat der Ständigen Konferenz der Kultusminister der Länder i. d. Bundesrepublik Deutschland (Hrsg.): Einheitliche Prüfungsanforderungen in der Abiturprüfung Informatik, Beschluss vom 1.12.1989 i. d. F. vom 5.2.2004. Wolters Kluwer, München, 2004.

[Kl03] Klieme, E. et al.: Expertise. Zur Entwicklung nationaler Bildungsstandards. Bonn, 2003.

[Lo05] Loidl, S.; Mühlbacher, J.; Schauer, H.: Preparatory Knowledge: Propaedeutic in Informatics. In: Mittermeir, R. (Ed.): From Computer Literacy to Informatics Fundamentals. Springer, Berlin, 2005; pp. 104 – 115.

[Ma05] Magenheim, J.: Towards a Competence Model for Educational Standards of Informatics. In: Proc. of the 8th IFIP WCCE, Cape Town, 2005.

[Me04] Meyerhöfer, W.: Zum Kompetenzstufenmodell von PISA. In: Journal für Mathematik-Didaktik 25 (2004) 3/4; S. 294 – 305.

[Ni95] Nievergelt, J.: Welchen Wert haben theoretische Grundlagen für die Berufspraxis? Gedanken zum Fundament des Informatik-Turms. In: Informatik Spektrum 18 (1995) 6; S. 342 – 344.

[Pu03] Puhlmann, H.: Informatische Literalität nach dem PISA-Muster. In: Hubwieser, P. (Hrsg.): Informatische Fachkonzepte im Unterricht. Köllen, Bonn, 2003; S. 145 - 154.

[Re03] Reichert, R.: Theory of Computation as a Vehicle for Teaching Fundamental Concepts of Computer Science. Dissertation 15035, ETH Zürich, 2003.

[Sc93] Schwill, A.: Fundamentale Ideen der Informatik. In: Zentralblatt für Didaktik der Mathematik 25 (1993) 1; S. 20 – 31.

[Sc01] Schöning, U.: Theoretische Informatik – kurzgefasst. Spektrum, Heidelberg, 2001.

[Sc05] Schubert, S.: From Didactic Systems to Educational Standards. In: Proc. of the 8th IFIP World Conference on Computers in Education, Cape Town, 2005.

[WS06] Witten, H.; Schulz, R.: RSA & Co. in der Schule. Moderne Kryptologie, alte Mathematik, raffinierte Protokolle. In: LOG IN 26 (2006) 140; S. 45 – 54.

Vergleichende Analysen zweier Problemlöseprozesse unter dem Aspekt des Problemlöseerfolgs

Bertold Kujath

Didaktik der Informatik
Universität Potsdam
August-Bebel-Straße 89
14482 Potsdam
kujath@uni-potsdam.de

Abstract: Problemlöseerfolg bei Informatikaufgaben ist nicht nur das Ergebnis von vorher erworbenem Wissen und Übungspraxis, sondern auch der Aktivitäten, die während der Aufgabenbearbeitung entwickelt werden. In der hier vorliegenden Studie sollen effiziente mit ineffizienten Problemlöseprozessen verglichen und Unterschiede herausgearbeitet werden. Ausgehend von der Annahme, bei informatischen Hochleistern häufiger effiziente Problemlöse-Prozesse nachweisen zu können, wurden im ersten Teil der Studie acht Endrundenteilnehmer des Bundeswettbewerbs Informatik beim Problemlösen mit der Methode des lauten Denkens beobachtet. Eine Versuchsreihe mit weniger erfolgreichen Problemlösern als Vergleichsgruppe steht unmittelbar bevor. Da bereits innerhalb der Gruppe der Hochleister deutliche Unterschiede in der Effizienz der Problemlöseprozesse beobachtet werden konnten, soll hier exemplarisch für die Vorgehensweise in der Gesamtstudie die Bearbeitungen eines Färbeproblems durch zwei der Probanden kontrastiert werden.

1 Einleitung

Die Frage, worin sich die Problemlöseprozesse Hochleistender von denen Niedrigleistender unterscheiden und welche unterschiedlichen Problemlöse- und Denkstrategien hierbei zum Einsatz kommen, soll mit der Methode des lauten Denkens untersucht werden. Bei dieser in der Problemlöseforschung seit langem etablierten, wenn auch kontrovers diskutierten Methode, werden die Probanden gebeten, beim Lösen von Informatikaufgaben alle Gedankengänge laut auszusprechen. Studien zur Validität dieses Datenerhebungsverfahrens belegen, dass diese Art des Problemlösens bei Beachtung gewisser limitierender Faktoren den Problemlöseprozess nicht qualitativ verändert [NS72, De84, ES93].

Versuche mit lautem Denken laufen in der Problemlöseforschung üblicherweise als sog. Kontraststudien ab, bei denen in der Regel zwei Extremgruppen hinsichtlich bestimmter Kriterien miteinander verglichen werden. Im hier vorliegenden Ansatz sollen Kontrast-

gruppen gleichen Wissensstands untersucht werden, die sich in ihrer Problemlöseleistung deutlich unterscheiden. Um eine vergleichbare Ausgangssituation herzustellen, sollen beide Gruppen während des Erarbeitens von Lösungswegen unbekannter Aufgabentypen beobachtet werden. Da diese Aufgabentypen in der Regel weder wissens- noch könnensbasiert gelöst werden, zielen sie auf Unterschiede in den kognitiven Fähigkeiten ab. Dies soll auch durch den Vergleich der Lösungen mit den Ergebnissen eines Intelligenzstrukturtests verdeutlicht werden.

Sollten die Untersuchungen ergeben, dass charakteristische Problemlösestrategien der Hochleister ursächlich für den höheren Problemlöseerfolg und als Strategiemuster beschreibbar sind, soll in einem weiteren Schritt geprüft werden, inwieweit diese Strategien an Niedrigleister vermittelt werden können.

Unter den Vorarbeiten anderer Autoren zu ähnlichen Fragestellungen sind besonders die folgenden zu erwähnen: R. Borromeo Ferri [Bo04] forschte zu mathematischen Denkstilen, indem sie 12 Schüler der 6. und 9. Jahrgangsstufe beim paarweisen Lösen mathematischer Aufgaben beobachtete. Sie identifizierte bei ihnen drei bereits von [Bu97] bei praktizierenden Mathematiklehrern und –lehrerinnen beschriebene mathematische Denkstile durch qualitative Datenanalyse. Da in unserer Studie ein direkter interindividueller Vergleich angestrebt ist, werden die Versuche in Einzelsitzungen durchgeführt.

G. Friege [Fr01] führte Experten-Novizen-Vergleiche beim Lösen elektrotechnischer Aufgaben durch. Die Auswertung der Problemlöseprozesse basierte auf den schriftlichen Lösungswegen der Versuchspersonen, die Ergebnisse wurden mit dem Berliner Intelligenzstrukturtest (BIS) abgeglichen. Vorangegangen waren Voruntersuchungen mit Teilnehmern der Physik-Olympiade. In unserer Studie sollen unmittelbar an die Bearbeitung der Probleme anschließende retrospektive Interviews zum Problemlöseerleben durchgeführt werden, um zusätzliche Information über die individuellen Strategien und Entscheidungsgründe zu gewinnen.

2 Hypothesen

Erfolgreiche, d.h. schnelle und effiziente Problemlöser lösen Informatikaufgaben mit adäquaten informatischen und allgemeinen Problemlösemethoden, Niedrigleister tun dies nicht oder nicht in ausreichendem Maße. In Anlehnung an die Fundamentalen Ideen der Informatik [SS04] kann man also in den Problemlöseprozessen informatischer Hochleister Prinzipien des Algorithmisierens, des strukturierten Zerlegens und sprachlicher Konzepte erwarten. Ebenso sind die Methoden des allgemeinen Problemlösens wie Zerlegen in Teilprobleme, planvolles Hypothesenbilden und –testen bei Hochleistern besser ausgeprägt. Als Konsequenz daraus sind die Problemlösestrategien von informatischen Hochleistern in der Regel effizienter, schneller und führen häufiger zu richtigen Ergebnissen. Bei der Analyse von Verbalprotokollen werden sich daher verstärkt Kategorien des allgemeinen und des informatikspezifischen Problemlösens in den Problemlöseprozessen Hochleistender identifizieren lassen. Erste Anhaltspunkte dafür gab es durch die

Ergebnisse einer im März 2006 abgeschlossenen Pilotstudie mit Studenten und wissenschaftlichen Mitarbeitern im Bereich Informatik der Universität Potsdam.

3 Probandenauswahl

Für den hier beschriebenen ersten Teil Studie wurden Probanden für die Gruppe der Hochleister aus Endrundenteilnehmern des Bundeswettbewerbs Informatik der letzten drei Jahre angeworben. Insgesamt acht dieser Endrundenteilnehmer nahmen bis Dezember 2006 an den Versuchen teil, sieben Versuchsteilnehmer waren zudem Bundessieger. Bei allen acht Teilnehmern wurde ein IQ von z.T. deutlich über 130 ermittelt, was allgemein als Schwellenwert für eine psychometrische Hochbegabung angesehen wird [WW90]. Da alle diese Testteilnehmer durch Erreichen der Endrunde des Bundeswettbewerbs offenbar zugleich über eine Leistungsexzellenz im Fach Informatik verfügen, erschienen sie als besonders geeignet für die Gruppe der Hochleister in dieser Studie. Die Auswahl der Probanden für die Gruppe der Niedrigleister ist zum aktuellen Zeitpunkt noch nicht abgeschlossen.

Der hier vorgestellte erste Teilnehmer T10 war zum Zeitpunkt des Versuchs 18 Jahre alt und hatte parallel zum Besuch der gymnasialen Oberstufe ein Informatik-Vordiplom absolviert. Der zweite Teilnehmer T16, ebenfalls zum Zeitpunkt des Versuchs 18 Jahre alt, hatte neben dem Besuch des Gymnasiums ein Praktikum an einem Institut für Informatik gemacht. Beide Teilnehmer haben mehrfach an nationalen oder internationalen Informatikwettbewerben teilgenommen. Diese beiden Probanden zeigten bei dem hier vorgestellten Färbeproblem die größten Leistungsunterschiede innerhalb der Gruppe, weshalb sie für einen direkten Vergleich geeignet erschienen.

4 Versuchsbeschreibung

4.1 Versuchsablauf und Datenauswertung

Die Datenerhebung und die Datenauswertung soll hier nur verkürzt dargestellt werden, eine genauere Beschreibung des Versuchsablaufs findet sich bei [Ku06].

Beiden hier besprochenen Probanden wurden, wie auch den übrigen Teilnehmern aus der Gruppe der Hochleister, bis zu acht schulbuchtypische Informatikaufgaben präsentiert, mit der Maßgabe, diese unter lautem Denken zu lösen. Interventionen seitens des Versuchsleiters beschränkten sich während des Problemlösens auf das Notwendige. Der gesamte Problemlöseprozess wurde mit einer Videokamera aufgezeichnet, die Verbalisierungen später zunächst wörtlich transkribiert und mit Zeitstempeln versehen. In einem weiteren Schritt wurden diese Transkriptionen unter Anwendung spezieller Textreduktionsoperatoren semantikerhaltend verkürzt und auf ein einheitliches Sprachniveau transformiert, um anschließend auffällige Textpassagen in ex-ante-Problemlösekategorien

einordnen zu können. Eine detaillierte Beschreibung dieser Kategorien findet sich ebenfalls in [Ku06].

4.2 Aufgabe

Zunächst soll zum besseren Verständnis die von den beiden Teilnehmern bearbeitete Aufgabe vorgestellt und ihre aufgabenspezifischen Problemlöse-Kategorien sowie die einzelnen Teillösungen kurz diskutiert werden. Der Aufgabentext, der wörtlich vorgelesen wurde und jedem Probanden während der gesamten Aufgabenbearbeitung zur Verfügung stand, befindet sich in Abb. 1.

Weiterhin sei zur Verdeutlichung der Aufgabenstellung in Abb. 2. das Beispiel aus der Aufgabe dargestellt, links die Ausgangskonfiguration der unteren Reihe, rechts die erste der sieben aufgeführten Färbemöglichkeiten für oben. Sämtliche Versuchspersonen die Aufgabe von links nach rechts betrachtet, daher werden die Felder stets von links nach rechts durchgezählt.

3-Färbung eines $2 * n$-Rechtecks

In einem $2 * n$-Rechteck soll jedes $1 * 1$-Quadrat gefärbt werden. Dabei sollen an den Kanten zusammenliegende Quadrate unterschiedliche Farben haben, insgesamt gibt es drei verschiedene Farben: weiß, grau und schwarz. Die untere Hälfte des längsliegenden Rechtecks ist bereits gefärbt, und zwar mit der Farbensequenz $C_1,...,C_n$. Wieviele unter feldes und dem Feld direkt darunter abhängt; im folgenden wird dieser Sachverhalt als schiedliche Möglichkeiten gibt es nun, die obere Hälfte zu färben? Diese Anzahl hängt von der Farbsequenz $C_1,...,C_n$ der unteren Hälfte ab.

Beispiel:
Sei n = 4 und die untere Sequenz sei <schwarz, weiß, schwarz, grau>. Dann gibt es in diesem Fall insgesamt sieben verschiedene Arten, die obere Hälfte korrekt zu färben:

<weiß, grau, weiß, schwarz>
<weiß, schwarz, grau, weiß>
<weiß, schwarz, grau, schwarz>
<weiß, schwarz, weiß, schwarz>
<grau, schwarz, grau, weiß>
<grau, schwarz, grau, schwarz>
<grau, schwarz, weiß, schwarz>

Sieben ist allerdings weder die maximale noch die minimale Anzahl der Möglichkeiten....

1. Welche ist die minimale und welche die maximale Anzahl von Möglichkeiten? (n ist dabei konstant, die Beschaffenheit der Farbsequenzen kann variieren).
2. Wie muss die untere Farbsequenz beschaffen sein, sodass man zum einen die minimale und zum anderen die maximale Anzahl von Möglichkeiten oben hat?

Abbildung 1: Aufgabentext des hier beschriebenen Färbeproblems

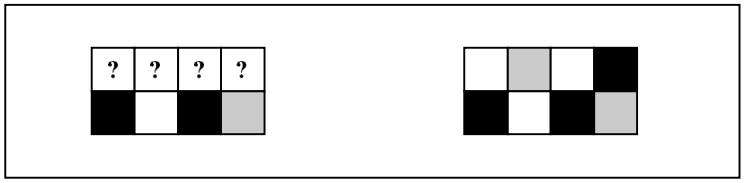


Abbildung 2: Erstes Färbebeispiel aus der Aufgabenstellung für n = 4

Wichtig für die Aufgabenbearbeitung ist die Schlüsselerkenntnis, dass die Anzahl der Färbemöglichkeiten eines beliebigen Feldes oben von den Farben des linken Vorgänger Diagonalkriterium bezeichnet. Sind in beiden Feldern die Farben gleich, hat man im zu färbenden Feld zwei Färbemöglichkeiten, andernfalls nur eine. Die maximale Anzahl von Färbemöglichkeiten erhält man also, wenn das Diagonalkriterium gleich möglichst oft erfüllt ist. Analog dazu erhält man die minimale Anzahl, wenn das Diagonalkriterium ungleich möglichst oft erfüllt ist.

Die Gesamtlösung der Aufgabe besteht aus 4 Teillösungen. Um eine Formel zu entwickeln, aus einem gegebenen n die minimale und die maximale Anzahl der Färbemöglichkeiten errechnen zu können, müssen zunächst die jeweiligen Konfigurationen für unten einerseits für den minimalen und andererseits für den maximalen Fall gefunden werden. Unter Berücksichtigung des oben geschilderten Diagonalkriteriums ergibt sich somit für den Minimalfall eine Konfiguration für unten, die alle drei Farben in Folge hintereinander enthält, z.B. weiß, schwarz grau, weiß, schwarz, grau usw. Die Anzahl der Färbemöglichkeiten für beliebige n errechnet sich dann nach der Formel n + 1. Um die maximale Anzahl an Färbemöglichkeiten oben zu erhalten, dürfen für die Farbkonfiguration unten nur zwei Farben im Wechsel verwendet werden, beispielsweise grau, weiß, grau, weiß usw., dann kann in jedem Feld zumindest schwarz verwendet werden. Die maximale Anzahl der Färbemöglichkeiten bei n Feldern ist Fibonaccizahl von n, also F(n).

Zusammengefasst stellen sich die Schlüsselerkenntnisse und die Teillösungen, also die aufgabenspezifischen Kategorien wie folgt dar:

- Diagonalkriterium gleich ergibt zwei Färbemöglichkeiten im Feld oben
- Diagonalkriterium ungleich ergibt nur eine Färbemöglichkeit im Feld oben
- Erstes Feld hat immer zwei Färbemöglichkeiten.
- Teillösung 1: Farbkonfiguration unten für den minimalen Fall: alle drei Farben immer abwechselnd.
- Teillösung 2: Formel für den minimalen Fall: n +1
- Teillösung 3: Farbkonfiguration unten für den maximalen Fall: nur zwei Farben immer im Wechsel
- Teillösung 4: Formel für den maximalen Fall: F(n)

5 Bisherige Ergebnisse der Hauptstudie

Es sollen nun die Problemlöseprozesse der Probanden T10 und T16 vorgestellt werden, einige Verbalisierungen werden zur Verdeutlichung an den entsprechenden Stellen wörtlich und mit Zeitangabe wiedergegeben.

5.1 Allgemeine Beschreibung T10

Sämtliche vier Teillösungen werden von dieser Versuchsperson in 13:08 Minuten gefunden, selbst innerhalb der Gruppe der Hochleister eine Rekordzeit. T10 beginnt seinen hochstrukturierten und nahezu sackgassenfreien Problemlöseprozess unmittelbar nach dem Vorlesen der Aufgabe mit der Problemanalyse, eine Problemrepräsentation ist nicht erkennbar. Der Einfluss des Diagonalkriteriums auf die Anzahl der Färbemöglichkeiten wird von T10 nach fünf Sekunden formuliert, wenig später auch der Zusammenhang zwischen gleicher bzw. ungleicher diagonaler Färbung und der daraus resultierenden Anzahl möglicher Farben im Feld darüber.

T10 wendet sich dann der Teillösung „Minimaler Fall“ zu, in der in Abb. 3 dargestellten ersten Skizze des Probanden wird ein minimaler Fall unter konsequenter Anwendung des Diagonalkriteriums ungleich in einem enaktiven Prozess entwickelt, d.h., die Farbkonfiguration für die untere Reihe wird sukzessiv konstruiert. T10 verwendet dabei die Zahlen 0, 1, 2 anstelle von Farbbezeichnungen, die Skizze wirkt dadurch prägnanter.

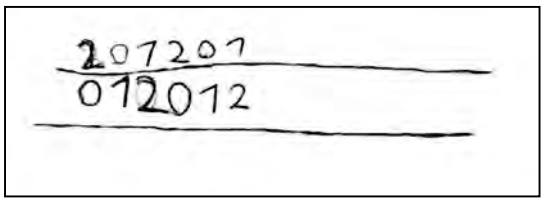


Abbildung 3: Skizze des Probanden T10 zur Entwicklung der unteren Farbkonfiguration im minimalen Fall

T10 erkennt in dieser Phase, dass – die untere Farbkonfiguration für den minimalen Fall vorausgesetzt – bei Erfüllung des Diagonalkriteriums ungleich im ersten Feld die weiteren Farben oben determiniert sind (vgl. Abb. 3).

Die nun folgende Vorgehensweise von T10 erinnert an die fundamentale Idee der Parametrisierung. Für den Fall, dass man im ersten Feld das Diagonalkriterium gleich erfüllt, ergeben sich im zweiten Feld oben zwei Färbemöglichkeiten, und diese Situation wiederholt sich für die nachfolgenden Felder, solange man im jeweils aktuellen Feld das Diagonalkriterium gleich erfüllt. Wählt man im zweiten Feld das Diagonalkriterium ungleich, so ist von da ab die Folge determiniert, denn es bestehen dann keine Farbwahlmöglichkeiten mehr für irgendeines der folgenden Felder.

Zitat des Probanden (sic!):

5:03 „Den Extremfall habe ich also entweder die Möglichkeit, gleich mit zu machen, oder erst beim zweiten Mal, oder erst beim dritten Mal, oder erst beim vierten Mal."

Aus dieser Erkenntnis leitet T10 dann auch die korrekte Formel für die Anzahl der Färbemöglichkeiten im minimalen Fall ab, womit die beiden Teillösungen für den minimalen Fall gefunden sind.

Zitat des Probanden:

5:46 „Das heißt, das minimale Beispiel dürfte... n+1 Möglichkeiten haben. So, und zu konstruieren als eine Folge von Farbe 1, Farbe 2, Farbe 3 und das wiederholt sich dann immer."

Ebenso konsequent geht T10 bei der Bearbeitung des maximalen Falls vor. Durch Kenntnis der Eigenschaften der unteren Farbsequenz für den minimalen Fall, gelingt es T10, auf Anhieb die korrekte untere Farbsequenz für den maximalen Fall zu deduzieren.

Es folgt die fundamentale Idee der strukturierten Zerlegung in Form eines Graphen, um die Anzahl der Färbemöglichkeiten im Maximalfall zu quantifizieren. Zunächst als Binärbaum geplant, wird von T10 die in Abb. 4 dargestellte Struktur entwickelt, indem jeweils gleiche Zustände einer Ebene des Binärbaumes zu einem Zustand zusammengefasst werden. Jede Ebene entspricht dabei einem zu färbenden Feld der oberen Reihe, jeder Knoten einer möglichen Farbe in diesem Feld. Die Anzahl der ausgehenden Kanten eines Knotens korrespondiert mit der Anzahl der Färbemöglichkeiten im folgenden Feld. Die Gesamtzahl der Färbemöglichkeiten für ein bestimmtes n ergibt sich aus der Summe aller zu dieser Ebene führenden Pfade, die sich wenigstens in einer Kante unterscheiden.

Zitat des Probanden:

8:47 „Ich schreibe mal ran, wie viele Möglichkeiten es gibt... so viele Pfade gibt es hier durch, auf jeder Ebene..."

Aufbauend auf dieser Erkenntnis und mithilfe der Skizze aus Abb. 4 entwickelt T10 letztendlich die rekursive Formel zur Berechnung der Färbemöglichkeiten für beliebigen im maximalen Fall, nämlich F(n).

Zitat des Probanden:

11:49 „...es gibt zum Beispiel hier zwei Wege und hier einen Weg... bei zwei... bei Dreien... gibt`s... ähm... hier... drei Wege und hier zwei."

12:17 „Das heißt, ähm... die Möglichkeiten bei *n* Zahlen... ist gleich Fibonaccizahl..."

Im Anschluss an die Lösungsfindung erfolgte noch eine empirische Lösungsprüfung der jeweiligen Formeln durch Beispielzahlen, die Lösung wurde von T10 als richtig befunden.

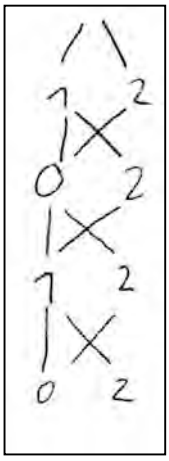


Abbildung 4: Graph zur Ermittelung der Anzahl der Färbemöglichkeiten im Maximalfall

5.2 Allgemeine Beschreibung T16

T16 beginnt seine Problembearbeitung mit einer kurzen Frage zum Aufgabenverständnis. In der sich anschließenden intensiven Problemanalyse formuliert T16 die theoretische Obergrenze 2^n an Färbemöglichkeiten, ausgehend von maximal zwei zulässigen Farben pro Feld.

Zitat des Probanden:

0:27 „O.K. Also.... hab` ich natürlich in der... in der oberen Hälfte, habe ich n Quadrate zu färben, für jedes davon kommen theoretisch erst einmal zwei davon in Betracht, nämlich die beiden, die in dem direkt darunter liegenden Quadrat noch nicht gewählt sind."

Es folgt die Schilderung des Diagonalkriteriums gleich und ungleich, zunächst in allgemeiner Form, dann aus Gründen der Selbsterklärung anhand konkreter Farbbeispiele. Unmittelbar darauf erkennt T16, dass im ersten Feld oben grundsätzlich zwei Färbemöglichkeiten bestehen. Korrekt ist auch die Aussage gegen Ende der Problemanalyse, dass man die maximale Anzahl von Färbemöglichkeiten erhält, wenn das Diagonalkriterium gleich so oft wie möglich erfüllt wird, ebenso wie die korrespondierende Aussage zur minimalen Anzahl von Färbemöglichkeiten.

Es folgt eine Formulierung, die als beabsichtigter Beginn der Problembearbeitung angesehen werden kann.

Zitat des Probanden:

5:40 „Also, ich färbe jetzt von links nach rechts oben diese Reihe mal sequentiell durch...“

Allerdings beginnt T16 nun, die bisherigen Erkenntnisse über das Diagonalkriterium sowie die Bedingungen aus der Aufgabe in unterschiedlichen Formen informatisch zu repräsentieren. Dabei greift er auf eine programmiersprachennahe Notation zurück. Wie aus Abb. 5 ersichtlich ist, fasst er als erstes seinen bisherigen Kenntnisstand über das Problem in seiner ersten Skizze zusammen.

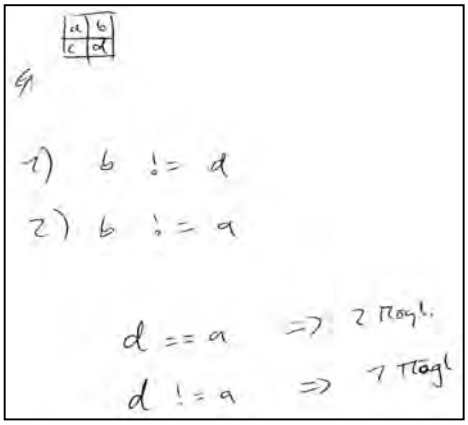


Abbildung 5: Skizze zur informatischen Repräsentation von T16

Die darauf folgende Skizze ist vom Inhalt her nahezu identisch , nur diesmal mit indizierten Feldbezeichnungen: untere Reihe a_1 bis a_n, obere Reihe b_1 bis b_n. Als T16 kurze Zeit später merkt, dass er mit diesem Ansatz nicht weiterkommt, vollzieht er einen Strategiewechsel.

Zitat des Probanden:

13:24 „Also, ob das so in der Form, wie ich das jetzt gemacht hab, ob das so toll ist, oder ob man nicht lieber die... die untere Sequenz, die Farbabfolge da an sich erst mal betrachtet, was es da für... für Möglichkeiten gibt, also, das jetzt eher an Beispielen durchgeht.“

Zu diesem Zeitpunkt hatte T10 bereits die vollständige Lösung gefunden. Nachdem T16 bis hierher ausschließlich auf einer abstrakten Ebene gearbeitet hat, versucht er nun einen konkreten Fall zu modellieren, dargestellt in Abb. 6. Dabei formuliert er korrekt die weiter oben als Teillösung 3 bezeichnete Farbkonfiguration für unten für den maximalen Fall.

Nur kurze Zeit später erfolgt ein erneuter Strategiewechsel, T16 wendet sich wieder der formalen Betrachtungsweise zu.

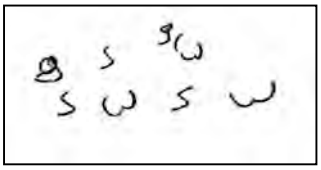


Abbildung 6: Skizze von T16 zur Modellierung eines konkreten Färbebeispiels

Zitat des Probanden:

16:18 „Aber, wahrscheinlich probiere ich das doch noch mal so hier... das so wirklich formal weiter... weiter zu machen."

Abb. 7 zeigt die Skizze, die T16 in dieser Phase anfertigt, mit Ausnahme einer Umgestaltung der Repräsentation hat sich nichts geändert.

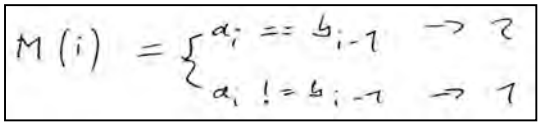


Abbildung 7: Rückkehr von T16 zur Repräsentationsphase

Diese Funktion soll nun rekursiv ausgedrückt werden, dies versucht T16 in der folgenden Phase mithilfe eines Entscheidungsbaums. T16 konstruiert diesen Baum bezogen auf die untere Farbfolge aus dem Aufgabenbeispiel, die bekanntermaßen weder die maximale noch die minimale Anzahl von Färbemöglichkeiten bewirkt. Kurze Zeit später wird auch die Konstruktion des Entscheidungsbaums wieder abgebrochen.

Zitat des Probanden:

25:36 Unter Umständen ist die... die Herangehensweise, das oben der Reihe nach auszufüllen... ist die nicht... nicht richtig."

T16 erklärt dann, Schwierigkeiten zu haben, einen formalen Ansatz zu finden, die Frage des Versuchsleiters nach den bisherigen Zwischenergebnissen wird ohne konkrete Aussage beantwortet.

6 Diskussion

Der Problemlöseprozess des Probanden T10 zeigt eine sehr frühe Ausrichtung der Aktivitäten auf Teilziele und Teillösungen. Die untere Farbsequenz für den Minimalfall wird in einer enaktiven Modellskizze unter konsequenter Umsetzung der aus der Problemanalyse stammenden Erkenntnis des Diagonalkriteriums ungleich konstruiert. Durch rationale Überlegung bildet er dann eine Hypothese über die allgemeine Formel des Minimal-

falles, die er anhand einer enaktiven Modellskizze überprüft. Folgerichtig geht T10 bei seinem Problemlöseprozess in einem Bottom-up-Ansatz stets vom Konkreten zum Allgemeinen, weshalb er die einzelnen Teillösungen scheinbar problemlos und ohne nennenswerte Sackgassensituationen der Reihe nach bearbeiten kann. T10 ist in der Lage, seine Strategien in unerwarteten Situationen zu adaptieren und braucht daher keine einmal begonnenen Lösungsansätze komplett zu verwerfen. Dies ist eine der Ursachen für seinen großen Zeitvorteil gegenüber anderen Problemlösern. Seine Skizzen haben einen hohen Informationsgehalt und sind gut strukturiert, was insbesondere im Fall der Skizze aus Abb. 4 zu einem leichteren Erkennen des zugrundeliegenden Bildungsprinzips führt. Im Gegensatz zu T16 finden sich bei T10 mehrere deutlich ausgeprägte fundamentale Ideen der Informatik wie die Parametrisierung, die strukturierte Zerlegung in Form eines Graphen und die Rekursion.

T16 beginnt sofort nach der Problemanalyse mit einem rein formalen Ansatz. Seine erste Skizze aus Abb. 5 ist aufgrund der wenig systematisierten Feldbezeichnungen a, b, c, d weder sehr aussagekräftig noch sinnvoll erweiterbar. Erst im zweiten Ansatz indiziert er Feldbezeichnungen, bleibt aber nach wie vor bei seiner nicht-enaktiven Modellierung. Erst viel später befasst er sich mit einem konkreten Einzelbeispiel und hat auch die richtige Farbfolge für den Maximalfall gefunden. Im Gegensatz zu T10 zieht er aber keinen Nutzen aus seinen Erkenntnissen, sondern verwirft sehr schnell die konkrete Herangehensweise um wieder auf der formalen Ebene zu arbeiten. Die offensichtliche Vorliebe dieses Teilnehmers zu rein formalen Betrachtungsweisen ist umso erstaunlicher, da er in der retrospektiven Befragung zu dieser Aufgabe seine prinzipiellen Schwierigkeiten mit formalen Sichtweisen schildert. Offenbar wählt T16 einen Bearbeitungsstil für diese Aufgabe, der ihm nach eigenen Angaben nicht liegt. Während des gesamten hier betrachteten Zeitraums trennt T16 die Aufgabenstellung nicht in Teilaspekte, sondern versucht in einer Top-down-Vorgehensweise jede neue Erkenntnis immer gleich für Maximal- und Minimalfall zu formulieren. Sein Leitgedanke ist, eine einzige Formel zu finden, die beide Fälle beschreibt. Letztlich hindert ihn das an einem wirklichen Fortschritt in der Problembearbeitung, da beide Teillösungen unterschiedlich bearbeitet wer-

Färbungsproblem	**T10**	**T16**
grundsätzlicher Ansatz	Bottom-up	Top-down
primäre Vorgehensweise	enaktiv	nicht enaktiv
Strategiewechsel	selten	häufig
Teilzielbildung	konsequent	nur ansatzweise
Skizzen	aussagekräftig	weniger aussagekräftig
Umsetzung gewonnener Erkenntnisse	vollständig	unvollständig
Fundamentale Ideen der Informatik	deutlich ausgeprägt	kaum vorhanden

Tabelle 1: Gegenüberstellung wesentlicher Merkmale der beiden Problemlöseprozesse

den müssen. In der nun folgenden Tabelle 1 sollen die wesentlichen Merkmale der beiden Problembearbeitungen gegenübergestellt werden.

7 Ausblick

Die Tatsache, dass Hochleister im Bereich Informatik, wie in diesem Beispiel gezeigt, durchaus auch ineffiziente Problemlöse-Prozesse beim Bearbeiten fachspezifischer Aufgaben verfolgen, macht deutlich, dass Problemlösen eine eigenständige Kompetenz darstellt, die mit spezifischen didaktischen Methoden vermittelt werden muss. Erste Erkenntnisse können allerdings nur bezogen auf die beiden hier geschilderten Fälle formuliert werden. Offensichtlich arbeiten erfolgreiche Problemlöser mit konkreten Anfangsbeispielen, die sie mit hoher Zielgerichtetheit bis zur fertigen Lösung weiter entwickeln. Ebenso sind erfolgreiche Problemlöser in der Lage, gewonnene Erkenntnisse nutzbringend umzusetzen. Weniger erfolgreiche Problemlöser scheinen ihre Lösungsansätze nicht konsequent genug zu verfolgen und wechseln in unerwarteten Situationen lieber ihre Strategie, als die gerade verwendete zu adaptieren. Fundamentale Ideen der Informatik finden sich deutlich ausgeprägter und konsequenter verfolgt in den Lösungswegen erfolgreicher Problemlöser. Eine didaktische Konsequenz daraus könnte sein, informatische Prinzipien nicht isoliert, sondern im konkreten Problemlösekontext zu vermitteln.

Literaturverzeichnis

[Bo04] R. Borromeo Ferri, Mathematische Denkstile, Franzbecker, 2004

[Bu97] L. Burton, Mathematicians and their Epistemologies – and the Learning of Mathematics, in: Inge Schwank, (Ed.), European Research in Mathematics Education Vol I, Osnabrück: Forschungsinstitut für Mathematikdidaktik, p. 87-102, 1999

[De84] G. Deffner, Lautes Denken - Untersuchungen zur Qualität eines Datenerhebungsverfahrens, Verlag Peter Lang, 1984

[ES93] K. A. Ericsson, H. A. Simon, Protocol Analysis, MIT Press, 1993

[Fr01] G. Friege, Wissen und Problemlösen, Logos Verlag, 2001

[Ku06] B.Kujath, Ein Test- und Analyseverfahren zur Kontrastierung von Problemlöseprozessen informatischer Hoch- und Niedrigleister – erste Ergebnisse einer Pilotstudie, GI-Edition-Lecture Notes in Informatics (LNI), 2006

[NS72] A. Newell, H.A. Simon, Human Problem Solving, Prentice-Hall, 1972

[SS04] S. Schubert, A. Schwill, Didaktik der Informatik, Spektrum Akademie Verlag, 2004

[WW90] M.R. Waldmann, F.E. Weinert, Intelligenz und Denken, Hogrefe 1990

Lesen im Informatikunterricht

Carsten Schulte

Didaktik der Informatik
Freie Universität Berlin
Takustr. 9
14195 Berlin
schulte@inf.fu-berlin.de

Abstract: Dass im Informatikunterricht Programme geschrieben werden, ist nicht ungewöhnlich. Ebenso werden Programme entworfen, getestet, analysiert und manchmal auch verändert und erweitert. Welche Rolle aber spielt das Lesen von Programmen?
Die These ist: Lesen ist zentraler Bestandteil der genannten Aktivitäten wie schreiben, analysieren, testen und erweitern. Lesen ist zudem Voraussetzung für das Verstehen von Programmen und – das ist der zentrale Punkt aus fachdidaktischer Sicht – Lesen ist eine recht gut beschreibbare Aktivität, sodass über den Zugriff des Lesens von Programmen lern- und leistungsdiagnostische Instrumente sowie Ansätze zu einer Theorie der Kompetenzentwicklung gefunden werden können. Solche Ansätze, ein Modell der Programmlesekompetenz sowie Anregungen für die Unterrichtspraxis werden im Artikel vorgestellt.

1 Programmieren lernen – Programmtexte lesen

In der informatikdidaktischen Literatur findet man kaum Treffer bei der Suche nach dem Begriff Lesen. Implizit jedoch wird das Lesen in verschiedenen informatikdidaktischen Ansätzen angesprochen: Im systemorientierten Ansatz wird vorgeschlagen, Systeme zu dekonstruieren. Die Autoren formulieren: „Software wird ‚interpretiert'" ([HMS99], S.161). Im informationszentrierten Ansatz gehört – ausgehend von der Feststellung, dass Information stets auf Repräsentation angewiesen ist (z.B.: [Hu99], [Br05]) – die „Interpretation von Repräsentationen" zum „inhaltlichen Kern des Schulfaches Informatik" ([Hu99], S. 168). Auch die Leitlinien informatischer Bildung der GI beziehen lesende Anteile ein: „Die Schülerinnen und Schüler eignen sich die Basiskonzepte ausgewählter Informatiksysteme durch Anwendung, Analyse, Modifikation und Bewertung an" ([GI00], S. 6). Dennoch finden sich kaum nähere Hinweise zum Lesen und Verstehen von Programmen. Nähere Angaben zum Lesen, etwa zu möglichen Lernhürden, Schwierigkeiten, Schwierigkeitsstufen, Übungen, Unterrichtsmethoden und zum Zusammenhang mit dem Schreiben von Programmen fehlen. Der vorliegende Artikel soll einen Beitrag leisten, diese Lücke zu schließen. Über den lesenden Zugang können möglicherweise didaktische Konzepte und Methoden zur Überprüfung von Lernfortschritten, Lernhürden bzw. dem aktuellen Leistungsniveau von Lernenden entwickelt werden.

Im Artikel wird nach der Diskussion verwandter Ansätze ein didaktisches Programmlesemodell vorgeschlagen, welches das verstehende Lesen von Programmtexten erklärt. Das Modell wird anschließend anhand der Analyse von Ergebnissen empirischer Studien aus dem Bereich Programmieren lernen, einigen unterrichtsmethodischen Vorschlägen für das Lesen von Quelltext und Konsequenzen für ein Kompetenzmodell auf seine Nützlichkeit untersucht.

2 Verwandte Ansätze

Verwandte Ansätze gibt es im Forschungsgebiet program comprehension, in Ergebnissen der informatikdidaktischen Forschung im Bereich Programmieren lernen sowie in der allgemeinen Theorie des Verstehens von W. Kintsch [Ki98].
Das Forschungsgebiet program comprehension (Programmverstehen, Softwarewartung) gibt es seit über 30 Jahren. Insbesondere sollen Wartungsarbeiten an (sehr großen) Softwaresystemen unterstützt werden. Forschungsansätze und -ergebnisse richten sich an Experten und nicht an Lernende (für einen Überblick siehe [MV94], [De01], [St05]). Die Arbeiten klären unter Anderem, welche Arten von Informationen aus einem Programmtext entnommen werden können bzw. müssen (etwa [Pe87], S. 298ff).
Einen Überblick zu informatikdidaktischen Forschungen zum Programmieren lernen bieten [SS89] und [RRR03]. Eine besondere Schwierigkeit betrifft das Verstehen des aus dem Programmtext resultierenden Ablaufs ([MR02] S. 55). Nach [LAJ05] fällt es Studenten schwer zu verstehen, dass jede Anweisung auf dem durch die vorangegangenen Anweisungen erzeugten Zustand operiert ([LAJ05], S.15). Dieselben Probleme stellen [RB05] auf der Schulebene fest. Für den Bereich Lesen sind insbesondere zwei Aspekte relevant: Das Verstehen der Programmausführung und das Zusammenfügen einzelner Teile zu einem sinnvollen Ganzen.
Nach Kintsch [Ki98] ist Verstehen ein regelbasierter Prozess, in dem aus einem Text Informationen extrahiert und unter Aktivierung relevanten Vorwissens und schlussfolgernden Denkens verschiedene mentale Repräsentationen gebildet werden, die letztendlich zu einem Gesamtverständnis führen ([Ki98], S.96f). Dieser Prozess führt schrittweise zu jeweils abstrakteren und von der ursprünglich wahrgenommenen Information unabhängigeren mentalen Repräsentationen ([Ki98], S.10-16). In diesem Prozess spielt das aktivierte und in das Verstehen integrierte Vorwissen eine große Rolle. Daher wird das (eigentlich holistische) Verstehen für analytische Zwecke in Textbasis (mit den extrahierten Informationen aus dem Text) und Situationsmodell (inklusive der gezogenen Schlussfolgerungen) getrennt ([Ki98], S.50).

3 Das didaktische Programmlesemodell

Das Modell soll bei möglichst geringer Komplexität helfen, Lern- und Lehrprozesse beim Erwerben von Programmierkompetenz zu analysieren, den Lernstand zu diagnostizieren sowie didaktische Konzepte zu planen, zu analysieren und zu bewerten.
Aus der Forderung nach Einfachheit folgt, so weit wie möglich auf paradigmen-, aufgaben-, domänen-, und werkzeugspezifische Aspekte des Programmverstehens zu verzich-

ten. Das Modell bezieht sich daher auf das an Schulen dominierende imperativ-objektorientierte Paradigma und den grundlegenden Prozess des verstehenden Lesens eines Programmtexts.

3.1 Verstehen

Verstehen kann als Prozess aufgefasst werden, in welchem dem vorliegenden Programmtext Bedeutung zugewiesen wird. Dazu wird beim Lesen des Programms eine interne mentale Repräsentation erzeugt, die die wesentlichen expliziten und impliziten Informationen aus dem Programmtext enthält. Als wesentlichen Aspekt des Verstehens fasse ich dabei die Fähigkeit auf, die im Programmtext enthaltene algorithmische Idee zu verstehen. Etwas abgeschwächt kann man im Falle einfacher Anfangsprogramme auch von Abläufen oder Verfahren sprechen. Verstehen bedeutet damit, sich von den wahrgenommenen Details des Programmtexts zu lösen. Erkennbar wird Verstehen beispielsweise an der Fähigkeit, das Programm mit eigenen Worten zusammenfassend zu erklären. Diese Erklärung muss qualitativ über das Nachvollziehen eines Programmablaufs hinausgehen, etwa indem die dahinterliegende algorithmische Idee erkannt wird und beschrieben werden kann. Folglich löst sich im Prozess des Verstehens die gedankliche Vorstellung von den gelesenen Einzelheiten des Programmtexts. Im optimalen Fall bleiben genau die wesentlichen Informationen erhalten, sodass auf Grundlage der so entstehenden Vorstellung das Programm erklärt werden kann. Wenn ein Programmtext verstanden wurde, dann kann man Schlussfolgerungen ziehen, Vorhersagen machen, den Programmtext in andere Darstellungen übersetzen, aber auch Details auf das Ganze

Der kommentierte Programmtext:

Die Funktion SortiereDurchEinfuegen sortiert das Array "A" mit *n* Elementen (z.B. Büchern) von klein nach groß.

```
SortiereDurchEinfuegen(A)
1  for i:=2 to n do                          {betrachte die Bücher an der Stelle i=2 bis n}
2      j:=i
3      while j ≥ 2 and A[j-1] > A[j]         {solange die Bücher A[j] und A[j-1] in der
   do                                         falschen Reihenfolge stehen...}
4          vertausche die Einträge A[j]      {...vertausche Bücher und...}
           und A[j-1]
5          j:=j-1                            {...gehe zum Bücherpaar eins weiter links}
6      endwhile
7  endfor
```

Beigefügte Erklärung, auf der Webseite oberhalb des Programmtextes präsentiert:

[1] „Wir bringen alle Bücher von links nach rechts im Regal fortschreitend an die richtige Position."
[2] „Das erste Buch bleibt zunächst an seinem Platz; dann nehmen wir das zweite Buch hinzu und vertauschen es mit dem ersten, falls es links von diesem stehen muss; dann nehmen wir das dritte Buch hinzu, vertauschen es mit dem zweiten, falls nötig, und vertauschen es anschließend gegebenenfalls mit dem ersten. Dann nehmen wir das vierte Buch hinzu, usw."
[3] „Allgemein nehmen wir also an, dass alle Bücher links im Bücherregal bereits sortiert sind; sodann nehmen wir eines hinzu und bringen es durch Vertauschen mit dem linken Nachbarn an die richtige Position."

Abbildung 1: http://www-i1.informatik.rwth-aachen.de/~algorithmus/algo2.php

beziehen, Auswirkungen von Änderungen vorhersagen und gezielt Änderungen am Programmtext vornehmen. Dies soll an einem Beispiel (Abbildung 1) erläutert werden.
Der Textauszug (Abbildung 1) ist dem Projekt Algorithmus der Woche [Al06] entnommen und beschreibt das Sortieren durch Einfügen am Beispiel eines Bücherregals. Das in Abbildung 1 gezeigte Beispiel bezieht sich auf die von den Autoren vorgelegte Erklärung des vorgestellten Algorithmus.
Verstehen umfasst damit das Verstehen der Ziele (Absatz [1]) und des Verfahrens (Absatz [2]). Das Verständnis des gelesenen Programmtexts – in Abbildung 2 durch das Oval dargestellt – kann analytisch in zwei Dimensionen getrennt werden: Die Textbasis enthält diejenigen Vorstellungen, die direkt den vorliegenden Informationen (dem Programmtext und eventuellen Eingabedaten) entnommen werden können. Das sind Vorstellungen über die konkrete Programmausführung sowie über den verallgemeinerten Ablauf. Die Textbasis enthält also das Verständnis des beschriebenen Verfahrens. Das Situationsmodell enthält darüber hinausgehende Schlussfolgerungen und das Verstehen der Ziele bzw. des Programmzwecks. Absatz [3] ist ein Beispiel für eine auf Basis des Verstehens gezogene Schlussfolgerung.
Verstehen beinhaltet insgesamt also (siehe Abbildung 2): a) relevante Aspekte der Programmausführung, b) relevante Aspekte des verallgemeinerten Ablaufs, c) über den Programmtext hinausgehende Schlussfolgerungen und d) die Beschreibung der allgemeinen algorithmischen Idee, die für unterschiedliche Eingaben gültig bzw. passend ist.

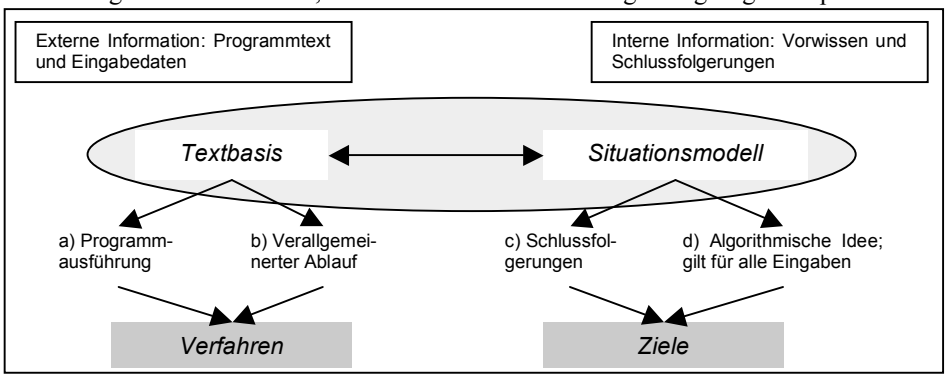


Abbildung 2: Textbasis und Situationsmodell

3.2 Der Verstehensprozess

Verstehendes Lesen von Programmtexten bezeichnet den Prozess der Wahrnehmung eines geschriebenen Quelltexts, die Extraktion relevanter Informationen, das Entwickeln einer Vorstellung von der Programmausführung sowie das Entwickeln eines generellen Verständnisses der algorithmischen Idee und der Ziele bzw. Zwecke des Texts.
Dieser Prozess kann in verschiedene Schritte unterteilt werden: Ausgehend von einzelnen Sprachkonstrukten, über kleinere Einheiten (Blöcke), deren zunehmende Integration bis zum Gesamtverstehen (siehe Abbildung 3). In Analogie zu Kintsch ([Ki98], S.101f) wird der schrittweise Verstehensprozess folgendermaßen konzipiert: Die dem Text entnommene Information wird sofort, d.h. Wort für Wort, in die mentale Repräsentation übernommen.

Auf der Konstruktebene wird die Bedeutung der einzelnen Konstrukte gebildet. Auf dieser untersten Ebene können nur Syntax und Semantik von Konstrukten sowie der Effekt der Ausführung betrachtet werden, erst ab der Blockebene sind Ziele erkennbar. Als Block kann man sich das vorstellen, was zwischen zwei Klammern oder „begin“ und „end“ steht. Unterprogramme, Methoden- und Schleifenrümpfe sind Beispiele für Blöcke. In den meisten Sprachen können Blöcke ineinander geschachtelt werden.
Blöcke übernehmen die Rolle, die Sätze in natürlichsprachlichen Texten einnehmen. Beim Lesen natürlichsprachlicher Texte wird die erste textnahe mentale Repräsentation am Satzende in das nächst abstraktere Modell integriert. Das Kurzzeitgedächtnis ist damit frei für den nächsten Lesezyklus. Es können jeweils nur wenige Einzelheiten, oder komprimierte bzw. abstrahierte Informationen übernommen werden. Daher müssen im Laufe des Verstehensprozesses die gebildeten mentalen Repräsentationen abstrakter werden.

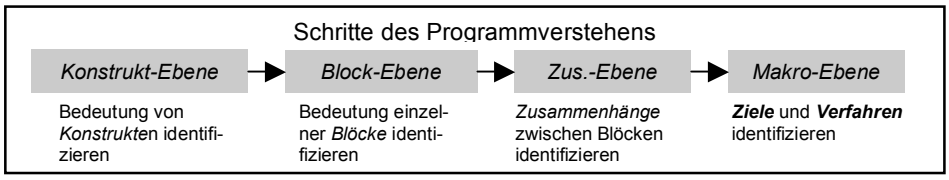


Abbildung 3: Ebenen bzw. Schritte des verstehenden Lesens eines Programms

In vielen Fällen können Blöcke benutzt werden, müssen aber nicht. Beispielsweise könnte in einem Sortierprogramm das Vertauschen zweier Variablen an verschiedenen Stellen jeweils als Folge einzelner Anweisungen geschrieben, oder in eine Methode „vertausche (a, b)“ ausgelagert werden, die jeweils aufgerufen wird. Wenn wir annehmen, dass ein Leser die Bedeutung einzelner Anweisungen, oder die einzelnen Zeilen des gelesenen Programms jeweils an einem Blockende in das nächst abstraktere Modell integrieren kann, dann ist ein Programm leichter zu verstehen, das logisch zusammengehörende Anweisungen in Blöcke verpackt. Dies liegt daran, dass die einzelnen Blöcke kürzer sind, sodass ein Leser sich nicht zu viele Einzelinformationen merken muss, bevor die Bedeutung einzelner gelesener Zeilen als Bedeutung eines Blocks zusammengefasst werden können.
Doch was bedeutet im zweiten Schritt, auf der Blockebene (siehe Abbildung 3), das Verstehen der Bedeutung eines Blocks? Die Bedeutung der Methode „vertausche (a, b)“ aus dem obigen Beispiel (Abbildung 1) kann man bereits an ihrem Namen ablesen. Die Bedeutung eines Blocks umfasst Ziel und Verfahren, während auf Konstruktebene meist noch kein Ziel erkennbar ist.
Im nächsten Schritt (Zusammenhangs-Ebene) werden Beziehungen und Zusammenhänge zwischen Blöcken in die mentale Repräsentation aufgenommen. Bleiben wir beim Beispiel oben: Dem Lesen des vertausche(a,b)-Blocks geht voran das Lesen der Bedingung im Schleifenkopf (Zeile 3 in Abbildung 1). Durch das Verknüpfen der beiden Blöcke wird verstanden, dass die Bücher nur dann vertauscht werden, wenn sie in der falschen Reihenfolge stehen (vgl. die Kommentare in Abbildung 1). Das Ziel ist also, die richtige Reihenfolge der Bücher herzustellen. Nebenbei: Die neben den Quelltextzeilen angebrachten Kommentare zeigen, dass beim Lesen diese Verstehensebenen gebildet werden; d.h. jede Quelltextzeile wird im Kontext des bis dahin Gelesenen verstanden.

Auf der obersten Ebene schließlich (Makroebene) wird der gesamte gelesene Programmtext verstanden, d.h. die Ziele sowie das Verfahren können erklärt werden.
Der Verstehensprozess läuft also bottom-up schrittweise von den unteren Ebenen zu den höheren. Eine Konsequenz dieses Verarbeitungsprozesses ist, dass Textbasis und Situationsmodell parallel konstruiert werden. Diese beiden Aspekte sind nur für Zwecke des Modells getrennt – tatsächlich entsteht eine einheitliche mentale Repräsentation.

3.3 Das Programmlesemodell

Das resultierende Programmlesemodell ist in Abbildung 4 dargestellt:

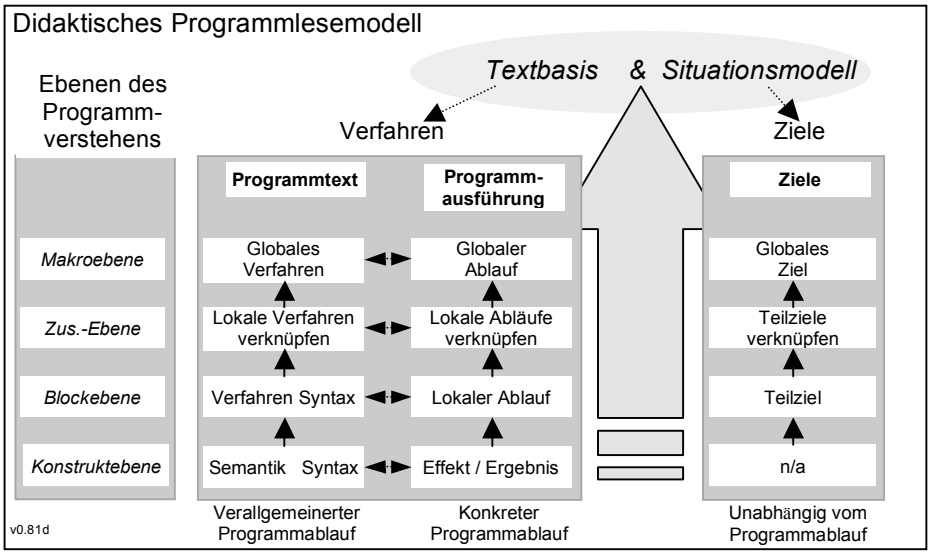


Abbildung 4: Didaktisches Programmlesemodell

Das Modell zeigt, wie auf den vier aufeinander folgenden Ebenen (linke Spalte) parallel Textbasis und Situationsmodell (oben rechts) aufgebaut werden. Die Textbasis enthält die Vorstellung des Verfahrens (mittlere Spalte). Dazu gehören die direkt dem Text entnehmbaren Informationen über den (verallgemeinerten und konkreten) Programmablauf. Wird etwa der Block j:=j+1 gelesen, wird zunächst die Syntax wahrgenommen und der Effekt (bzw. die Semantik) ermittelt: Als Semantik etwa „dekrementieren des Werts der Variablen j um 1“; der konkrete Effekt hängt vom Variablenwert ab. Mit Hilfe dieses Ablaufs wird im obigen Beispiel das Verfahren umgesetzt, die Bücherpaare von rechts nach links zu vergleichen (vgl. Kommentar in Zeile 5, Abbildung 1).
Auf den höheren Ebenen enthält die zunehmend abstraktere mentale Repräsentation nicht mehr direkte Darstellungen des gelesenen Texts, der eigentlichen Syntax, sodass dieser Aspekt nur auf Ebene 1 und 2 vorkommt. Stattdessen wird vom eigentlichen Programmtext abstrahiert und auf den gebildeten mentalen Repräsentationen operiert. Das Programmlesemodell unterscheidet in der Textbasis zwischen dem Verstehen des allgemeinen Verfahrens, wie es im Programmtext beschrieben ist, und dem Verstehen des konkreten Ablaufs, der von den konkreten Eingabedaten bestimmt wird.

Zum Situationsmodell gehören das Verstehen der algorithmischen Idee, der Ziele sowie weiterer Schlussfolgerungen (rechte Spalte in Abbildung 4). Die Integration von Textbasis und Situationsmodell stellt den schwierigen Teil dar (oben rechts). Denn hier gilt es, vom (beobachtbaren / anhand des Vorwissen rekonstruierbaren) Verhalten des Programms auf die mit dem Programmtext verbundene Absicht (Ziele) zu schließen.
Der Verstehensprozess verläuft parallel zum Lesen, also zyklisch. D.h. während des Lesens des nächsten Konstrukts werden die Stufen jeweils wieder neu durchlaufen, soweit das möglich ist. Falls etwa schon Blöcke gelesen wurden, werden beim Lesen des nächsten Blocks mögliche Zusammenhänge gebildet. Am Blockende wird jeweils das gebildete mentale Modell soweit abstrahiert, dass von der konkreten Textgestalt und einzelnen Anweisungen abstrahiert wird, sodass das sensorische Kurzzeitgedächtnis die nächsten zu lesenden Konstrukte wieder aufnehmen kann. Obwohl dieser Prozess streng bottom-up verläuft, gibt es Aspekte des intentionalen top-down-Vorgehens durch Hypothesen, die beim Lesen gebildet werden, oder durch Arbeitsaufträge, die vor dem Lesen gegeben werden. Der Verstehensprozess wird so durch aktiviertes Vorwissen bzw. Hypothesen beeinflusst. Diese Beeinflussung wird stärker, wenn Lernende die unteren Schritte (das Verarbeiten und Verstehen der Bedeutung einzelner Konstrukte) bereits automatisiert bewältigen können. Zudem werden vermutlich Teile des Programmtexts wiederholt gelesen, insbesondere von Anfängern. Damit wird das Lesen nochmals durch die bis dahin aufgebaute mentale Repräsentation beeinflusst.
Bezüglich der Kompetenzentwicklung wird angenommen, dass sich die Verstehens-Kompetenz der Lernenden von unten nach oben entwickelt, dass also die unteren Ebenen schneller automatisiert verarbeitet werden als die oberen. Ein kompetenter Leser unterscheidet sich von einem Anfänger vor allem dadurch, dass die notwendigen grundlegenden Verarbeitungsschritte stärker automatisiert ablaufen können. Zudem passieren beim Aufbau des Verstehens weniger Fehler und es kann mehr relevantes Vorwissen aktiviert werden. Dadurch bleiben mehr kognitive Ressourcen für schlussfolgerndes Denken und tiefer gehende Verarbeitungsschritte übrig.

4 Anwendungen des Modells

In diesem Kapitel werden beispielhaft Anwendungen des Modells gezeigt. Diese dienen auch als Test der Gebrauchstauglichkeit des Modells im Sinne der Erklärungskraft, inneren Stimmigkeit und externen Gültigkeit.

4.1 Analyse einer Studie zur Programmierkompetenz

Zunächst soll das Modell benutzt werden, um die Ergebnisse einer internationalen Studie zur Programmierkompetenz zu interpretieren: Kann das Modell die Ergebnisse erklären?
In der so genannten Lister-Studie [Li04] hat eine internationale Arbeitsgruppe einen Test konzipiert, der ca. 550 Studenten aus 7 Ländern vorgelegt wurde. Der Test besteht vor allem aus Lese-Aufgaben, bei denen zum Teil eine fehlende Quelltextzeile ergänzt werden musste. Insgesamt beschreiben die Autoren die Ergebnisse folgendermaßen ([Li04], S. 128): Das beste Viertel hat ein gefestigtes Verständnis der Grundlagen der Programmierung, das schlechteste Viertel habe „fundamentale Probleme“. Die 50 % in der Mitte

verstehen Schleifen und Felder (arrays). Insgesamt gelte: „Their weakness is the inability to reliably work their way through the long chain of reasoning required to hand execute code, and/or an inability to reason reliably at a more abstract level to select the missing line of code.“ ([Li04], S. 128). Als problematisch erweist sich, die Ergebnisse zu verstehen und Schlussfolgerungen zu ziehen ([Li04], S. 128). In einem neueren Ansatz ([Li06]) verwenden die Autoren ein theoretisches Modell, mit dem sie ihre Ergebnisse interpretieren. Der Erkenntnisgewinn dieser Herangehensweise besteht in einen Interpretationsrahmen für die Ergebnisse, aus dem heraus sich das Leistungsniveau erklären lassen kann und von dem leistungsfördernde Maßnahmen abgeleitet werden können. Die Schlussfolgerung der Autoren lautet nun, dass die Schwierigkeiten vor allem darin bestehen, einzelne Teile eines Programmtexts miteinander in Beziehung zu setzen.
Zur Erklärung der Ergebnisse anhand des Programmlesemodells: Viele Studierende haben einen Kenntnisstand, der es ermöglicht die in der Studie verwendeten Programmtexte auf der Ebene 3 (Zusammenhangs-Ebene) zu verstehen. Sie können Konstrukte verstehen, einzelne Blöcke erklären und Beziehungen verstehen. Dies gelingt jedoch nicht so gefestigt, dass sie sicher operieren. Daher schleichen sich bei anspruchvollen und versteckten Wechselwirkungen zwischen Blöcken beim Lesen Verarbeitungsfehler bzw. vorschnelle Schlussfolgerungen ein, die zu einem falschen Gesamtverstehen führen. Die didaktische Schlussfolgerung wäre, das Verstehen des Zusammenspiels zwischen Teilen eines Programms zu üben. Zusätzlich könnte man die Verarbeitungsschritte auf den niedrigeren Ebenen trainieren, damit diese stärker automatisiert ablaufen, sodass mehr kognitive Ressourcen für komplexere Verarbeitungsschritte frei bleiben.

4.2 Leseaufgaben im Informatikunterricht

In diesem Abschnitt werden Leseaufgaben für den Informatikunterricht untersucht, die in einem fachdidaktischen Seminar für Lehramtsstudierende erprobt und anhand des Programmlesemodells analysiert wurden. Bei den drei Aufgabentypen handelt es sich um: „Lesepuzzle“, „Finde den Fehler“ und „Lesen mit verteilten Rollen“.

Lesepuzzle

Der zu lesende Quelltext wird in einzelne Zeilen oder Abschnitte aus mehreren Zeilen zerschnipselt. Die Aufgabe ist, das Quelltextpuzzle wieder richtig zusammenzusetzen. Die Lernenden bekommen jeweils ein eigenes Puzzle. Nachdem die Teile ausgepackt und auf dem Tisch verteilt sind, versuchen einige, zunächst verschiedene größere Teile zusammenzusetzen. Andere versuchen sofort, einen einheitlichen Text zu erstellen. In allen Fällen werden Papierschnipsel aufgenommen, miteinander verglichen, auf dem Tisch verschoben und ausgetauscht. Das Lösen der Aufgabe war schwerer als erwartet. Als einzige Methode wurde sie zweimal erprobt, da beim ersten Mal fast niemand in der geplanten Zeit die Lösung gefunden hat.
Die Analyse anhand des Programmlesemodells: Bei der ersten Erprobung bestand das Puzzle überwiegend aus einzelnen Zeilen, und damit sind die Ziele des Textes nicht bzw. kaum zu erkennen. Die Lernenden bekommen nur Informationen auf der untersten Ebene des Programmlesemodells (siehe Abbildung 4). Beim zweiten Versuch entsprachen die Puzzleteile eher der Block-Struktur des Programms. Dieses Mal konnte das Puzzle gelöst werden, da nun für das Zusammenfügen der Teile die Beziehungen zwischen

Blöcken geklärt werden mussten. Damit setzt diese Variante voraus, dass die Lernenden die einzelnen Blöcke verstehen.
Bewertung: Es hat sich herausgestellt, dass die meisten Studenten das Puzzle anhand einfacher Hinweise im Programmtext gelöst haben: Puzzleteile, in denen Variablen deklariert werden, gehören tendenziell eher an den Anfang, Teile mit Ausgaben für den Benutzer eher ans Ende. Obwohl die Aufgabe motivierte und die Lernenden aktivierte, hat das verwendete Puzzle nicht durchgehend die gewünschte Verarbeitung auf den Ebenen 2 und 3 gefordert. Im Ergebnis konnten trotz richtiger Lösung nicht alle die Intention des Programms erklären, haben also kein Gesamtverständnis aufgebaut. Für einen effektiven Einsatz muss die Methode also sehr sorgfältig vorbereitet werden (bis hin zum Anfertigen der Teile: Die Schnittkanten dürfen nicht verraten, wie die einzelnen Teile zusammengehören).

Finde den Fehler

Bei dieser Methode wird ein den Lernenden an sich bekanntes Programm in veränderter Fassung der gesamten Klasse präsentiert. Aufgabe ist, die Fehler möglichst schnell zu finden und zu erklären. Dies können syntaktische und semantische Fehler sein. Für jeden gefundenen und richtig erklärten Fehler gibt es Punkte. Sofort mit der Präsentation des veränderten Quelltextes versuchen die Lernenden, Fehler zu finden und sich möglichst schnell zu melden. Einige rufen sogar die Lösung laut in die Klasse.
Die Analyse anhand des Programmlesemodells: Die Methode bezieht sich bei syntaktischen Fehlern auf die unterste Ebene. Die Erklärung semantischer Fehler setzt voraus, die Verbindung des einzelnen Blocks zum Gesamtziel erklären zu können. So werden auch die Ebenen 3 und stärker noch Ebene 4 trainiert. Die Erklärungen unterstützen zudem die Verbindung von Textbasis und Situationsmodell.
Bewertung: Den Studenten hat diese Methode sehr gut gefallen; vermutlich auch aufgrund des Wettbewerbscharakters durch die Punktevergabe. Allerdings gab es daher auch störende Zwischenrufe, wenn jemand einen Fehler zwar identifiziert hat aber nicht gleich erklären kann. Diese hereingerufenen Erklärungen haben den Verstehensprozess unterbrochen. Da recht einfach Fehler bzw. interessante algorithmische Abwandlungen eingebaut werden können, ist diese Methode flexibel und recht einfach vorzubereiten.

Lesen mit verteilten Rollen

Bei dieser Methode handelt es sich um ein vereinfachtes Rollenspiel: Verschiedene Sprecher lesen einzelne Textabschnitte vor und müssen dabei die Programmausführung anhand vorgegebener Eingabedaten inszenieren. Im Seminar gab es zudem die Rolle, an der Tafel eine Wertebelegungstabelle parallel zum vorgelesenen (bzw. durchgespielten) Programmtext mitzuführen. Die verschiedenen Rollen wurden wie folgt verteilt: Pro Block liest ein Sprecher alle Vergleichoperationen (und das Resultat mit den aktuellen Werten); der andere den Rest. Bei jedem erneuten Methodenaufruf lesen neue Sprecher.
Die Analyse anhand des Programmlesemodells: Da jeweils ein Programmblock gelesen und anschließend die Ausführung erläutert wird, trainiert diese Methode den Aufbau der Textbasis, insbesondere die Verbindung zwischen Programmausführung und Programmtext. Durch das wiederholte Aufrufen derselben Methode mit verschiedenen Werten wird auch der Aufbau des Situationsmodells gefördert, da zunehmend die Ziele der einzelnen Blöcke und ihre Beziehungen untereinander deutlich werden.

Bewertung: Die Durchführung ist recht zeitaufwändig, weil der Quelltext manuell ausgeführt wird. Zudem ist jeweils nur ein Teil der Gruppe aktiv, auch wenn die Zuhörer die Aufgabe bekommen haben, die Arbeit der Sprecher zu prüfen. Daher wirkt die Methode mitunter zäh, allerdings wird durch Sprecherwechsel und zunehmend schnelleres Lesen bei wiederkehrenden Passagen deutlich, dass das Programm besser verstanden wird. Bei der Erprobung im Seminar wechselten einige Leser z.B. unbeabsichtigt die Ebene und lasen nicht mehr die Anweisungen vor, sondern erklärten sofort Verfahren und Ziele des Textabschnitts; als wenn sie z.B. in Abbildung 1 nicht mehr nur den Quelltext, sondern die Kommentare sehen und diese lesen würden.

Leseaufgaben und Programmlesemodell

Die Diskussion der verschiedenen Unterrichtsmethoden mit Hilfe der Begrifflichkeit des didaktischen Programmlesemodells sollte zeigen, wie dieses zur Analyse und Planung von Aufgaben genutzt werden kann. Es liefert Erklärungen bzw. Thesen für den Lerneffekt der Methoden sowie Hinweise für die Gestaltung und Bewertung. Beispielsweise ist nach dem Modell das Puzzeln möglicherweise doch nicht sehr lernwirksam, obwohl alle Lernenden aktiv sind. Die Effektivität der Methode hängt stark von der Vorbereitung (Zuschnitt der Puzzle-Teile) ab. Dagegen scheint das Lesen mit verteilten Rollen zwar recht langwierig und für die meisten Lernenden eher passiv zu sein, unterstützt aber das Aufbauen des Programm-Verständnisses, da die einzelnen Verstehenselemente nacheinander sichtbar bzw. hörbar gemacht werden. Insbesondere wird der Zusammenhang zwischen Textbasis (Verfahren) und Situationsmodell (Ziel) deutlich. Die dritte vorgestellte Methode (Finde den Fehler) hängt stark von der Durchführung ab: Wenn Fehler nicht nur gefunden, sondern erklärt werden müssen, dann wird an verschiedenen (vom Lehrer vorher ausgewählten Stellen) der Zusammenhang zwischen einem Detail und dem Ganzen deutlich, etwa zwischen Block und Gesamtziel, Zusammenhänge zwischen Blöcken oder Zeile/Konstrukt und Block oder Textbasis und Situationsmodell.
Des Weiteren kann das Modell auch zum Erfinden bzw. zum Anpassen der Lesemethoden für die konkrete Unterrichtssituation herangezogen werden.
Insgesamt können die Lesemethoden in Präsentations-, Inszenierungs- und Transformationsmethoden unterteilt werden. Präsentationsmethoden präsentieren den Text auf ungewöhnliche Art, etwa als Puzzle. Inszenierungsmethoden erwecken den Text zum Leben, indem sie ihn mit der Programmausführung verbinden. Beispiele sind die oben vorgestellte Methode des Lesens mit verteilten Rollen, Rollenspiele u.ä. . Transformationsmethoden regen zum genauen Lesen an, indem sie das Ändern bzw. Ergänzen der Darstellung erfordern. Beispielsweise das Übersetzen des Quelltextes in grafische Darstellungsformen, die Methode Finde den Fehler oder Kommentierungs-Methoden, bei denen Kommentare zum Quelltext ergänzt, benotet, verändert werden müssen. Das Programmlesemodell kann zur Ausgestaltung der Unterrichtsmethoden genutzt werden. So könnte es helfen, Kriterien für Kommentare zu finden (sollen Kommentare sich auf die Textbasis oder das Situationsmodell beziehen?) oder zu überlegen, ob auch ein Kommentarpuzzle (Ordne die Kommentare ihren Quelltextzeilen zu) eine sinnvolle Methode wäre.

4.3 Programmtexte lesen und Programmierkompetenz

Eine weitere interessante Anwendung ist der Bereich der Kompetenzmessung, der bereits im Abschnitt 4.1 gestreift wurde. Dort wurden die Ergebnisse eines (zugespitzt formuliert) „Programmlese-Tests“ anhand des Modells interpretiert. Es scheint nun nahe zu liegen, die Stufen des Leseprozesses (siehe Abbildung 3) als Kompetenzstufen zu benutzen. Dabei würde jedoch die Schwierigkeit des gelesenen Programmtextes zu wenig beachtet werden – denn das Verstehen eines Programmtexts hängt natürlich von seiner Komplexität ab (sowie von weiteren Faktoren wie der Vertrautheit des Lesenden mit entsprechenden Algorithmen oder Problembereichen). Das Programmlesemodell beschreibt also keine Kompetenzstufen, sondern den Verstehensprozess beim Lesen eines Programmtexts. Ein Kompetenz(stufen-)modell müsste beschreiben, bis zu welcher Komplexität Programme verstanden werden können.
Das Programmlesemodell ließe sich nutzen, um Hinweise für die Komplexität von Programmen abzuleiten. Beispielsweise bestehen simple Programme aus einem oder wenigen Blöcken, etwas anspruchsvollere ggf. aus mehreren Blöcken, welche direkt verbunden sind. Weitere Komplexität würde durch komplexere Zusammenhänge zwischen Blöcken entstehen, etwa durch geschachtelte Blöcke usw. Die genauen Komplexitätsstufen eines solchen Modells würde man empirisch festlegen. Dazu notwendige Aufgaben ließen sich vorab anhand des Modells konstruieren. Ein Beispiel für eine solche Herangehensweise findet sich in [BS06].
Ebenso ließe es sich zur Lerndiagnostik nutzen, um anhand des Lernstands geeignete Fördermaßnahmen zu entwickeln (vgl. die Argumentationsweise am Ende von Abschnitt 4.1). Ausgehend vom Lesen des Quelltexts zeigt das Lesemodell Prozesse und Dimensionen, die zum Verstehen führen. Diese ermöglichen es, gezielt diejenigen Kenntnisse und Fähigkeiten zu trainieren, die nach den Ergebnissen der Forschung zum Programmieren Lernen besondere Probleme bereiten: Das Verstehen der Programmausführung und das Zusammenfügen einzelner Teile zu einem sinnvollen Ganzen (vgl. Abschnitt 0).

5 Schlussbemerkung

Insgesamt spielt die Programmausführung eine entscheidende Rolle beim Verstehen von Programmtexten. Aussagen über Programmtexte beziehen sich in vielen Fällen tatsächlich auf dessen Ausführung und nicht auf den Text im eigentlichen Sinne. Das Schwierige am Verstehen eines Programms ist dementsprechend, den Zusammenhang zwischen den verschiedenen möglichen konkreten Programmausführungen und der gemeinsamen allgemeinen Idee herstellen zu können (vgl. Abbildung 2: Textbasis und Situationsmodell). Es scheint wenig sinnvoll zu versuchen, zunächst etwa die erste Stufe zu lehren, dann die zweite usw. Dies liegt daran, dass Ziele nicht auf der ersten Ebene (bzw. im ersten Verstehensschritt) sichtbar werden. Für das sinnvolle Lernen aber ist die Kenntnis der Programmziele unerlässlich. Ebenso wie das Schreiben eines Programmtexts auf die spätere Ausführung des Programms zielt, ist auch das Lesen eines Programmtexts auf die Programmausführung bezogen – es kann sogar als integraler Bestandteil des Schreibens gesehen werden. Nicht zuletzt deshalb verwenden empirische Studien zur Programmierkompetenz (vgl. das Beispiel aus Abschnitt 4.1) Leseaufgaben.

Literaturverzeichnis

[Al06] Fakultätentag Informatik: Webseite Algorithmus der Woche: http://www-i1.informatik.rwth-aachen.de/~algorithmus/index.php (Letzter Zugriff April 2007)

[BS06] Bennedsen, J.; Schulte, C.: A Competence Model for Object-Interaction in Introductory Programming. In: PPIG 2006. http://www.ppig.org/workshops/18th-programme.html

[Bo89] du Boulay, B. (1989). Some difficulties of learning to program. In: Soloway, E. and Spohrer, J. C. (Eds): Studying the novice programmer, S. 57-73

[Br05] Breier, Norbert: Informatik im Fächerkanon allgemein bildender Schulen - Überlegungen zu einem informationsorientierten didaktischen Ansatz. INFOS 2005 S. 67-78

[BDW02] Burkhardt, J.-M., Detienne, F., Wiedenbeck, S.: Object-Oriented Program Comprehension. In: Empirical Software Eng., Nr. 2, 2002, S. 115-156.

[De01] Détienne, Francoise: "Software Design – Cognitive Aspects", Springer, 2001.

[ET05] A. Eckerdal and M. Thune. Novice java programmers' conceptions of object and class, and variation theory. In ITiCSE '05, 2005.

[GI00] Gesellschaft für Informatik: Empfehlungen für ein Gesamtkonzept zur informatischen Bildung an allgemein bildenden Schulen. 2000.

[Ha99] Hampel, T.; Magenheim, J.; Schulte, C.: Dekonstruktion von Informatiksystemen als Unterrichtsmethode. Zugang zu objektorientierten Sichtweisen. INFOS 1999, S. 149-164.

[Hu99] Hubwieser, P.: Informatik als Pflichtfach an bayerischen Gymnasien. INFOS 1999, S. 165-174.

[Ki98] Kintsch, W.: Comprehension. A Paradigm for Cognition. Cambridge Univ. Press, 1998.

[LAJ05] Lahtinen, E., Ala-Mutka, K., and Järvinen, H. 2005. A study of the difficulties of novice programmers. In ITiCSE 2005

[Li04] Lister, Raymond et al. A multi-national study of reading and tracing skills in novice programmers. In ITiCSE-WGR '04, 2004.

[Li06] Lister, R., Simon, B., Thompson, E., Whalley, J. L., Prasad, C. Not seeing the forest for the trees: novice programmers and the SOLO taxonomy. In ITICSE 2006. S. 118-122.

[MV94] Van Mayrhäuser, A.; Vans, A.M.: Program Understanding – A Survey. Technical Report CS-94-120. Colorado State University 1994. (Manuskript)

[MR02] I. Milne & G. Rowe. Difficulties in Learning and Teaching Programming - Views of Students and Tutors. Education and Information Technologies, 7(1):55-66, 2002.

[Pe87] Pennington, Nancy: Stimulus structures and mental representations in experts comprehension of computer programs. Cognitive Psychology 19, 295–341, 1987.

[RB05] Ragonis, N.; Ben-Ari, M: On understanding the statics and dynamics of object-oriented programs. In SIGCSE '05, 2005, S. 226-230.

[RRR03] A. Robins, J. Rountree, and N. Rountree. Learning and teaching programming: A review and discussion. Computer Science Education, 13(2):137–172, 2003.

[SS89] E. Soloway and J. Spohrer. Studying the Novice Programmer. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1989

[SS86] Spohrer, J. C.; Soloway, E.: Novices Mistaces: Are the folk wisdom correct. In: Communications of the ACM, 29,Nr. 7. 1986, S. 624-632.

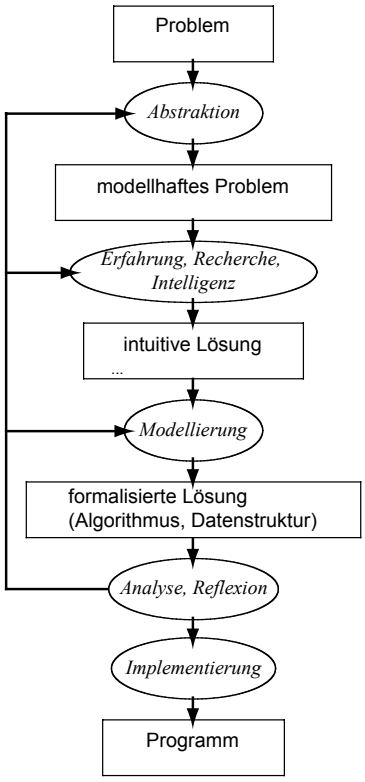
[St05] Storey, M.-A.: Theories, Methods and Tools in Program Comprehension: Past, Present and Future. In: IWPC 2005

Abenteuer Informatik oder „hands on“ bei Problemlösemethoden

Jens Gallenbacher, Didaktik der Informatik
Fachbereich Informatik, Technische Universität Darmstadt
gallenba@informatik.tu-darmstadt.de

Abstract. Paradigmen der Informatik, die sich an ein "natürliches" Weltmodell anlehnen, etwa die Objektorientierung, eignen sich selbstverständlich für eine entsprechend vom Computer losgelöste, handlungsorientierte Einführung. Der handlungsorientierte Ansatz eignet sich jedoch auch für Themen, die traditionell direkt in einer Programmiersprache oder abstrakt abgehandelt werden. Analogien und Modelle können von Lehrer und Schülern zusammen erarbeitet werden. Auf diese Weise kann auch in der Oberstufe ein weniger implementierungslastiger Unterricht etabliert werden.

In den Naturwissenschaften und der Mathematik gibt es eine lange Tradition für anschaulichen Unterricht bzw. populärwissenschaftlich aufbereitete Präsentationen mit entsprechender Museumsdidaktik. Museen wie das Mathematicum in Gießen oder das Deutsche Museum in München mit ihren Mitmach-Exponaten dienen nicht nur dazu, das entsprechende Thema zu erklären - sie wecken vielmehr auch das Gesamtinteresse an der jeweiligen Wissenschaft und bauen Berührungsängste ab.



Für Ingenieurwissenschaften und Informatik stehen zwar eindrucksvolle Technologie-Demonstrationen zur Verfügung, die aber keine echten Einsichten in die Materie erlauben. Ganz ähnliche Probleme stellen sich im schulischen Informatik-Unterricht: Ein großer Teil des Informatik-Unterrichts in höheren Klassen läuft entweder abstrakt ab oder am Computer.

Für den Aspekt der Modellierung wurden bereits einige gängige und praktikable Ansätze vorgeschlagen, der Aspekt des Problemlösens wird hier jedoch weitgehend ausgeklammert (z. B. [Br05b]).

Die in der Abbildung gezeigten Schritte des allgemeinen Problemlöseprozesses lassen sich allerdings ebenfalls sehr gut handlungsorientiert vermitteln. Beispiele hierfür sind Routenplaner, Sortieren in $O(n^2)$, $O(n \log n)$ und $O(n)$.

Besonders allgemeinbildenden Charakter hat die Kombination mit weiteren Fachdisziplinen. So kann etwa der Informationsbegriff auch im Kontext der deutschen Sprache vermittelt werden.

Im Rahmen der Informatik-Lehrerausbildung sind an der TU Darmstadt etliche Szenarien entstanden und dort sowie an Partnerschulen erprobt worden. Eine genaue Beschreibung kann im Rahmen dieses Praxisbandes leider nicht erfolgen. Ansätze können aber aus [Ga06] entnommen werden, hier finden sich auch geeignete Vorlagen für Lehrmaterial. Sie illustrieren, dass das Lernziel „Problemlösemethoden und ihre Bewertung“ in allen Phasen handlungsorientiert durchgeführt werden kann: Mit minimalem Coaching des Lehrers entwickeln die Schüler selbständig Ideen zur Problemlösung, die sie dann formalisieren und analysieren.

Die Implementierung ist optional! „Vom Problem zum Programm“, etwa in [Br05a] wird ersetzt durch „Vom Problem zum Algorithmus“. Das Unterrichtsziel verschiebt sich dadurch von einer Kompetenzbildung in Bezug auf die Lösung von Problemen mit dem Computer in Richtung einer allgemeinen Problemlösekompetenz.

Literaturverzeichnis

[Br05a] Norbert Breier „Informatik im Fächerkanon allgemein bildender Schulen - Überlegungen zu einem informationsorientierten didaktischen Ansatz“ in Proceedings der INFOS 2005 - 11. GI-Fachtagung Informatik und Schule, 2005, S. 67 - 78

[Br05b] Peter Brichzin, Ulrich Freiberger, Klaus Reinold, Albert Wiedemann: „Ikarus Natur und Technik. Schwerpunkt: Informatik“ Oldenbourg Schulbuchverlag, ISBN 3-486-88286-4, München, 2. Auflage 2005

[Ga06] Jens Gallenbacher: „Abenteuer Informatik: IT zum Anfassen von Routenplaner bis Online-Banking“ Elsevier Spektrum Akademischer Verlag, ISBN 3-8274-1635-3, Heidelberg, 2006

Objektorientierte Softwareentwicklung an der Realschule mit SEMI-OOS[1]

Robert Pütterich
Institut für Informatik, Fachgebiet Didaktik der Informatik
Technische Universität München
robert.puetterich@in.tum.de

Seminar für Informatik, Sophie-La-Roche-Realschule Kaufbeuren
robert.puetterich@gmx.de

Zielsetzung der in der Überschrift genannten Entwicklungsumgebung ist es, bereits in der Sekundarstufe I und zwar auch mit Schülerinnen und Schülern des mittleren Bildungsweges und unter Berücksichtigung gesicherter Erkenntnisse der Didaktik der Informatik, objektorientierte Software zu erstellen.

Die Entstehung der Idee und die Notwendigkeit einer solchen Umgebung, deren Planung, Erstellung und Evaluation Inhalt meines laufenden Promotionsvorhabens ist, kann kurz wie folgt zusammengefasst werden: Eigene Unterrichtserfahrungen zeigten zunächst, dass die Erarbeitung grundlegender Inhalte zum Thema Objektorientierung anhand einfacher Beispiele aus der Alltags- bzw. Softwarebedienungserfahrung die Schülerinnen und Schülern vor keine zu großen Probleme stellt. Die Anwendung der erworbenen Kenntnisse im Rahmen der objektorientierten Softwareentwicklung gestaltete sich dann aber in der Regel vor allem in der Implementierungsphase deutlich schwieriger, da die Flut der Syntaxspezifikationen und –besonderheiten der von mir gewählten Sprache Java, die in dieser Hinsicht kaum als Sonderfall zu betrachten ist, große Probleme bereitete.

Aus diesen Erfahrungen, die mir von einigen Kolleginnen und Kollegen durchaus bestätigt wurden, und der vergeblichen Suche nach einer optimalen Entwicklungsumgebung für die Realschule entstand das Vorhaben der Gestaltung einer für den genannten Einsatzzweck optimierten Software. Eine erste Version wird voraussichtlich bis zum Sommer 2007 fertig gestellt sein. Aktuelle Informationen und eine Möglichkeit zum Download werden unter http://robert.pütterich.de zur Verfügung stehen.

Die Entwicklung von SEMI-OOS basiert auf folgenden Grundsätzen:

Unter Berücksichtigung moderner fachdidaktischer Prinzipien ist ein entscheidendes Kriterium einer solchen Umgebung die Zentrierung hin zur Semantik und weg von der einer spezifischen Syntax (siehe [Hu03]). So werden beispielsweise Klassen sowie Vererbungsbeziehungen nicht textuell definiert, sondern durch die oberflächengesteuerte Festlegung ihrer Bestandteile. Die Darstellung in UML erledigt SEMI-OOS umgehend nach jeder Eingabe.

Eine möglichst reduzierte Anzahl an Modellierungstechniken ist neben einer übersichtlichen und leicht zu bedienenden Oberfläche ebenfalls ein wichtiges Kriterium, um die

[1] Schulgemäße Entwicklungsumgebung zur Modellierung und Implementierung objektorientierter Software

Schülerinnen und Schüler gerade an der Realschule nicht zu überfordern und damit mehr Verständnis, Klarheit und Freude im Umgang mit der Software zu schaffen. Ich habe mich daher für eine Minimalkombination aus Klassendiagrammen und Struktogrammen zur Definition der Operationen entschieden.

Die Eingabe der Anweisungen und booleschen Ausdrücke wird vom System laufend überwacht und mit einer stets aktiven Textvervollständigung zur Vermeidung von Tippfehlern optimiert.

Oberflächen sind in der modernen Softwareentwicklung nicht mehr wegzudenken. Deren Gestaltung erfolgt der Einfachheit halber mit einem Editor, welcher zudem die Attribute mit ihren Werten und die verfügbaren Methoden der einzelnen Objekte anzeigt. Im vom System erzeugten Code ist die Zusammensetzung der Oberflächen aus objektorientierter Sicht ebenso nachvollziehbar.

Das System implementiert in einer intuitiv verständlichen, neu entwickelten „Schüler"-Programmiersprache und zum Vergleich zusätzlich in der verbreiteten Sprache Java.

Der Programmstart erfolgt durch die Ausführung einer vom Benutzer gewählten Methode. Mit Hilfe dieser Option kann auch die Funktionalität jeder anderen Methode separat getestet werden, indem sie als Startmethode festgelegt und ausgeführt wird.

Um dem verschiedenen Kenntnisstand gerecht zu werden, ist es zur Differenzierung möglich, bestimmte Entwicklungsbereiche an- und abzuschalten.

Zur besseren Analyse der Fähigkeiten und Fertigkeiten der Schülerinnen und Schüler werden alle Aktionen und Eingaben auf Festplatte mit Datum, Uhrzeit und jeweiliger Zeitspanne zwischen den einzelnen Tätigkeiten mitprotokolliert.

Didaktische Reduzierungen sind im Sinne der oben genannten Grundsätze unvermeidbar, SEMI-OOS ist aber so gestaltet, dass viele, grundlegende Problemstellungen gelöst werden können.

Literaturverzeichnis

[Hu03] Hubwieser, P.: Didaktik der Informatik. Springer-Verlag, 2. Auflage, 2003; S. 78-90.

Ein Konzept zum (re)integrierenden Lernen in der Schulinformatik an Hand komplexer Systeme

Daniel Michael Meyer
Freie Universität Berlin
danielmichaelmeyer@gmx.de

Abstract. Die Förderung informatischen Verständnisses über die Programmierung hinaus wird global über die gesamten Lerninhalte der Oberstufe hinweg betrachtet. Dabei orientiert sich der Ansatz an den höheren Bildungszielen und am Paradigma der Komplexitätsreduktion. Statt separierenden Ansätzen wird ein integrierender verfolgt, auf Basis direkten Unterichtens an Hand komplexer Systeme oder mit Hilfe noch zu suchender Gütekriterien für thematische Anknüpfungspunkte.

Einleitung und Problemanalyse. Vorliegender Ansatz verfolgt eine integrierende Idee über Themata der gesamten Sek.II, der an Hand der höheren Bildungsziele wie „Förderung der Abstraktionsfähigkeit", „Thematisierung realistischer Beispiele", „Paradigmenwechsel" und informatikspezifischer „Modellbildung", „Abstraktion", „Kapselung", „Problemreduktion", „Kompartimentierung" und „Engineering" motiviert ist.

Komplexe Systeme als methodisch-integrative Klammer informatischer Bildung und das Komplexitätsparadigma. Diese Ziele sind auf die Zukunft ausgerichtet, jedoch der Historie entwachsen. Viele Entwicklungen in der Informatik resultieren aus dem Zwang, Komplexität zu reduzieren oder beherrschbar zu machen. Erst damit sind Konzepte der Modellierung, der OOP, Datenbanken usw. überhaupt zu rechtfertigen. Damit ist ein begrifflicher Rahmen gefunden, unter den verschiedene Aspekte der Oberstufeninformatik ohne „Kunstkrücken" fallen. Sätze wie „Warum extra eine Klasse bilden? Ich packe das einfach in eine Funktion und fertig..." sind ja absolut berechtigt, wenn es rein um die Funktion geht. Damit der Rahmen nicht gedanklich-begrifflicher Natur bleibt bietet es sich an, die Problematik der Komplexität auch an Hand eines komplexen Problems darzustellen. Selbiges soll die reale didaktische Klammer bilden, an Hand derer ein Großteil des gesamten Schulstoffs der Sek.II angesprochen werden kann (nicht muß). Dazu gehören Datenbanken, Hardware, die Programmierung, Modellierung und Technikkonsequenzen. Dieses Umsetzen verschiedener Sichten (didaktischer Linsen, vgl. [SM06]) auf ein und dasselbe Problem soll dabei helfen zu begreifen, dass Realweltprobleme zumeist nur im Methodenverbund lösbar sind und ein Problem mehrere Ebenen hat, die sich gegenseitig bedingen und Perspektivabhängig sind.

Ansatzpunkte für Verbindungselemente. Als Ziel eines Ansatzes zur (re)Integration von eher isoliert gelerntem und vermitteltem Wissen wäre eine Indikatorliste wünschenswert, der diese Ansatzpunkte möglichst allgemein benennt bzw. die Kriterien aufführt, die einen sinnvollen Ansatzpunkt ausmachen. Dazu sollte die Lernpsychologie konstruktive Hilfestellung liefern können, da das Problem größerer Kontexte auf Basis isolierten Lernens auch dort untersucht wird. Entsprechende Untersuchungen folgen. Als prinzipielle Stoßrichtungen für die Suche nach Kritierien für geeignete Verbindungslemente wären zwei Wege - auch in Kombination - denkbar, die beide im :Promotionsprojekt verfolgt, unterrichtsmateriell konkretisiert und empirisch evaluiert werden:

1. Verbindungspunkte aus dem realen Unterrichtsinhalten heraus schaffen.
2. An Hand des komplexen Systems (KS) problemorientiert unterrichten und so Verbindungspunkte a priori planbar machen.

Robotik als exemplarisches komplexes System. Ein geeignetes KS muß neben seiner Komplexität auch beherrschbar sein, die Schüler motivieren (zentral!), übergreifenden Lehrzielen - z.B. hinsichtlich des Zentralabiturs - gerecht werden, viele der gefordern Lernbereiche abdecken und stark modular aufgebaut sein, damit ein Einsatz in verschiedenen Stufen bei unterschiedlichen Vorkenntnissen und Neigungen möglich ist. (Autonome mobile) Roboter liefern ein mögliches System. So kann z.B. der Roboterentwurf im Rahmen von Hardware thematisiert werden oder aber als gegebenes Modell übernommen werden. Fächerübergreifende Aspekte der Elektrotechnik, Physik und der Fächer Deutsch-Philosophie können direkt angesprochen und unterliegen einem handlungsorientierten und konstruktivistischen Zugang. Gleichfalls aber wäre es möglich, beispielsweise eine Einführung in die Programmierung gleich vor dem Hintergrund der Robotik zu motivieren. So könnten etwa Variablen durch die Notwendigkeit motiviert werden, Sensordaten abzulegen, Dateien für einen Trace eingeführt werden und Kontrollstrukturen vor dem Hintergrund eines Navigationsproblems dargestellt werden. So wird aus dem komplexen System selbst der Unterricht motiviert und bei verschiedenen Themen (Datenbanken, GUI, . . .) als umgreifendes Konstrukt begriffen und Eingeführt. Ein hybrider Ansatz ist ebenso denkbar und praktizierbar.

Literaturverzeichnis

[SM06] Schulte; Magenheim: Social, ethical and technical issues in informatics – an integrated approach, Springer verlag, 2006, url: http://www.springerlink.com-/content/658435n54601048x/

Ein Beitrag zur informatischen Bildungsforschung „Informatikunterricht zahlt sich aus“

Peter Micheuz
Institut für Informatik Systeme, Universität Klagenfurt
peter.micheuz@uni-klu.ac.at

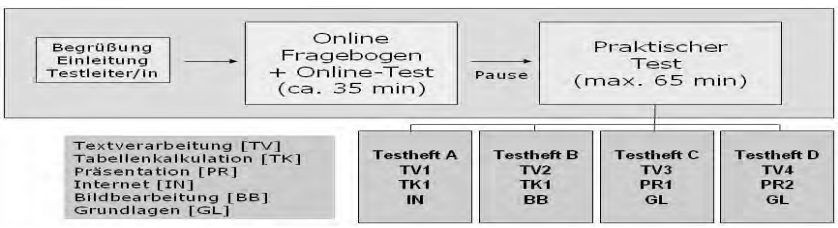
Es fällt auf, dass es unter den Publikationen zur Schulinformatik bis jetzt wenige empirische Studien zur Evaluation informatischer Kompetenzen gibt. In diesem Beitrag wird in der gebotenen Kürze ein Überblick über eine Evaluationsstudie gegeben, die in der Sekundarstufe I an den Gymnasien in Kärnten/Österreich durchgeführt wurde.

Seit 2002 ist der Informatikunterricht an 11 von 15 Kärntner Gymnasien in der 5. und 6. Jahrgangsstufe im Ausmaß jeweils einer Wochenstunde etabliert. Die Erfahrungen aus den Jahren 2002-2004, in denen der Informatik-Unterricht auf Grundlage unterschiedlicher schulautonomer Lehrpläne erteilt wurde, flossen in die Konzeption eines Informatik-Standards [Mi04] ein. Dieser wurde von den beteiligten Schulen ausgehandelt und stellt derzeit die gemeinsame curriculare Basis für die informatische Unterrichtserteilung in den ersten beiden Jahrgangsstufen der Gymnasien Kärntens dar. In Anlehnung an den ECDL wurden ca. 130 operationalisierte Lehrziele für die Module Grundlagen, Textverarbeitung, Präsentation, Tabellenkalkulation, Bildbearbeitung sowie Kommunikationstechnologie erarbeitet. Damit wurden die Weichen in Richtung verstärkter Verbindlichkeit, Vergleichbarkeit und Messbarkeit gelegt.

Im Sommersemester 2006 wurde das Projekt „Evaluation des Informatikunterrichts in den 1. und 2. Klassen der AHS in Kärnten“ durchgeführt [vgl. auch http://imst.uni-klu.ac.at/materialien/index2.php?content_id=207706]. In einem (relativ) aufwändigen Verfahren wurde die Nachhaltigkeit der in den 5./6. Jgst. erworbenen „informatischen Bildung“ evaluiert. Wichtige Fragen waren unter anderem:

- Welche schulischen Rahmenbedingungen begleiten den Informatikuntericht?
- Wie gut wurden die im Standard definierten Kompetenzen (nachhaltig) erfüllt?
- Wie korreliert das Ausmaß des Informatikunterrichts mit dem Grad „informatischer Bildung“?

Die Zielgruppen (ProbandInnen) waren ca. 20% der SchülerInnen der 7. Jgst. (3. Klasse) und involvierte LehrerInnen. Die auf für zwei Unterrichtsstunden konzipierten Testungen an den 15 Gymnasien fanden im Mai 2006 nach folgendem Ablaufschema statt:



Das Projekt ist unter http://www.schulinformatik.at/eva2006 dokumentiert, wo alle Testaufgaben (siehe Beispielaufgabe auf dieser Seite), exemplarische Auswertungen und weitere Projektdaten vorliegen. Unmittelbar nach den Testungen wurden die von den SchülerInnen bearbeiteten Aufgaben von den studentischen Evaluationsteams an den Schulen gesammelt, bewertet und in einer webbasierten Datenbank gespeichert, wo sie zusammen mit den Daten der Online-Fragebögen und des Wissenstests aufbereitet und ausgewertet wurden.


Testheft D

(2) Aufgabe zur Präsentation und Internetrecherche

Erstelle mittels einer Präsentationssoftware 2 Folien über das neue Flugzeug Airbus 380.

EIN VOGEL HEBT AB

DER AIRBUS 380

- Größe
  - Länge ??? m
  - Spannweite ??? m
  - Höhe ??? m
  - Flügelfläche ??? m²
- Reichweite ??? km
- Kapazität ??? Pers.

Viel Vergnügen mit oder ohne Vogelgrippe!

Beachte: Die im Angabeordner vorhandene Grafik airbus.jpg ist „ohne Landebahn" einzufügen.
Die Flugzeugdaten auf der 2. Folie sind im Internet zu suchen und entsprechend einzutragen.

Alle ProjektmitarbeiterInnen konnten im Zuge dieses Projekts wertvolle Theorie- und Praxiserfahrungen in Schulen sammeln. Weiters wurden wesentliche Erkenntnisse im Zusammenhang mit den Methoden der empirischen Forschung, der Fragebogen - und Testkonstruktion gewonnen. Und nicht zuletzt liefert diese empirische Untersuchung wertvolles Feedback für die lokale Schulbehörde, Schulen, LehrerInnen und SchülerInnen. Ein wesentliches Ergebnis dieser Studie ist die hohe Korrelation zwischen dem Ausmaß informatischer Unterrichtserteilung und den Testleistungen. Ein Blick auf und in die Schulwirklichkeit im Zuge von proaktiver Bildungsforschung ist auch im Bereich der Informationstechnologie mit Informatik als Bezugsfach unverzichtbar. Bildungsforschung könnte - neben ihrem beschreibenden Charakter – durch ihre normative und standardisierende Wirkung die Schulinformatik ein Stück voranbringen.

Literaturverzeichnis

[Mi04] Micheuz P., Standards at an Early Stage, Students Assessment, GI, Bonn, 2004 und Micheuz, P., Informatics Education at Austria`s Lower Secondary Schools between Autonomy and Standards. Beitrag zur ISSEP 2005 Klagenfurt, http://issep.uni-klu.ac.at

Einführung in visuelle Programmiersprachen und Mobile Endgeräte

Büdding, Hendrik
Institut Didaktik der Mathematik und Informatik
Fachbereich 10, Westfälische Wilhelms-Universität
info@m-learning.info

Wie lassen sich kerninformatische Inhalte im Informatikunterricht mit mobilen allgegenwärtigen Endgeräten vermitteln? Was ist der Mehrwert? Döring und Kleeberg [DK06] berichten über Mobiles Lernen in verschiedenen Projekten und beschreiben die technischen und didaktischen Aspekte des mobilen Arbeitens mit portablen Endgeräten im schulischen Umfeld. Aufgrund der positiven Bilanz, die dieser Text zulässt, aber auch aufgrund anderer Ergebnisse, beispielsweise des „Palm Education Pioneers Program“ bzw. Arbeiten von P. Haller („PDA macht Schule“), scheint die Entscheidung, Mobile Endgeräte für den Informatikunterricht zu nutzen, als grundlegend sinnvoll. Im Rahmen eines fachdidaktischen Ansatzes, der die Entwicklung des Ubiquitous und Pervasive Computing für den Informatikunterricht in der Schule anhand von Handheld Computern [ML06] (in der Hand gehaltene Computer / PDA) mit einer entsprechenden Schulungssoftware aufgreift, wird der Informatikunterricht um weitere Aspekte und Unterrichtsmethoden bereichert. Erfahrungsberichte zur Integration von PDAs in die Informatik-Ausbildung findet man z.B. von Luckin, Brewster et al bei Kukulska-Hume und Traxler.[1] Kukulska-Hume und Traxler beschreiben das Mobile Lernen als spontan, persönlich, tragbar und situativ; aber es kann auch informell, unaufdringlich, allgegenwärtig und durchschlagend passieren.[2] Um einen real existierenden Prozess, der bislang unvollständig oder falsch wahrgenommen wurde und dessen mentale Fehlvorstellung der Schülerinnen und Schüler (SuS) existiert, richtig zu begreifen, müssen auch mobile Lernszenarien berücksichtigt werden, die außerhalb des schulischen Computerraums stattfinden. Dabei liegt die Bedeutung der mobilen Programmentwicklung darin, innerhalb der Modellierungsphase ein möglichst genaues, abstrahiertes Modell abzubilden und den entwickelten Algorithmus anhand „realer Situationen“ zu erproben, d.h. mit realen Prozessen direkt zu vergleichen und somit gesichert zu verifizieren. Ebenso ist die VOR-ORT Programmentwicklung und -verifikation durch den Besuch des späteren realen Einsatzgebiets der IT vorteilhaft, da die SuS darin einen verstärkten Realitätsbezug bei ihrer Programmentwicklung erkennen. Erste Ergebnisse unter Verwendung der Entwicklungsumgebung „PocketCoder“ [PC06] (erprobt mit dem Fujitsu Siemens Pocket LOOX 720) stimmen optimistisch. Vor-Ort-Programmierung scheint den Modellierungsprozess zu unterstützen. Es gelingt besser, die grundsätzliche Lösungsidee für ein algorithmisches Problem zu finden und abzubilden als in einer stationären Situation. In der Praxisphase des hier vorgestellten Projektes wurde und wird untersucht, ob es möglich ist, traditionelle Inhalte des Informatikunterrichts mit Handheld PDAs zu vermitteln. Neben den technischen Möglichkeiten mobiler Computer sollen insbesondere Auswirkungen auf die inhaltliche und methodische Gestaltung des Unterrichts analysiert werden. Visuelle Programmiersprachen können den Einstieg in die Programmierung erleich-

[1] Vgl.: [KH05] S. 116-123.
[2] Vgl.: [KH05] S. 25-44.

tern. Zwar bietet laut Schiffer [Sc98] die visuelle Programmierung im Regelfall eine eingeschränktere Ausdrucksweise als ein textuelles System, aber in der Einstiegsphase im Informatikunterricht hat ein visuelles Programmiersystem den Vorteil, dass sich die Fehlerarten und -zahlen reduzieren lassen. Die SuS sollen die Syntax verstehen, die der Programmiersprache zugrunde liegt, sollen sich aber hauptsächlich mit der Modellierung und deren Implementierung in die Programmiersprache beschäftigen, um die globalen Programmierparadigmen/-konzepte zu verstehen. Eine „interaktive“ visuelle Programmiersprache, wie PocketCoder, zeigt automatisch Alternativen an, die syntaktisch an der gerade bearbeiteten Stelle zulässig sind. Durch den speziellen Aufbau der visuellen Programmiersprache, die nur eingeschränkte, syntaktisch korrekte Auswahlmöglichkeiten zulässt, sind die SuS in der Lage, schneller zu lernen und erhalten aufgrund der geringeren Fehlerquote bei der Codeerstellung ein schnelleres Erfolgserlebnis (weitere Bespiele in [Th99]). PocketCoder wurde am Arbeitsbereich Didaktik der Informatik an der Universität Münster entwickelt und basiert auf Erkenntnissen einer Bedarfsbefragung von Informatiklehrern in NRW. Das System besitzt eine ikonische Programmierumgebung, bei der mit Hilfe einer "Klicken und Auswählen"-Bedienoberfläche ein an Python angelehntes Programm erstellt werden kann. Die Idee ist, PocketCoder in den Anfangsunterricht zu integrieren, um SuS an die allgemeinen und pythonspezifischen Programmierkonzepte heranzuführen und den Brückenschlag zu komplexen Python-Entwicklungen im Informatikunterricht ggf. auf Desktop-PC zu erleichtern bzw. vorzubereiten. Die Programmiersprache ermöglicht es, mehreren PDAs bzw. Pockt-PCs unterschiedliche Aufgaben zuzuordnen und diese im Rahmen der Verteilten Programmierung miteinander über WiFi arbeiten zu lassen.

Literaturverzeichnis

[DK06] Döring, N.; Kleeberg, N.: Mobiles Lernen in der Schule. Entwicklungs- und Forschungsstand. Unterrichtswissenschaft - Zeitschrift für Lernforschung, 34 (1), S. 70-92, 2006.

[Hu06] Humbert, L.: „Stifte und Mäuse“ auf mobilen Systemen. In: Humbert, L.: If Fase (2006), Ausgabe 8, 1. April 2006;

[KH05] Kukulska-Hulme, A. et al: Mobile learning : a handbook for educators and trainers, London, 2005.

[ML06] M-Learning Projekt Handhelds@School, http://www.m-learning.info, 01.04.2007.

[PC06] PocketCoder – Projekt Universität Münster, www.pocketcoder.de, 01.04.2007.

[Sc98] Schiffer, S.: Visuelle Programmierung – Grundlagen und Einsatzmöglichkeiten. Addison-Wesley-Longman, Bonn, 1998.

[Th99] Thomas, M.: http://ddi.uni-muenster.de/personen/marco/visu/FolienVP3/index.htm (31.01.2007).

ali – Aachener eLeitprogramme der Informatik

Ulrik Schroeder, Nils J. van den Boom
{Ulrik.Schroeder, Nils.van-den-Boom}@rwth-aachen.de

Ziele. Mit dem Projekt „ali – Aachener *e*Leitprogramme der Informatik“ verfolgen wir mehrere Ziele, die dazu beitragen sollen, Lehrkräften gute Unterrichtsbeispiele zur Verfügung zu stellen und die Arbeitsweise einer „fremden“ Unterrichtsvorbereitung sowie den Einsatz digitaler Medien zu etablieren. Zum einen sollen aktuelle Unterrichtsinhalte der Informatik mit konkret ausgearbeiteten Beispielen und zahlreichen Aufgaben sowie innovativen, didaktischen Methoden entstehen. Diese Einheiten sollen im Unterricht erprobt und evaluiert werden. Darüber hinaus sollen die Materialien im Internet zur Verfügung gestellt werden und mit einer Möglichkeit versehen sein, Kommentare und Erfahrungsberichte zum eigenen Unterrichtseinsatz sowie Anregungen für die Weiterentwicklung anzuheften. Schließlich werden die Unterrichtsbeispiele durch eLearning-Komponenten angereichert und ermöglichen den Einsatz digitaler Medien und selbstgesteuerter Lernfortschrittskontrolle im Unterricht. Ein viertes Ziel verfolgen wir dadurch, dass das Projekt eng mit der Lehramtsausbildung an der RWTH Aachen verzahnt ist und durch die Ausarbeitung konkreter Unterrichtsbeispiele und die Diskussion didaktischer Entscheidungen Praxiserfahrung in die universitäre Ausbildung eingebunden sind. Die Projektmitarbeiter sind Lehramtsstudierende, die die Unterrichtseinheiten im Rahmen ihrer Fachdidaktikausbildung erarbeiten und den schulischen Einsatz begleiten.

Im Rahmen einer fachdidaktischen Projektlehrveranstaltung wurden die theoretischen Grundlagen zu Leitprogrammen und weiteren didaktischen Methoden, z.B. dem „Informierenden Unterrichtseinstieg“, die Entwicklung und Formulierung von Lernzielen und Prüfungsfragen, Prinzipien des aktiven und handelnden Lernens sowie die Abbildung von Lerneinheiten in digitalen eLearning-Sequenzen vorgestellt. In den begleitenden Übungen wurden die didaktischen Entwürfe, die gewählten Beispiele und Lernfortschrittsaufgaben diskutiert, bevor diese als komplette Unterrichtseinheiten umgesetzt wurden.

Die Lerneinheiten wurden passgenau für den Unterrichtseinsatz entwickelt. Dazu waren als Begleitung aktive Lehrkräfte von fünf Aachener und einem Bonner Gymnasium beteiligt. Die didaktische Methode „Leitprogramme“ wurde gewählt, da diese geeignet sind, einfach von Lehrkräften genutzt zu werden, die diese nicht selbst ausgearbeitet haben. Die Ergebnisse wurden auf der eLearning-Plattform „iTac.teach&learn“ der regio IT realisiert und stehen so weiteren Aachener Gymnasien zum Einsatz bereit.

eLeitprogramme. Die Leitprogramm-Methode wurde in den 90er Jahren an der ETH Zürich durch Kombination und Weiterentwicklung der Keller-Plan-Methode und dem Prinzip des Mastery-Learning entwickelt. Es handelt sich um einen komplett schriftlich vorgefertigten, aber aktiven und abwechslungsreichen Unterricht. Die Schüler-innen erhalten ein Heft zum Selbststudium mit detaillierten Arbeitsanleitungen zu einem Thema. Ein Leitprogramm organisiert das Lernen durch eindeutige Zielvorgaben und regelmäßige Fortschrittskontrollen. Der Stoff wird relativ kleinschrittig präsentiert und durch Beispiele und Aufgaben gefestigt. Dabei soll das Selbstbewusstsein und Vertrauen in das eigene Können aufgebaut werden. Durch die Selbstkontrolle wird zudem erreicht, dass Schüler-innen lernen, ihre Lernprozesse zu überwachen und mitbekommen, dass sie

Fortschritte machen und dadurch ihre Motivation gesteigert wird. Lediglich am Ende eines Kapitels gibt es einen Test, der vor der Lehrkraft (i.d.R. mündlich) abgelegt wird. Nur wenn dieser Test erfolgreich bestanden wird, darf ein-e Schüler-in gemäß des Mastery-Prinzips mit dem nächsten Kapitel fortfahren.

Einen großen Vorteil für die Lehrkräfte stellt der entspannte Unterricht dar, der mehr Zeit für Beobachtungen und gezielte individuelle Beratungen gibt. Sie kann sich nun intensiv mit einzelnen Schüler-innen befassen, was in einem normalen Unterricht eher schwierig umzusetzen wäre. Die Schüler-innen können ihr Lerntempo selbst bestimmen und eine Differenzierung ist vorbereitet.

In der eLearning-Version erzielen wir Mehrwerte durch die multimediale Präsentation einiger Sachverhalte (z.B. Simulatoren für dynamische Prozesse oder die Einbindung von Werkzeugen zum Modellieren) die interaktiven Tests, deren Lösung und Erläuterung direkt bei der Aufgabenstellung eingeblendet werden kann sowie die automatische Auswertung von Lernfortschrittskontrollen und die Freischaltung von Addita und Folgekapiteln nach erfolgreichem Absolvieren vorgeschalteter Tests.

Ergebnisse. Es wurden sechs Leitprogramme in Print- und Online-Versionen erstellt, von denen fünf (z.T. mehrfach) innerhalb der Projektschulen zum Einsatz kamen. Weitere Anfragen und Downloads von den ImaS-Seiten[1] zeigen, dass großes Interesse an den eLeitprogrammen besteht. Die ersten Rückmeldungen bestätigen den Erfolg der Methode und geben Anregungen für die weitere inhaltliche Gestaltung. Schüler-innen bleiben konzentriert bei der Sache, loben die Gestaltung, die klar vorgegebenen Lernziele, das individuelle Lerntempo und die vielen Rückmeldungen und wünschen sich insgesamt öfter einen Unterricht mit Leitprogrammen. Die Lehrkräfte konnten entspannten Unterricht durchführen, allerdings müssen noch herausfordernde Lernfortschrittskontrollen und ausführlichere Addita erarbeitet werden, um Leistungsunterschiede noch besser auszugleichen. Eine ausführliche Version der Projektbeschreibung mit Diskussion der Vor- und Nachteile gibt es unter http://lufgi9.informatik.rwth-aachen.de/infos07.

[1] Informatiklehrmaterialien für den Schulunterricht: http://lehramt.informatik.rwth-aachen.de/imas

Projekt Internetworking und E-Learning

Kirstin Schwidrowski, Christian Eibl, Sigrid Schubert
Didaktik der Informatik und E-Learning, Universität Siegen
{schwidrowski, eibl, schubert}@die.informatik.uni-siegen.de

Ziel. Dieses DFG-Projekt befasst sich mit „Internetworking“ zu den Schwerpunkten (A) Strukturen des Internet, (B) Kommunikationsbeziehungen im Internet und (C) Informationssicherheit im Internet. Dabei verstehen wir unter dem Begriff Internetworking den Zusammenschluss von eigenständigen, möglicherweise heterogenen Rechnernetzen zu einem transparent zu verwendenden Rechnernetz sowie Anwendungen und ihre Merkmale, für deren Funktion vernetzte Rechnernetze essentiell sind. Konzepte und Anwendung von Schnittstellen dienen zur Kapselung und Überbrückung unterschiedlicher Rechnernetzkonzepte in den einzelnen Subnetzen hinsichtlich der Protokolle, Adressen, Übertragungsmedien und Topologie. E-Learning wird als Lernform gewählt, um das Bildungsangebot Berufstätigen zugänglich zu machen. Ziel ist ein Didaktisches System „Internetworking“, das konzipiert und evaluiert wird, um dessen Komponenten Wissensstrukturen, Aufgabenklassen und Explorationsmodule zu verfeinern ([Fr06], [FS07]). Diese auf einander abgestimmten Komponenten dienen den Lehrenden als Gestaltungsmittel des E-Learning und zur Unterstützung des fachdidaktischen Austausches.

Forschungsmethodik. Ausgehend von der fachlichen Analyse der Schwerpunkte (A)-(C) wurde eine Auswahl von Lernzielen getroffen und deren Relevanz in der beruflichen Tätigkeit mittels einer schriftlichen Befragung von Personalverantwortlichen in kleinen und mittelständischen Betrieben ermittelt. Es zeigte sich, dass Informationssicherheit besonders wichtig ist. Auf die Analysephase (2005/2006) folgte die Konzeption einer Wissensstruktur durch Verfeinerung der Lernziele. Zudem wurden entsprechende Aufgaben erstellt und in Aufgabenklassen strukturiert gesammelt ([Co02], [KR01], [Ta04]). Diese beiden Komponenten des Didaktischen Systems werden hinsichtlich der Didaktik der Informatik auf Anwendungsorientierung und Lebensweltbezug im Berufsfeld bewertet. Kriterien des E-Learning wurden herangezogen, um in den Aufgabenklassen die Besonderheiten des selbstorganisierten Lernens zu berücksichtigen. Empirisches Arbeiten mit Probanden ermöglichte, anhand der Lösungen zu Aufgaben die Komponenten zu prüfen. Diese Arbeiten fanden im Zeitraum 2006/2007 statt. Es können Explorationsmodule eingesetzt werden, um die Kompetenzen der Wissensstrukturen und Aufgabenklassen zu unterstützen. Die Evaluationsergebnisse wurden zur Überarbeitung von Lernprozess und Lernmaterialien verwendet.

E-Learning-Prozess. Während der Vorbereitung des Arbeitens mit Probanden wurden Lernmaterialien erstellt, da für die Zielgruppe geeignete Materialien, insbesondere anwendungsorientierte Aufgaben, fehlten. Im E-Learning ist das experimentelle Arbeiten mit dem Lerngegenstand unverzichtbar. Die erste Erprobung fand in einem mittelständischen Unternehmen von Oktober 2006 bis Januar 2007 statt. Daran nahmen vier Teamassistentinnen und eine Seniorin teil. Gestartet wurde mit einer Präsenzveranstaltung im Unternehmen, um den Kontakt zwischen Lernenden und Lehrenden zu fördern und die Verteilung der Lernmaterialien zu üben. Im Anschluss folgte eine E-Learning-Phase, die mit den Lernzielen zu den Themen „E-Mail“ und „World Wide Web“ begann, um einen anwendungsorientierten Zugang mit Lebensweltbezug zu sichern. Es folgte das Thema „Passwörter“, das diese Kriterien ebenso erfüllte und somit den Einstieg in den komple-

xeren Schwerpunkt C „Informationssicherheit“ unterstützte. Wöchentlich erhielten die Teilnehmerinnen ein Lernpaket mit Lerntext, Selbsttest und Aufgaben, zu deren Lösungen per E-Mail Rückkopplung durch die Betreuerin erfolgte. Schon zu einem frühen Zeitpunkt zeigte sich, dass sich für die Teamassistentinnen die Integration des Lernprozesses in den Arbeitsprozess sehr schwierig gestaltete. Aufgrund dieser Probleme wurden die Lernziele nach sieben Wochen wiederholt, um einen Wiedereinstieg zu ermöglichen. In einer Abschlussdiskussion konnten aus der Akzeptanz Schlussfolgerungen für die Überarbeitung des Didaktischen Systems „Internetworking“ gezogen werden.

Evaluation. Die Probanden kannten nur das Eingabe- und Ausgabeverhalten von Internetanwendungen, welches aber nur Anfang und Ende eines Kommunikationsprozesses im Internet ist. Die dazwischen stattfindenden Kommunikationsschritte waren unbekannt. Der anwendungsorientierte Einstieg wurde positiv von den Lernenden aufgenommen. Es zeigte sich, dass die Teilnehmerinnen Fachbegriffe der Informatik nicht kannten und auch keine intuitive Vorstellung hatten (z. B. Zugriffsrechte) oder auch Begriffe falsch anwendeten. Die Auswahl der zu erlernenden Fachbegriffe wurde sehr sorgfältig vorgenommen. Die fehlende Kommunikation zwischen den Lernenden erwies sich als Problem. Die Lernerfolge der Seniorin zeigten, dass die fachlichen Anforderungen korrekt gewählt wurden. Bei den Berufstätigen lag durch den Druck der Vorgesetzten zur Teilnahme eine geringe Motivation zur informatischen Weiterbildung vor.

Reflexion. Aus den vorliegenden Zwischenergebnissen leiten wir für die nächste Etappe bis Ende 2007 drei wichtige Forschungsaufgaben ab. (1) Die Klassifizierung der vorliegenden Aufgaben und die Erweiterung für eine strukturierte Aufgabensammlung entsprechend den Anforderungen an das Didaktische System. (2) Die Erarbeitung eines speziellen Zugangs zur Fachsprache für diese Lernenden ohne Informatikvorkenntnisse ist realisierbar durch eine Brücke zwischen der umgangssprachlichen Verwendung und der fachlichen, die durch eine schrittweise Erweiterung der Informatikbegriffe erreicht wird. (3) Um den Erfahrungsaustausch zwischen den Lernenden zu fördern, kommt eine Kommunikationssoftware mit Moderation zum Einsatz. Diese drei Teilaufgaben münden in eine Überarbeitung der Komponenten des Didaktischen Systems „Internetworking“, das insgesamt Bildungsstandards der Informatik, in diesem Fall in der Erwachsenenbildung, etablieren hilft. Der Schwerpunkt liegt dabei auf dem Verstehen von Informatiksystemen, die mit Aufbau und Funktionsweise charakterisiert werden, aber auch deren kompetente Anwendung einschließlich Interaktion und Kommunikation umfassen.

Literaturverzeichnis

[Co02] Comer, D.: Computernetzwerke und Internets: mit Internet-Anwendungen. Pearson Studium, München 2002.

[Fr06] Freischlad, S.: Beitrag des Informatikunterrichts zur Entwicklung von Medienkompetenzen. In: Schwill, A.; Schulte, C. ; Marco, T.: Didaktik der Informatik: 3. Workshop der GI-Fachgruppe „Didaktik der Informatik“, 19.-20. Juni 2006, Potsdam, 2006.

[FS07] Freischlad, S.; Schubert, S.: Towards High Quality Exercise Classes for Internetworking. Joint IFIP-Conference Informatics, Mathematics and ICT (IMICT2007): A golden triangle. Boston, USA, 27th–29th June 2007.

[Ta04] Tanenbaum, A.: Computernetzwerke. Pearson Studium, München 2002.

[KR01] Kurose, J. F.; Ross, K. W. (2001) Computer Networking: A Top-down Approach Featuring the Internet. Addison-Wesley Longman, Amsterdam, 2001.

Analog denken – analog programmmieren

Michael Weigend
Institut für Didaktik der Mathematik und der Informatik
Westfälische Wilhelms-Universität Münster
michael.weigend@uni-muenster.de

Aristoteles (384-322) verwendete den Begriff „geometrische Analogie“ zur Beschreibung der Gleichheit zweier quantitativer oder qualitativer Verhältnisse [Co02]. Beispiele: „1 verhält sich zu 2 wie 5 zu 10“ oder „Die Federn sind für den Vogel, was die Schuppen für den Fisch sind“. Eine informatische Modellierung beginnt manchmal mit der Formulierung qualitativer geometrischer Analogien. Ein Beispiel ist die Wiedergabe von Ferne und Nähe in einem Programm mit zweidimensionaler Grafik: (1) Entfernte Objekte befinden sich weiter oben und nahe Objekte weiter unten auf dem Bildschirm. (2) Entfernte Objekte erscheinen klein und nahe Objekte groß. (3) Entfernte Objekte sehen blass und bläulich aus, nahe Objekte haben eine hohe Farbbrillanz (Farbperspektive). Um diese Analogien in das Programm einfließen zu lassen, müssen sie quantifiziert werden. Für die Beschreibung eines plausiblen Zusammenhangs zwischen räumlicher Ausdehnung und vertikaler Position auf dem Bildschirm kann man auf den Strahlensatz zurückgreifen. Die Farbperspektive lässt sich so simulieren, dass über die Abbildung eines Objektes als Filter eine deckungsgleiche blaue Fläche gelegt wird, deren Transparenz in Abhängigkeit von der Position auf der Ebene eingestellt wird. Je „näher“ das Objekt ist, desto durchlässiger ist der Filter [We07].

Gebräuchlich ist die Unterscheidung zwischen analoger und digitaler Repräsentation von Wissen [Ma05]. Eine digitale Repräsentation verwendet Symbole einer Sprache und stellt die abzubildende Entität explizit durch ein Literal dar. So kann die Farbe des Himmels durch das Wort „blau“ oder das RGB-Tupel (30, 30, 250) beschrieben werden. Analoge Repräsentationen gründen auf geometrischen Analogien. Ein hoher Balken in einem Balkendiagramm verhält sich zu einem niedrigen Balken wie eine große Zahl zu einer kleinen. Analoge Repräsentationen werden unmittelbar „erlebt“, während digitale Repräsentationen (unter Rückgriff auf ein Symbolsystem) zuerst kognitiv verarbeitet werden müssen um wahrgenommen zu werden [Ma05]. Graphische Benutzungsoberflächen (GUI) enthalten analoge Ein- und Ausgabemechanismen. Dazu gehören z.B. die Veränderung der Fenstergröße, das Verstellen von Schiebereglern und andere direkte Manipulationen auf dem Bildschirm mit der Maus. Im Informatikunterricht können verschiedene Aspekte analoger Repräsentation von Daten im Rahmen eines Projektes aufgegriffen werden, bei dem die GUI-Gestaltung im Zentrum steht.

Analoges Denken spielt im Zusammenhang mit Polymorphie in der OOP eine Rolle. Eine Form der Polymorphie ist das Überladen von Operatoren. Das bedeutet, dass ein Operator je nach dem Objekt, auf das er angewendet wird, die Ausführung einer anderen Operation bewirkt (z.B. Operator + für Addition von Zahlen und Konkatenation von Strings). Polymorphie kann zu verständlicheren Programmtexten führen. Der kognitive Vorteil liegt darin, dass vertraute Konzepte (wie die Addition) analog auf neue Domänen angewendet werden. Eine Herausforderung im Rahmen eines Programmierprojektes in der Schule ist es, für eine selbst definierte Klasse eine Methode zu erfinden, die einer aus anderen Kontexten bereits bekannten Operation analog ist, und den entsprechenden

Operator zu überladen. Beispiel: Definiere eine Klasse „Color“, die Farben repräsentiert. Implementiere eine plausible Methode für die Addition zweier Farbobjekte.

Bei komplexeren Analogien werden nicht nur einzelne Begriffe oder Größen sondern strukturierte Systeme aus mehreren Komponenten zueinander in Beziehung gesetzt. Beispiele für den Mathematikunterricht findet man in [En97]. Solche strukturellen Analogien werden häufig bei Programmentwicklungen herangezogen, wenn es darum geht, eine grundsätzliche Lösungsidee für ein algorithmisches Problem zu finden. Im agilen Programmieren beginnt eine Softwareentwicklung mit der Formulierung einer Metapher, welche die Idee des gesamten Projektes beschreibt [Be99]. Für das Problem der Suche nach einem Objekt auf einer Fläche können beispielsweise folgende Alltagssituationen herangezogen werden: (1) Eine Gruppe von Polizisten durchkämmt linear ein Waldstück um ein Beweisstück zu suchen. (2) In ein Planschbecken mit schwimmenden Gummienten wirft jemand einen Stein. Es entsteht eine kreisförmige Welle, die sich in alle Richtungen ausbreitet. Die Gummiente, die der Einwurfstelle am nächsten liegt, wackelt zuerst. (3) Ein Wanderer steht auf einem Hügel. Er sucht sein Ziel, indem er sich langsam dreht und den Blick schweifen lässt.

Im Rahmen des Modellbildungsprozesses müssen solche Situationen, die Lösungsideen repräsentieren, zunächst einmal gefunden, dann hinsichtlich Struktur und Dynamik analysiert und im Hinblick auf Brauchbarkeit (Implementierbarkeit, Effizienz etc.) bewertet werden. Schließlich wird ein System entwickelt, das (in gewissen Grenzen) der gewählten Situation analog ist.

Literaturverzeichnis

[Be99] Beck, Kent (1999): Extreme Programming Explained. Addison Wesley, Boston u.a. 1999.

[Co02] Coenen, Hans Georg: Analogie und Metapher. Grundlegung einer Theorie der bildlichen Rede. Walter de Gruyter, Berlin 2002.

[En97] English, Lyn D. (Hrsg.): Mathematical Reasoning. Analogies, Metaphors, and Images. Lawrence Erlbaum Associates, Publishers, Mahwah New Jersey, London 1997.

[Ma05] Matthen, Mohan: Seeing, Doing, and Knowing: A Philosophical Theory of Sense Perception. Oxford University Press, Oxford 2005.

[We07] Weigend, Michael: Verwendung geometrischer und struktureller Analogien bei der Gestaltung von Flash-Animationen – Galerie. URL: http://www.creative-informatics.de/analog/

Workshop

„Didaktik der Informatik – aktuelle Forschungsergebnisse“

Zielorientierte Didaktik der Informatik – Kompetenzvermittlung bei engen Zeitvorgaben

Nicole Weicker

Lehrbeauftragte für Didaktik der Informatik
an der Universität Leipzig
nicole@weicker.de

Abstract: Die Informatiklehre steht vor der Aufgabe, in kürzerer Zeit mehr Kompetenzen zu vermitteln. Die zielorientierte Didaktik der Informatik bestehend aus den didaktischen Prinzipien Lernzielorientierung, Anknüpfung an Vorwissen, enge Verzahnung von Input und Aktivität, Motivierung und frühzeitiges Feedback bzgl. aller Lernziele wird allgemein und in ihrem Einsatz in der Hochschullehre vorgestellt.

1 Einleitung

Die Rahmenbedingungen für die Vermittlung von Informatik an Hochschulen haben sich in den letzten Jahren stark verändert: Auf der einen Seite sollen mehr Kompetenzen vermittelt werden, da sich die Informatik von einer vorrangigen Erforschung der Möglichkeiten der Informationsverarbeitung zur Dienstleistungswissenschaft mit starkem Ingenieurscharakter für nahezu alle anderen Disziplinen entwickelt hat. Auf der anderen Seite sind die zeitlichen Vorgaben, in denen die Lernenden diese Kompetenzen erwerben sollen, verschärft worden [GI05]. Durch die politisch gewollte Vorgabe, 70% der Studierenden nach dem Bachelorabschluss ins Berufsleben zu schicken, entsteht die Notwendigkeit, die entsprechenden berufsbefähigenden Kompetenzen in wenigen Semestern zu vermitteln. Diesen Herausforderungen hat sich die Informatiklehre ohne größere Personalaufstockungen zu stellen.

Um der Anforderung – Vermittlung von mehr Kompetenzen in Informatik in kürzerer Zeit – gerecht werden zu können, ist die Didaktik der Informatik gefragt, neue passende Antworten zu finden. Die in dieser Arbeit vorgestellte zielorientierte Informatiklehre ist eine mögliche Reaktion auf diese Herausforderungen. Sie beinhaltet die enge Kombination der didaktischen Prinzipien: Lernzielorientierung, Anknüpfung an Vorwissen, enge Verzahnung von Input und Aktivität, Motivierung und frühzeitiges Feedback bzgl. aller Lernziele. Die Umsetzung der zielorientierten Informatiklehre wird an fünf Beispielen aus der Hochschullehre aufgezeigt: Workshopseminar, kooperatives Lernen im Übungsbetrieb des Grundstudium, aktivierende Methoden in Großveranstaltungen, Kombination von Spezialvorlesung und Übung sowie ideenorientiertes Spiralkonzept für Vorlesungen. Allen Beispielen ist gemein, dass sie ohne zusätzliches Personal durchgeführt wurden und sich der zeitliche Aufwand für die Lehrenden im Rahmen hielt.

2 Didaktische Antworten und ihre Grenzen

Zwei wichtige Lerntheorien, die kombiniert vielen didaktischen Aspekten der zielorientierten Lehre zugrundeliegen, sind der Kognitivismus und der Konstruktivismus [KW01].

Kognitivistisches Lernen

Dem kognitivistischen Lernmodell liegen Modelle der neuronalen Verarbeitung von Wissen zugrunde. Lernen wird im Kognitivismus als streng regelhaft verlaufender Prozess des Wissenserwerbs verstanden, der erfolgreich von Außen gesteuert werden kann. Die Lerninhalte werden als fertiges System vom Lehrenden vermittelt. Die Lernenden sind während dieser Vermittlung passiv. Beim kognitivistischen Ansatz steht der Wissensinput durch den Lehrenden im Mittelpunkt. Das eigentliche Lernen findet dabei in der Regel nicht während der Lehrveranstaltung sondern in Eigenarbeit statt. Ein hauptsächlich kognitivistisches Vorgehen impliziert, dass die Lernenden unabdingbar viel Zeit zum Reifen ihres kognitiven Wissens benötigen. Kompetenzen in affektiver und pragmatischer Hinsicht werden vom Kognitivismus nicht explizit gefördert. Ihr impliziter Erwerb kann damit nicht von allen Lernenden im gleichen Maße vorausgesetzt werden.

Für die Informatiklehre ist die strukturierte Wissensvermittlung in Form von Frontalunterricht in Vorlesungen insbesondere im üblichen Massenbetrieb in den ersten Semestern unverzichtbar. Um jedoch in kürzerer Zeit (sprich mit weniger individueller benötigter Reifungszeit) kognitive Kompetenzen erreichen zu können, sind die Prinzipien des Kognitivismus um weitere didaktische Elemente zu erweitern.

Konstruktivistisches Lernen

Im konstruktivistischen Ansatz wird Wissen als individuelle Konstruktion aufgefasst. Lernen ist vor diesem Hintergrund ein aktiver, konstruktiver Prozess in einem bestimmten Handlungskontext. Dazu hat die Lernumgebung vor allem Situationen anzubieten, in denen der Lernende eigene Konstruktionsleistungen erbringen kann. Unterrichten bedeutet in diesem Kontext mehr Unterstützen, Anregen und Beraten. Der Lehrende tritt in einer reaktiven Rolle hinter der aktiven Rolle des Lernenden zurück.

In diesem lerntheoretischen Modell findet das Lernen während der Veranstaltungen statt. Dadurch können die Lernenden neues Wissen direkt in ihrem individuellen Vorwissen verankern. Durch die eigene Auseinandersetzung wird Praxis und Wissensbildung miteinander verbunden. Zusätzlich hat jeder Lernende die Möglichkeit, eigene, dem eigenen Lerntyp angepasste Lernwege zu beschreiten. Allerdings bedeutet diese Art des Vorgehens, dass im Zweifel jeder Lernende „das Rad neu erfinden“ muss. Die Lerngeschwindigkeiten der Lernenden sind in aller Regel sehr unterschiedlich und die inhaltlichen Ziele sind im Vornherein nicht abschätzbar. Zusammenfassend ist der Konstruktivismus sehr zeitaufwändig. Durch zusätzliche Wissensinputphasen (wissensbasierter Konstruktivismus) ist es möglich, dass die Lernenden zumindestens auf einer Wissensbasis aufbauend lernen können, statt sich alles selbst erarbeiten zu müssen. Dennoch bleiben die anderen Nachteile des Konstruktivismus bestehen.

Im Rahmen der Informatiklehre ist der Konstruktivismus trotz seiner unbestreitbaren Nachteile von wichtiger Bedeutung, da durch konstruktivistische Lernphasen insbesondere pragmatische und affektive Lernziele angestrebt werden können, die im Kognitivismus der individuellen Auseinandersetzung des Einzelnen überlassen bleiben.

3 Zielorientierte Didaktik

Die im weiteren beschriebene zielorientierte Didaktik für Informatik folgt der Anforderung, in kürzerer Zeit fundierte Kompetenzen in allen Bereichen der Informatik zu vermitteln. Während der wissensbasierte Konstruktivismus kognitivistische Ansätze (Wissensinput) mit der Theorie des Konstruktivismus verbindet, verknüpft die zielorientierte Didaktik kognitivistisches Vorgehen, wann immer kognitive Lernziele angestrebt werden, mit konstruktivistischen Aspekten für die pragmatischen und affektiven Lernziele.

Lernzielorientierung

Ein wichtiges Standbein der zielorientierten Didaktik ist die Bestimmung der jeweils angestrebten Lernziele bzgl. aller Lernbereiche, damit ausgehend von den Lernzielen die Inhalte und deren Strukturierung festgelegt und geeignete Lehrmethoden bestimmt werden können.

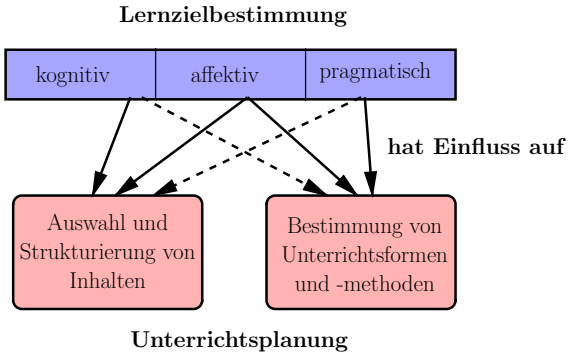


Abbildung 1: Zusammenhang zwischen Lernzielbestimmung und Unterrichtsplanung

Wie in Abbildung 1 dargestellt, beeinflussen vorwiegend die kognitiven und affektiven Lernziele die Auswahl der Inhalte und deren Strukturierung, während die affektiven und pragmatischen Lernziele bei der Bestimmung geeigneter Unterrichtsformen und -methoden berücksichtigt werden sollten. Didaktische Anhaltspunkte für die Auswahl von Inhalten bietet das Konzept des exemplarischen Lernens. Mögliche Strukturierungen sind ideen- oder problemorientierte Spiralgliederungen von Vorlesungen. Unterrichtsformen können neben Frontalunterricht und Übungsgruppen auch die Kombination von Vorlesung und Übung sein. Eine weitere Unterrichtsform, die in der Informatiklehre von besonderer Bedeutung ist, ist die Organisation von Lern- oder Arbeitsgruppen über kooperative Kon-

zepte. Unterrichtsmethoden sind beispielsweise der Einbau von Lernstopps in den Vorlesungsfluss oder Gruppenarbeitsphasen im kombinierten Vorlesungs-/Übungskonstrukt.

Anknüpfung an Vorwissen

Eine weitere wichtige Komponente der zielorientierten Didaktik ist die explizite Thematisierung und Nutzung von vorhandenem Vorwissen. Aus der Gehirnforschung ist bekannt, dass sinnvolles Lernen nur unter Einbeziehung des individuellen Vorwissens möglich ist [Ve04]. Ein großes Problem gerade in großen Veranstaltung wie z.B. die Vorlesung „Einführung in die Informatik“ im ersten Semester des Informatikstudiums besteht darin, dass die Vorkenntnisse der Studierenden unbekannt sind. Zusätzlich besuchen nicht alle Studierenden die gleichen Parallelveranstaltungen, so dass sich der Dozent nicht darauf verlassen darf, dass bestimmte Inhalte in anderen Vorlesungen wie z.B. „Mathematik für Informatiker“ behandelt werden. Die wesentlichen Aspekte zur Nutzung von Vorwissen ist die Feststellung des Vorwissens, eventuell die Schaffung von Vorerfahrungen und die tatsächliche Aktivierung des Vorwissens.

Um explizit Vorwissen aktivieren zu können, ist es notwendig, dass der Dozent die unterschiedlichen Vorwissens- und Vorerfahrungstiefen der Studierenden erfragt. In kleinen Veranstaltungen kann dies direkt abgefragt und beispielsweise an der Tafel festgehalten werden. In größeren Veranstaltungen kann Vorwissen über Zettelabfrage, sprechende Wand (Wandplakate, in die die Studierenden über Klebepunkte oder Kreuze ihre Vorerfahrungen in vorgegebene Kategorien eintragen) oder über die Nutzung von Online-Tools (z.B. OPAL an der HTWK Leipzig) erfragt werden. Interessant ist für die „Einführung in die Informatik“ unter anderem in welcher Programmiersprache in welchem Umfang Kenntnisse vorhanden sind und welche Methoden und Darstellungformen der objektoriertierten Modellierung bei wie vielen Studierenden vorausgesetzt werden können.

In vielen Fällen ist auch ohne explizite Befragung Vorwissen z.B. aus früheren Veranstaltungen bekannt. Dennoch genügt es nicht, dass das Vorwissen theoretisch vorhanden ist. Vielmehr ist es notwendig, das vorhandene Vorwissen in Gedächnis zu rufen und damit zu aktivieren. Nur dann können die neuen Inhalte und Methoden direkt mit dem Bekannten verknüpft werden. Anderenfalls findet diese Verknüpfung zu einem individuell unterschiedlich späteren Zeitpunkt statt. Im Sinne der zielorientierten Didaktik wird angestrebt, dass derartige Verknüpfungen so bald wie möglich hergestellt werden. Aktiviert werden kann Vorwissen z.B. über eine kurze Wiederholung. Je aktiver die Studierenden bei dieser Wiederholung sind, desto wahrscheinlicher liegt das Vorwissen danach präsent im Gedächnis vor.

Eine andere Form der Nutzung von Vorwissen ist die explizite Schaffung von Vorerfahrung auf die im weiteren Verlauf der Veranstaltung direkt zurückgegriffen wird.

Enge Verzahnung von Input und Aktivität

Für eine zielorientierte Didaktik ist es notwendig, den Abstand zwischen kognitivem Input zu bestimmten Inhalten und deren praktische Umsetzung so gering wie möglich zu halten. Durch die Praxisphasen können die kognitiven Inhalte besser im Gedächnis verankert und

um eigene Erfahrungen vertieft werden. Zusätzlich tauchen dadurch Fragen früher auf und Verständnisprobleme können gezielt angegangen werden.

Aus diesem Grund sollten in jede Veranstaltung Praxismöglichkeiten integriert werden. In Vorlesungen kann dies über aktivierende Methoden (z.B. Lernstopp oder Bienenkorb) umgesetzt werden. Bei kleineren Veranstaltungen kann dies über eine Kombination von Vorlesung und Übung stattfinden (4(V+Ü) statt 2V + 2Ü).

Im Kontext der Informatiklehre besteht insbesondere bei theoretischen oder abstrakten Konzepten das Problem, dass die Studierende das Konzept in der Vorlesung zwar nachvollziehen können, zuhause bei einer eigenen Umsetzung in Form der Lösung von Übungsaufgaben daran scheitern. Der Abstand zwischen Vorlesung und der praktischen Übung durch Übungsaufgaben ist häufig so groß, dass kognitive Inhalte bzw. das in der Vorlesung vorhandene Verständnis bereits wieder vergessen wurden. Im Gegensatz zu anderen Fächern ist in der Informatik (fast) immer ein Praxisbezug vorhanden, der explizit herausgearbeitet werden sollte.

Motivierung

Unter Motivierung wird in dieser Arbeit in erster Linie die Motivierung der Lernenden und nicht die Einführung eines Themas verstanden. Die wesentlichen Einflussfaktoren (siehe Abbildung 2), durch die Lehrende die Lernenden motivieren können sind

- Vermittlung von Begeisterung durchs eigene Vorbild
- ansprechende (interessante) Themen
- Alltagsbezug aus der Lebenswelt der Lernenden
- Übertragung von Verantwortung (beispielsweise innerhalb eines stark selbstbestimmten Projekts)
- sozialer Kontext (Gruppenarbeit oder Wettbewerbssituationen)
- Scheinbedingungen oder Prüfungsleistungen.

Abbildung 2: Einflussfaktoren zur Motivierung von Lernenden

Besonders die letztere Motivierungsmöglichkeit sollte nicht außer Acht gelassen werden. Auch wenn es sich dabei um eine nicht inhaltlich begründete Motivation handelt, kann diese hilfreich sein, so dass die Lernenden sich überhaupt auf die Aufgaben oder eine aktive Auseinandersetzung mit den Inhalten einlassen und die anderen Motivierungspunkte greifen können.

Frühzeitiges Feedback bzgl. aller Lernbereiche

Ein wichtiger Bestandteil der zielorientierten Lehre ist die frühzeitige Rückmeldung an die Lernenden, in welchen Punkten sie wie gut sind und welche Verbesserungsmöglichkeiten es noch gibt. Ein derartiges Feedback auf den Leistungsstand erfolgt in der üblichen kognitiven Lehre durch die Rückmeldungen auf die Lösung von Übungsaufgaben und durch Prüfungsergebnisse. Diese Feedback-Praxis hat einige Nachteile. Zum einen gelingt es immer wieder Studierenden trotz z.T. scharfer Scheinbedingungen sich durch die Übungsaufgabenverpflichtungen zu „mogeln“, sei es durch die Mitarbeit in einer Gruppe, bei der sie nicht tatsächlich zur Lösung der Aufgaben beitragen oder sei es durch Abschreiben von fertigen Lösungen. Auf diese Art und Weise bringen sich diese Studierende selbst um eine Rückmeldung auf ihre tatsächliche Leistungen und bei den Prüfungen werden sie von ihrem eigenen Unvermögen überrascht. Auf der anderen Seite erhalten die Studierenden kaum oder keine Rückmeldung auf ihren Stand bzgl. affektiver oder pragmatischer Lernziele.

Die Basis zu einem funktionierendem Feedbackprozess besteht darin, dass der Lehrende alle Lernziele seiner Veranstaltung für sich kleinschrittig und bzgl. aller Lernebene formuliert. Dies sollte in aller Regel vor der eigentlichen Veranstaltung stattfinden. Während der Durchführung der Veranstaltung sollte der Lehrende die jeweils aktuellen Lernziele thematisieren und längerfristige Lernziele immer wieder wiederholen, so dass diese den Lernenden präsent sind. Zusätzlich sollte jeder Lernende regelmäßig alle paar Wochen ein Feedback auf seine Fähigkeiten (kognitiv), Fertigkeiten (pragmatisch) und Haltungen (affektiv) erhalten. In Abbildung 3 werden beispielhaft einige Möglichkeiten aufgezeigt, wie ein regelmäßiges Feedback auch in größeren Veranstaltungen möglich ist.

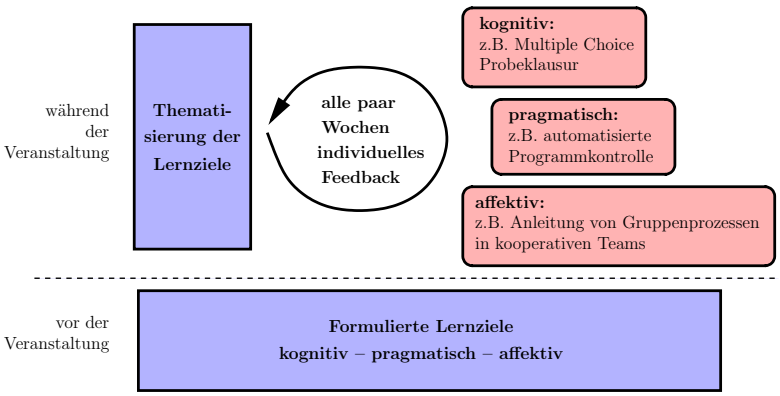


Abbildung 3: Funktionierender Feedbackprozess auch für große Vorlesungen mit Übungsbetrieb

4 Beispielhafte Umsetzungen in der Informatiklehre

In diesem Abschnitt wird anhand von einigen konkreten Beispiele vorgeführt, wie die sich zielorientierte Didaktik der Informatik konkret umsetzen lässt. Alle diese Beispiele sind an Universitäten und/oder Fachhochschulen exemplarisch umgesetzt worden. Für viele der Beispiele gibt es Evaluationen über Fragebögen.

Seminar als Workshop

Die wesentlichen Lernziele eines Seminars sind neben der vertieften Beschäftigung mit fachlichen Inhalten die folgenden.

- wissenschaftliche Texte (zunehmender Komplexität) lesen und verstehen,
- fundierte Literaturarbeit (Recherche) durchführen,
- wissenschaftliche Inhalten in eigenen Worten verständlich wiedergeben,
- die Fachsprache Informatik verwenden,
- eine wissenschaftliche Arbeit präsentieren,
- passende Präsentationsmedien einsetzen,
- die Verteidigung der eigener Arbeit üben (für Studien-, Bachelor-, Master- oder Diplomarbeit im Studium, für eine eigene Arbeit gegenüber Kollegen, Geldgebern, Kunden, Vorgesetzten im späteren Arbeitsleben),
- eine Ausarbeitung erstellen, die die wissenschaftliche Arbeit, beschreibt, erklärt und kritisch hinterfragt,
- die eigene Ausarbeitung und Vortrag kritisch beurteilen.

Im Bestreben, das Erreichen dieser Lernziele innerhalb eines Seminars an der Hochschule so effektiv wie möglich zu fördern, wurde an jeweils zwei Universitäten und Fachhochschulen ein spezielles Workshopseminarkonzept durchgeführt [WDW06]. In Anlehung an übliche wissenschaftliche Workshops und Konferenzen findet das eigentliche Vortragsseminar als Blockveranstaltung gegen Ende des Semesters statt. Zusätzlich zu den üblichen Aktivitäten wie der Einreichung eines Exposés, dem Peer-Review, dem Verfassen einer wissenschaftlichen Ausarbeitung für den Tagungsband und dem eigentlichen Vortrag finden eine Reihe von Vorträgen des Dozenten statt, in dem Wissen zu ausgewählten Kompetenzen vermittelt wird. Im Laufe des Semesters werden die Studierenden parallel zu ihrer eigenen Tätigkeit bzgl. der jeweils anfallenden Arbeitsaufgaben geschult. In Abbildung 4 sind die drei Ebenen dargestellt, die das Workshopseminar charakterisieren: Input, Praxis und Feedback. Die gestrichelten Elemente zeigen, welche Elemente nur bei einigen Workshopseminaren an den Universitäten Stuttgart und Leipzig, der FH Braunschweig-Wolfenbüttel und der HTWK Leipzig durchgeführt wurden. Benotet wurden jeweils die endgültige Ausarbeitung, die Gutachten, der Vortrag und die Teilnahmen an den Diskussionen im Verhältnis 30:30:30:10.

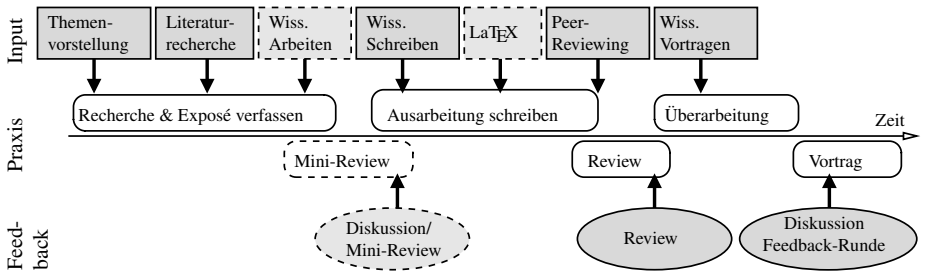


Abbildung 4: Ablauf des Seminars als Workshop

Inhaltlich erhalten die Studierenden je nach Titel des Seminars einen speziellen Auftrag. Im Seminar „Informatik und Gesellschaft“ an der Universität Leipzig sollen sie beispielsweise gegensätzliche Positionen zu ihren speziellen Vortragsthema gegenüberstellen. Im Seminar „Evolutionäre Algorithmen“ oder „Selbstreproduzierende Programme, Viren und Würmer“ an der HTWK Leipzig sollen sie die Techniken, die sie vorstellen, selbst umgesetzt haben und über diese Umsetzung berichten. Im Fachseminar „Didaktik der Informatik“ ist die Aufgabe der Studierenden Unterrichtsentwürfe aus der Literatur vorzustellen und anschließend eine eigene Position dazu zu beziehen bzw. einen alternativen Unterrichtsentwurf vorzulegen. Die Kernaufgabe wird thematisiert als die aktive Aufbereitung von Literatur und der Frage des Mehrwerts der Seminararbeit: „Warum sollte jemand die Seminararbeit statt der Originalliteratur lesen?“

Bereits früh kann im Unterschied zu anderen Seminaren festgestellt werden, dass sich die Studierenden aktiver mit ihren Themen und insbesondere mit der Literatur auseinander setzen. Die Diskussion der Exposés wird von den Studierenden als anstrengend wahrgenommen. Auf der anderen Seite wird von vielen das Feedback begrüßt, dass ihnen hilft, ihre eigene Position zu schärfen. Die Studierenden sind trotz des Mehraufwands stark motiviert und die Ausarbeitungen und Vorträge sind qualitativ besser als übliche Seminarbeiträge. Zusätzlich fällt es leichter, inhaltliche Diskussionen anzuregen, da die verschiedenen Vortragsthemen durch die Vorstellung der Zusammenfassungen allen bekannt sind und jede Ausarbeitung von mehreren Studierenden begutachtet wurde. Im Feedback am Ende der Seminare wird einheitlich die inhaltliche Vermittlung von Grundlagen zur Wissenschaftsarbeit sowie das intensive Erleben am Workshop-Tag gelobt.

Der Mehraufwand für den Dozenten ist die Vorbereitung und das Halten der Inputsequenzen. Dieser Aufwand amortisiert sich bei Wiederholung dieses Konzepts schnell. Weitere zusätzliche Zeit ist für die Diskussion der Exposés sowie die Bewertung der Gutachten notwendig.

In diesem Beispiel der zielorientierten Didaktik wurden die Konzepte der Lernzielorientierung, der engen Verzahnung von Input und eigener Aktivität sowie einem frühzeitigen und umfassenden Feedback direkt durch organisatorische und strukturelle Vorgaben umgesetzt. Neben einer Motivation über den sozialen Kontext der Seminargruppe kann über ansprechende Themen und Alltagsbezug motiviert werden. Allerdings hat auch die extrinsische Motivation über die Benotung einen wesentlichen Anteil.

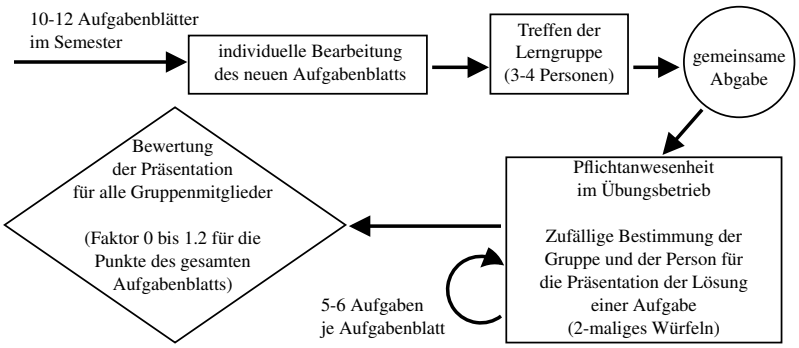


Abbildung 5: Ablauf des kooperativen Lernens im Übungsbetrieb

Kooperatives Lernen im Grundstudium

In den ersten Semestern eines Informatikstudiums sind die Einführungsveranstaltungen häufig durch große Studierendenzahlen, verschiedene Studiengänge und z.T. stark heterogenen kulturellen Hintergrund der Studierenden gekennzeichnet. Je nach Studienort fällt es den Studierenden leichter oder schwerer, sich selbstorganisierend zu Arbeits- und Lerngruppen zusammenzufinden. Aus individuellen Befragungen von Studierenden in mündlichen Nachprüfungen im Studiengang Informatik hatten viele im Studium keine Kommilitonen, mit denen sie zusammen gearbeitet oder gelernt hätten. Neben der Übung fachlicher und methodischer Informatikinhalte im Übungsbetrieb sollen die Studierenden in den ersten Semestern lernen, sich im Studium selbst zu organisieren und individuell effektive Lernstrategien anzuwenden. Daneben ist es wichtig, dass sie sozialen Kontakt zu ihren Kommilitonen aufbauen und lernen, im Studium von einander zu profitieren (Entwicklung von Kommunikation- und ersten Teamkompetenzen).

An der Universität Stuttgart wurde im WiSe 2005/06, SoSe 2006 ein verpflichtend kooperativer Übungsbetrieb zur Veranstaltung „Einführung der Informatik I+II“ mit knapp 300 Studierenden eingeführt. Das Konzept (positive Abhängigkeit innerhalb von vorgegebenen Kleingruppen, individuelle Verantwortlichkeit für die eigene Arbeit und das Produkt der Kleingruppe, direkte Zusammenarbeit in der Kleingruppe, Notwendigkeit der Entwicklung von sozialen Kompetenzen und regelmäßige Gruppenreflexionen) [FB01], seine Durchführung (siehe Abbildung 5) und Ergebnisse von Akzeptanzbefragungen sind ausführlicher in Weicker et al. [WDW06] beschrieben. Den dort berichteten Erfahrungen ist noch hinzu zu fügen, dass die Studierenden im folgenden Semester in der Veranstaltung „Einführung in die Informatik III“ weiter kooperativ zusammen arbeiteten und ohne Scheinzwang die Übungsaufgaben weiterhin lösten. Im Vergleich zu früheren Jahren erwarben ca. 60% (sonst üblich 15–20%) der Studierenden den Schein zu dieser Veranstaltung.

Aktivierende Methoden

Insbesondere in großen Vorlesungen, in denen Frontalunterricht zwangsläufig die bestimmende Unterrichtsform ist, kann über kurze „Denk- bzw. Diskussionspausen“ eine enge

Verzahnung von Input und Aktivität erreicht werden. Dabei stellt der Dozent eine kurze Frage oder Aufgabe, die die Studierenden entweder allein oder in Zweier- bis Dreiergruppen lösen sollen [FB03]. Dabei gibt es zwei wichtige Prinzipien, die bei der Durchführung derartiger aktivierender Methoden zu beachten sind, damit tatsächlich eine aktivierende Wirkung auf möglichst viele Studierende erreicht wird. Auf der einen Seite sollte einer solche Pause nicht länger als 2–3 Minuten angesetzt werden, damit die Diskussionen in den Kleingruppen sachbezogen bleiben. Wer in dieser Zeit keine Idee/Lösungsansatz o.ä. gefunden hat, wird auch nach 10 Minuten nicht weiter sein. Auf der anderen Seite ist es entscheidend, dass der Dozent direkt Studierende um ihre Lösung bittet. Wenn nur die Studierenden aufgerufen werden, die sich selbst melden, werden die übrigen beim nächsten Mal wenig Veranlassung haben, sich selbst Gedanken zu machen. Besonders gut ist es, mehrere Studierende auch aus den hinteren Reihen um ihre Antwort zu bitten, bevor diese kommentiert werden.

Die Beispiele in Abbildung 6 stammen aus der Vorlesung „Algorithmen und Datenstrukturen“ im SoSe 2007 an der HTWK Leipzig.

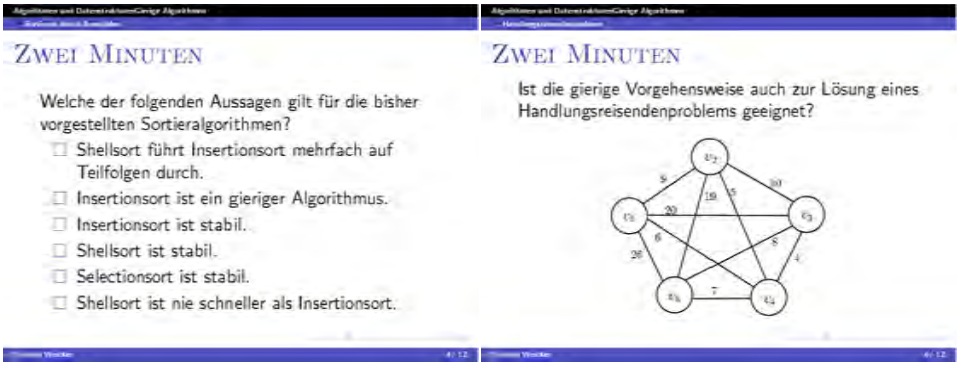


Abbildung 6: Aktivierende Fragen aus der Vorlesung „Algorithmen und Datenstrukturen“.

Kombination von Vorlesung und Übung

An den Universitäten Stuttgart und Leipzig wurde in verschiedenen Kontexten eine Kombination Vorlesung und Übung umgesetzt, wie sie in Abschnitt 3 zur engen Verzahnung von Input und Aktivität vorgeschlagen wird.

„Formale Methoden“ für Wirtschaftsinformatiker (Stuttgart) mit 2V+1Ü (= 3(V+Ü)) fand im WiSe 2004/05 mit 24 Studierenden statt. Ein wichtiges Lernziel dieser Veranstaltung ist, pragmatische Fertigkeiten im Kontext mit formalen Methoden der Informatik zu fördern. Durch die direkten Übungsphasen mit Gruppen- und Diskussionsaufgaben war es möglich, direkten Einfluss auf die Motivation der Studierenden zu nehmen und über ein unmittelbares Feedback den Lernerfolg zu steigern.

„Didaktik der Informatik“ für Lehramtsstudierende (Leipzig) mit 2V+2Ü (= 4(V+Ü)) fand im WiSe 2005/06 mit 14 und im WiSe 2006/07 mit 11 Studierenden statt. Neben der Vermittlung fachdidaktischer Inhalte und Methoden sind wichtige Lernziele dieser Veranstaltung affektiver Natur. Informatikunterricht an Schulen ist in vielerlei Hinsicht

besonders (falsches Bild von Informatik bei Kollegen, Schülern und Schülerinnen, stark heterogene Vorkenntnisse/-erfahrungen und Motivation, überdurchschnittlich viel Einzelbetreuung im Unterricht durch ausgedehnte Praxisphasen). Deshalb ist es wichtig, dass die angehenden Lehrer und Lehrerinnen sich dieser Besonderheiten bewusst werden und für sich entsprechende Strategien zum Umgang damit entwickeln. Die Integration von Gruppenarbeiten, Rollenspielen und entwickelndem Unterrichtsgespräch, die jeweils einzelne Besonderheiten im Zusammenhang mit Unterrichtsplanung und -durchführung thematisieren, erleichtern die Schulung derartiger affektiver Lernziele.

Spiralkonzept in „ADS“

Der übliche Aufbau der Vorlesung „Algorithmen und Datenstrukturen“ gliedert sich in die drei bzw. vier große Teilabschnitte 0. Grundlagen, (falls notwendig), 1. Suchen, 2. Sortieren und 3. Graphalgorithmen. Diese Gliederung ergibt sich logisch aus einer inhaltlichen Sortierung. Der Zusammenhang zwischen gewählten Datenstrukturen, darauf möglichen Algorithmen und dem jeweils notwendigen Zeit- bzw. Platzbedarf wird für jeden Aufgabenbereich getrennt betrachtet. Auf Gemeinsamkeiten im dahinterliegenden Entwurfsprinzip wird bei einem derartigen Aufbau der Vorlesung in der Regel nur mündlich verwiesen. Der Linearansatz entspricht der üblichen Vorgehensweise, bei der der Vorlesungsaufbau hauptsächlich durch die Inhalte festgelegt wird. Wenn z.B. alle Verfahren zur Aufgabenstellung Suchen nacheinander behandelt werden, geht für die Studierenden oft das Verständnis verloren, warum jetzt noch ein anderea Suchverfahren behandelt wird. Die Querbezüge zwischen den Algorithmen werden nur innerhalb des Abschnittes Suchen gezogen und die grundlegende Entwurfsmuster werden nur implizit vermittelt.

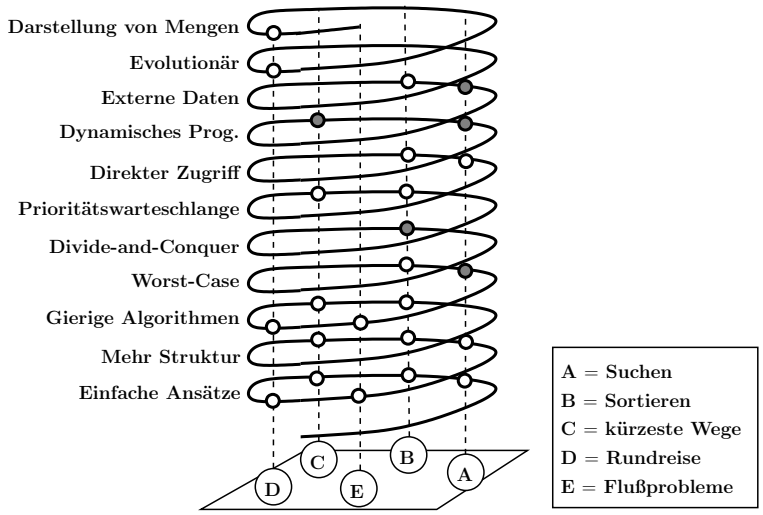


Abbildung 7: Spiralaufbau der Vorlesung „Algorithmen und Datenstrukturen“ an der HTWK Leipzig SoSe 2006 nach dem Spiralansatz. ○ bedeutet, dass das entsprechende Thema als Lösungsansatz für das jeweilige Problem A–E umgesetzt wird. ● kennzeichnet die Inhalte, die von den Studierenden erfahrungsgemäß als schwierig eingestuft werden.

Gerade für die Vorlesung „Algorithmen und Datenstrukturen“ bietet sich eine ideenorientierte Spiralstruktur für die Vorlesungsgliederung an, da anhand weniger grundsätzlicher Aufgabenstellungen (Suchen, Sortieren, etc.) verschiedene Algorithmenentwurfsmuster vorgestellt und analysiert werden (vgl. Abbildung 7). Dem Spiralentwurf geht die Überlegung voraus, welche wesentlichen Inhalte und Ideen die Studierenden erlernen sollen. Durch die Ideenorientierung wird ein besonderes Gewicht auf die Entwurfsmuster wie z.B. gierige Verfahren oder direkter Zugriff gelegt”. Die Studierenden sollen die Prinzipien und Denkweisen, die den Entwurfsmustern zugrundeliegen verstehen und anwenden lernen. Ein Nachschlagekatalog, welche Sortier- oder Suchverfahren es mit welchen Laufzeiten und welchen hauptsächlichen Einsatzgebieten gibt, wird durch die Zusammenfassung am Ende der Vorlesung geliefert.

5 Zusammenfassung und Ausblick

Die zielorientierte Didaktik der Informatik verwendet die didaktischen Prinzipien der Lernzielorientierung, der Anknüpfung an Vorwissen, die enge Verzahnung von Input und Aktivität, die Motivierung und frühzeitiges Feedback bzgl. aller Lernbereiche. Die in dieser Arbeit aufgeführten Beispiele der Umsetzung zielorientierter Informatiklehre zeigen, dass es möglich ist, auch unter der Einschränkung begrenzter Zeit- und Personalresourcen die Kompetenzvermittlung im Informatikstudium deutlich zu verbessern. Weitere Bespielen im Bereich der E-Learning-gestützte Programmierlehre, der Betreuung von Softwareprojekten, der Vermittlung von Projektleitungskompetenzen und der Betreuung von Abschlussarbeiten werden derzeit erarbeitet bzw. ausgewertet.

Literaturverzeichnis

[FB01] R. M. Felder und R. Brent. Effective strategies for cooperative learning. J. Cooperation & Collaboration in College Teaching, 10(2):69–75, 2001.

[FB03] R. M. Felder und R. Brent. Learning by doing. Chem. Engr. Education, 37(4):282–283, 2003.

[GI05] Gesellschaft für Informatik e. V., Hrsg. Empfehlungen für Bachelor- und Masterprogramme im Studiengang Informatik an Hochschulen. Gesellschaft für Informatik e. V., Bonn, 2005.
http://www.gi-ev.de/service/publikationen/empfehlungen/ (Stand 26.5.2007).

[KW01] Andreas Krapp und Bernd Weidenmann. Pädagogische Psychologie. Beltz, Weinheim, 4. Auflage, 2001.

[Ve04] Frederic Vester. Denken, Lernen, Vergessen. dtv, München, 30. Auflage, 2004.

[WDW06] Nicole Weicker, Botond Draskoczy und Karsten Weicker. Fachintegrierte Vermittlung von Schlüsselkompetenzen der Informatik. In Peter Forbrig, Günter Siegel und Markus Schneider, Hrsg., HDI 2006: Hochschuldidaktik der Informatik – Organisation, Curricula, Erfahrungen, Seiten 51–62, Bonn, 2006. GI-Edition Lecture Notes in Informatics. 2. GI-Fachtagung, 7./8.12.2006 in München.

Empirisches Untersuchungsdesign zum Medieneinsatz im objektorientierten Anfangsunterricht

Michael Dohmen

Didaktik der Informatik
Universität Paderborn
Fürstenallee 11
33095 Paderborn
dohmen@uni-paderborn.de

Abstract: Für einen objektorientierten Informatikunterricht in der Jahrgangsstufe 11 spielen mentale Vorstellungen der Schüler zu den wichtigsten Grundbegriffen eine große Rolle. Zur Veranschaulichung zentraler Begriffe der Objektorientierung sind geeignete mediale Werkzeuge wie etwa das CASE-Tool FUJABA sicher sinnvoll, jedoch ergeben sich daraus zusätzliche Schwierigkeiten beim Umgang mit einer doch recht komplexen Software. Um diese Probleme zu untersuchen, ist eine empirische Vergleichsuntersuchung notwendig, bei der möglichst gleichartige Lerngruppen benötigt werden. In diesem Artikel werden mögliche Lösungen für eine optimale Homogenisierung der Gruppen vorgestellt.

1 Problemlage

Im Informatikunterricht der Jahrgangsstufe 11 hat sich in den letzten Jahren weitgehend der objektorientierter Ansatz durchgesetzt. Diese zunehmende Angleichung der Unterrichtsinhalte, die den Informatikunterricht bisher gegenüber anderen Fächern deutlich unterschied, wird zusätzlich durch zentrale Abschlussprüfungen und durch Entwicklung von Lernstandards bis hin zur Entwicklung von PISA-Items verstärkt.

Probleme der Schülerinnen und Schüler beim objektorientierten Ansatz gerade zu Beginn der Ausbildung kann ich aus eigener Unterrichtserfahrung immer wieder feststellen. Sie werden durch die Abbildung des Softwareentwicklungsprozesses und der Behandlung von relativ großen Projekten mit Anspruch der eigenständigen Modellierung noch weiter verstärkt. Daraus ergeben sich für den Anfangsunterricht zentrale Probleme, die Moll in den folgenden Fragen zusammenfasst:

1. Wie kann die Vorstellung von einzelnen Objekten und deren Zusammenwirken sowie das Zusammenspiel der verschiedenen Klassen besser gefördert werden?
2. Wie können grafische Methoden der Modellierung von den Schülerinnen und Schülern besser genutzt werden bei der Erstellung und Weiterentwicklung von Modellen?

3. Wie können die unterschiedlichen Ebenen der sachbezogenen Modellierung (Modellierung des Anwendungskontextes) und der darstellungsbezogenen Modellierung (Darstellung auf dem Bildschirm) den Lernenden deutlicher gemacht werden?

4. Wie können die Schwierigkeiten in der programmiersprachlichen Umsetzung von statischen Modellen verringert werden?

5. Wie können die Bedeutungen einzelner Attribute und Methoden den Schülerinnen und Schülern in einem Softwareprojekt längerfristig präsent sein? (vgl. [Mo02])

Ein Großteil dieser Probleme wird durch den im Anfangsunterricht sehr generischen Ansatz „Objects First“ [BK06] beseitigt. Dabei spielt der Einsatz geeigneter Werkzeuge eine wichtige Rolle. In der Schule ist das von Michael Kölling entwickelte Werkzeug BlueJ weit verbreitet, da es sehr schnell das Erzeugen von Klassen und Objekten erlaubt und dadurch die Schritte vom Modell zum vom Schüler erstellten Programm möglichst einfach gestaltet. Es gibt jedoch auch ein paar Nachteile, die hauptsächlich in nicht UML-konformen Darstellungen zu sehen sind.

Eine weitere Alternative für den Ansatz „Objects First“ wurde im Rahmen des Projekts Life³ [1] entwickelt. Hierbei handelt es sich nicht nur um ein Werkzeug, sondern um eine evaluierte Unterrichtskonzeption, die sowohl methodisch als auch medial neue Wege geht. Insbesondere der Einsatz des CASE-Tools FUJABA ist für den Anfangsunterricht ungewöhnlich und noch wenig verbreitet. Es ist möglich, ohne jegliche Kenntnisse des Quellcodes, ein Programm zu erstellen. In Kooperation mit der FG Didaktik der Informatik und FG Softwaretechnik der Universität Paderborn hat Schulte dieses Werkzeug für den Anfangsunterricht angepasst, evaluiert und ein positives Feedback erhalten (vgl. [Sc04]). Leider ist dieser Ansatz recht weit von dem klassischen Anfangsunterricht entfernt, was sicher eine der Hauptgründe für die geringe Verbreitung darstellt.

Um eine größere Verbreitung dieses Ansatzes zu erreichen, reicht es nicht, ihn isoliert vom weiteren Unterricht zu betrachten, sondern es muss eine Einbettung dieses Konzepts in die bisherigen klassischen Unterrichtskonzepte durchgeführt werden. Dabei lernen Schüler verschiedene Sichten auf ein Softwaresystem kennen und müssen zwischen diesen Sichten wechseln können. Gerade der Prozess dieses Wechsels kann zu Problemen und Blockaden der Schüler führen. Jedoch scheinen verschiedene Sichten auf ein Informatiksystem aus lerntheoretischer Sicht sicher sinnvoll. Es stellt sich also nicht vorrangig die Frage, ob ein solcher Wechsel im Unterricht eingesetzt werden soll, sondern wie er unterrichtlich vollzogen werden soll. Hierbei bieten sich zwei Modelle für den Anfangsunterricht an, die beide auf dem Vorgehensweise „Objects First“ basieren:

1. Die Modellierung und Grundkonzepte werden nach dem Life³-Konzept eingeführt und unter Verwendung von BlueJ implementiert und getestet. Erst in einem zweiten Schritt wird ein Werkzeug wie FUJABA benutzt, um mit Hilfe von Diagrammen eine graphische Modellierungssicht der Software zu erhalten.

[1] Das Projekt Life³ ist ausführlich unter http://life.uni-paderborn.de dokumentiert.

2. Die Modellierung und Grundkonzepte lassen sich viel besser mit FUJABA veranschaulichen. Deshalb ist der Einsatz schon zu Beginn der Jahrgangsstufe sinnvoll. Ein Wechsel des Werkzeugs und die Einführung der „Codesicht“ finden dann nach Abschluss der ersten Projekte statt. Dies ist notwendig, um etwa den Unterricht für das Zentralabitur zusammenzuführen.

Im Rahmen des Life³-Projekts und darauf aufbauenden Arbeiten (vgl. [Di07]) wurde bisher nur die zweite Alternative näher untersucht Die Probleme des Sichtwechsels auf das Softwaresystem wurden jedoch nicht genauer erforscht. Hier besteht also noch Handlungsbedarf. Somit ergeben sich zwei zentrale Fragen:

1. Wie weit werden Modellierungskompetenzen durch geeignete Modellierungswerkzeuge bei einer inhaltlich sonst gleichen Kurssequenz mit gleichen Einstiegsbeispielen unterstützt? Welche Auswirkungen haben die medialen Angebote und Sichtweisen auf die Vorstellung der Schüler von Informatiksystemen?

2. Welche Probleme treten bei einem Wechsel der Modellierungssichtweise von graphischer zu codeorientierter Sicht bzw. umgekehrt auf? Gibt es also bestimmte Reihungseffekte, die sich positiv auf den Lernerfolg auswirken?

2 Untersuchungsdesign

Für eine empirische Vergleichsuntersuchung beider Ansätze ist es notwendig, möglichst viele Einfluss nehmende Faktoren zu erfassen und ggf. durch geeignete Maßnahmen anzugleichen oder mindestens zu kontrollieren. Da sich bei einer langfristigen Untersuchung über ein Schuljahr eine Laborsituation nicht anbietet, sollte die Homogenität der Lerngruppe in Bezug auf die intervenierenden Variablen durch geeignete Instrumente ermittelt werden. Falls dabei Ungleichgewichte in den Gruppen festgestellt werden, muss durch Selektion in den einzelnen Gruppen versucht werden, gleiche Voraussetzungen zu schaffen. Dadurch könnten sich die Größen der beiden Untersuchungsgruppen jedoch deutlich reduzieren.

Für die empirische Untersuchung wurden zwei Kurse der Jahrgangsstufe 11 an einem Gymnasium ausgewählt. Da an dieser Schule in der Jahrgangsstufe 11 das Fach Informatik normalerweise nicht koedukativ unterrichtet wird, bestand die Wahl, entweder zwei reine Mädchenkurse oder zwei Jungenkurse zu wählen. Bei der Untersuchung spielten Genderaspekte keine Rolle. Weiterhin waren die beiden Jungenkurse auch von der Schülerzahl[2] gleichmäßiger verteilt, so dass ich mich für die Jungenkurse entschieden habe. Bei Befragungen, etwa in [MS05], wurden darüber hinaus auch geschlechtsspezifische Unterschiede in der Interessenlage festgestellt. Es ist also zu vermuten, dass die Jungenkurse ein höheres Interesse am Programmieren und softwaretechnischen Fragen haben.

Die Beeinflussung der Kurswahl der Schüler ist ein weiterer wichtiger Schritt zur Homogenisierung der Lerngruppen. Hierbei hat man im laufenden Schulbetrieb jedoch nur

[2] Die beiden Jungenkurse haben 20 bzw. 14 Schüler. In dem größeren Kurs 1 waren jedoch auch 2 Mädchen, die aufgrund von Kurskollisionen diesen Kurs wählen mussten.

geringen Einfluss, da aufgrund von Kursbindungen die Schüler nur sehr eingeschränkt den Kurs wechseln konnten. Eine solche Umwahl ist zwar theoretisch in der ersten Woche noch möglich, würde aber nach Beginn der Unterrichtreihe zu Verzerrungen führen, da die Schüler dann eine thematische Wahl durchführen würden. Aus diesem Grund wurden die Gruppen zunächst entsprechend der Schülerwahl belassen.

Zur Ermittlung von Vorerfahrungen der Schüler wurde eine Eingangsbefragung durchgeführt, die prüfen sollte, ob die Schüler im Mittel der beiden Kurse gleiche Vorbedingungen mitbringen. Diese Eingangsuntersuchung wurde schon im Rahmen des Life³-Projekts eingesetzt und basiert in wesentlichen Teilen auf dem Test von [RNG01], dessen Reliabilität getestet ist. Die Zahl der Fragekomplexe wurde jedoch reduziert, um nur für diese Untersuchung relevante Faktoren zu bestimmen. Es wurden deshalb folgende Vorbedingungen untersucht:

- Vorwissen im Bereich Programmiersprachen und Objektorientierung
- Interessenlage und Erwartungen an das Fach
- Vorwissen und Kenntnisse im Umgang mit dem Computer
- Selbsteinschätzung gegenüber dem Fach und speziell dem Umgang mit dem Computer

Inhaltlich gliedert sich die Befragung in die folgenden Fragenkomplexe

- Personendaten
- Vertrautheit mit verschiedenen Computeranwendungen (VECA)
- Vorstellungen von Informatik und Informatikunterricht
- Sicherheit im Umgang mit Computern und Computeranwendungen (SUCA)
- Computer- und Internet-Nutzungsmotive (CIM)
- Praktisches Computerwissen (PRACOWI)
- Wissen in Bezug auf den Softwareentwicklungsprozess

Gerade die Vorstellungen des Softwareentwicklungsprozesses spiegeln wichtige Eingangsvoraussetzungen der Schüler für diese Unterrichtsreihe wieder. Hierbei zeigte sich, dass jeweils nur ein Schüler pro Lerngruppe diese Frage zu beantworten versuchte und dabei nur eine sehr rudimentäre Lösung angab. Deshalb kann man davon ausgehen, dass in diesem Bereich kein fundiertes Vorwissen existiert.

Mit Hilfe dieser Tests wurde gezeigt, dass die Lerngruppen sowohl im Mittel als auch in der Streuung in den wichtigsten Tests keine messbare Abweichung zeigen Die wichtigsten Ergebnisse einiger Teiltests finden sich exemplarisch in der Abbildung 1.

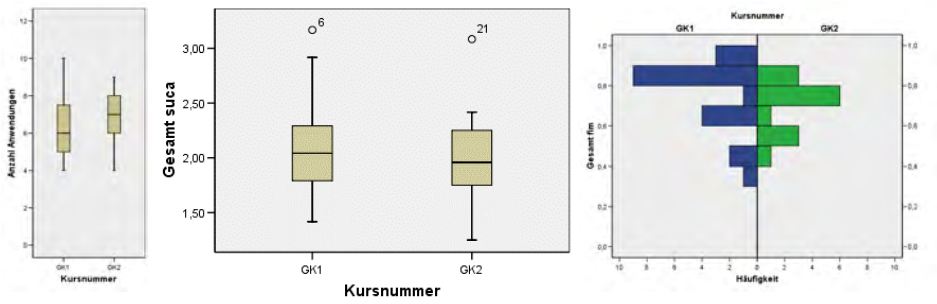


Abbildung 1: Exemplarische Teilergebnisse der Test VECA, SUCA und PRACOWI aufgeschlüsselt nach den beiden Kursen

Bei den dargestellten Ergebnissen im Bereich SUCA handelt es sich um zusammengefasste Items auf einer Skala von 1 bis 5. Bis auf zwei Ausreißer ergibt sich ein identischer Mittelwert und Streuung in den beiden Kursen. Aus der Zahl von 12 Anwendungen konnten die Schüler im einem Teil des Testes VECA anwählen: Auch hier gibt es zwar eine Abweichung im Mittelwert, doch ist diese bei der Anzahl von Versuchsteilnehmern noch nicht relevant. Auch die neun Multiple-Choice-Fragen zum Computerwissen wurden im Mittel in beiden Gruppen gleich gut bearbeitet.

Für die Vergleichsuntersuchung wurde ein Zeitraum von einem Schuljahr mit wöchentlich 3 Schulstunden gewählt. Abbildung 2 zeigt den zeitlichen Aufbau der Untersuchung.

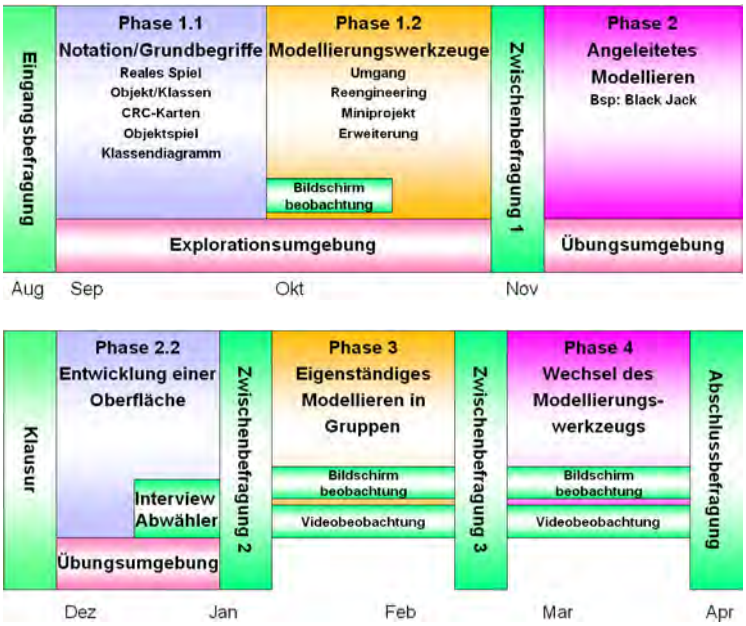


Abbildung 2: Zeitlicher Ablauf der Untersuchung

Bis zum Januar wurde in den beiden Kursen parallel der gleiche Unterricht mit gleichen Beispielen durchgeführt mit dem einzigen Unterschied, dass jeweils ein anderes Werkzeug eingesetzt wird. Dabei wurde im größeren Kurs 1 zunächst Fujaba eingesetzt, im Kurs 2 dagegen BlueJ. Es ist sicher nicht praktikabel, insgesamt pro Kurs 110 Unterrichtsstunden zu beobachten. Aus diesem Grund werden besonders die beiden letzten Phasen beobachtet, da hier die Eigenständigkeit der Schüler im Vordergrund steht. In der Phase 3 sollte dem Einfluss des Werkzeugs auf die Modellierungskompetenz nachgegangen werden. Im der letzten Phase steht die Frage dagegen der Sichtwechsel im Vordergrund.

Ein weiterer Faktor beider Ansätze ist ein Unterricht, der sich nur durch das eingesetzte Werkzeug und die dadurch ergebene unterschiedliche Sicht auf die Software unterscheidet. Es muss also sichergestellt sein, dass weitere Faktoren möglichst ausgeschaltet werden. Hierzu gehört im besonderen Maße die Person des Lehrers, der erheblichen Einfluss auf den Unterrichtsverlauf hat. Die Lehrperson muss nicht nur in beiden Ansätzen Erfahrungen mitbringen, sondern es ist auch notwendig, dass durch persönliche Vorlieben des Lehrers keine zusätzliche Beeinflussung des Unterrichts stattfindet. Aus diesem Grunde habe ich mich entschlossen, den Unterricht selbst durchzuführen. Daraus ergibt sich jedoch das Problem der Selbstkontrolle und Selbstevaluation der Lehrperson. Zur Überprüfung, ob das Handeln des Lehrers den oben genannten Anforderungen entspricht, wäre eine Videoauswertung denkbar gewesen. Alternativ wäre sicher auch eine Teilnahme von Kollegen oder Referendaren am Unterricht denkbar gewesen. Ich halte jedoch eine Beurteilung der einzelnen Unterrichtsphasen durch die Schüler für geeigneter, da ein externer Beobachter nur sehr unzureichend den Lernerfolg der Schüler beurteilen kann. Bei einer ersten Zwischenbefragung der Probanden nach der ersten Phase zeigte sich, dass im allgemeinen Teil für beide Kurse die Qualität des Unterrichts von den Schülern gleich bewertet wurde. Kleine Unterschiede ergaben sich nur im speziellen auf das Werkzeug bezogenen Teil. Dies zeigt, dass der Unterricht in beiden Lerngruppen von den Schülern als qualitativ gleichwertig empfunden wurde. Diese Ergebnisse der zusammengefassten Items sind in der Abbildung 3 dargestellt.

Neben der Beurteilung des Unterrichts durch die Schüler sind auch Tests zur Ermittlung von Kompetenzen notwendig, um den Erfolg einer Unterrichtsreihe zu beurteilen. Um auch bei den Tests eine Vergleichbarkeit sicherzustellen, wurden sie zeitparallel mit gleichen Themen durchgeführt. Sie gliedern sich in einen allgemeinen Modellierungsteil, den die Schüler werkzeugunabhängig bearbeiten können und in einen werkzeugabhängigen Teil. Der allgemeine Teil stellt sicher, dass die beiden Lerngruppen ähnliche Fähigkeiten erworben haben, während im speziellen Teil Unterschiede im Werkzeuggebrauch evaluiert werden können. Diese Tests sind ähnlich den Befragungen regelmäßig durchzuführen, um ein Auseinanderdriften der beiden Lerngruppen möglichst frühzeitig festzustellen und vielleicht zeitnah Faktoren dafür zu ermitteln. Die erste Klausur zeigt zwar unterschiedliche Streuung beim Umgang mit dem Werkzeug, jedoch noch keine signifikanten Unterschiede auf der Notenskala. Dabei handelt es sich bei dieser Klausur auch um eine Wahlpflichtklausur, wobei das Wahlverhalten sicher auch von der Leistungsfähigkeit der Schüler abhängig ist. Somit liegt hier sicher keine repräsentative Stichprobe der Kurse vor. Aus diesem Grunde sollten die Ergebnisse der ersten Klausur nicht überbewertet werden.

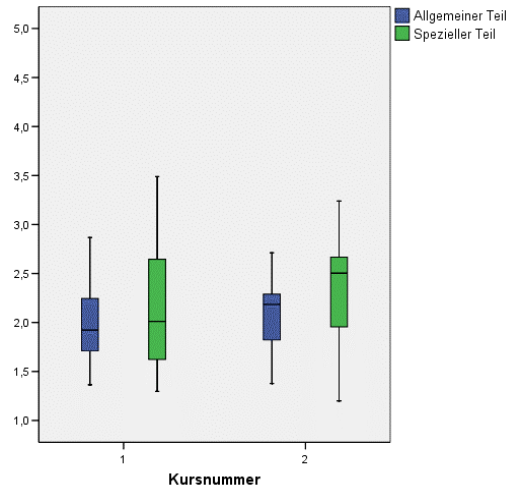


Abbildung 3: Ergebnisse der ersten Zwischenbefragung aufgeschlüsselt nach allgemeinen und speziellen, vom Werkzeug abhängigen Teilen des Unterrichts auf einer Notenskala von 1 bis 5

Die Kompetenzmessung mit Hilfe von Klausuren ist in der didaktischen Forschung nicht unumstritten. Leider gibt es bisher kein probateres und valideres Werkzeug zur Kompetenzmessung im Bereich der objektorientierten Modellierung. Jedoch könnten die hier verwendeten Aufgaben aus dem Anfangsunterricht ähnlich den Ergebnissen aus dem Life³-Projekt eine allgemeine werkzeugunabhängige Basis hierfür schaffen, um objektorientierte Fähigkeiten zu testen.

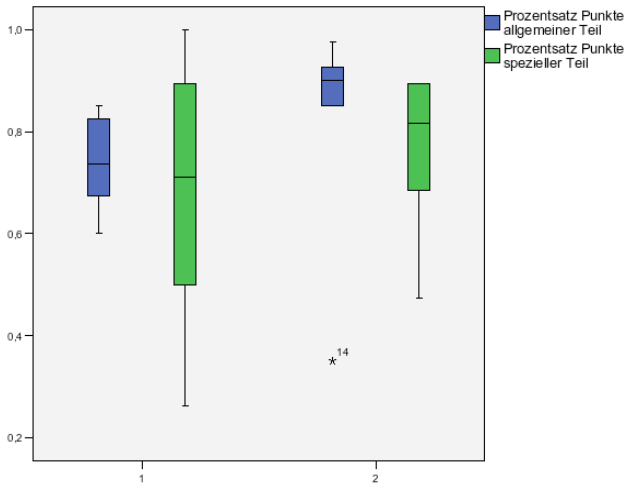


Abbildung 4: Ergebnisse der ersten Klausur aufgeschlüsselt nach allgemeinen und speziellen, vom Werkzeug abhängigen Aufgabenteilen in Kurs 1 und 2

In Nordrhein-Westfalen können die Schüler auch am Ende des Schulhalbjahres Kurse abwählen. In den beiden Kursen gab es insgesamt 6 Abwähler, wobei bezeichnenderweise die beiden Mädchen des Kurses 1 dazu gehörten. Um mehr über die Gründe der Abwahl zu erfahren und daraus Konsequenzen für etwaige Probleme des Unterrichts zu ziehen, haben diese Schüler noch einmal den bisherigen Unterricht beurteilt und unterzogen sich einem Leitfrageninterview mit den folgenden Themen:

- Vorstellungen über den Softwareentwicklungsprozess
- Was bedeutet Modellierung?
- Wie sind Sie mit den Werkzeugen klargekommen?
- Entsprach der Unterricht den Erwartungen?
- Was waren die Hauptgründe für die Abwahl?

Die wichtigsten Ergebnisse dieser Befragung waren, dass das Fach als Lückenfüller diente und die Unterrichtsinhalte nicht mit den Erwartungen übereinstimmten. Damit verbunden waren die recht geringen fachlichen Kenntnisse, die auf eine mangelnde Motivation schließen lassen. Wichtig jedoch war, dass sich bei der Befragung erneut keine Unterschiede zwischen den beiden Kursen ergaben.

3 Fazit und offene Fragen

Durch geeignete Maßnahmen und Untersuchtungsinstrumente wurde sichergestellt, dass die beiden Schülerpopulationen zum einen möglichst gleiche Vorerfahrungen mitbringen und zum anderen auch gleichen Bedingungen in einer normalen Unterrichtssituation ausgesetzt wurden. Der einzig zu variierende Faktor war der Einsatz des Modellierungswerkzeugs bzw. die zeitliche Abfolge des Einsatzes der beiden Werkzeuge. Es wurden Tests zur Ermittlung der Modellierungskompetenz und Befragungen der Probanten zum Unterricht durchgeführt. Die Ergebnisse aller dieser Tests zeigen auf, dass die Lerngruppen zwar nicht leistungshomogen sind, aber trotz einer großen Streuung in wichtigen Faktoren gleichartig aufgeteilt sind. Somit kann man davon ausgehen, dass auch soziale Prozesse – etwa Zusammensetzung der Gruppen innerhalb der Gruppenarbeit – in beiden Lerngruppen gleichartig ablaufen werden.

Um diese These zu untermauern, werden im Laufe der weiteren Untersuchung in Gruppenarbeitsphasen die Diskussionen und Bildschirme mitgeschnitten um den Entstehungsprozess der Software zu dokumentieren und bewerten zu können.

Erste Ergebnisse der weiteren Untersuchungen, die sich im Laufe des weiteren Unterrichtreihe angeschlossen haben, zeigen, dass das Werkzeug einen entscheidenden Einfluss auf die mentalen Modelle und die Modellierungskompetenz der Schüler hat. Welche Fähigkeiten genau vorhanden sind, müssen die Auswertungen der Videos und weitere Tests bis Juni 2007 genauer untersuchen. Welchen Einfluss die Reihenfolge auf die objektorientierten Vorstellungen der Schüler hat, ist eine weitere wichtige Untersu-

chungsfrage, die ebenfalls erst gegen Ende des Schuljahres geklärt werden kann. Es ist aber in meinen Augen die entscheidende Frage, um die Unterrichtssequenz des Life³-Konzepts in den bisherigen „klassischen“ Informatikunterricht möglichst lerneffizient einbetten zu können.

Literaturverzeichnis

[BK06] Barnes, David; Kölling, Michael: Java lernen mit BlueJ. Eine Einführung in die objektorientierte Programmierung, Pearson Studium; Auflage: 3. Aufl. 2006

[Di07] Diethelm, Ira: “Strictly models and objects frst” – Unterrichtskonzept und –methodik für objektorientierte Modellierung im Informatikunterricht, Fachbereich Elektrotechnik/Informatik der Universität Kassel, 2007 (unveröffentlicht)

[Mo02] Moll, Stefan: Objektorientierte Modellierung unter Einsatz eines CASE-Tools im Informatikunterricht der Jahrgangsstufe 11, GI-Workshop, Bommerholz, 11.10.2002

[MS05] Magenheim, J. Schulte, C.: Erwartungen und Wahlverhalten von Schülerinnen und Schülern gegenüber dem Schulfach Informatik – Ergebnisse einer Umfrage in: Friedrich, S. (Hrsg.) Unterrichtskonzepte für informatische Bildung, infos2005-11. GI - Fachtagung Informatik und Schule 28.-30.September 2005 in Dresden, Proceedings S. 111 - 122

[Sc04] Schulte, Carsten: Lehr-Lernprozesse im Informatik-Anfangsunterricht – Theoriegeleitete Entwicklung und Evaluation eines Unterrichtskonzepts zur Objektorientierung in der Sek. II. Fachbereich Elektrotechnik, Informatik und Mathematik, Universität Paderborn, 2004

[RNG01] Richter, T., Naumann, J. & Groeben, N. (2001). Das Inventar zur Computerbildung (INCOBI): Ein Instrument zur Erfassung von Computer Literacy und computerbezogenen Einstellungen bei Studierenden der Geistes- und Sozialwissenschaften. Psychologie in Erziehung und Unterricht, 48, 1-13.

Autorenverzeichnis