Didaktisches System für objektorientiertes Modellieren im Informatikunterricht der Sekundarstufe II

Vom Fachbereich 12 – Elektrotechnik und Informatik

der Universität Siegen

zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
(Dr. rer. nat.)

genehmigte Dissertation

von

Diplom-Informatiker Torsten Brinda

1. Gutachter: Professorin Dr. S. Schubert
2. Gutachter: Prof. Dr. P. Hubwieser

Tag der mündlichen Prüfung: 15.03.2004

urn:nbn:de:hbz:467-585

Zusammenfassung

In Deutschland befindet sich die informatische Bildung in der Sekundarstufe in einem konzeptionellen Wandel von starker Betonung der prozeduralen Programmierung hin zum objektorientierten Modellieren (OOM). Obwohl von Informatikdidaktikern empfohlen, betont und bereits in einigen Curricula der Bundesländer verankert, hat dieser Wandel die Mehrzahl der Schulen noch nicht erreicht. Ein scheinbarer Wandel fand statt von prozeduraler hin zur objektorientierten Programmierung. Als eine wesentliche Ursache für die noch seltene Berücksichtigung von OOM in der Schulpraxis wurde im Rahmen der vorliegenden Arbeit, ausgehend von einer Literaturanalyse des fachdidaktischen Standes zum OOM, ein Mangel an verfügbaren Unterrichts- und Übungsbeispielen, unterrichtsgeeigneten Informatiksystemen zur Förderung der Aneignung von Fachkonzepten, Strukturierungshinweisen für den Unterricht und damit verbundenen Einsatzkonzepten identifiziert. Eine ähnliche Ausgangslage zeigt sich in anderen, wichtigen Themenfeldern des Informatikunterrichts (z.B. Wirkprinzipien von Informatiksystemen), die einerseits empfohlen werden, für die andererseits aber noch zu wenige Lehr-Lern-Materialien existieren bzw. zugänglich sind. Mit den Zielen, einerseits den identifizierten Bedarf beim OOM nachhaltig zu decken und andererseits auch in anderen Bereichen der informatischen Bildung zu einer Bereicherung der Lehr-Lern-Prozesse beizutragen, wurde ein als didaktisches System bezeichneter Verbund, bestehend aus traditionellen und neuen Komponenten des Lehr-Lern-Prozesses, konzipiert. Gestaltungsleitend war hierbei der zuvor dargelegte Bedarf. Begründet und exemplarisch erprobt wurde, wie die vorgeschlagenen Systemkomponenten Aufgabenklassen, Explorationsmodule und Wissensstrukturen zu einer Bereicherung traditioneller Lehr-Lern-Prozesse beitragen können, durch Anwendung in, als Gestaltungsmittel für und durch Förderung der fachdidaktischen Kommunikation und Diskussion zu Lehr-Lern-Prozessen. Bei Aufgabenklassen handelt es sich um abstrakte Aufgabenrahmen, die aus einem zu diesem Zweck entwickelten, Auswahl-, Abstraktions- und Strukturierungsprozess, hervorgingen, der auf über 320 Übungsaufgaben in fachwissenschaftlichen Lehrbüchern zum OOM angewandt wurde. Damit verbunden wurde eine Methodik zur Gestaltung von Aufgaben zum OOM zur Erfüllung vielfältiger fachdidaktischer Funktionen entwickelt, um diesen Prozess speziell für mit OOM wenig erfahrene Lehrende zu erleichtern. Lernenden geben Aufgabenklassen Orientierung, indem sie abstrakte Aufgabenrahmen mit im Unterricht erarbeiteten Lösungsstrategien verbinden. Die Bewältigung von Problemen mit ähnlichen Aufgaben wird damit vereinfacht. Bei Explorationsmodulen handelt es sich um softwarebasierte Lernangebote zur Anregung der Exploration von fachlichen Zusammenhängen in „Blended Learning“-Szenarios. Lernenden wird damit ein neuer, handlungsorientierter Zugang zu Fachkonzepten eröffnet, indem sie mit Repräsentationen von Fachkonzepten unter Verwendung systematischer Explorationsstrategien interagieren, anstatt ausschließlich zu programmieren. Dadurch schärfen sie nicht nur ihre Fach-, Methoden- und Sozial-, sondern auch ihre Lernkompetenz. Wissensstrukturen dienen als grafische Repräsentationen von fachlichen Erarbeitungsstrukturen unter Verwendung von Und-Oder-Graphen der Veranschaulichung von Vorkenntnisstrukturen zwischen Fachkonzepten für Akteure in Lehr-Lern-Prozessen. Sie erleichtern die fachdidaktische Analyse bzw. Diskussion von Lehr-Lern-Prozessen sowie die Kontrolle des erreichten Bildungsstands für Lehrende und Lernende. Die Vorgehensweise bei der Entwicklung der Komponenten des didaktischen Systems wurde ausführlich dokumentiert und abstrahiert. Dadurch ist ein Leitfaden gegeben, mit dem weitere didaktische Systeme für andere Themenbereiche oder andere Zielgruppen gestaltet werden können. Gezeigt wurde weiterhin, wie eine exemplarische Erprobung durch Integration in den Informatikunterricht der Sek. II sowie in die Informatiklehreraus- und -weiterbildung erfolgen kann. Involviert in diesen Prozess waren über 100 Lernende im Informatikunterricht der Sek. II und in Studienwerbeveranstaltungen, 10 Lehramtsstudierende in der Informatiklehrerausbildung sowie über 200 Lehrende in Veranstaltungen zur Informatiklehrerfortbildung.

Abstract

In Germany a conceptual change from strong emphasis of procedural programming to object-oriented modelling (OOM) is on the way within the field of secondary Informatics education. Though recommended and emphasized by Informatics didactists and already embodied in some of Germany's federal Informatics curricula, the change has not yet reached the majority of schools. A seeming change was carried out from procedural to object-oriented programming. As part of the thesis at hand, based on a literature analysis of the status of didactics concerning OOM, a lack of available teaching and exercise examples, class suitable Informatics systems supporting the acquisition of Informatics concepts, advice for class structuring and associated concepts of use were identified as essential reasons for the still rare consideration of OOM in school practice. A similar situation is evident in other important subject areas of secondary Informatics education (e.g. principles of Informatics systems), which, on the one hand, are recommended, for which, however, on the other hand, there are still too few teaching and learning materials existent respectively accessible. With the aim of, on the one hand, covering the identified need regarding OOM lastingly, while, on the other hand, contributing to the learning and teaching processes of other areas of Informatics education as well, a compound referred to as didactic system, consisting of both traditional and new components of the learning and teaching process, was designed. In this context the deciding motive regarding the design was the need mentioned above. It was founded and proven exemplarily, how the proposed system components exercise classes, exploration modules, and knowledge structures could contribute to the enrichment of traditional learning and teaching processes through the use in, as means of creation of as well as through the support of communication concerning didactics and the discussion of learning and teaching processes. Exercise classes are abstract exercise templates, which have been derived from a choosing-, abstraction- and structuring process, especially developed for this purpose, which was applied to more than 320 exercises from scientific textbooks on OOM. In this context a methodology for creating exercises on OOM to perform a multitude of didactic functions was developed in order to facilitate this process particularly for teachers with little experience with OOM. Exercise classes give learners orientation in combining abstract exercise templates with solution strategies worked out in class. Mastering problems with similar exercises is this way simplified. Exploration modules are software-based learning offers, which are to motivate exploration of Informatics concepts in "blended learning"-scenarios. This way a new, learning-by-doing-oriented approach to Informatics concepts is opened up to learners, having them interact with representations of such concepts, while using systematic exploration strategies, instead of exclusively programming. In doing so they do not only sharpen up their professional-, method-, and social competence, but also their learning competence. As graphic representation of professional acquisition structures using and-or-graphs, knowledge networks serve as an illustration of pre-knowledge relationships between concepts of a subject for actors in learning and teaching processes, facilitate the didactic analysis respective discussion of learning and teaching processes as well as the checking of the educational level reached for teachers and students. The approach to the development of the components of the didactic system was documented and abstracted at full length. Thus, a guideline is given, according to which more didactic systems regarding other subject areas or other target groups may be designed. Furthermore, it was shown, how an exemplary trial may take place by the means of integration into secondary Informatics education as well as into Informatics teacher education and in-service teacher training. More than 100 learners in secondary Informatics education and in university promotion events, ten student teachers of Informatics as well as more than 200 teachers participating in Informatics teacher trainings were involved in this process.

Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter in den Fachgruppen „Didaktik der Informatik“ der Universität Dortmund und „Didaktik der Informatik und E-Learning“ an der Universität Siegen.
Ich möchte mich an dieser Stelle bei allen Personen bedanken, die in unterschiedlicher Weise zum Entstehen dieser Arbeit beigetragen haben. Die Arbeit wurde betreut von Frau Prof. Dr. Sigrid Schubert. Ihr danke ich herzlich für die vielfältigen, gewinnbringenden Diskussionen, ihre Unterstützung und stete Förderung meiner wissenschaftlichen Arbeit. Herrn Prof. Dr. Peter Hubwieser danke ich für die Übernahme des Koreferats.
Den beiden Informatiklehrern Herrn Thomas Daub und Herrn Wolfram Vorsmann danke ich für die engagierte und für mich sehr lehrreiche Zusammenarbeit im Rahmen des Tagespraktikums der Informatiklehrerausbildung an der Universität Dortmund. Durch sie erhielt ich in der Anfangsphase meiner Tätigkeit als wissenschaftlicher Mitarbeiter einen fundierten und nachhaltigen Einblick in die Berufspraxis von Informatiklehrerinnen und -lehrern.
Ich danke den Mitgliedern der Projektgruppe LEO (Nr. 403) des Fachbereichs Informatik der Universität Dortmund sowie den Mitgliedern des Siegener LEO-Projektteams, insbesondere Herrn Peter Droste, ohne deren Zusammenarbeit das LEO-System in seiner jetzigen Form nicht denkbar wäre.
Weiterhin danke ich den Studenten, deren Diplom- bzw. erste Staatsexamensarbeiten ich betreuen durfte. Hier gilt mein Dank den Herren Dirk Steinkamp, Tobias Ortmann und Andreas Hoffmann.
Schließlich danke ich meinen ehemaligen Kolleginnen und Kollegen aus Dortmund, insbesondere Herrn Dr. Ludger Humbert, sowie meinen Kolleginnen und Kollegen aus Siegen für das stets angenehme Arbeitsklima, die vielfältige Unterstützung und die gute Zusammenarbeit in gemeinsamen Projekten.
Wertvolle Hinweise inhaltlicher und formaler Natur für die endgültige Fassung des Manuskripts verdanke ich insbesondere den beiden Informatiklehrenden Frau Barbara Leipholz-Schumacher und Herrn Ralf Sagorny-Schwarz. Bedanken möchte ich mich weiterhin bei Herrn Karsten Meininghaus für das sorgfältige Korrekturlesen der Arbeit.
Nicht zuletzt danke ich meiner Freundin Simone sowie meinen Eltern Liesel und Gerhard für ihre vielfältige Unterstützung im persönlichen Bereich.

Siegen, im Dezember 2003 Torsten Brinda

Inhaltsverzeichnis

Zusammenfassung ..... **iii**

Abstract ..... **iv**

Vorwort ..... **v**

Inhaltsverzeichnis ..... **vii**

1 Einleitung **1**

1.1 Problemlage 1

1.2 Forschungsmethodische Vorgehensweise 4

1.3 Beitrag der Arbeit 6

1.4 Gliederung der weiteren Arbeit 8

2 Fachdidaktischer Stand zum objektorientierten Modellieren **9**

2.1 Überblick 9

2.2 Informatisches Modellieren 10

2.2.1 Betonung der Modellierung 10

2.2.2 Programmierung und Modellierung 12

2.2.3 Stellenwert der Modellierung 13

2.3 Objektorientiertes Modellieren 15

2.3.1 Nationale Diskussion 15

2.3.1.1 Empfehlungen, Konzepte und Ergebnisse 15

2.3.1.2 Unterrichtserfahrungen und -beispiele 24

2.3.2 Internationale Arbeiten 32

2.3.3 Kritische Positionen zur Objektorientierung 37

2.4 Fazit 39

2.4.1 Zusammenfassung 39

2.4.2 Schlussfolgerungen 40

2.4.3 Wissenschaftliche Fragestellungen 43

3 Konzeption eines didaktischen Systems für OOM **44**

3.1 Überblick 44

3.2 Zielsetzung und Definition 44

3.3 Aufgabenklassen 47

3.3.1 Didaktische Funktionen von Aufgaben 47

3.3.2 Entwicklung und Erprobung von Aufgabenklassen und Aufgaben 48

3.3.3 Didaktische Funktionen von Aufgabenklassen 48

3.4 Explorationsmodule 49

3.4.1 Grundprinzipien des OOM mit Lern-Software entdecken 49

3.4.2 Entwicklung und Erprobung von Explorationsmodulen ... 50
3.4.3 Didaktische Funktionen von Explorationsmodulen ... 51
3.5 Wissensstrukturen ... 52
3.5.1 Repräsentation von Wissensstrukturen ... 52
3.5.2 Entwicklung von Wissensstrukturen ... 53
3.5.3 Didaktische Funktionen von Wissensstrukturen ... 54
3.6 Verknüpfung der Komponenten ... 55
3.7 Konzept für die Lehrerbildung ... 56
3.8 Abgrenzung von anderen Ansätzen ... 58
3.9 Zum weiteren Aufbau der Arbeit ... 63

4 Aufgabenklassen ... 64
4.1 Überblick ... 64
4.2 Stellenwert der Aufgabenklassen im didaktischen System ... 65
4.3 Entwicklung einer strukturierten Sammlung von Aufgabenklassen ... 66
4.3.1 Auswahl von Übungsaufgaben aus Lehrbüchern ... 66
4.3.2 Abstraktion von Übungsaufgaben zu Aufgabenklassen ... 69
4.3.3 Strukturierung der Aufgabenklassen ... 72
4.4 Gestaltung von Aufgaben mittels Aufgabenklassen ... 80
4.4.1 Vorüberlegungen ... 80
4.4.2 Niveaustufungen ... 80
4.4.3 Auswahl des Kontextes ... 81
4.4.4 Fachdidaktische Diskussion ... 84
4.4.5 Analogie zu anderen Unterrichtsfächern ... 86
4.5 Empirische Fallstudien im Informatikunterricht der Jgst. 11 und 12 ... 86
4.5.1 Konzeption der Fallstudien ... 86
4.5.2 Auswertung der Beobachtungsstunden ... 91
4.5.3 Befragungsergebnisse ... 96
4.6 Erprobung in der Informatiklehrerbildung ... 99
4.6.1 Lehramtsstudium ... 99
4.6.1.1 Anwendung des Aufgabenklassenkonzepts bei der Gestaltung von Übungsaufgaben ... 99
4.6.1.2 Identifikation von Aufgabenklassen und Kontexten ... 103
4.6.2 Lehrerfortbildungen ... 103
4.7 Zusammenfassung, Fazit und offene Fragen ... 105
4.7.1 Zusammenfassung ... 105
4.7.2 Fazit ... 108
4.7.3 Offene Fragen ... 109

5 Explorationsmodule 111
5.1 Überblick 111
5.2 Stellenwert der Explorationsmodule im didaktischen System 111
5.3 Entwicklung von Explorationsmodulen 113
5.3.1 Grundlagen eines Konzeptes für Explorationsmodule 113
5.3.1.1 Exploratives / entdeckendes Lernen 113
5.3.1.2 Exploratives Lernen mit Informatiksystemen 115
5.3.2 Fachdidaktische Analyse von Werkzeugen zum objektorientierten Modellieren 122
5.3.3 Konzept für Explorationsmodule zum OOM 128
5.3.4 Explorationsmodule zum OOM 135
5.3.4.1 Zur Entwicklungsreihenfolge 135
5.3.4.2 Animationen mit Interaktionsmöglichkeiten: Puzzles und Experimentierumgebungen 136
5.3.4.3 Explorer für geometrische Objekte – EGO 140
5.3.4.4 Lernumgebung für objektorientiertes Modellieren im Informatikunterricht – LEO 142
5.3.4.5 Vergleich der Entwicklungsergebnisse 145
5.3.5 Architekturkonzept für Software zur Exploration im Bildungskontext 147
5.4 Konzept für das Lernen mit Explorationsmodulen 150
5.4.1 Pädagogische Doppelfunktion von Explorationsmodulen 150
5.4.2 Explorationsstrategien für objektorientiertes Modellieren 151
5.4.3 Verknüpfung mit Handlungsphasen problemorientierten Unterrichts 154
5.4.4 Exploration in Hochschulen 156
5.5 Erprobungen in der informatischen Bildung 158
5.5.1 Vorbemerkung 158
5.5.2 Sekundarstufe 158
5.5.3 Informatiklehrerbildung 160
5.5.3.1 Lehramtsstudium Informatik 160
5.5.3.2 Lehrerfortbildungen 167
5.5.4 Informatikstudium 167
5.6 Zusammenfassung, Fazit und offene Fragen 168
5.6.1 Zusammenfassung 168
5.6.2 Fazit 171
5.6.3 Offene Fragen 172

6 Wissensstrukturen 174
6.1 Überblick 174
6.2 Fachwissenschaftliche und fachdidaktische Erarbeitungsstrukturen 174
6.2.1 Vorüberlegungen 174
6.2.2 Fachwissenschaftliche Lehrbücher 174

6.2.3 Fachdidaktische Problembereiche ... 176
6.2.4 E-Learning ... 178
6.3 Theoretischer Hintergrund ... 178
6.4 Repräsentation der Struktur von Lehr-Lern-Prozessen ... 180
6.4.1 Fachdidaktische Funktionen ... 180
6.4.2 Anforderungen an die Darstellungsform ... 181
6.4.3 Analyse ausgewählter Darstellungsformen ... 182
6.5 Vorgehensweise zur Entwicklung einer didaktischen Landkarte ... 186
6.6 Zusammenfassung, Fazit und offene Fragen ... 189
6.6.1 Zusammenfassung ... 189
6.6.2 Fazit ... 191
6.6.3 Offene Fragen ... 192

7 Zusammenfassung, Fazit und offene Fragen ... 193
7.1 Zusammenfassung ... 193
7.2 Fazit ... 194
7.2.1 Zu den wissenschaftlichen Fragestellungen der Arbeit ... 194
7.2.2 Zur Vorbereitung von Bildungsstandards ... 195
7.3 Offene Fragen ... 196

A Hospitation von Unterricht (Konzept „Von Stiften und Mäusen“) ... 201

B Verzeichnis der ausgewählten Übungsaufgaben ... 204

C Strukturierte Sammlung von Aufgabenklassen ... 206
C.1 Aufgabenklassen zum statischen Modell (1. Fassung) ... 206
C.2 Aufgabenklassen zum statischen und dynamischen Modell (verfeinerte und überarbeitete Fassung) ... 208
C.2.1 Überblick ... 208
C.2.2 Wissens- und Verständnisfragen zu objektorientierten Konzepten ... 208
C.2.3 Statisches Modell ... 208
C.2.3.1 Objekte und Objektstrukturen ... 208
C.2.3.2 Klassen und Klassenstrukturen ... 209
C.2.3.3 Verknüpfung von Objekt- und Klassenstrukturen ... 210
C.2.4 Dynamisches Modell ... 211
C.2.4.1 Anwendungsfälle ... 211
C.2.4.2 Funktionale Abhängigkeiten ... 211
C.2.4.3 Objektinteraktion ... 211
C.2.4.4 Objektzustände ... 212
C.2.4.5 Verknüpfung von Objektinteraktion und Objektzuständen ... 212
C.2.5 Verknüpfung des statischen und dynamischen Modells ... 212
C.2.5.1 Anforderungsanalyse ... 212

C.2.5.2 Verknüpfung von Teilmodellen 213
C.2.5.3 Analyse- und Entwurfsmuster 213
C.2.5.4 Prototyp einer Benutzungsoberfläche 214
C.2.6 Metamodellierung 214

D Materialien der empirischen Studien zu Aufgabenklassen 215
D.1 Arbeitsblatt und Musterlösungen zum „Traktor“ 215
D.2 Arbeitsblatt und Musterlösungen zum „Zug“ 218
D.3 Ergebnisse der schriftlichen Befragung 222
D.4 Arbeitsblatt zur Anwendung des Aufgabenklassenkonzepts 225
D.5 Übungsaufgaben im Rahmen der Lehrerfortbildungen 226

E Materialien der empirischen Studien zu Explorationsmodulen 231
E.1 Referenzmodell bei der Analyse von Modellierungswerkzeugen 231
E.2 Aufgabe im Rahmen der Schnupperuni 2001 232
E.3 Schreiben zu LEO 233
E.4 Arbeitsblätter zum Explorationsmodulkonzept 235

F Dokumentation zu Explorationsmodulen 239
F.1 Überblick 239
F.2 Animationen 239
F.2.1 Puzzles 239
F.2.2 Experimentierumgebungen 244
F.3 Anwendungen 247
F.3.1 Explorer für geometrische Objekte – EGO 247
F.3.2 Lernumgebung für objektorientiertes Modellieren – LEO 250

Abkürzungsverzeichnis 253

Abbildungsverzeichnis 254

Tabellenverzeichnis 256

Literaturverzeichnis 258

1 Einleitung

1.1 Problemlage

Hochkomplexe Informatiksysteme[1] begegnen uns heute in allen Bereichen des Lebens und führen zu beachtlichen Veränderungen unserer Lebens- und Arbeitsweisen. Die Informatiksysteme werden immer kleiner, leistungsfähiger, die mit ihnen verbundenen Technologien immer komplexer und damit oft immer schwieriger zu verstehen und zu beherrschen, auch wenn die Produktwerbung gerne einen anderen Eindruck vermitteln möchte.

> „Die Menschen benötigen in naher Zukunft ein tief greifendes Verständnis dieser Technologien, um deren zukünftige Entwicklung verantwortungsbewusst steuern und zielsicher abschätzen sowie die Systeme ökonomisch und effizient nutzen zu können." (Hubwieser 1999a, 165)

Hierfür werden neue Kompetenzen erforderlich, die es vorher nicht gab (Schubert 2001a, 15). Lernort hierfür ist der allgemein bildende Informatikunterricht.
Bereits seit Anfang der siebziger Jahre wird im Rahmen eines Unterrichtsfaches Informatik[2] versucht, Lernende[3] fundiert informatisch auszubilden. Da damals weder informatikdidaktische Theorien noch empirische Untersuchungsergebnisse aus der Schulpraxis vorlagen, wurden die Unterrichtsinhalte stark an der Fachwissenschaft orientiert. Mit der Zeit bildeten sich verschiedene Lehrmethoden heraus, wie der hardwareorientierte-, der algorithmenorientierte-, der anwendungsorientierte- und der systemorientierte Ansatz. Alle Ansätze und ihre Umsetzungen in die Unterrichtspraxis wurden intensiv diskutiert, jedoch keiner dieser Ansätze blieb unumstritten. Das wesentliche Problem war, dass die Lernenden eigentlich von jedem dieser Schwerpunkte etwas verstehen sollten, insofern also keiner der Ansätze dazu geeignet war, ein ganzes Curriculum zu strukturieren.
Seit Mitte der siebziger Jahre dominierte der algorithmenorientierte Ansatz die informatische Bildung an Schulen. In der Umsetzung dieses Ansatzes kam es verbreitet zu Missverständnissen derart, dass die Bedeutung von Programmiersprache und Codierung zu stark in den Vordergrund gerückt wurde bei gleichzeitiger Vernachlässigung der eigentlich zu vermittelnden informatischen Konzepte aus den Bereichen Algorithmen und Datenstrukturen. Als Konsequenz dieser Überbetonung von Programmiersprache und Codierung wurde Ende der achtziger Jahre davon gesprochen, dass der Informatikunterricht in einer Krise stecke, weil man in der Vermittlung von Programmiersprachendetails keinen Beitrag mehr zur Allgemeinbildung sah (Peschke 1989). Überreste der Missinterpretation der damaligen Bildungsziele lassen sich bis heute noch an Schulen beobachten.
Die Kritik an diesem und anderen klassischen informatikdidaktischen Ansätzen sowie ihren Umsetzungen in die Schulpraxis und das Bewusstsein einer sich rasant weiter entwickelnden Fachwissenschaft führten zur breiten Diskussion unter Informatikdidaktikern, welches die langlebigen, übertragbaren Kenntnisse, Fähigkeiten und Fertigkeiten zu hochkomplexen Informatiksystemen seien, die Lernende in allgemein bildendem Informatikunterricht erwerben sollten, um diese neuen Technologien beurteilen und beherrschen zu können. Im Ergebnis führte dies zu einem konzeptionellen Wandel der Didaktik der Informatik vom imperativen

---

[1] „Als Informatiksystem bezeichnet man die spezifische Zusammenstellung von Hardware, Software und Netzverbindungen zur Lösung eines Anwendungsproblems." (Claus / Schwill 2001, 301)
[2] Das Unterrichtsfach *Informatik* startete in der Sekundarstufe II. Später folgten Angebote im Wahlpflichtbereich der Jahrgangsstufen 9 und 10 und eine so genannte, fächerintegrierte *informationstechnische Grundbildung (ITG)* in der Sekundarstufe I. Das fächerintegrierte ITG-Konzept wird aus heutiger Sicht von der Mehrheit der Informatikdidaktiker und Unterrichtspraktiker als gescheitert angesehen.
[3] Im Text werden, abgesehen von Zitaten, überall dort, wo es möglich ist, geschlechtsneutrale Formulierungen gewählt. Wenn die männliche Form verwendet wird, so sind damit, im Sinne des generischen Maskulinums, immer Frauen und Männer gleichermaßen gemeint, sofern nicht explizit das Gegenteil behauptet wird. Frauen mögen sich dadurch nicht ausgeschlossen fühlen.

Problemlösen mit starker Betonung der Programmiersprache zum informatischen Modellieren[4] im Informatikunterricht[5]. Diese neue Orientierung wurde wesentlich beeinflusst durch Schwill und Hubwieser, die Vorschläge für informatische Bildung basierend auf fundamentalen Ideen und Modellierungstechniken beschrieben und begründeten (Schwill 1993a, 1997; Hubwieser et al. 1997, Hubwieser 2000a, 2000b).
Unabhängig von diesem konzeptionellen Wandel gewannen objektorientierte Problemlösungsmethoden seit Anfang der 90er Jahre zunehmend an Bedeutung in der informatischen Bildung[6]. 1995 begründete Schwill eine Einführung in die Informatik nach dem objektorientierten Ansatz, da „informatisch orientierte Forderungen nach zeitgemäßem Unterricht mit mächtigen Konzepten, wie Erweiterbarkeit, Kapselung, evolutionäre Software-Entwicklung, u.a. erfüllt werden und das didaktische Prinzip der Fortsetzbarkeit im Sinne eines Spiralcurriculums angewandt werden kann“ (Schwill 1995, 183). Weiterhin betonte Schwill, dass die objektorientierte Sichtweise der menschlichen Wahrnehmungsweise ihrer Umwelt im Vergleich zu anderen Programmierparadigmen sehr nahe komme und damit für die Anfangsausbildung besonders geeignet sei. Ferner wurde der objektorientierten Problemlösemethoden inne liegende Schwerpunkt auf Analyse und Entwurf herausgestrichen, der sehr gut mit dem Ziel harmoniert, die starke Programmiersprachenbindung des Informatikunterrichts zu überwinden und stärker informatisches Modellieren in den Mittelpunkt zu stellen (Crutzen / Hein 1995). Neuere Forschungsarbeiten, z.B. (Hampel et al. 1999, Magenheim 2001), betonen die Modellierungsmöglichkeiten im Bereich der Objektorientierung als Unterrichtsschwerpunkt.
Inzwischen wurde objektorientiertes Programmieren (OOP) in Lehrplänen verankert, z.B. (MSWF 1999), mit teilweise umstrittenen Beispielen, die für Lernende der Sekundarstufe II (Sek. II) nicht altersgerecht sind. In den Fachzeitschriften für Informatiklehrende *Informatik betrifft uns* und *LOG IN* wurden[7] und werden regelmäßig Unterrichtsbeispiele zu OOP publiziert. Meist handelt es sich dabei um gut dokumentierte Lösungen zu objektorientierten Programmieraufgaben, bei denen der Stellenwert der objektorientierten Modellierung nicht genug deutlich wird. Diese Publikationen zeigen, dass es oft weiterhin die Implementierung – früher imperativ heute zunehmend objektorientiert – ist, die den Unterricht dominiert. Schubert (2001b) spricht in diesem Zusammenhang von einem scheinbaren Wandel zu neuen Programmiersprachen, aber nicht zu neuen Lehr-Lern-Prozessen. Der in Forschungsarbeiten und Bildungsempfehlungen vollzogene konzeptionelle Wandel vom Programmieren zum Modellieren hat die Schulpraxis erst punktuell erreicht. Die große Popularität der objektorientierten Programmiersprache Java und die Motivation der Lernenden, die sich aus deren verbreitetem Einsatz bei der Gestaltung von multimedialen und interaktiven Webseiten im World Wide Web und verteilten Systemen ergibt, wirkt dem angestrebten konzeptionellen Wandel zusätzlich entgegen.
Informatisches- und speziell objektorientiertes Modellieren (OOM) werden also von der Fachdidaktik betont und für den Unterricht empfohlen, im Informatikunterricht aber noch zu wenig umgesetzt. Der Autor dieser Arbeit hält objektorientiertes Modellieren zwar für einen

[4] Der Modellbegriff in der Informatik ist sehr facettenreich (z.B. Wedekind et al. 1998, Thomas 2002a). Während Modellierung in der Schule zunächst oft mit Modellbildungswerkzeugen und Simulation verknüpft war (Modell von etwas), gewann zunehmend die Modellierung im Sinne der Software-Entwicklung (Modell für etwas) an Bedeutung. Wegen der vielschichtigen Ausprägungen, auch z.B. im Hinblick auf den Modellbegriff der Mathematik, gibt es sogar Autoren, die empfehlen, den Modellbegriff gar nicht mehr zu verwenden (Schefe 1999, 132).
[5] „Im Informatikunterricht bedeutet ‚Modellierung’ im Wesentlichen die Abgrenzung eines für den jeweiligen Zweck relevanten Ausschnittes der Erfahrungswelt, die Herausarbeitung seiner wichtigen Merkmale unter Vernachlässigung der unwichtigen sowie seine Beschreibung und Strukturierung mithilfe spezieller Techniken aus der Informatik.“ (Breier et al. 2000b, III)
[6] Ab der zweiten Hälfte der neunziger Jahre wurde diese Entwicklung nochmals verstärkt durch die Verbreitung anschaulicher, grafischer Modellierungssprachen in der Fachwissenschaft, wie der *Unified Modelling Language* (UML, Booch et al. 1998) und ihrer Vorläufer.
[7] Die Fachzeitschrift *Informatik betrifft uns* wurde 2000 eingestellt.

zentralen Inhalt modernen Informatikunterrichts, ist aber keineswegs der Auffassung, dass OOM der alleinige Unterrichtsinhalt sein sollte.
Die Ursachen für die bislang geringe Berücksichtigung werden u.a. in folgenden Problembereichen vermutet:

- Es gibt ein generelles Missverhältnis zwischen fachdidaktischen Erfordernissen und anwendbaren Forschungsergebnissen (Schubert 1999, 2001c).
- Es gibt einen Mangel an publizierten Unterrichtsbeispielen, -konzepten und Aufgaben (z.B. Übungsaufgaben) zum objektorientierten *Modellieren*. Bis vor kurzem dominierte die Diskussion der Unterrichtseignung von ausgewählten, objektorientierten Programmiersprachen (z.B. Penon / Spolwig 1998, von Lavergne 1998, 1999; Baumann 2000, 2001a, 2001b[8]) sowie die Erläuterung von Quelltexten zu objektorientiert gelösten Programmieraufgaben konzeptionelle Überlegungen. Es finden sich zwar Publikationen zu Unterrichtsprojekten (z.B. Rollke 1994, 1995; Modrow 1995, 1996; Spolwig 1995; Hermes 1996a, 1996b; Stobbe 1998), kaum aber zu einführenden Beispielen, die geeignet sind, schrittweise an die neuen Techniken heranzuführen oder neu eingeführte Konzepte zu üben oder zu vertiefen.
- Es fehlt an geeigneten Systemen und Werkzeugen für den Lehr-Lern-Prozess (Schwill 1995, Füller 1999, Hubwieser 2000a). Frühzeitig werden hier oft professionelle Software-Entwicklungsumgebungen eingesetzt, die insbesondere in der Anfangsphase kaum dazu geeignet sind, den Lernprozess zu fördern. Mittels geeigneter Werkzeuge, die objektorientierte Prinzipien und Konzepte veranschaulichen und zunächst einen experimentellen Zugang zu diesen ermöglichen, sollen mentale Modelle zur Objektorientierung bei den Lernenden vorbereitet werden. Gerade im Bereich der Verknüpfung von Realitätsausschnitten mit ihren Darstellungen in objektorientierten Modellen sowie in der synchronisierten Verknüpfung von verschiedenen Modellsichten mit dem Ziel, grundsätzliche Wirkprinzipien objektorientiert gestalteter Informatiksysteme als Gesamtbild zu verstehen, eröffnet sich ein neuer, handlungsorientierter und die Codierung zunächst zurück stellender Zugang zu zentralen Informatikkonzepten.
- Ebenso besteht ein Bedarf an Konzepten, die diese neuen Aufgaben und Systeme in den Unterricht integrieren.

Diese skizzierten Problembereiche waren Motivation und Ausgangspunkt für die in dieser Arbeit beschriebene Entwicklung eines *didaktischen Systems für objektorientiertes Modellieren* und dessen exemplarische Erprobung im Informatikunterricht der Sekundarstufe II sowie in der Informatiklehreraus- und -weiterbildung (vgl. Brinda 2000a, 2000b, 2001, 2002, 2003; Brinda / Schubert 2001, 2002a, 2002b, 2003; Brinda / Ortmann 2002). Unter einem didaktischen System wird hierbei ein informatikdidaktisch begründeter Verbund, bestehend aus neuen und traditionellen Komponenten des Lehr-Lern-Prozesses, mit Einsatzempfehlung im Informatikunterricht verstanden (vgl. 3.2). Ziel dieses didaktischen Systems ist es, den Aneignungs- und Vermittlungsprozess zum objektorientierten Modellieren auf verschiedenen Ebenen zu unterstützen und weiter zu entwickeln und damit einen Beitrag zur Entwicklung und Förderung von Bildungsstandards für den Informatikunterricht zu leisten.
Weitere, vieldiskutierte Problembereiche, wie

- die fehlende Tradition in der grundständigen Informatiklehrerausbildung und daraus resultierende Folgeerscheinungen und
- die Tatsache, dass es bislang kein verpflichtendes Informatikfundament in der Sekundarstufe I (Sek. I) für alle Lernenden gibt,

[8] vgl. (Böszörményi 1998) für einen korrespondierenden Beitrag aus dem Bereich der Hochschulbildung

wurden bei der Konzeption des didaktischen Systems als für den Entwicklungszeitraum unveränderbare Einflussgrößen angenommen, deren Anpassung außerhalb des Gestaltungsbereiches des Autors dieser Arbeit liegt.

1.2 Forschungsmethodische Vorgehensweise

Phase 1: Vorbereitung

Den Auftakt bildeten eine Reihe von Unterrichtsbesuchen in den Schuljahren 1998/1999 und 1999/2000 zur Erkundung und Beschreibung des traditionellen Informatikunterrichts und zum Schwerpunkt OOM (Tabelle 1, Phase 1). Im Einzelnen waren dies im:

- *Wintersemester 1998/99*: Prädikatives Modellieren mit Prolog, Gesamtschule, Grundkurs Sek. II,
- *Sommersemester 1999*: Internet-Recherche und elektronisches Publizieren, Hauptschule, Wahlfach Sek. I,
- *Wintersemester 1999/2000*: Einführung in das objektorientierte Modellieren mit Delphi, Gymnasium, Grundkurs Sek. II,
- *Sommersemester 2000*: punktuelle Unterrichtsbesuche zum objektorientierten Modellieren, Grundkurse in der Sek. II.

Betreut durch den Autor entstand 1999 im Rahmen einer Diplomarbeit ein Konzept für „Informatik-Experimente im Schullabor“ (Steinkamp 1999), ein neuer, handlungsorientierter Zugang zu Wirkprinzipien von Informatiksystemen, der die spätere Entwicklung des didaktischen Systems im Bereich der „Explorationsmodule“ (vgl. Kapitel 5) mit beeinflusste (Tabelle 1, Phase 1).

Phase 2: Literaturstudie

Die 2. Phase (Wintersemester 1999/2000) des Forschungsprojektes bildeten Literaturstudien zur Analyse von Lehr-Lern-Konzepten zum objektorientierten Modellieren und Programmieren im Informatikunterricht. Als Ergebnis dieser Studie wurde die Notwendigkeit eines Konzeptes für didaktische Systeme begründet (Brinda / Schubert 2001)[9].

Phase 3: Spezifikation – Theoretische Konzeption des didaktischen Systems

Im Sommersemester 2000 erfolgte die theoretische Konzipierung eines didaktischen Systems für OOM (Brinda 2000a) unter Anwendung und Weiterentwicklung von fachwissenschaftlichen und fachdidaktischen Grundlagen (Tabelle 1, Phase 3). Parallel dazu wurden Werkzeuge für objektorientiertes Modellieren im Rahmen einer vom Autor betreuten Projektarbeit auf ihre Eignung für den Informatikunterricht untersucht und fachdidaktische Anforderungen abgeleitet (Koch / Ortmann 2000). Aufgrund eines Mangels an unterrichtsgeeigneten Übungsaufgaben zum OOM wurden ca. 330 Übungsaufgaben aus der Fachwissenschaft analysiert, Kriterien für deren Unterrichtseignung entwickelt, angewandt und anschließend zu Aufgaben- und Kontextklassen verdichtet und strukturiert (Brinda 2000b). Um die Orientierung von Lernenden und Lehrenden im Bildungsprozess zum OOM zu verbessern, wurden abstrakte und anschauliche Darstellungsformen für Wissensstrukturen und didaktische Landkarten analysiert (Brinda / Schubert 2002a). Hierzu wurden Kriterien entwickelt, auf verschiedene, informatische Darstellungsformen angewandt, und im Ergebnis schließlich so genannte Und-Oder-Graphen ausgewählt (Tabelle 1, Phase 3).

[9] Da diese Literaturstudie zusammen mit ersten Ergebnissen zum Konzept im Rahmen eines Forschungsberichts der Universität Dortmund veröffentlicht wurde, liegt das Veröffentlichungsjahr ein Jahr nach der Erstellung.

Phase	Beschreibung	Aktivitäten	1999	2000	2001	2002	2003	Veröffentlichung(en)
1	Vorbereitung	Erkundung und Beschreibung des traditionellen Informatikunterrichts und zum Schwerpunkt OOM						
		Informatik-Experimente im Schullabor (Diplomarbeit)						Steinkamp 1999
2	Literaturstudie	Analyse von Lehr-Lern-Konzepten zum OOM						Brinda / Schubert 2001
3	Spezifikation - Theoretische Konzeption des didaktischen Systems	Analyse von Werkzeugen zum objektorientierten Modellieren für den Informatikunterricht (Projektarbeit)						Koch / Ortmann 2000
		Theoretische Konzeption eines didaktischen Systems für OOM						Brinda 2000a
		Analyse von Übungsaufgaben zum OOM und Entwicklung und Strukturierung von Aufgabenklassen						Brinda 2000b, Brinda / Schubert 2002a
		Auswahl von Und-Oder-Graphen zur Strukturierung von Fachkonzepten für den Bildungsprozess						Brinda 2002, Brinda / Schubert 2002a
		Entwicklung von Gestaltungskriterien für Explorationsmodule						Brinda / Schubert 2003
		Entwicklung eines Konzeptes zur Einbettung von Explorationsmodulen in den Lehr-Lern-Prozess						Brinda / Schubert 2002b, Brinda 2003
4	Entwicklung von Explorations-modulen und exemplarische Erprobung	Workshops zur Informatiklehrerfortbildung						
		Erprobung von Komponenten des didaktischen Systems in der Informatiklehrerausbildung						
		Entwurfsmuster im Informatikunterricht und als Bestandteil des didaktischen Systems für OOM (Projektarbeit)						Netzer 2002
		Entwicklung eines Explorationsmoduls für geometrische Objekte (SHKs)						
		Entwicklung von LEO - Lernumgebung für objektorientiertes Modellieren im Informatikunterricht (Projektgruppe)						Alex et al. 2002
		Unterrichtliche Einbettung ausgewählter Aufgabenklassen (1. Staatsarbeit)						Ortmann 2002, Brinda / Ortmann 2002
		Präsentation und Diskussion auf Fachtagungen						
		Exemplarische Evaluation von Explorationsmodulen in der Medieninformatikausbildung						
5	Überarbeitung und Verfeinerung	Überarbeitung und Verfeinerung des Konzeptes						Brinda / Ortmann 2002
		Überarbeitung der Explorationsmodule						
		Zusammenfassung und Publikation der Ergebnisse						Brinda 2004

Tabelle 1: Vorgehensweise bei der Entwicklung des didaktischen Systems für OOM

Fortgesetzt wurden die Arbeiten an der theoretischen Konzeption in der zweiten Hälfte 2001 mit der Entwicklung von Gestaltungskriterien für Software-Bausteine zur Exploration objektorientierter Basiskonzepte und eines Lernkonzeptes zur Einbettung in den sozialen Bildungsprozess (Brinda / Schubert 2002b, 2003).

Phase 4: Entwicklung von Explorationsmodulen und exemplarische Erprobung
Schwerpunkte der 4. Phase waren der Entwurf und die Implementierung von so genannten Explorationsmodulen (Alex et al. 2002) sowie die exemplarische Erprobung zur Prüfung der Tragfähigkeit des Konzeptes im Bereich der Informatiklehrerbildung (hier im Bereich der Informatikdidaktiklehre des Lehramtsstudiums an der Universität Dortmund (ca. 10 Studierende) sowie in Workshops zur Weiterbildung von ca. 200 Informatiklehrenden) mit dem Ziel der Gestaltungsrückkoppelung sowie ersten Einsatztests im Informatikunterricht (ca. 100 Lernende) der Sekundarstufe II (Ortmann 2002, Brinda / Ortmann 2002). Diese Phase verlief in Teilen parallel zur 3. Phase, da bereits entwickelte Komponenten des didaktischen Systems direkt in der Lehrerbildung erprobt werden konnten.
Die Evaluation ausgewählter Explorationsmodule wurde Anfang 2003 in der E-Learning-Ausbildung von Medieninformatikern an der Universität Siegen fortgesetzt.

Phase 5: Überarbeitung und Verfeinerung
In der abschließenden 5. Phase wurde auf Basis der aus der exemplarischen Erprobung gewonnenen Erkenntnisse eine Überarbeitung und Verfeinerung des in der 3. Phase erarbeiteten theoretischen Konzepts mit Anpassungen an den Explorationsmodulen vorgenommen und die vorliegende Abschlussarbeit angefertigt.

1.3 Beitrag der Arbeit

Der wesentliche Beitrag der vorliegenden Arbeit liegt im Bereich der Informatiklehrerausund -weiterbildung sowie in der informatischen Bildung an Schulen. Orientiert an der Klassifikation von Forschungsarbeiten der ACM von 1998[10] ist dies der folgende Schwerpunkt (lt. ACM „Categories and subject descriptors“):

- **K.3.2 [Computers and Education]: Computer and Information Science Education---*Computer science education, Curriculum, Literacy, Self-assessment***

Gegenstand des beschriebenen Bildungskonzeptes sind Fachkonzepte aus dem Bereich der objektorientierten Software-Entwicklung. Bezogen auf die ACM-Klassifikation sind dies im Wesentlichen folgende Schwerpunkte:

- D.2.2 [Software Engineering]: Design Tools and Techniques---*Computer-aided software engineering (CASE), Object-oriented design methods,*
- D.3.3 [Programming Languages]: Language Constructs and Features---*Classes and objects, Control structures, Data types and structures, Inheritance, Patterns, Polymorphism, Procedures, functions, and subroutines,*
- D.1.5 [Programming techniques]: Object-oriented Programming.

Tabelle 2 gibt einen Überblick über wesentliche Ergebnisse der Arbeit im Hinblick auf die Informatiklehrerbildung bzw. Beiträge zur Didaktik der Informatik.

[10] URL: http://www.acm.org/class/1998/ (aufgerufen am 09.11.03)

Didaktisches System	Aufgabenklassen (AKn)	Explorationsmodule (EMe)	Wissensstrukturen (WSen)
Vorgehensweise zur Konzeption eines didaktischen Systems • Analyse des Bildungsbereiches (hier: OOM im Informatikunterricht) • Identifikation und Spezifikation von Problembereichen (Mangel an: unterrichtsgeeigneten Aufgaben, unterrichtsgeeigneter Software zur Förderung des Aneignungsprozesses, Empfehlungen zur Strukturierung des Lehr-Lern-Prozesses (LLPes) • Entwurf eines didaktischen Systems als erweiterbarer Verbund von Komponenten des LLPes (AKn, EMe, WSen) zur Bewältigung der Problembereiche • Begründung der informatikdidaktischen Funktionen der Komponenten (Gestaltungsmittel für LLPe, Anwendung in LLPen, Förderung der fachdidaktischen Kommunikation und Diskussion zu LLPen, Vorbereitung von Bildungsstandards) • Erprobung der Komponenten in der informatischen Bildung • Konzept für die Informatiklehrerbildung • Überarbeitung der Komponenten aufgrund der Erprobungsergebnisse	**Vorgehensweise zur Entwicklung von AKn** • Auswahl- bzw. Transformationskriterien für Übungsaufgaben aus fachwissenschaftlichen Lehrbüchern für den Informatikunterricht • Methodik zur Abstraktion von Aufgaben zu AKn • Identifikation von Aufgabentypen • Klassifikation von Aufgaben (Fachkern, Gegenstand, Aufgabentyp) • Strukturierte Sammlung von AKn • Repräsentationsmittel für die Struktur von AKn zur Kommunikation **Methodik zur Gestaltung von Aufgaben mittels AKn** • Gestaltung von Niveaustufungen bei Aufgaben zum OOM • Auswahlkriterien für Kontexte • Bewertung von Kontextklassen **Erprobungen in der informatischen Bildung (Sek. II, Lehrerbildung)** • Konzeption (Fallstudien, schriftliche Befragung) • Ergebnisse	**Vorgehensweise zur Entwicklung von EMen** • Fachdidaktische Analyse von OOM-Werkzeugen (Auswahl, Unterrichtseignung, erkenntnisförderliche / -hemmende Funktionen) • Konzept für EMe zum OOM (Sichtenkonzept, Konkretisierung für den Bereich der OOM) • Begründung und Entwicklung von Lernsoftware für den Unterricht und die Lehrerbildung (Puzzles, Experimentierumgebungen, EGO[11], LEO[12]) • Klassifizierung und Verallgemeinerung der EMe • Architekturkonzept für EMe **Konzept für das Lernen mit EMen** • pädagogische Doppelfunktion von EMen • Explorationsstrategien • Verknüpfung mit problemorientiertem Unterricht **Erprobungen in der informatischen Bildung (Sek. II, Lehrerbildung, Informatikstudium)** • Konzeption • Ergebnisse	**Repräsentation der Struktur von Lehr-Lern-Prozessen** • Theoretische Begründung von fachdidaktischen Funktionen der Repräsentation • Begründung von Anforderungskriterien an eine Repräsentationsform • Fachdidaktische Analyse von Begriffsnetzen, Klassendiagrammen, Objektdiagrammen, Semantischen Netzen, Und-Oder-Graphen • Vorgehensweise zur Entwicklung einer Didaktischen Landkarte

Tabelle 2: Ergebnisse für die Informatiklehrerbildung / Didaktik der Informatik

[11] EGO – Explorer für geometrische Objekte, vgl. 5.3.4.3
[12] LEO – Lernumgebung für objektorientiertes Modellieren im Informatikunterricht, vgl. 5.3.4.4

1.4 Gliederung der weiteren Arbeit

Kapitel 2: Fachdidaktischer Stand zum objektorientierten Modellieren

Im Kapitel 2 wird der fachdidaktische Stand zum objektorientierten Modellieren im Informatikunterricht anhand einer Literaturstudie ausführlich analysiert (vgl. 2.2, 2.3). Es werden Schlussfolgerungen für die Gestaltung informatischer Bildung im Bereich des objektorientierten Modellierens gezogen (vgl. 2.4.2) und die wissenschaftlichen Fragestellungen der vorliegenden Arbeit abgeleitet (vgl. 2.4.3).

Kapitel 3: Konzeption eines didaktischen Systems

Ein so genanntes „didaktisches System für OOM“ zur Bewältigung der in Kapitel 2 identifizierten Problembereiche wird im Kapitel 3 konzipiert. Nach Präzisierung der Zielsetzung und Definition des didaktischen Systems (vgl. 3.2) werden die Systemkomponenten „Aufgabenklassen“ (vgl. 3.3), „Explorationsmodule“ (vgl. 3.4) und „Wissensstrukturen“ (vgl. 3.5) konkretisiert und im Abschnitt 3.6 miteinander verknüpft. Im Abschnitt 3.7 wird das im Rahmen der Arbeit entwickelte und exemplarisch erprobte Konzept für die Informatiklehrerbildung vorgestellt. Abschließend wird das Konzept des didaktischen Systems für OOM von anderen Arbeiten zum objektorientierten Modellieren im Informatikunterricht abgegrenzt, die parallel zu dessen Entwicklung veröffentlicht wurden (vgl. 3.8).

Kapitel 4: Aufgabenklassen

Eine Vorgehensweise zur Entwicklung einer strukturierten Sammlung von Aufgabenklassen zum OOM wird im Kapitel 4 begründet und angewandt (vgl. 4.3). Auf der Basis dieser Sammlung wird eine Methodik zur Gestaltung von (Übungs-)Aufgaben zum objektorientierten Modellieren für unterschiedliche Niveaustufungen entwickelt (vgl. 4.4). Ergebnisse der Erprobung des Konzeptes in der Sekundarstufe II und in der Lehrerbildung werden in den Abschnitten 4.5 und 4.6 dargestellt und analysiert, welche Konsequenzen diese Ergebnisse für die Weiterentwicklung des Konzepts implizieren.

Kapitel 5: Explorationsmodule

Im Kapitel 5 wird eine Vorgehensweise zur Entwicklung von Software zur Anregung explorativen Lernens im Bereich objektorientierter Basiskonzepte (so genannte Explorationsmodule) vorgestellt und angewandt (vgl. 5.3). Dieses Gestaltungskonzept wird anschließend mit einem Konzept für das Lernen mit Explorationsmodulen verknüpft (vgl. 5.4). Ergebnisse der exemplarischen Erprobung des Lernkonzepts in der informatischen Bildung werden im Abschnitt 5.5 präsentiert und ausgewertet.

Kapitel 6: Wissensstrukturen

Den Ausgangspunkt von Kapitel 6 stellt die Analyse fachwissenschaftlicher und fachdidaktischer Erarbeitungsstrukturen dar (vgl. 6.2). Auf der Basis von Ergebnissen zu „Mapping Techniken“ (vgl. 6.3) werden verschiedene, fachdidaktische Funktionen einer Repräsentation der Struktur von Lehr-Lern-Prozessen begründet, Anforderungen an Darstellungsformen präzisiert und verschiedene graphbasierte Darstellungsformen im Hinblick auf die Erfüllung der Anforderungen analysiert (vgl. 6.4). Ein im Rahmen einer studentischen Projektgruppe entwickeltes Vorgehensmodell zur Gestaltung einer didaktischen Landkarte wird im Abschnitt 6.5 präsentiert.

Kapitel 7: Zusammenfassung, Fazit und offene Fragen

Den Abschluss der Arbeit bildet Kapitel 7 mit Zusammenfassung (vgl. 7.1), Fazit (insb. zu den wissenschaftlichen Fragestellungen der Arbeit, vgl. 7.2) und offenen Fragen (vgl. 7.3).

2 Fachdidaktischer Stand zum objektorientierten Modellieren

2.1 Überblick

Seit Anfang der neunziger Jahre wird objektorientiertes Modellieren für den Informatikunterricht in deutschsprachigen, informatikdidaktischen Schriften zunehmend diskutiert und betont. In diesem Kapitel wird der fachdidaktische Stand zum objektorientierten Modellieren im Informatikunterricht dargestellt und analysiert. Dazu wird zunächst im Abschnitt 2.2 der in der Didaktik der Informatik gewachsene Stellenwert der Modellierung herausgestellt und damit die fachdidaktischen Rahmenbedingungen für die informatische Bildung in Schulen beschrieben. Zur Charakterisierung der nationalen Ausgangslage zum objektorientierten Modellieren im Informatikunterricht zu Beginn des hier beschriebenen Forschungsprojektes um 2000/2001 wurden Habilitations- und Promotionsschriften zur Didaktik der Informatik, Beiträge in den Tagungsbänden der Fachtagung *Informatik und Schule – INFOS*[13] der Gesellschaft für Informatik e.V. (GI) und der Web-Zeitschrift zu fachdidaktischen Grundlagen der Informatik *Informatica Didactica*[14] sowie ausgewählte Bildungsempfehlungen, Unterrichtserfahrungen und -beispiele (letztere publiziert in den beiden deutschsprachigen Fachzeitschriften für Informatiklehrerinnen und -lehrer *LOG IN* und *Informatik betrifft uns*) analysiert (vgl. 2.3.1). Wesentliche, parallel zur Entwicklung des im Rahmen dieser Arbeit beschriebenen Konzeptes publizierte, Beiträge zum OOM im Informatikunterricht werden im Abschnitt 3.8 erörtert. Die internationale fachdidaktische Diskussion zum objektorientierten Modellieren wird bislang überwiegend für die Hochschulausbildung geführt, z.B. auf den Fachkonferenzen der *Special Interest Group for Computer Science Education (SIGCSE)* der Association for Computing Machinery (ACM). Da einige der Ergebnisse im Bereich der Hochschulausbildung auch für die Entwicklung von Konzepten für die Sekundarstufe II geeignet sind, werden ausgewählte Arbeiten aus diesem Bereich berücksichtigt (vgl. 2.3.2). Im Abschnitt 2.4 werden aus den gewonnenen Erkenntnissen Schlussfolgerungen für die Entwicklung eines didaktischen Systems gezogen. Es erfolgt im Wesentlichen eine chronologische Darstellung, wobei Arbeiten derselben Autoren oder Arbeiten zum selben Gegenstand zusammengefasst wurden.
Zu beachten ist, dass der Fachbegriff *objektorientiertes Modellieren* sowie verwandte Termini in der fachdidaktischen Literatur nicht einheitlich verwendet werden. Während einige Autoren darunter den Gestaltungsprozess objektorientierter Software verstehen, beziehen andere dies nur auf die Analyse- und Entwurfsphase oder noch spezifischer auf die Arbeit mit grafischen Modellierungstechniken. Fachdidaktische Arbeiten, die sich in Teilen auch auf objektorientiertes Modellieren beziehen, diesen Begriff aber nicht verwenden, benutzen anstatt dessen oft verwandte Termini, wie z.B.

- *objektorientierte Analyse* und *Entwurf*,
- *objektorientierter Ansatz*,
- *objektorientierter Programmierstil*,
- *objektorientierte Programmierung*,
- *objektorientierte Methode*,
- *objektorientiertes Paradigma* oder
- *Objektorientierung*.

Welches genaue Begriffsverständnis sich jeweils dahinter verbirgt, bleibt zumeist offen. Aus

[13] Auf den INFOS-Tagungen wird im Zweijahrestakt der Forschungsstand zur Didaktik der Informatik in Schulen diskutiert. Die ersten INFOS-Tagungen fanden 1984 in Berlin und 1986 in Kaiserslautern statt. Der Zweijahrestakt gilt seit der 1989er Tagung in München.
[14] URL: http://www.informaticadidactica.de/ (aufgerufen am 12.11.03)

diesem Grund werden in der nachfolgenden Literaturanalyse die von den jeweiligen Autoren verwendeten Begriffe beibehalten. An allen anderen Stellen der vorliegenden Arbeit verwendet der Autor den Modellierungsbegriff im Sinne der allgemein gehaltenen Definition aus den Empfehlungen für ein Gesamtkonzept der informatischen Bildung an allgemein bildenden Schulen (Breier et al. 2000b, III):

> „Im Informatikunterricht bedeutet ‚Modellierung' im Wesentlichen die Abgrenzung eines für den jeweiligen Zweck relevanten Ausschnittes der Erfahrungswelt, die Herausarbeitung seiner wichtigen Merkmale unter Vernachlässigung der unwichtigen sowie seine Beschreibung und Strukturierung mithilfe spezieller Techniken aus der Informatik."

Im Sinne dieser Definition sind auch Programmiersprachen als Beschreibungsmittel nicht ausgeschlossen. Soll eine explizite, sprachliche Abgrenzung von der Implementierungsphase, der Verwendung von Programmiersprachen zur Codierung etc. zum Ausdruck gebracht werden, so wird hierfür der Begriff objektorientiertes Programmieren (OOP) verwendet.

2.2 Informatisches Modellieren

2.2.1 Betonung der Modellierung

Obwohl sich bereits frühe Forderungen nach „Methoden der Modellbildung und Problemlösung" (Brauer / Brauer 1973) für den Informatikunterricht finden lassen, konnten sich der Modellierungsbegriff und dessen Thematisierung in der informatischen Bildung zunächst nicht durchsetzen. In den ersten ***Empfehlungen der Gesellschaft für Informatik e.V.*** (GI) für Informatikunterricht[15] ***von 1976*** wurde er nicht verwendet. Nach diesen Empfehlungen sollten die Lernenden aber Kompetenzen erwerben, die heute der Modellierung zugeordnet werden:

> „Der zunehmende Einsatz von Rechenanlagen verlangt von jedem Schulabgänger, dass er die Fähigkeit besitzt, einfache Probleme zu analysieren, deren Lösungsabläufe zu entwickeln und diese so zu beschreiben, daß sie schließlich auf einer Rechenanlage ausgeführt werden können." (Brauer et al. 1976, 35)

Der Erwerb von Kompetenzen zu den Bereichen Analyse, Entwurf und Implementierung war also auch zu dieser Zeit schon ein wichtiges Bildungsziel. Damals verstand man unter Entwurf aber im Wesentlichen die Entwicklung von Algorithmen und Datenstrukturen, heute sind es hingegen zumeist Software-Pakete und Klassenstrukturen.
Für ***Koerber*** und ***Peters*** (1989) steht „im Zentrum der Informatik: Problemlösen und Modellbilden". Sie beschreiben die Modellbildung als eine Phase eines fünfschrittigen „Problemlösungs[...][prozesses] mit Hilfe des Werkzeugs Computer", in dem sie Methoden der Informatik als Inhalt und didaktisches Prinzip realisieren. Diese Strategie findet sich auch in späteren Arbeiten, z.B. bei Hubwieser (2000b). Die Bedeutung der Modellierung wird betont.
In den ***GI-Empfehlungen von 1993*** (Schulz-Zander et al. 1993) wird das Modellieren den Zielen und Inhalten des Informatikunterrichts im Bereich „A): Wechselwirkung Mensch – Computer"[16] zugeordnet. Lernende sollen demnach „exemplarische Methoden und Verfahren der Modellierung eines Ausschnittes der Wirklichkeit kennenlernen, diese anwenden und kritisch hinterfragen" und „einen Überblick [...] [über] unterschiedliche Ansätze und Darstellun-

[15] Die verschiedenen Empfehlungen für Informatikunterricht der GI bündeln zu ihren jeweiligen Erscheinungszeitpunkten die Positionen der mitwirkenden Fachdidaktiker und anderer Personen mit dem Ziel der Einflussnahme auf die weitere Bildungsplanung im Bereich der Informatik. Für die vorliegende Arbeit ist dieser Bündelungsaspekt wichtiger, als die schlussendliche Bedeutung der einzelnen Empfehlungen im Prozess der Weiterentwicklung von Richtlinien und Lehrplänen.
[16] Die weiteren Bereiche waren *B) Formalisierung und Automatisierung geistiger Arbeit* und *C) Informatiksysteme, Gesellschaft und Umwelt.*

gen der Modellierung mit dem Computer [gewinnen]“ (ebd., 208). Modellierung wird hier als ein wichtiger Unterrichtsinhalt betont.

Schwill veröffentlichte 1993 seine Ergebnisse zu „fundamentalen Ideen der Informatik“ (Schwill 1993a). Auf Basis der Vorarbeiten von Bruner (1960) lieferte er Kriterien (Horizontal-, Vertikal-, Zeit-, Sinnkriterium) zur Auswahl von Lerninhalten für allgemein bildenden Informatikunterricht, wandte diese auf Informatikfachkonzepte an und identifizierte die Masterideen „Algorithmisierung“, „Strukturierte Zerlegung“ und „Sprache“. Auch wenn er in seiner Analyse im Wesentlichen von Fachkonzepten aus den Bereichen Algorithmen und Datenstrukturen ausging, stützen die identifizierten Masterideen dennoch die Bedeutung des Modellierungsbegriffs (hier am Beispiel der objektorientierten Modellierung), da die strukturierte Zerlegung mit der Entwicklung von Software-Paketen und Klassenstrukturen korrespondiert, die Algorithmisierung mit der Entwicklung von Methoden und die Sprache im Sinne von Modellierungs- und Programmiersprachen für die Dokumentation und Kommunikation in diesem Zusammenhang von großer Bedeutung ist. Von Schwill wurde somit ein Fundament für die Begründung des allgemeinen Bildungswertes der Modellierung im Informatikunterricht gelegt. Im Abschnitt 2.4.2 werden die Fundamentalitätskriterien von Schwill auf den Themenbereich der Objektorientierung angewandt.

In der Beschreibung der Ergebnisse der zweiten fachdidaktischen Gespräche zur Informatik[17] fasst ***Friedrich*** (1995) Vorarbeiten zur Entwicklung eines Gesamtkonzeptes für die informatische Bildung in der Sek. I und II zusammen. Modellierung und Modelle spielen in zwei der vier beschriebenen Leitlinien für informatische Bildung eine zentrale Rolle. Diese sind das „Problemlösen mit Informatiksystemen“ und das „Arbeiten mit Modellen“[18] (Friedrich 1995, 31).

Vor dem Hintergrund der gewachsenen Bedeutung des Modellierungsbegriffes, der vielfältigen Erfahrungen mit den klassischen informatikdidaktischen Ansätzen, die alle nicht geeignet waren, den Lernenden die erforderlichen Kompetenzen zur Beherrschung und Bewertung von hochkomplexen Informatiksystemen angesichts einer sich rasant weiter entwickelnden Fachwissenschaft zu vermitteln, entwickelte ***Hubwieser*** in seinen Arbeiten seit 1996 einen neuen fachdidaktischen Ansatz, den „informationszentrierten Ansatz“ (Hubwieser / Broy 1996, 1997, 1999; Hubwieser et al. 1997; Hubwieser 1999a, 1999b, 2000a, 2000b), der informatisches Modellieren als Lerninhalt („Erlernen von Modellierungstechniken zur Beschreibung komplexer Systeme“) und Lernmethode („Erarbeitung der grundlegenden Prinzipien von Informatiksystemen durch ihre Modellierung“) in den Mittelpunkt des Unterrichts rückt (Hubwieser 2000b, 50f) und der sich als Vereinigung und Abstraktion früherer fachdidaktischer Ansätze versteht (ebd., 47). Schwerpunkte des Lehr-Lern-Prozesses bilden die Repräsentation, Verarbeitung, Transport und Interpretation von Information. Ferner begründet er, warum sich die Behandlung des Modellierungsvorganges bislang im Unterricht nicht durchsetzen konnte:

> „Man [verfügte] bis vor kurzem nicht über geeignete Techniken, um diesen Modellierungsvorgang im Unterricht systematisch und in angemessener Tiefe umsetzen zu können. Aus diesem Mangel heraus gerieten die Betrachtungen zu diesem Thema im Unterrichtsgeschehen oft zu rein philosophischen, wenig schülergemäßen Exkursen. Inzwischen haben sich jedoch auf dem Gebiet der Softwareentwicklung Modellierungstechniken durchgesetzt, die aufgrund ihrer Anschaulichkeit und Beschreibungsmächtigkeit geeignet scheinen, genau diese methodische Lücke zu schließen.“ (ebd., 50)

Er bezieht sich dabei u.a. auf die Unified Modeling Language (UML, Booch et al. 1998) und ihre Vorläufer (Rumbaugh et al. 1991, Booch 1994). Weiterhin liefert Hubwieser Argumente

[17] Diese finden als offener und informeller Erfahrungsaustausch zwischen Informatikdidaktikern und Informatiklehrenden im Jahrestakt in Königstein in der sächsischen Schweiz unter der Leitung von Prof. Dr. S. Friedrich von der TU Dresden statt.

[18] Weitere Leitlinien waren: „Umgang mit Informationen“ und „Wirkprinzipien von Informatiksystemen“

für den allgemein bildenden Wert des informatischen Modellierens. Dazu teilt er den Anwendungsbereich von Lerninhalten in Bezug auf Informatiksysteme in vier Klassen ein (Hubwieser / Broy 1997a, 44). Modellierungstechniken und Problemlösungsstrategien ordnet er den breitesten Anwendungsbereich zu, da er sie auch für außerhalb des Bereiches der EDV für anwendbar hält. Syntaxelemente einer konkreten Programmiersprache, die Menüstruktur eines Anwendersystems oder die Architektur eines Mikroprozessors, typische Repräsentanten der ersten fachdidaktischen Ansätze, betreffen nur ein konkretes System und haben daher in seiner Klassifikation die geringste Allgemeingültigkeit[19]. Das Konzept von Hubwieser wird gegenwärtig an über sechzig bayerischen Gymnasien in den Jahrgangsstufen 6, 10 und 11 erprobt (Hubwieser 1999a, Frey et al. 2001) und ein Pflichtfach Informatik an bayerischen Schulen damit fachdidaktisch vorbereitet. Es startet 2004 in der 6. Jahrgangsstufe (Jgst.) als zweistündiges Unterrichtsfach und wird an den naturwissenschaftlich-technologischen Gymnasien darüber hinaus in den Jahrgangsstufen 9 bis 11 fortgesetzt (Frey et al. 2001, 20) [20].

2.2.2 Programmierung und Modellierung

Die Diskussion zum Stellenwert der *Programmierung* im Informatikunterricht wird ebenfalls seit längerem geführt (für eine Zusammenfassung wesentlicher Positionen siehe z.B. Hubwieser 2000a, 87ff). ***Hubwieser*** kommt unter Bezugnahme auf ***Schubert*** (1991) zu dem Schluss, dass diese Diskussion in Teilen auf unterschiedliche Interpretationen des Begriffs *Programmierung* zurückzuführen ist. Der Duden Informatik definiert Programmierung wie folgt[21]:

> „Unter Programmierung versteht man zum einen den Vorgang der Programmerstellung und zum anderen das Teilgebiet der Informatik, das die Methoden und Denkweisen beim Entwickeln von Programmen umfasst." (Claus / Schwill 2001, 512)

Programmierung ist also deutlich mehr als Codierung. Dennoch werden diese Begriffe oft fälschlicherweise synonym verwendet und damit Missverständnisse ausgelöst. Der Autor stimmt mit den Positionen von Schubert (1991) und Hubwieser (2000b) überein, die

> „Programmierung für einen wesentlichen, wenn auch nicht den wichtigsten Teil einer allgemein bildenden Schulinformatik [halten], allerdings nur, solange zwei Bedingungen erfüllt sind:
> - Es muss sich (im Sinne von Duden Informatik (1993) wirklich um die *Implementierung* eines vorher entwickelten Modells handeln, also keineswegs um Stegreifprogrammierung nach dem Prinzip ‚Versuch und Irrtum'.
> - Die *Syntax* der verwendeten Sprache darf nicht in den Vordergrund treten. Die Schüler dürfen also keinesfalls durch die Eigenheiten der jeweiligen Sprache von der Problematik der Modellierung abgelenkt werden. Ein ‚Programmierkurs', der von Sprach- anstatt von Problemstrukturen ausgeht, ist im Pflichtfachbereich fehl am Platze.
>
> Unter Einhaltung dieser Regeln kann die Programmierung als Implementierung von abstrakten Modellen dafür sorgen, dass diese durch Simulation veranschaulicht und überprüft werden können, was vielen Schülerinnen und Schülern erst den eigentlichen Zugang zur Modellierung ermöglichen wird." (Hubwieser 2000b, 52f, Hervorhebungen im Original)[21]

Programmierung als Umsetzung zuvor entworfener Modelle und als Werkzeug zu deren Überprüfung hat im allgemein bildenden Informatikunterricht eine wesentliche Bedeutung. Es ist aber zu untersuchen, inwieweit insbesondere der zweite Aspekt durch andere Formen der Validierung (z.B. mittels Modellierungswerkzeugen oder Rollenspielen) ersetzbar ist.

---

[19] Die beiden weiteren Klassen werden beschrieben mit „charakteristisch für alle elektronischen Informatiksysteme" und „Anwendung beschränkt sich auf eine Klasse von Informatiksystemen". Zugeordnet werden diesen im ersten Fall „Komplexitätsklassen, Grenzen der Berechenbarkeit" und im zweiten „Programmierparadigmen, spezielle Datenstrukturen, Netzwerktopologien, Sortieralgorithmen" (Hubwieser 2000b, 48).

[20] URL: http://www.isb.bayern.de/gym/lehrplaene/lehrpl.htm (aufgerufen am 01.12.03)

[21] Hubwieser bezieht sich in seiner Argumentation auf die Definition der Programmierung lt. Duden Informatik in der Ausgabe von 1993. In der Ausgabe von 2001 (Claus / Schwill 2001) blieb diese Definition unverändert.

2.2.3 Stellenwert der Modellierung

Zum Stellenwert der Modellierung im Informatikunterricht gibt es inzwischen einen breiten fachdidaktischen Konsens, wie aktuelle Bildungsempfehlungen und fachdidaktische Forschungsarbeiten belegen. Im Februar 2000 veröffentlichte[22] die ***Kommission „Informatik"***[23] ***des MSWF***[24] erstmals eine Gliederung zur Didaktik der Informatik. Diese stützte sich auf die vier Säulen:

- „Didaktische Konzeption des Unterrichts,
- Informatisches Modellieren[25],
- Interaktion und Kommunikation mit Hilfe von Informatiksystemen[25],
- Gestaltung von Curricula und Unterrichtsszenarien".

Das informatische Modellieren wird als ein Schwerpunktbereich betont, der sich in die Unterpunkte:

- „Gestaltung und Bewertung informatischer Modelle,
- Wirkprinzipien von Informatiksystemen,
- Bewertung von Entwurfsprinzipien und Sprachklassen und
- Konstruktion und Dekonstruktion"

aufgliedert. Die ***Empfehlungen der Gesellschaft für Informatik e.V***. für ein Gesamtkonzept zur informatischen Bildung an allgemein bildenden Schulen (GIFA 1999; Breier et al. 2000a, 2000b) begründen moderne Bildungsleitlinien für Informatikunterricht, wie „Wirkprinzipien von Informatiksystemen" und „Informatisches Modellieren"[26]. Gezeigt wird, wie allgemein bildender Informatikunterricht realisiert werden kann, wenn sich dieser an den vorgeschlagenen Leitlinien orientiert. Ziel der Leitlinie zum informatischen Modellieren ist es, dass

> „Schülerinnen und Schüler verstehen, dass jedes Informatiksystem als Kombination von Hard- und Software-Komponenten das Ergebnis eines informatischen Modellierungsvorgangs ist, das nach seiner Fertigstellung als Bestandteil der realen Welt mit allen Eigenschaften eines unvollständigen, künstlichen Systems wirkt. Sie kennen informatische Modellierungstechniken und können sie zur Beschreibung der Struktur von Informatiksystemen und zur Lösung komplexerer Probleme anwenden. Die bei der Analyse von Informatiksystemen kennen gelernten Modellierungstechniken ermöglichen den Schülern dabei auch ganz allgemein die Strukturierung umfangreicher Datenbestände und die Orientierung in komplexen Informationsräumen." (Breier et al. 2000b, IIIf)

Es zeigt sich eine große Vielfalt an unterrichtlichen Möglichkeiten. Die Empfehlungen für die Sek. II betonen informatisches Modellieren mit verschiedenen Modellierungsverfahren für unterschiedliche Problemlösestrategien:

> „Das Aufzeigen der Struktur eines Problems durch Modellierung führt zum geeigneten Lösungsmodell. In das informatische Modellieren wird mit den Phasen Problemgewinnung, informelle Problembeschreibung, formale Modellierung, Realisierung von Lösungsansätzen und Bewertung eingeführt. Verschiedene Modellierungsverfahren gehören dabei zu verschiedenen Problemlösestrategien. Die Schülerinnen und Schüler vertiefen ihre Kenntnisse und Fähigkeiten an ausgewählten Prinzipien, Methoden und Werkzeugen für die Simulation der Modelle durch Programme oder andere geeignete Mittel." (ebd., VI)

Es zeigt sich also, dass informatisches Modellieren zu einem festen Bestandteil moderner Bildungsempfehlungen geworden ist. Soweit wie Hubwieser, der ein ganzes Curriculum ab der

---

[22] http://www.didaktik-der-informatik.de/DIE_BIB/Bildungsempfehlungen/kom_inf/ (aufgerufen am 12.11.03)

[23] In der Kommission zur Neugestaltung der Landeslehrerprüfungsordnung (Teil Informatik) waren drei der zu diesem Zeitpunkt vier Hochschullehrer zur Didaktik der Informatik aus Deutschland vertreten. Dadurch erhielten die erzielten Ergebnisse bundesweite Bedeutung.

[24] Ministerium für Schule, Wissenschaft und Forschung des Landes Nordrhein-Westfalen

[25] unter fachdidaktischen Gesichtspunkten

[26] Weitere Leitlinien sind die „Interaktion mit Informatiksystemen" und die „Wechselwirkung zwischen Informatiksystemen, Individuum und Gesellschaft".

Jahrgangsstufe 6 basierend auf der Thematisierung von Modellierungstechniken entwirft, gehen diese Empfehlungen aber nicht.
Verschiedene neuere Forschungsarbeiten zur Didaktik der Informatik befassen sich mit Aspekten der Modellierung im Informatikunterricht. So arbeitete ***Thomas*** (1998, 1999, 2001, 2002a, 2002b) zur Klärung des Modellierungsbegriffs. Er analysierte deutschsprachige Vorlesungsskripte der Informatik mit dem Ziel, verschiedene Verwendungen des Modellierungsbegriffs in Teilgebieten der Informatik zu identifizieren und zu strukturieren. Er beschreibt fünf Hauptmodelltypen als Metamodelle (Architekturmodelle, Vorgehensmodelle, Entwurfsmodelle, Untersuchungsmodelle und mentale Modelle) und fordert, viele dieser Facetten des Modellierungsbegriffs im Informatikunterricht zu thematisieren. ***Humbert*** (2001a, 2001b, 2003) befragte sechzehn Informatiklehrpersonen aus sieben Bundesländern mit Gesprächsleitfaden u.a. zu ihrer Position zur informatischen Modellierung. Ausgewählt wurden praktizierende Lehrende, die aktive Beiträge zur (Weiter-)Entwicklung der Fachdidaktik leisten (z.B. durch Publikationstätigkeit im Rahmen der INFOS[27]-Tagungen) und denen deshalb ein Expertenstatus zugeschrieben werden kann. Er kommt dabei u.a. zu folgenden Ergebnissen:

> „Informatische Modellierung wurde als zentrales Konzept von allen interviewten Expertinnen und Experten hervorgehoben. [...] Es ist darüber hinaus festzustellen, dass [...] der objektorientierten [Modellierung] eine Schlüsselrolle zugesprochen wird. [...] Viele Expertinnen und Experten machten in ihren Ausführungen deutlich, dass die Verkürzung auf eine Art der informatischen Modellierung im Informatikunterricht vermieden werden sollte. In der Umsetzung komme es dabei weniger auf Vollständigkeit (vor allem bezogen auf Implementierung) als vielmehr auf das Wissen um alternative Modellierungsmöglichkeiten an.“ (Humbert 2001b, 73f)

Auch dies belegt den fachdidaktischen Konsens zum Stellenwert der informatischen Modellierung. Zusammenfassend lässt sich also feststellen, dass informatisches Modellieren von Fachdidaktikern und Expertinnen und Experten der Schulpraxis zunehmend betont und gefordert wird, verbunden mit dem Hinweis, informatische Modellierung nicht nur auf eine Art zu verkürzen. Die Programmierung als Implementierung und Werkzeug zur Überprüfung zuvor entworfener Modelle wird weiterhin als wesentlicher Teil der Schulinformatik betrachtet.
Generell ist zu beachten, dass ein Verzicht auf eine starke Programmiersprachenorientierung zur Folge hat, dass andere Darstellungsformen informatischer Modelle, z.B. grafische, wie die Unified Modeling Language (UML) im Bereich des objektorientierten Modellierens (vgl. Booch et al. 1998), für die Kommunikation im Unterricht stärkeres Gewicht gewinnen. Ist das Ziel ein konzeptioneller Wandel, so muss sich dieser Wechsel allerdings auch auf die Gestaltung der Lehr-Lern-Prozesse niederschlagen. Ansonsten besteht die Gefahr, dass lediglich die Programmiersprache durch eine andere Metasprache ausgetauscht wird, anhand deren Sprachelementen wiederum eine Strukturierung von Lehr-Lern-Prozessen erfolgen könnte. Damit würde letztlich aber lediglich die vielfach kritisierte (Programmier-)Sprachenorientierung auf einer anderen Darstellungsebene wiederholt.
Weiterführende Informationen zur Modellierung im Informatikunterricht, z.B. zum Stellenwert der Modellierung in den Informatikcurricula der Bundesländer, finden sich bei Hubwieser (2000b) und Thomas (2002a).

[27] GI-Fachtagung „Informatik und Schule – INFOS“

2.3 Objektorientiertes Modellieren

2.3.1 Nationale Diskussion

2.3.1.1 Empfehlungen, Konzepte und Ergebnisse

Auf den INFOS-Tagungen 1989 und 1991 werden noch keine Forschungsergebnisse zum OOM im Informatikunterricht beschrieben. Objektorientierung wird als Gestaltungstechnik für Lehr-Lern-Software eingesetzt (Rauch 1989, Schnitzler et al. 1991) und als ein fakultativer Lehrinhalt in der Universitätsausbildung im Nebenfach Informatik (Matthäus / Schleiff 1991) erwähnt.

Husch (1993) stellt ein auf der Entwicklung von Prototypen basierendes Konzept für die Durchführung von Software-Projekten im Informatikunterricht vor, das motiviert ist durch die Schwierigkeit, die Größe schulischer Software-Projekte richtig zu dimensionieren: einerseits sei die Durchführung von vollständigen, realitätsnahen Software-Projekten in der Schule aufgrund der Projektkomplexität oft nicht möglich, andererseits führe eine zu starke didaktische Reduktion zu einer ungewünschten Verzerrung der Realität. Husch verbindet seine Problematisierung mit der Frage, ob der klassische Software-Lebenszyklus die geeignete Projektmethode für den Unterricht sei, selbst wenn den analysierenden und entwerfenden Phasen stärkeres Gewicht gegenüber der Implementierung eingeräumt werde. Für die gründliche Planung eines komplexeren Systems sei der schulische Zeitrahmen zu eng. Deshalb empfiehlt er die Entwicklung von Prototypen im Unterricht und den schulischen Software-Projekten Musterarchitekturen zugrunde zu legen. Er stellt dazu ein Schichtenmodell für Software-Systeme vor und nennt als didaktische Vorteile für den Unterricht, dass einzelne Schichten sich gut vorproduzieren, wiederverwerten und einzeln realisieren ließen. Prototyping werde dadurch gut unterstützt. Husch schlägt (1993, 103) vor, in den analytischen Phasen „Demonstrationsprototypen“ zu erstellen, um die grundsätzliche Durchführbarkeit eines Projekts zu untersuchen, und später „horizontales Prototyping“ zu verwenden, bei dem einzelne Schichten des Systems weiter ausgestaltet werden. Von den schnell vorhandenen anschaulichen Zwischenergebnissen erwartet er eine hohe Motivation der Lernenden. Besonders geeignet für den Informatikunterricht ist nach seiner Auffassung die Schicht der Benutzungsschnittstelle. Aus heutiger Sicht ist diese Aussage stark in Frage zu stellen. Aufgrund der Anschaulichkeit motiviert die Entwicklung von Benutzungsschnittstellen die Lernenden sicherlich. Da dies heute aber halbautomatisch mit Entwicklungswerkzeugen für Benutzungsschnittstellen (GUI-Buildern) geschieht, besteht die Gefahr, dass der Schwerpunkt der Betrachtung zu sehr auf der Darstellung bei Vernachlässigung der Fachkonzepte liegt (vgl. hierzu die Arbeiten von Penon / Spolwig, S. 25). Als Mindestanforderung an schulische Software-Projekte fordert er, die Verfahren des objektorientierten Entwurfs (OOE) anzuwenden, „weil in ihm die Prinzipien der gemeinsamen Betrachtung von Algorithmen und Datenstrukturen sowie der Wiederverwendbarkeit von Software wiederzufinden sind“ (ebd., 104). Husch hält schulische Software-Projekte basierend auf der Entwicklung von Prototypen im objektorientierten Paradigma für möglich, weist allerdings explizit auf die Gefahr hin, dass aufgrund von komplexen Entwicklungswerkzeugen die Codierungsphase einen zu großen Stellenwert erhalten könne. Ferner warnt er vor dem Missverständnis, die Anwendung einer objektorientierten Klassenbibliothek mit einer objektorientierten Entwurfsstrategie gleichzusetzen, und vor Problemen mit unklar definierten Begriffen:

> „Die Schwierigkeit besteht zur Zeit immer noch in der nur rudimentär vorhandenen Einheitlichkeit der Begriffsbildung und -verwendung. Dies läßt sich an Definitionen der Begriffe Objekt und Klasse [...] ablesen.“ (ebd., 104f)

Husch lässt offen, mit welchen Techniken Analyse und Entwurf durchgeführt werden und wie die Basiskonzepte eingeführt und vertieft werden sollen. Da jedoch die Entwicklung von Pro-

totypen zur durchgängigen Methode erhoben wird, liegt die Vermutung nahe, dass in Analyse und Entwurf bereits viel Implementierungsarbeit von den Lernenden zu leisten ist, wenngleich auf einem abstrakteren Niveau.

Crutzen und ***Hein*** (1995) stellen vier mit objektorientiertem Denken zusammenhängende didaktische Linien vor, die sich bei der Entwicklung eines 100-Stunden-Fernkurses zur Einführung in die Informatik an der Open University der Niederlande und bei der Durchführung von Informatikprojekten auf Schülerakademien des „Bildung und Begabung e.V." bewährt haben. Diese sind im Einzelnen (Crutzen / Hein, 156f):

- Linie „Sehen – Verstehen – Ändern – Selbertun",
- Linie „Analyse – Entwurf – Implementation",
- Linie „Evaluation, Evaluation und nochmals Evaluation",
- Linie „simulierend – registrierend – regelnd – autonom".

Sie betonen die Wichtigkeit objektorientierten Denkens für den Informatikunterricht und fordern, es „von Anfang an und zyklisch durch alle Klassenstufen immer wieder" zu unterrichten (ebd., 149). Sie sehen verschiedene didaktische Vorteile. Am Anfang sei nur wenig Begriffliches aus der Informatik erforderlich. Das ermögliche informell zu beginnen und dann stufenweise zu formalisieren. Beispielsweise halten sie die Vererbung für Anfänger für entbehrlich, da das Erstellen einer Klassenhierarchie nicht zwangsweise objektorientiertes Denken nachweise (ebd., 152). Konstruktives, abstrahierendes und problemlösendes Denken werden gefördert und Analysieren, Beschreiben und Entwerfen sind zyklisch möglich:

> „Während der analytische Schritt nur rezeptiv ist und ein zweckentsprechend ausreichend genaues Eins-zu-Eins-Modell der Realität anstrebt, kommen beim Entwurf synthetische und konstruktive Überlegungen hinzu [...]." (ebd., 151)

> „Aus der objektorientierten Sicht wird nämlich durch Implementation die bisherige Realität um ein neues Objekt ‚Informationssystem' und neue Interaktionen zwischen ihm und anderen Objekten erweitert. Diese erweiterte Realität kann man nun wieder analysieren (z.B. um den bisherigen Entwurf des Informationssystems zu verbessern) – objektorientiertes Denken ist grundsätzlich zyklisch angelegt." (ebd., 152)

Crutzen und Hein kommen zu dem Ergebnis, dass objektorientiertes Denken besonders gut zu erlernen sei, wenn entsprechend den von ihnen vorgestellten Linien objektorientierte Modelle geändert, verfeinert, erweitert oder mit anderen Modellen kombiniert werden. Durch die Betonung objektorientierten Denkens erwarten sie eine Schwerpunktverlagerung des Informatikunterrichts von der Programmier- hin zu Modellierungs- und Evaluationskompetenz:

> „Das Ziel, ein Informationssystem ‚aus'nutzen zu können, tritt hinter die Fähigkeit zurück, ein modelliertes oder implementiertes Informationssystem auf seine Realitätstreue und Zweckmäßigkeit hin zu betrachten, sowie konstruktive Erweiterungs- und Verbesserungsvorschläge machen zu können. Die Schüler sollen erfahren, daß interaktives softwaretechnisches Operieren in komplexen Informationssystemen möglich ist; daß man dazu nicht das ganze Informationssystem, insbesondere nicht seine Implementation verstanden haben muß [...]." (ebd., 158)

Mit ihrem Beitrag, mit dem sie die Objektorientierung zur „didaktischen Basis der Informatik" (Crutzen / Hein 1995, 149) erhoben, regten Crutzen und Hein die fachdidaktische Diskussion an. Es gibt bis heute keinen Konsens zu einer ausschließlich objektorientierten Ausrichtung der Didaktik der Informatik, wenngleich verschiedene Autoren Objektorientierung als Erklärungsmodell für informatische Erscheinungen empfehlen (Knapp / Fischer 1998, Spolwig 1999, Hubwieser 2000b, Voß 2003). Viele informatische Modelle lassen sich zwar angemessen mit der objektorientierten Methode erstellen, aber neben anderen Modellierungsmethoden erfordern auch Wirkprinzipien von Informatiksystemen eigene, angemessene Formen der unterrichtlichen Behandlung (vgl. Breier et al. 2000b).

Schwill (1993b, 1995) analysiert die Eignung von Programmierstilen für den Informatikanfangsunterricht. Als wichtiges Motivationsproblem bei der imperativen Programmierung nennt er, dass die Lernenden „zunächst mit einer großen Menge von Konzepten vertraut wer-

den [müssen] (‚Lernen auf Vorrat'), bevor sie ein vernünftiges Programm mit Anwendungsbezug und ohne Wegwerf-Charakter entwickeln können" (Schwill 1995). Aufgrund von informatischen und pädagogischen Gründen solle die imperative Programmierung dennoch bis auf weiteres im Unterricht beibehalten werden (ebd., 180). Bei der prädikativen Programmierung sieht er Schwierigkeiten ausgelöst durch fehlendes Vorwissen der Lernenden zur Prädikatenlogik (ebd., 182). Zur funktionalen Programmierung referenziert Schwill Unterrichtsbeispiele zu LISP, Scheme und Logo, stellt aber in Frage, ob die positiven Erfahrungen auf moderne funktionale Sprachen, wie ML etc., übertragen werden können (ebd., 183). Für eine Einführung in die Informatik mit dem objektorientierten Ansatz sprechen nach Schwill (1995, 183) drei Gründe:

1. Das Paradigma erfüllt „unbestrittenermaßen informatorisch orientierte Forderungen nach einem zeitgemäßen Unterricht mit mächtigen Konzepten, wie Erweiterbarkeit, Anpaßbarkeit, Rekonfiguration, Vererbbarkeit, Kapselung, evolutionäre Softwareentwicklung sowie Wünsche nach einer stärkeren Anwendungsorientierung durch Betonung der Nutzung des Computers anstelle von einem vertieften Verständnis seiner Funktionsweise".
2. „[...] dieser Ansatz [ist] im Sinne des didaktischen Prinzips der Fortsetzbarkeit, ein zentrales Merkmal eines nach dem Spiralprinzip organisierten Curriculums, auf höherem Niveau beliebig ausbaufähig. Eine Umstellung von einer speziellen Anfangssprache auf eine ‚echte' Programmiersprache mit den damit verbundenen Reibungsverlusten, wie sie beim Einstieg mit Roboter NIKI oder LOGO notwendig werden, ist nicht erforderlich."
3. „[...] der objektorientierte Stil [ordnet sich] in besonderer Weise harmonisch den elementaren kognitiven Prozessen unter, die beim Denken, Erkennen und Problemlösen im menschlichen Gehirn ablaufen."

Den dritten Punkt hält Schwill aus pädagogischer Sicht für den wichtigsten Pluspunkt. Die objektorientierte Programmierung sei der geeignetste Ansatz, um Informatik im Anfangsunterricht zu vermitteln, weil sie fundamentale kognitive Prozesse widerspiegele und damit der natürlichen Denkweise am nahesten komme (ebd., 186). Schwill belegt dies anhand von Erkenntnissen zum Dunkerschen Kerzenproblem[28] aus der Kognitionspsychologie (Duncker 1966), in denen er eine objektorientierte Denkweise bei den Probanden identifiziert:

> „Gegenstände werden zuerst unter dem Gesichtspunkt in Klassen eingeteilt, welche Operationen mit ihnen möglich sind. Eine Schachtel gehört damit für die erste Gruppe zur Klasse der Objekte, auf denen Operationen wie ‚öffnen' und ‚schließen' erlaubt sind und die einen Zustand wie ‚leer' oder ‚gefüllt' besitzen. Diese Operationen bestimmen fortan das Denken. Dabei übersieht die Gruppe im Gegensatz zur zweiten, daß Schachteln auch als Objekte einer Oberklasse mit allgemeineren Eigenschaften aufgefaßt werden können, etwa als Objekte der Klasse quaderförmiger, flacher Gegenstände mit Operationen wie ‚als Unterlage verwenden', ‚stapeln' usw. [...] Funktionale Gebundenheit ist aus objektorientierter Sicht also die mangelnde Fähigkeit, zwischen unterschiedlichen Verfeinerungen einer Klassenhierarchie gedanklich hin und her zu wechseln." (Schwill 1995, 185f)

Offen blieb damals der empirische Nachweis dafür, dass sich ein objektorientierter Ansatz besonders eigne für eine Einführung in die Informatik. Die Verankerung von OOM im und Ergebnisse der externen Evaluation zum informationszentrierten Ansatz (vgl. S. 20) liefern einen nachträglichen Beleg für diese Position. Schwill warnt davor, als Schlussfolgerung sei-

[28] „Mehrere Versuchspersonen, die sich jeweils allein in einem Versuchsraum befanden, erhielten die Aufgabe, an einer Wand in Augenhöhe nebeneinander drei Kerzen zu befestigen und anzuzünden. Hierfür standen den Probanden eine Reihe von willkürlich auf dem Tisch verteilten Gegenständen zur Verfügung. Unter den meist nutzlosen Gegenständen befanden sich auch einige für die Lösung brauchbare: Heftzwecken, Streichhölzer und drei kleine in Farbe und Größe etwas unterschiedliche Pappschachteln von der Form einer Streichholzschachtel. Lösung der Aufgabe: Mit je einer Heftzwecke werden zunächst die Pappschachteln an der Wand befestigt; sie dienen den Kerzen als Standflächen. Anschließend werden die Kerzen angezündet und mit etwas Wachs auf den Schachteln festgeklebt. Die Versuchspersonen mußten diese Aufgabe in zwei leicht unterschiedlichen Ausgangssituationen lösen: Bei den Personen der ersten Gruppe waren die drei Pappschachteln mit Versuchsmaterialien gefüllt [...]. Bei der zweiten Gruppe waren die Schachteln leer [...]. Erstaunlicherweise wurde die Aufgabe von der zweiten Gruppe signifikant häufiger und schneller gelöst als von der ersten." (Schwill 1995, 184f)

ner Argumentation den Unterricht einfach mit einer OOP-Sprache zu beginnen. Der prognostizierte Lernerfolg hänge nicht nur von der Sprache, sondern auch von der verwendeten Programmierumgebung ab, die die objektorientierte Denkweise sichtbar machen müsse. Diese müsse eine künstliche Welt generieren, in der möglichst viele, klar visualisierte Objekte zur Verfügung stehen und eine Benutzungsschnittstelle bieten, über die Lernende „interaktiv und spielerisch explorativ Objekte auf ihre Funktionen analysieren und sie mit zunehmendem Niveau manipulieren, kombinieren, rekonfigurieren, evolutionär erweitern und schließlich neu entwickeln können“ (ebd., 186). Dies korrespondiert mit den „didaktischen Linien“ von Crutzen und Hein (1995)[29]. Als Beispiele, die seiner Vision nahe kommen, nennt Schwill das HyperCard- und das Smalltalk-80-System. Bemerkenswert ist, dass er von Informatikunterricht basierend auf objektorientierter Programmierung schreibt, in der Vision aber schon ein Werkzeug zum objektorientierten Modellieren skizziert.

Humbert (1999) zeigt Möglichkeiten zur Umsetzung von Grundkonzepten der Informatik im Unterricht auf. Er beschreibt eine Kurssequenz, in der mit Lernenden ein Informatiksystem zur automatischen Erzeugung von HTML-Seiten objektorientiert gestaltet wurde und fordert „die Ablösung der strukturierten Programmierung durch die objektorientierte Modellierung“ (Humbert 1999, 187). Humbert ist Mitautor eines Unterrichtskonzepts für objektorientierte Programmierung („Von Stiften und Mäusen“, s. S. 29). Er zeigt verschiedene Felder für Problemstellungen auf, die sich im Schulfach Informatik für eine objektorientierte Modellierung anbieten (Humbert 2001a). Als Beispiele benennt er das Teilgebiet Rechnernetze und verteilte Systeme, ereignisgetriebene Systeme, grafische Benutzungsoberflächen, Interaktion und Kommunikation sowie Simulation. Dabei seien die verpflichtenden Schwerpunkte „einfache Algorithmen und Datenstrukturen“ und der „Variablenbegriff“ explizit zu thematisieren (Humbert 2001a, 126). Humbert fordert, sich bei der Modellierung nicht nur auf den objektorientierten Ansatz zu beschränken (vgl. S. 14). Er sieht verschiedene Verknüpfungsmöglichkeiten mit anderen Ansätzen:

> „Aus dem Bereich der objektorientierten Modellierung bieten sich an mehreren Stellen Übergangsmöglichkeiten zu anderen Modellierungen an: z.B. kann mit der Anbindung von Datenbankschnittstellen auf umfangreiche Datenbestände zugegriffen werden; außerdem sind Erweiterungen verfügbar, die die Nutzung von Elementen der funktionalen Modellierung erlauben. Als fakultative Schwerpunkte im Zusammenhang mit der Objektorientierten Modellierung bieten sich die Bereiche Nebenläufigkeit, Dokumentenbeschreibungssprachen (Äquivalenz von Dokumenten- und Datenstruktur) geradezu an.“ (Humbert 2001a, 126)

Er befasste sich auch mit der Integration von Elementen grafischer Modellierungssprachen in den Unterricht:

> „Dazu wurden u.a. ausgewählte Unterrichtsstunden des Informatikunterrichts der Jahrgangsstufe 11 zur Objektorientierten Modellierung mit Hilfe der Videoanalyse dokumentiert und untersucht. Die Analyse zeigt, dass ausgewählte Elemente der graphischen Beschreibung zur Unterstützung der Orientierung im Lehr-/Lernprozess fruchtbar gemacht werden können. Die Abstraktionsleistungen werden im dokumentierten Unterrichtsprozess von den Schülerinnen erbracht. Es wird deutlich, dass der Modellierungsprozess neben der Abstraktionsleistung eine begriffliche Sicherheit erfordert. Die begriffliche Exaktheit konnte von den Schülerinnen nur mit Mühe erreicht werden. Bei Lernerfolgskontrollen zeigte sich, dass dennoch eine große Unsicherheit in der Definition von zentralen Begriffen, wie Klasse und Objekt, Attribut und Wert eines Attributs, allgemeines Fachkonzept und konkrete Umsetzung besteht.“ (ebd., 128f)[30]

Mit Elementen grafischer Modellierungssprachen konnte den Lernenden zwar Orientierung im analysierten Lehr-Lern-Prozess gegeben werden, die Sicherheit in der Fachsprache wurde aber nicht erreicht.

---

[29] vgl. S. 16

[30] Humbert verwendet in seinen Arbeiten das generische Femininum für geschlechtsbezogene Begriffe. Mit dem Begriff „Schülerinnen“ meint er Schülerinnen und Schüler.

Borg und ***Wiehenstroth*** (1999) beschreiben Objektorientierung als einen Schwerpunkt im Lerngebiet „Algorithmen, Datenstrukturen, Programmierung“ in der beruflichen Bildung. Nach Beobachtung der Autoren besitzen die Lernenden am Anfang keine Vorkenntnisse in der Software-Erstellung und ihr Abstraktionsvermögen ist wenig entwickelt. Aus Zeitgründen können sich die Lernenden nicht sowohl die Grundlagen der strukturierten als auch der objektorientierten Programmierung aneignen. Deshalb wird der objektorientierte Zugang bei der strukturierten Vorgehensweise durch eine Reihe formaler und inhaltlicher Regelungen vorbereitet, wie z.B. „weitestgehender Verzicht auf globale Variablen und Konstanten; Erstellung weitgehend unabhängiger Moduln (Prozeduren, Funktionen, Units) mit dem Ziel der Wiederverwendbarkeit; Analyse und Entwurf der Datenstrukturen als Ausgangspunkte der Softwareentwicklung“ (Borg / Wiehenstroth 1999, 208). Für das Erstellen von Methoden beim OOM sind Kenntnisse über Algorithmenentwurf und Kontrollstrukturen erforderlich. Offen bleibt, ob diese durch strukturierte Programmierung besser vorbereitet werden können als verzahnt mit dem objektorientierten Konzept.

Dekonstruktion von Informatiksystemen als Unterrichtsmethode

Das Konzept der Dekonstruktion von Informatiksystemen als Unterrichtsmethode von ***Hampel, Magenheim*** und ***Schulte*** (Hampel et al. 1999) richtet sich an Lernende in der Sekundarstufe II mit Vorkenntnissen zu objektorientierten Basiskonzepten. Es geht zurück auf Dekonstruktionskonzepte der Philosophie und Literaturwissenschaften, bei denen die Meinung des Autors aus Textanalysen erschlossen werden soll (Magenheim 2001). Motivation für das hier beschriebene Konzept ist die meist geringe Komplexität der Aufgaben und Beispiele zum Programmieren im Informatikunterricht, an denen sich Problemlösestrategien, Schulung analytischen Denkens, Abstraktionsvermögen, Zergliedern von Problemen in Teilprobleme etc. nicht angemessen einüben lassen und die dazu führen, dass der Informatikunterricht seine allgemein bildende und wissenschaftspropädeutische Funktion kaum erfüllen kann:

> „In der fachwissenschaftlichen Diskussion spielen Argumente wie Sicherheit und Stabilität von Software, leichte Wartbarkeit, Wiederverwendbarkeit und Qualitätssicherung eine gewichtige Rolle, wenn es um die Begründung von objektorientierter Softwareentwicklung geht. [...] Gerade bei komplexen Softwareprojekten entfalten objektorientierte Maximen wie Kapselung, Abstraktion, Vererbung oder Polymorphie ihre entscheidenden Vorteile gegenüber imperativen Konzepten. Eine derartige Komplexität bei der Softwareentwicklung, die das Verwenden objektorientierter Methoden erforderlich machte, ist im schulischen Unterricht kaum zu realisieren.“ (Hampel et al. 1999, 151)

Hampel et al. empfehlen eine systemorientierte Didaktik der Informatik, die „die Modellierung und Analyse oder besser die Dekonstruktion von Informatiksystemen“ zum Ausgangspunkt hat (ebd., 150). Verwendet wird für die Dekonstruktion ein speziell vorbereitetes, gut dokumentiertes Informatiksystem mittlerer Größe (hier: Schulkiosk-Software). Ziel ist es, dass die Lernenden das der Software implizite Modell soweit wie möglich explizieren, es mit dem realen System vergleichen und sich auf diese Weise objektorientierte Sichtweisen und Prinzipen sowie programmiertechnische Codierung in Java erschließen. Dazu erstellen sie zunächst ein Analysemodell der Software, indem sie Anwendungsfälle identifizieren und diese objektorientiert analysieren (Anwendungsfallanalyse nach Jacobson et al. 1992). Mit der CRC-Kartenmethode (CRC – class, responsibilities, collaborators) nach Beck und Cunningham (1989) erstellen sie dann kooperativ im Team eine Vorstufe eines Klassendiagramms. Anschließend überprüfen sie ihr Modell mittels eines Rollenspiels, indem sie feststellen, inwieweit das Modell und die zuvor identifizierten Anwendungsfälle zusammenpassen. In der Entwurfsphase werden das CRC-Kartenmodell und die Anwendungsfälle in ein Klassendiagramm und Interaktionsdiagramme transformiert. Die Dekonstruktionsphase der Software ersetzt die Implementierungsphase:

> „Anstelle der Neuimplementation der eigenen Modellierung dekonstruieren die Lernenden eine Implementation, die auf den Analyse-, Design- und Implementationsentscheidungen eines anderen ‚Entwicklungsteams' beruht." (Hampel et al. 160)

Die Lernenden analysieren den Quellcode mit dem Ziel, Entwurfsentscheidungen der Entwickler partiell nachzuvollziehen, Software also zu „interpretieren". In dieser Phase kommt es zu einem permanenten Wechsel zwischen Konstruktion (konstruktives produktorientiertes Programmieren im Kleinen) und Dekonstruktion (analytischer Zugang zum Programmieren im Großen). Hier sollen die Lernenden Erkenntnisse über die Programmierung mit Java sammeln, um später selbstständig „Elemente des ‚Programmieren[s] im Kleinen' [...] umzusetzen" (ebd., 160), Zusatzmodule für die Software zu entwickeln und anschließend ein anderes Projekt von der Modellierung bis zur Implementierung konstruktiv zu entwickeln. Den Bildungserfolg machen Hampel et al. abhängig von geeigneten Beispielsystemen und Entwicklungsumgebungen, die den Modellierungs- und Implementierungsprozess unterstützen. Nachteilig sind die bisher eingesetzten Entwicklungsumgebungen (CASE-Tools, computer aided software engineering), die für die professionelle Software-Entwicklung konzipiert wurden und für den schulischen Einsatz viel zu mächtig und damit nicht einsetzbar sind. Die unterrichtspraktische Erprobung und Evaluation des Konzeptes von Hampel et al. stand zum Vortragszeitpunkt noch aus. Inzwischen zeigte sich aber, dass die Umkehr des Entwicklungsprozesses vom Produkt hin zu Entwurfsentscheidungen der Entwicklerinnen und Entwickler in der Praxis nicht wie beabsichtigt gelingt. In neueren Forschungsarbeiten von Magenheim und Schulte wurde das Konzept erweitert und verfeinert, siehe hierzu Abschnitt 3.8. Beim Konzept der Dekonstruktion handelt es sich um ein anspruchsvolles und ambitioniertes Konzept, das für Lernanfänger der Informatik aber nur bedingt geeignet ist. Entwickelt wird ein Vorgehensmodell für das Erlernen von Software-Entwicklung und Systemgestaltung im Informatikunterricht. Offen bleibt, wie Beispiele, Übungen und einführende Systeme aussehen können, um Lernende an objektorientierte Basiskonzepte heranzuführen bzw. um erlernte Konzepte zu vertiefen.

OOM im informationszentrierten Ansatz

In seiner Habilitationsschrift fasste ***Hubwieser*** (2000b) seine Ergebnisse zum „informationszentrierten Ansatz" zusammen (vgl. auch S. 11). Er entwickelte ein geschlossenes Konzept zum informatischen Modellieren für den Informatikunterricht, das sich an den Phasen des Software-Entwicklungsprozesses orientiert, wobei der Schwerpunkt auf der Modellierungsphase und dem Erlernen und Anwenden von Modellierungstechniken liegt. In den vorgestellten Unterrichtsmodulen für die Jahrgangsstufen 6 bis 13 am Gymnasium spielen Aspekte der Objektorientierung in den Modulen „Fundamentum" und „Mittelstufe" eine wichtige Rolle. Für den Anfangsunterricht der Jgst. 6 (Modul Fundamentum) wird die Objektorientierung herangezogen als „Grundstein für den Aufbau angemessener mentaler Modelle und die Verwendung einer sauberen, ausdrucksstarken Terminologie in den späteren Jahren" (ebd., 59). Dies geschieht am Begriff Dokument und den damit verbundenen Operationen:

> „Dokumente und ihre Komponenten werden als Objekte mit charakteristischen Attributen und zugeordneten Methoden aufgefasst. Im Zentrum der unterrichtlichen Betrachtung steht vor allem die Analyse von typischen Dokumentklassen wie Texten, Tabellen, Grafiken." (Hubwieser 2000b, 60)

Spolwig (1995) berichtet von einem „unschätzbaren Nutzen" einer ähnlichen Vorgehensweise für einen nachfolgenden Einstieg in die objektorientierte Modularisierung in der Sek. II. Auch von Knapp / Fischer (1998) und dem Gesamtkonzept der informatischen Bildung der GI (Breier et al. 2000b) wird dieses Vorgehen für die Sek. I empfohlen:

> „Auf den ersten Erfahrungen aufbauend kann ein altersgemäßes Objektmodell helfen, eine Vielzahl von Phänomenen im Zusammenhang mit Informatiksystemen zu verstehen und zu systematisieren: Bei der Gestaltung von Grafiken und Texten lassen sich Objekte identifizieren, ihre Eigen-

> schaften benennen, Zusammenhänge aufspüren sowie mögliche Operationen analysieren.“ (Breier et al. 2000b, V)

Hubwieser betont, dass es in dem Modul „Fundamentum“ darum gehe, objektorientiert zu analysieren und zu modellieren, keinesfalls aber zu programmieren[31]. Eine externe Evaluation des Ansatzes (Hubwieser et al. 2001, Frey et al. 2001) belegte das Erreichen des Bildungsziels dieses Moduls:

> „Die Evaluationsfrage lautete: Bildet das Konzept der objektorientierten Modellierung im konkreten Vermittlungszusammenhang für die Lernenden eine brauchbare Metapher, die ihnen hilft, Neues mit Bekanntem so zu verbinden, dass eine fachgerechte Sicht auf Informatiksysteme im Unterricht erkennbar wird? Sie wurde mit folgender Hypothese beantwortet: Die provozierte Verbindung zwischen dem bereits Gelernten und Neuen, die durch die künstliche Brücke der objektorientierten Modellierung zusammengebracht werden soll, ist für die Lernenden nützlich, bleibt aber im Unterrichtsprozess für die Lernenden unsichtbar. Diese Hypothese wurde durch die externe Evaluation des Unterrichts verworfen. Die Fachsprache des objektorientierten Modellierens wurde konsequent angewendet. Sie förderte bei den Schülerinnen und Schülern die Einsicht in die wesentlichen Prinzipien der Informatik.“ (Hubwieser et al. 2001, 213f)

Der Erfolg spiegelt sich auch in den Ergebnissen einer Umfrage der Technischen Universität München unter 429 Lernenden der Klasse 6, die große Begeisterung für das Unterrichtsfach signalisierten. Lediglich fünf Lernende gaben an, dass ihnen das Fach nicht gefiel. 64.3% fanden Informatik leichter als Mathematik, 25.6% gleich schwierig und nur 8.6% fanden es schwerer (Frey et al. 2001, 22f).
In dem von Hubwieser vorgeschlagenen Modul der Mittelstufe (Jgst. 9 bis 11) ist das Ausbildungsziel „die Vermittlung einer *vertieften informatischen Allgemeinbildung* ‚zur Vorbereitung auf Studium und Beruf’ entsprechend den gesetzlichen Vorgaben“ (Hubwieser 2000b, 62, Hervorhebungen im Original). Neben der Repräsentation von Information, Datenmodellierung, zustandsorientierte Modellierung, funktionale Modellierung und Software-Projekten ist die objektorientierte Modellierung hier ein Schwerpunkt mit den Inhalten:

> „Objektmodell: Identifikation von Objekten, Klassenbildung
> Objekte als Automaten: Objekte vereinen Datenstrukturen (Attribute) und Verarbeitungsvorschriften (Methoden); Attributname und -wert;
> Objekt als Instanz seiner Klasse; Datenkapselung; Klassenhierarchien
> Kommunikation zwischen Objekten: Gegenseitiger Aufruf von Methoden durch Botschaften
> Ausblick auf nebenläufige Ausführung von Methoden“ (ebd., 63)

Thematisiert werden Basiskonzepte objektorientierter Modelle. Eine Implementierung der zuvor entworfenen Modelle mittels einer objektorientierten Programmiersprache ist vorgesehen. Hubwieser fordert hierfür spezielle Lernumgebungen, die aus Modellen lauffähige Quelltexte erzeugen können, um diese zu simulieren. In seinem Konzept wird OOM als eine neben mehreren Modellierungstechniken thematisiert, die Verkürzung auf eine Art der Modellierung (vgl. Abschnitt 2.2.3) somit vermieden. Die Arbeiten von Hubwieser (2000b) zeigen also, dass sich OOM für eine Einführung in die Basiskonzepte der Informatik für Anfängerinnen und Anfänger in der Sek. I eignet. Da zum Zeitpunkt der Niederschrift dieser Arbeit noch keine publizierten Evaluationsergebnisse seines Konzepts aus den Jgst. 9 bis 11 vorlagen, lassen sich damit zunächst noch keine Aussagen für die Sek. II verknüpfen. Klar ist, dass sich das Konzept nicht ohne weiteres auf die Situation in anderen Bundesländern übertragen lässt, in denen im Informatikunterricht der Sek. II noch auf kein verbindliches Fundament aus der Sek. I zurückgegriffen werden kann[32]. Dort müssen in der Sek. II einerseits noch informatische Grundlagen geschaffen werden, andererseits soll anspruchsvoller, altersgerechter Informatikunterricht stattfinden.

---

[31] Unterrichtsbeispiele zu diesem Ansatz finden sich unter der URL:
http://ddi.in.tum.de/unterricht/anfangsunterricht.html (aufgerufen am 12.11.03)

[32] In Bayern wird das Informatikfundament in der Jgst. 6 für alle Lernenden (ab 2004) verpflichtend (vgl. 2.2.1, S. 11, und Fußnote 20).

Im Hinblick auf die Sek. II wirft ***Schubert*** (2001a) die Frage auf, ob sich OOM für Anfängerinnen und Anfänger besser als die strukturierte Programmierung zur Einführung in die Basiskonzepte der Informatik eignet. Unter Bezug auf Appelrath et al. (1998) kommt sie zu dem Schluss:

> „Dagegen spricht vorerst, dass es Aufgabenklassen gibt, die für OOM schwer zugänglich sind und OOM nicht für das ‚Programmieren im Kleinen' empfohlen wird (Appelrath u.a. 1998). Widersprüchliche Veröffentlichungen verunsichern die Lehrenden. Das objektorientierte Problemlösen wurde als besonders intuitiv und einfach dargestellt. In Hospitationen und Unterrichtsvideoaufzeichnungen bestätigte sich das vorerst nicht. Das Variablenkonzept, die Datentypen und Datenstrukturen, die imperativen Strukturen (Zyklen und Alternativen) sind Basiskonzepte der objektorientierten Lösungen und müssen parallel zum Verständnis von Klassenhierarchien, Vererbung und dem Austausch von Botschaften erlernt werden." (Schubert 2001a, 18)

Schulte (2001) befasst sich mit der Frage, „welche Chancen für die Unterrichtskonzeption der Paradigmenwechsel in der Softwaretechnik [...], die Hinwendung zur Objektorientierung, mit sich bringt". Unter Bezug auf Möller (1999) kommt Schulte zu dem Schluss, dass Objektorientierung im Unterricht vermutlich vernetztes Denken trainiere, denn die interagierenden Objekte sollen ein benutzbares Software-System ergeben, so dass eine sinnvolle Vernetzung der Objekte hergestellt werden müsse:

> „Fokussiert man die formalen Bildungsziele auf das problemlösende Denken, dann kann folgende Hypothese aufgestellt werden: Man kann zwischen einfachen und komplexen Problemen sowie (damit verknüpft) zwischen linear-analytischem und vernetztem Denken unterscheiden. Objektorientierte Methoden fordern und fördern Letzteres stärker als algorithmenorientierte Herangehensweisen. Was ist vernetztes Denken? In einer Arbeitsdefinition kann vernetztes Denken ‚als ein situationsbezogenes Denken verstanden werden, bei dem die in einer spezifischen Denksituation relevanten Elemente zueinander in Relation gesetzt, integriert und zu einem Gesamtzusammenhang verflochten werden' (Möller, 1999 S.28). Gegenüber ‚linear-analytischem – auf logische-deduktive-Kausalketten zurückführbarem – Denken' berücksichtigt vernetztes Denken Wechselwirkungen einzelner Teilaspekte und Gesamtzusammenhänge (Möller, 1999, S.28)." (Schulte 2001)

Objektorientierung könne also „möglicherweise komplexere Denkweisen fordern und fördern". Dennoch sei es nicht möglich, „von *den* Auswirkungen oder *den* Möglichkeiten durch die Objektorientierung zu sprechen" (Hervorhebungen im Original). Es gebe jedoch Hinweise, „dass durch die Hinwendung zu offenen Fragestellungen (im Gegensatz zu algorithmisch lösbaren Problemen) sowohl die kognitiven Lernziele (Denkfähigkeiten), die informatischen Lernziele (Fragestellungen und Werkzeuge, Methoden der Softwaretechnik) und der Aspekt der Wechselwirkung zwischen Informatik und Gesellschaft profitieren können" (ebd.). Schulte lieferte somit weitere Argumente für die Thematisierung von objektorientierten Denk- und Arbeitsweisen im Informatikunterricht. Aktuellere Arbeiten von Schulte werden erörtert im Abschnitt 3.8.

Zwischenzusammenfassung 2.1

Da die Didaktik der Informatik ein noch junges Teilgebiet der Informatik mit einer bislang noch sehr überschaubaren Anzahl von Wissenschaftlerinnen und Wissenschaftlern ist, liegen bislang nur wenige Forschungsarbeiten zum Themenfeld objektorientiertes Modellieren im Informatikunterricht vor. Durch vorhandene Arbeiten wurden verschiedene *Argumente für eine Einbeziehung* dieses Themenfeldes in Lehr-Lern-Prozesse der Sek. I und II geliefert (vgl. Tabelle 3):

Argumente für eine Einbeziehung	**Quelle**
• am Anfang nur wenig Begriffliches aus der Informatik erforderlich • konstruktives, abstrahierendes und problemlösendes Denken werden gefördert • Analysieren, Beschreiben und Entwerfen sind zyklisch möglich • Schwerpunktverlagerung des Informatikunterrichts von der Programmierkompetenz hin zu Modellierungs- und Evaluationskompetenz	Crutzen / Hein 1995

Argumente für eine Einbeziehung	Quelle
• objektorientiertes Paradigma erfüllt informatorisch orientierte Forderungen nach einem zeitgemäßen Unterricht mit mächtigen Konzepten sowie Wünsche nach einer stärkeren Anwendungsorientierung durch Betonung der Nutzung des Computers anstelle von einem vertieften Verständnis seiner Funktionsweise • objektorientierter Ansatz ist im Sinne des didaktischen Prinzips der Fortsetzbarkeit (Spiralcurriculum) auf höherem Niveau beliebig ausbaufähig • objektorientierter Stil ordnet sich in besonderer Weise harmonisch den elementaren kognitiven Prozessen unter, die beim Denken, Erkennen und Problemlösen im menschlichen Gehirn ablaufen	Schwill 1995
• Objektorientierung als Grundstein für den Aufbau angemessener mentaler Modelle und die Verwendung einer ausdrucksstarken Fachsprache	Hubwieser 2000b
• Objektorientierung fordert und fördert möglicherweise komplexe Denkweisen (vernetztes Denken vs. linear-analytisches Denken)	Schulte 2001

Tabelle 3: Argumente für eine Einbeziehung der Objektorientierung in Lehr-Lern-Prozesse

Ebenso wurde eine Reihe von *Problemen und Schwierigkeiten* thematisiert (vgl. Tabelle 4).

Probleme und Schwierigkeiten einer Einbeziehung	Quelle
• Gefahr der Vernachlässigung der Fachkonzepte bei Betonung der Benutzungsschnittstelle • Codierung könnte durch Verwendung professioneller Entwicklungsumgebungen zu großen Stellenwert erhalten • fachliche Missverständnisse (z.B. widersprüchliche Begriffsverwendungen, Gleichsetzen der Anwendung einer objektorientierten Klassenbibliothek mit einer objektorientierten Entwurfsstrategie)	Husch 1993
• empirischer Nachweis, dass sich ein objektorientierter Ansatz besonders für eine Einführung in die Informatik eignet, ist offen • Programmierumgebung fehlt, die objektorientierte Denkweise sichtbar macht	Schwill 1995
• Begründung objektorientierter Software-Entwicklung über Komplexität ist im Informatikunterricht kaum realisierbar • professionelle Software-Entwicklungsumgebungen sind für den schulischen Einsatz viel zu mächtig und damit nicht einsetzbar	Hampel et al. 1999
• Variablenkonzept, Datentypen, Datenstrukturen, Kontrollstrukturen als Basiskonzepte der objektorientierten Lösungen müssen parallel erlernt werden	Borg / Wiehenstroth 1999, Humbert 2001a, Schubert 2001a

Tabelle 4: Probleme und Schwierigkeiten einer Einbeziehung der Objektorientierung in Lehr-Lern-Prozesse

Unterrichtskonzeptionen wurden entwickelt für die Sek. I basierend auf objektorientierter Analyse dokumentenorientierter Standardanwendungen (Hubweiser 2000b) und für die Sek. II eine Projektmethode, basierend auf objektorientiertem Entwurf und Prototyping (Husch 1993), sowie ein Vorgehensmodell schulischer Systemgestaltung (Hampel et al. 1999). Ein Ziel der Arbeiten von Husch und Hampel et al. ist es dabei, die Thematisierung komplexerer Systemstrukturen im Unterricht zu ermöglichen, um die Vorteile des objektorientierten Ansatzes besser ausnutzen zu können. Borg und Wiehenstroth (1999) beschrieben, wie sie in der beruflichen Bildung die Vermittlung von imperativen und objektorientierten Konzepten vereinbaren. Für Lehr-Lern-Prozesse wurden verschiedene *Strukturierungsmöglichkeiten* vorgeschlagen. Crutzen und Hein empfehlen hierzu vier didaktische Linien (vgl. S. 16), entsprechend denen objektorientierte Modelle geändert, verfeinert, erweitert oder mit anderen Modellen kombiniert werden sollen. Schwill stellte eine hiermit korrespondierende Forderung für *unterrichtsgeeignete Software-Werkzeuge* auf, deren Fehlen mehrere Autoren beklagen (Husch 1993, Schwill 1995, Hampel et al. 1999). Von *Problemen mit dem korrekten Einsatz der Fachsprache* berichten Husch (1993) und Humbert (2001b). Die Ergebnisse von Hubwieser (2000b) zeigen hier, dass sich mit dem von ihm verfolgten Ansatz eine solide Fachsprache

bei den Lernenden entwickeln lässt. Offen bleibt, wie die objektorientierten Basiskonzepte vermittelt und vertieft werden.

2.3.1.2 Unterrichtserfahrungen und -beispiele

Mit dem Ziel, die nationale Diskussion in der fachdidaktischen Forschung zum objektorientierten Modellieren (vgl. Abschnitt 2.3.1.1) mit der tatsächlichen Situation im Informatikunterricht in Relation setzen zu können, werden hier publizierte Unterrichtserfahrungen, -beispiele und -empfehlungen aus der Schulpraxis vorgestellt.
Modrow (1992, 310ff)[33] äußert sich in seinem frühen Lehrbuch „Zur Didaktik des Informatik-Unterrichts" noch skeptisch bzgl. der zukünftigen Bedeutung von Objektorientierung für den Informatikunterricht, obwohl er der Auffassung ist, dass „die Verwendung von Objekten [...] einen erstrebenswerten Arbeitsstil [bewirkt]" (Modrow 1992, 311):

> „Der systematische Entwurf von Objektklassen mit den daraus abgeleiteten virtuellen Methoden macht die Situation nun nicht leichter, sondern scheint mir die ‚Kopfarbeit' eher zu verstärken. Es kann damit nicht die Aufgabe des Informatikunterrichts sein, (im Normalfall) vollständig neue OOP-Modelle zu entwickeln. [...] Die Brauchbarkeit des Verfahrens kann sich also eigentlich erst dann erweisen, wenn in einem entwickelten OOP-System mit einer Vielzahl von bereitgestellten Objektklassen eher experimentierend als konzipierend von den Schülerinnen und Schülern gearbeitet werden kann." (ebd., 312f)

Lehrenden, Lernenden und Didaktikern fehle es an praktischer Unterrichtserfahrung mit OOP-Systemen, um deren Unterrichtseignung beurteilen zu können. Die Kenntnisse und Fähigkeiten zu den bisherigen Programmentwicklungssystemen seien „so weit gestreut [...], dass es sinnvoller erscheint, die schon erworbenen Kenntnisse ‚abzurunden', statt in einem völlig anderen System auf [...] Anfängerniveau wieder neu zu beginnen" (ebd., 313). Ein zu schneller Wechsel der Lernumgebung erzwinge, sich permanent mit technischen, statt mit pädagogischen Fragen zu beschäftigen. Trotz Unzulänglichkeiten seien die um „OOP-Fähigkeiten erweiterten prozeduralen Sprachen [...] vorläufig [...] das Werkzeug der Wahl" (ebd., 313). Er empfiehlt eine behutsame Einführung in objektorientierte Spracherweiterungen als „syntaktische Eigenheiten" (ebd., 313), und diese dann später thematisch wieder aufzunehmen.
Ähnliches findet sich im Lehrbuch zur „Didaktik der Informatik" in der Auflage von 1996 von ***Baumann***[34], in dem objektorientiertes Modellieren und Entwerfen ebenfalls nur kurz vorgestellt werden. Unter Bezug auf u.a. Crutzen und Hein (1995)[35] fasst Baumann Argumente für Objektorientierung im Informatikunterricht zusammen, wie z.B. die erwartete Verlagerung von der Programmier- hin zu einer Modellierungs- und Evaluationskompetenz, um dann zu dem Schluss zu kommen:

> „Das klingt überzeugend; die gegenwärtige Situation ist allerdings dadurch gekennzeichnet, daß kaum jemand die Entschlußkraft aufbringt, von der vertrauten Sprache Pascal abzugehen." (Baumann 1996, 281)

Diese „Verharrungstendenz" auf langjährig bewährten Unterrichtskonzepten ist bis heute zu beobachten. Nach Baumann scheint es sich beim Wechsel vom imperativen zum objektorien-

[33] Modrow veröffentlichte 2003 seine Dissertationsschrift (vgl. Modrow 2003). Die darin vertretene Position zur Objektorientierung wird im Abschnitt 3.8 erörtert.
[34] Baumann gehörte 1990 zu den frühen Autoren zum objektorientierten Programmieren (Baumann 1990, Baumann et al. 1990) in der Zeitschrift LOG IN. Die ersten Artikel zum OOP stellten die neuen Konzepte zunächst vor, befassten sich aber noch nicht mit einem Transfer in den Informatikunterricht. Das gilt auch für einen frühen LOG IN-Artikel mit starken Bezügen zum objektorientierten Modellieren mit grafischen Modellierungstechniken von Eirund (1993), in dem Klassenstrukturdarstellungen sowie Visualisierungen von Objektkommunikation zu finden sind.
[35] vgl. S. 16

tierten Programmieren zunächst nur um eine Änderung der Terminologie zu handeln (ebd., 281f, Tabelle 5):

Imperativische Programmierung	Objektorientierte Programmierung
Datentyp	Objekttyp, Klasse
Variable	Objekt, Exemplar (engl.: instance) der Klasse
Wert der Variablen	Zustand des Objekts
Variablenvereinbarung	Objektvereinbarung
Belegung mit einem Anfangswert	Erzeugung eines Objekts
Prozedur, Funktion	Methode
Prozeduraufruf, Funktionsaufruf	Senden einer Nachricht

Tabelle 5: Imperativische vs. objektorientierte Programmierung nach Baumann (1996, 282)

Das Spezifikum der Objektorientierung beginne mit dem Vererbungskonzept. Dem ist energisch zu widersprechen. Sicherlich korrespondieren die Zeileneinträge in Tabelle 4 miteinander und das Wissen um diese Analogie mag für viele Lehr-Lern-Situationen von Vorteil sein. Dennoch handelt es sich um zwei grundverschiedene Sichtweisen, die sich nicht nur in ihrer Terminologie unterscheiden. Als Konsequenz solcher nur terminologischen Veränderungen kam es zu Unterricht, der moderne objektorientierte Techniken zum Gegenstand haben sollte, in dem tatsächlich aber imperative Programme mit objektbasierten oder objektorientierten Sprachen geschrieben wurden. Dieses Problem trat auch in der Hochschulausbildung auf (vgl. Mitchell 2000 auf S. 35).

Penon und ***Spolwig*** publizierten eine Reihe von Unterrichtserfahrungen mit OOP und OOM (Spolwig 1995, 1999, 2000; Penon / Spolwig 1998). Spolwig (1995) stellt Schwierigkeiten der Lernenden beim Transfer von Erkenntnissen zu „Miniprogrammen“ auf den Entwurf größerer Programme ab 500 Zeilen fest (für den Autor dieser Arbeit steht die Behandlung umfangreicher Programme in der Sek. II nicht im Mittelpunkt). Ferner scheitern diese regelmäßig am Parameter- und Modulkonzept und schaffen den Übergang vom ablauforientierten hin zu einem objektbasierten[36] Programmentwurf mit abstraktem Datentyp nicht. Als Lösung empfiehlt er, eine Strukturierung von Programmen nach dem EVA-Prinzip (Eingabe – Verarbeitung – Ausgabe) zu vermeiden, und im Unterricht von Anfang an objektbasiert zu beginnen. Hierbei rät auch Spolwig zur Verwendung vertrauter Sprachen für den Unterricht mit objektorientierten Erweiterungen, jedoch bei starker Betonung von OOA und OOD und deren Trennung von der Implementierungsphase. Hierin liegt der Unterschied zu Modrow (1992) und Baumann (1996). Im skizzierten Unterrichtsprojekt „Fahrscheinautomat“ wird Programmieren als Verbindung von Modellbildung und Implementierung verstanden. Komplexe Zusammenhänge werden von den Lernenden erkannt, Datenstrukturierung (Objektstruktur) als Mittel der Komplexitätsreduzierung kennen gelernt. Als sehr vorteilhaft stuft Spolwig die vorgelagerte Behandlung von Textverarbeitung mit Textobjekten und den darauf möglichen Operationen ein, um einen intuitiven Objektbegriff vorzubereiten.

Die Beiträge von Penon und Spolwig ab 1998 (Penon / Spolwig 1998; Spolwig 1999, 2000) haben verschiedene Unterrichtserfahrungen mit Java und Object Pascal zum Gegenstand. Programmierumgebungen für diese Sprachen (wie z.B. das Delphi-System) fanden im Informatikunterricht große Verbreitung, da mit den in ihnen integrierten GUI-Buildern professionell aussehende Benutzungsoberflächen erstellbar sind (vgl. Fußnote 42 auf S. 27). Nach Penon und Spolwig führt die Anwendung dieser Werkzeuge oft zu einer „völlig willkürliche[n] und kaum nachvollziehbare[n] Melange von Zugriffen auf GUI-Komponenten [...] und Datenobjekte des Fachkonzepts innerhalb aller möglichen Programmblöcke“ (Penon / Spolwig 1998, 40). Verursacht werde dies durch eine unzulässige Verkürzung des Software Life Cycle auf folgende Vorgehensweise:

[36] Spolwig verwendet diesen Begriff, „weil hier der Akzent auf der Entwurfstechnik (OOA und OOD) und nicht so sehr auf dem technischen OOP Konzept liegt“ (Spolwig 1995, 43).

1. „Oberfläche mit dem GUI-Builder gestalten,
2. Ereignisprozeduren auf die Komponenten legen,
3. Überlegen, was in den Prozeduren geschehen soll." (Penon / Spolwig 1998, 41)

Weil die Modellierung des Fachkonzepts völlig außer Acht gelassen werde, zeige der Quelltext das oben beschriebene Erscheinungsbild, dem nur vorzubeugen ist, wenn Fachkonzept und Darstellung strikt getrennt modelliert werden (Model-View-Controller-Konzept – MVC). Zur Vermeidung sollen in den ereignisbehandelnden Methoden der GUI nur Methoden des Fachkonzepts aufgerufen werden, nicht aber Verarbeitungen stattfinden (Spolwig 2000, 53). Penon und Spolwig beschrieben damit ein Kernproblem des Informatikunterrichts. Durch die zu starke Ausrichtung des Unterrichts auf die Erfordernisse der jeweiligen Implementierungssprache und -umgebung werden Entwurfskonzepte unzureichend erlernt.
Spolwig (1999) stellt die Eignung von Sprachen, wie Java und Object Pascal, für den Unterricht in Frage:

> „Wenn man also DELPHI u.a. für OOP einsetzen will, wird man als Lehrer einiges für einen erfolgreichen Unterricht für die Schüler [...] umbrechen müssen. Dieses Problem ist meines Erachtens bisher nicht gesehen oder als nebensächlich erachtet worden. Es ist aber das Schlüsselproblem für den unterrichtlichen Erfolg von OOP. Andersherum ausgedrückt, wenn es nicht gelingt, die wenigen aussagekräftigen Begriffe Klasse, Objekt, Attribut, Methode, Botschaft genauso einfach zu notieren, dann gehört OOP nicht in den Unterricht, zumindest aber nicht eine solche Sprache." (Spolwig 1999, 41)

Um mit diesen Sprachen erfolgreich Unterricht zu gestalten, müssen Lehrende also geeignete didaktische Reduktionen vornehmen. Penon und Spolwig betonen die Konsequenzen für die Unterrichtsplanung:

> „Wer sich noch nie mit OOP beschäftigt hat, sollte zuerst in der ihm vertrauten Programmiersprache – soweit diese es zulässt – objektbasierte modulare Programme schreiben [...] und nach entsprechender Erfahrung auf andere Sprachen umsteigen." (Penon / Spolwig 1998, 45)[37]

> „Bei konsequentem objektorientierten Vorgehen führen Ereignisorientierung und grafische Oberfläche zu einer erheblichen Zunahme der Komplexität. Die gewonnene Zeit, die bei einer schnelleren Entwicklung der (eleganten) Oberflächen herausgearbeitet wird, wird jedoch wieder beim Realisieren und Verknüpfen der komplexen Klassenstrukturen mehr als verbraucht." (Penon / Spolwig 1998, 46)

Spolwig weist auch auf die Auswirkungen hin, die sich durch eine Verlagerung des Unterrichtsschwerpunkts vom Programmieren zum Modellieren ergeben:

> „Bei den letzten halbjährigen Softwareprojekten ergab sich, dass die OOM-Phasen (OOA, OOD einschließlich der vollständigen Spezifikation) bis zu 3/4 der verfügbaren Zeit in Anspruch nahmen, sodass in der Kodierungsphase nur noch ein Prototyp mit einem GUI-Builder erstellt werden konnte, um die Benutzungsschnittstelle zu realisieren. Praktisch bedeutete es, dass die Schüler ca. 12-15 Wochen lang kaum eine Zeile kodiert haben und es ist darüber kein Unmut ausgebrochen, wie man vielleicht befürchten würde. [...] Der Verzicht auf die Programmierphase fiel den Schülern nicht schwer, denn es war allen klar geworden, dass die eigentliche Aufgabe, ein Fachkonzept zu erarbeiten, gelöst war und was jetzt kommen würde, nur noch eine Fleißarbeit ist. Es ist eindeutig zu beobachten, dass die Verlagerung des Unterrichtsschwerpunktes auf OOM mit einem dramatischen Verlust an Programmierkompetenz (Kodierung und Kenntnisse der Programmiersprache) einhergeht, während die Analyse- und Modellierungskompetenzen deutlich besser ausgeprägt sind. [...] Das ist eine Situation, der man sich stellen muss [...] weil letztlich nur mit hinreichenden Programmierkenntnissen überprüft werden kann, ob das Modell vernünftig konzipiert war. Ich bin skeptisch, ob Versuche, die Programmierung aus dieser Funktion herauszunehmen und durch andere Verifizierungsinstrumente zu ersetzen, gelingen werden." (Spolwig 2000, 54)

Die beschriebene Kompetenzverlagerung wurde von Crutzen und Hein (1995)[38] bereits vermutet. Im Sinne aktueller Bildungsempfehlungen für Informatikunterricht (vgl. Breier et al. 2000b) ist sie durchaus gewünscht.

---

[37] Diese Vorgehensweise wurde auch von Modrow (1992) und Baumann (1996) empfohlen, vgl. S. 24.
[38] vgl. S. 16

Im Hinblick auf Lehrerfortbildungen kritisieren Penon und Spolwig, dass anstatt neuer objektorientierter Konzepte oft neue Programmiersprachen vermittelt würden:

> „Nach unseren Erfahrungen im Unterricht und in der Lehrerfortbildung resultieren *die Schwierigkeiten beim Einsatz dieser Sprachen in erster Linie aus der Datenmodellierung durch Klassenbildung und der Ablaufsteuerung durch Ereignisse, die vielen ungewohnt ist.* Der Fortbildungsbedarf liegt unseres Erachtens daher bei diesen Schwerpunkten und nicht – wie häufig angeboten und nachgefragt – bei reinen Programmierkursen in JAVA oder DELPHI“ (Penon / Spolwig 1998, 46, Hervorhebungen im Original).

Es zeigt sich der große Bedarf, auch Lehrende an objektorientierte *Konzepte* heranzuführen[39]. Aufgrund von Schwierigkeiten mit den Kategorien der „neuen Informatikdidaktik“ (Komplexität, Abstraktion, Modellbildung, Modularisierung und Klassenbildung) seien hierzu nur wenige Unterrichtsbeispiele publiziert[40] (Spolwig 2000, 53). Lehrende müssen besser als bisher bei der Gestaltung von Unterricht zu diesen Inhalten unterstützt werden.

Hermes plädiert in seinem zweiteiligen Beitrag (1996a, 1996b) für einen gleitenden Übergang von strukturierter Programmierung zur OOP. Hermes beklagt, dass es „bis auf wenige Ausnahmen an überzeugenden Unterrichtsbeispielen“ fehle. Vorhanden seien aber zahlreiche Publikationen über die OOP-Techniken. Ziel der von ihm vorgeschlagenen Unterrichtsreihe ist eine Einführung in die OOP und in abstrakte Datentypen. Er erprobte den Umstieg von Pascal bzw. Modula auf Oberon in zwei Lerngruppen mit geringen Programmierkenntnissen. Objektorientierte Begriffe und Prinzipien wurden eingeführt anhand der Animation (Darstellen – Löschen – Versetzen – Darstellen) zusammengesetzter geometrischer Objekte (z.B. stilisierte Lokomotive). Diese Objekte sind aus Vierecken, Kreisen und Linien zusammengesetzt, die in einer Liste verwaltet werden. Hermes berichtet von positiven Erfahrungen mit diesem Ansatz, der von der Anwendungsdomäne her stark an das Konzept „Von Stiften und Mäusen“ erinnert (vgl. S. 29)[41]. Die Einführung von Konzepten der Objektorientierung bis hin zur Polymorphie gelang.

Füller (1999) berichtet von ambivalenten Erfahrungen mit objektorientierter Programmierung in Java in der Schulpraxis, die er im Informatikunterricht der Jgst. 11 und im Leistungskurs Informatik an einer gymnasialen Oberstufe sammelte. Er kommt zu folgendem Resümee (Füller 1999, 190, Hervorhebung im Original):

- „OOP erlaubt es, Programme auf ‚natürliche’ Weise zu strukturieren. Die Steuerung der Programme mittels *Ereignissen* befreit vom Denken in Abläufen und erlaubt Programme mit komplexem Verhalten.
- Es wird Schülerinnen und Schülern ermöglicht, Programme mit modernen Benutzeroberflächen zu schreiben, und so den Anschein von Professionalität zu erzeugen (Produktorientierung).
- Ein objektorientierter Ansatz kann verwendet werden, um Anwendersysteme zu analysieren und neutral zu vergleichen.
- Es stellt sich jedoch heraus, dass eine gründliche Objektorientierung sehr viel, vielleicht zu viel systematische Analyse der Aufgabenstellung verlangt.
- Die üblichen objektorientierten Sprachen verwenden eine überfrachtete Syntax.
- Eine überzeugende objektorientierte [Entwicklungs-]Umgebung für die Schule liegt nicht vor.“[42]

---

[39] Aus diesem Grund wurden die im Rahmen dieser Arbeit erarbeiteten Konzepte in die Informatiklehreraus- und -weiterbildung einbezogen (vgl. 3.7, 4.6, 5.5.3).

[40] Penon / Spolwig veröffentlichen ihre Unterrichtsmaterialien im Internet unter der Adresse: http://www.schule.de/schulen/oszhdl/gymnasium/faecher/informatik/material (aufgerufen am 12.11.03)

[41] Weitere Unterrichtsbeispiele zum OOP wurden publiziert z.B. für folgende Anwendungsdomänen:

- Entwicklung eines Modellrechners (Rollke 1994),
- Unterstützung und Simulation von Messungen im Physikunterricht (Modrow 1995),
- Ampelsteuerung (Rollke 1995, mit Bezug zum Konzept „Von Stiften und Mäusen“, vgl. S. 29).

[42] Der erste Punkt stützt die Aussagen von Schwill (1995), vgl. S. 16. Der Anschein von Professionalität durch moderne Benutzungsoberflächen (zweiter Punkt) ist kritisch zu bewerten. Mittels Werkzeugen zur Gestaltung

Füller beschreibt ein Unterrichtsprojekt (Memory-Spiel) und stellt Probleme bei der Umsetzung dar. In der Jahrgangsstufe 11 wird Programmieren (Anweisungen, Variablen, Ablaufsteuerung, Modularisierung, Unterprogramme) „gelehrt als Werkzeug für Modellbildung, als Voraussetzung für einen produktorientierten Unterricht (im Sinne eines konstruktivistischen Lernens) [...]. Es geht uns also nicht darum, Schülerinnen und Schüler zu professionellen Programmierern auszubilden." (Füller 1999, 190f). Mit diesem Vorgehen sieht er u.a. folgende Probleme verbunden (ebd., 192, Hervorhebung im Original):

- „Es besteht die Gefahr, dass im Unterricht langweilig eine Sprachstruktur nach der anderen behandelt wird. Dagegen setzen wir ein arbeitsteiliges Vorgehen [...]. Von Zeit zu Zeit können die Schülerinnen und Schüler nach der *Expertenmethode* ihre Kenntnisse austauschen.
- Schüler, die bereits programmieren können, werden unterfordert. Wir haben versucht, diesen Schülern eine ihnen bisher unbekannte (funktionale) Programmiersprache anzubieten oder eine Programmentwurfsmethode (objektorientiert), die sie in stärkerem Maße fordert. Beide Ansätze nivellieren die Unterschiede in den Vorkenntnissen und tragen so zur Notengerechtigkeit bei."[43]

Bei seinem Entwurf des Projektes Memory-Spiel stellte er fest, dass für den Spielkern nur wenige „*objektorientierte* Eigentümlichkeiten verwendet [werden]. [...] Es findet eine erhebliche Abstraktion statt, die nicht gerade der Motivation der Schüler dient. Das ist jedenfalls kein ‚Bastelansatz', den man spontan am Rechner erprobt und verfeinert" (ebd., 198, Hervorhebung im Original). Bei der technischen Umsetzung müsse man sich stark mit dem Mehrfachvererbungsmechanismus von Java auseinandersetzen, den man nur verstehen könne, wenn man die Probleme der C++-Mehrfachvererbung kenne. Dies sei ein für die Schule unlösbares Problem. Leider legt Füller nicht offen, wie genau der Spielkern modelliert wurde, so dass hier nicht tiefer gehend analysiert werden kann, wodurch die Probleme verursacht wurden. Das Problem der Thematisierung von Sprachdetails, wie z.B. Javas *implements*, wäre durch vorbereitete Klassenrahmen vermeidbar gewesen. Praktische Erfolge ließen sich laut Füller bei der Realisierung als Applet erzielen. Mit dem Ziel, unterschiedliche Benutzungsschnittstellen zu erzeugen, sollten Gemeinsamkeiten aller GUIs in eine abstrakte Basisklasse ausgelagert werden. Allerdings war es schwierig, das Konzept der abstrakten Klassen zu motivieren, da „zu ihrer Implementierung [...] einige Stunden Überlegung und praktischer Arbeit am Computer notwendig [sind], an deren Ende das Programm die gleiche Funktionalität hat wie vorher, nur ist es jetzt intern besser organisiert. Im betreffenden Kurs hat keiner der Schüler diese Abstraktionsleistung erbracht. Spontan wurden die Quellcodes einfach kopiert und verändert." (ebd., 199). Programme mit einer intern besseren Organisation sind in der Informatik durchaus erstrebenswert. Es ist gut zu verstehen, dass Erweiterungen der Funktionalität für Lernende attraktiver sind als Strukturverbesserungen. Geeignete Unterrichtsbeispiele müssen so gewählt sein, dass Lernende durch eine bessere Strukturierung von Klassenhierarchien in ihrer weiteren Arbeit einen Vorteil verspüren[44]. Nach Füller verlangt „die Objektorientierung analytisch-planendes, an Strukturen orientiertes Vorgehen" (ebd., 200), worin er Vor- und Nachteil (Vermittlung systematischer Arbeitsweisen vs. Überforderung der Schüler) zugleich sieht. Ein spielerisch-experimentierender Ansatz gehe verloren und die objektorientierte Strukturierung eines Problems zahle sich möglicherweise erst so spät aus, dass der Gewinn im Unterricht nicht mehr eingefahren werden könne. Füller wirft folgende Leitfragen für die Bildungsplanung auf (ebd., 200):

- „Welchen Beitrag kann die Informatik zum konstruktivistischen, selbsttätigen Lernen leisten?

---

von Benutzungsschnittstellen können diese heute durchaus sehr unsystematisch (quick and dirty) erstellt werden. Durch zu starke Betonung der Darstellung können Fachkonzepte vernachlässigt werden. Diese Problematik findet sich auch bei Husch (1993), vgl. S. 15.

[43] Der erste Punkt (Lernen auf Vorrat) wird auch von Schwill (1995) thematisiert, vgl. S. 16.

[44] Analog führte beim imperativen Problemlösen die beste Datenstruktur zur Komplexitätsreduzierung des Algorithmus (z.B. 8-Damen-Problem).

- Welchen Stellenwert hat die induktiv-experimentierende, spielerische Arbeitsweise?
- Welche Rolle spielt analysierendes theoretisch-deduzierendes Vorgehen?
- Welche Werkzeuge werden für verschiedene Arbeitsweisen gebraucht und wann sollen diese eingesetzt werden?"

Füllers Beitrag zeigt die Probleme, die auftreten, wenn objektorientiertes Programmieren (OOP) Unterrichtsschwerpunkt ist. Diese Probleme lassen sich nicht mit der Wahl geeigneter Beispiele lösen, da sie prinzipieller Natur sind. Daraus folgt der große Bedarf an Lehr-Lern-Konzepten und geeigneten Beispielen für objektorientiertes Modellieren im Informatikunterricht.

Das Konzept „Von Stiften und Mäusen"

Mit dem Konzept „Von Stiften und Mäusen" in der Version von 1999 (LSW 1999) von ***Czischke, Dick, Hildebrecht, Humbert, Ueding*** und ***Wallos*** lag das erste geschlossene Konzept für eine Unterrichtsreihe zur Einführung in die Grundlagen objektorientierter Programmierung vor. Bereits vor der Veröffentlichung des Gesamtkonzepts wurden an verschiedenen Schulen Erfahrungen mit Komponenten (Czischke 1995a, 1995b, 1996, 1997, 1998a, 1998b, 2000) gesammelt[45]. Der vorgeschlagene Unterricht basiert auf einer unter didaktischen Gesichtspunkten zusammengestellten, zunächst sehr einfach gehaltenen, Klassenbibliothek, deren Basisklassen wesentliche Objekte der Mensch-Maschine-Interaktion heutiger Informatiksysteme modellieren (Bildschirm, Stift, Maus, Tastatur) und die im weiteren Unterrichtsverlauf schrittweise ausgebaut wird. Angestrebt wird, die Einführung in das imperative Programmieren zu ersetzen, Kontrollstrukturen und Variablenkonzept in einen ansonsten durchgängig objektorientierten Ansatz zu integrieren. Die Klassenbibliothek wurde für verschiedene Programmiersprachen und Zielplattformen implementiert. Sie dient insbesondere zur Bereitstellung einer einfachen Programmierschnittstelle, hinter der die vielen Syntaxdetails von professionellen Sprachen vor den Lernenden verborgen bleiben, und um mit einer Bibliothek überschaubarer Größe (im Vergleich zu professionellen Programmierumgebungen) Prinzipien von deren Konstruktion, Wiederverwendung und Weiterentwicklung vermitteln zu können. Das ist überzeugend gelungen. Grundkonzepte der Objektorientierung, wie Objekt, Klasse, Methode, und die Verwendung der Klassenbibliothek werden mit schrittweise komplexer werdenden Beispielen aus dem Bereich der Erzeugung einfacher Liniengrafiken eingeführt. In diesem Beispielrahmen liegt ein Schwachpunkt des Konzepts, da lebensweltliche Gegenstände (Tisch, Auto, Zug) oft stark vereinfacht durch Linien, Rechtecke und Kreise dargestellt werden. Weiterhin sind die in den Quelltexten verwendeten Methodenbezeichnungen (z.B. ZeigeDich, LöscheDich) nicht immer altersgerecht für die Sek. II. Hospitationen zeigten, dass sich auf der Basis des Konzepts dennoch anspruchsvolle Unterrichtsgespräche entwickeln lassen (vgl. z.B. Anhang A). Der Modellierung wird in dem Konzept noch ein zu geringer Raum zugestanden. Gegebene Klassenstrukturen werden grafisch unter Verwendung von Klassendiagrammen in der Coad-Yourdan-Notation (Coad / Yourdon 1991) dargestellt. Lösungen zu Aufgaben werden aber oftmals direkt auf Quelltextebene entworfen. Modellierungstechniken für die Systemdynamik (z.B. Sequenzdiagramme) haben in dem Konzept nur eine Randbedeutung (vgl. z.B. Czischke 1998a). Generell lässt sich feststellen, dass die objektorientierte Analyse zu kurz kommt. Das Konzept wird zur Umsetzung des Lehrplans für Informatik in der Sekundarstufe II in Nordrhein-Westfalen (MSWF 1999, 47f) empfohlen.
Inzwischen wird an dem Konzept weitergearbeitet. Die Klassenbibliothek wurde verbessert, UML wird als Darstellungsform empfohlen. Neue Unterrichtsprojekte für die Sekundarstufe

[45] Humbert (2001a), Mitautor des Konzepts, sieht einen Schwachpunkt der Handreichung (LSW 1999) in der fehlenden systematischen Evaluation: „Eine unterrichtliche Nutzung der Materialien ist nur auf einer fachlichen Grundlage möglich. Die angebotenen Unterrichtshilfen für den Informatikunterricht zeichnet aus, dass sie typischerweise in wenigen Kursen erprobt wurden; i.d.R. wird keine empirisch gestützte Evaluation der Materialien durchgeführt, bevor sie breit eingesetzt werden." (Humbert 2001a, 128)

II sind auf dem nordrhein-westfälischen Bildungsserver dokumentiert[46].

Objektorientiertes Modellieren und Robotik

Die Vermittlung von Fachkonzepten der Informatik über die Steuerung von virtuellen und realen Robotern hat im Bereich der Informatik-Ausbildung, insbesondere in der Programmierausbildung von Anfängerinnen und Anfängern, inzwischen eine gewisse Tradition. Arbeiten zur Turtle-Grafik mit Logo nach Papert (1980), „Karel the Robot" (Pattis 1981) und darauf basierende Derivate[47], wie z.B. „NIKI der Roboter" oder „Kara der Marienkäfer" (vgl. 2.3.2), belegen dies. Neuer hingegen sind Arbeiten zur Verknüpfung von objektorientiertem Modellieren und Robotik.

Hirsch, Magenheim und ***Reinsch*** (Hirsch et al. 2000) entwerfen einen Zugang zur Informatik mit Lego-Robotern (Mindstorms). Durch die Verwendung von Robotern erwarten sie verschiedene Vorteile:

> „Die Mindstorms-Roboter besitzen besonders für den Informatikunterricht in der Sekundarstufe I, aber auch für den Anfangsunterricht in der Sekundarstufe II eine hohe motivationale und mediale Qualität. Sie genügen in ihrer medialen Funktion im Unterricht und hinsichtlich der mit ihnen erschließbaren Themenbereiche einerseits Kriterien wie Altersgemäßheit, Bezug zum Erfahrungshorizont der Schülerinnen und Schüler, Wissenschaftsorientierung und Zukunftsbedeutung. Andererseits repräsentieren sie in der Erfahrung vieler Schülerinnen und Schüler auch einen am Spiel orientierten außerschulischen Freizeitkontext, so dass der Informatikunterricht von den damit verbundenen motivationalen Effekten profitieren kann. Voraussetzung dafür ist, dass entdeckendes und selbstgesteuertes Lernen [...] im Informatikunterricht nicht durch belehrende und frontale Unterrichtsformen [...] dominiert wird." (Hirsch et al. 2000, 36f)

Angestrebt wird auch die Einübung objektorientierter Sichtweisen:

> „Zugänge zur Idee des Modellierens lassen sich durchaus sinnvoll mit objektorientierten Sichtweisen verbinden." (ebd., 39)

Die Lernenden erstellen dazu ein Papiermodell des Roboters als ersten Entwurf einer Klasse Roboter mit assoziierten Klassen Motor und Sensor. Hiervon werden konkrete Roboter mit spezifischen Aufgabenstellungen (und dementsprechend angepassten Methoden) abgeleitet. In einem ersten Experiment mit 16 altersheterogenen Lernenden (Kompaktveranstaltung an der Universität Paderborn, zweimal vier Zeitstunden) gelang die Entwicklung objektorientierter Sichtweisen zunächst nicht (ebd., 45).

Dietzel und ***Rinkens*** (2001) berichten von Unterrichtserfahrungen in einem Informatik-Differenzierungskurs der Jahrgangsstufe 10 (16 Jungen und 4 Mädchen mit Vorkenntnissen in der Arbeit mit Logo) zur Einführung in Konzepte der Objektorientierung (Objekt, Klasse, Vererbung, Ereignis) mit Lego Mindstorms Robotern. In der Unterrichtsreihe bauten Lernende zunächst in Kleingruppen einen Roboter. Diesen sollten sie anschließend auf Papier präzise beschreiben. Ohne weitere Vorgaben enthielten die Beschreibungen Eigenschaften und Funktionalität, an denen die Begriffe Objekt, Attribut und Methode eingeführt wurden. Aus den vielfältigen Beschreibungen, von denen nicht alle Elemente relevant waren, wurde gemeinsam eine Schablone zur Beschreibung von Robotern erarbeitet und damit der Klassenbegriff eingeführt. Die Begriffsbildung wurde durch dieses Vorgehen gut unterstützt. Anschließend erhielten die Lernenden eine fertig gestellte Roboter-Klasse (in Java), die von ihnen analysiert, getestet und modifiziert werden konnte. Vererbung (Erweiterung der Funktionalität des Roboters) und Ereignisbehandlung (Roboter soll durch Labyrinth fahren) wurden angewandt. Wie Hirsch et al. (2000) berichten auch sie von einer hohen Motivation der Lernenden:

> „Die reale Existenz der Roboter-Objekte und die Möglichkeit, sie in die Hand zu nehmen, waren sowohl eine einträgliche Ausgangsbasis für die notwendige Abstraktion in der Objektorientierten Programmierung als auch ein gutes Anwendungs- und Testmedium für die entwickelten Pro-

[46] URL: http://www.learn-line.nrw.de/angebote/oop/ (aufgerufen am 12.11.03)

[47] Freiberger (2002) gibt eine Übersicht über verschiedene 2D- und 3D-Varianten von „Karel the Robot".

> gramme. Schwierig zu vermitteln war zwar nicht das Ereigniskonzept an sich, aber seine programmiertechnische Umsetzung." (Dietzel / Rinkens 2001, 198f)

Ein neuerer Ansatz zur Verknüpfung von objektorientiertem Modellieren und der Steuerung von Lego-Robotern (Diethelm et al. 2003) wird im Abschnitt 3.8 erörtert.
Der große Vorteil von (Lego-)Robotern als Medium zur Vermittlung von Informatikkonzepten liegt also in der zu erwartenden großen Motivation der Lernenden und daraus resultierenden Lernvorteilen. Für Schulen problematisch sind die nach wie vor hohen Anschaffungskosten sowie der zu erwartende Aufwand beim Zusammenhalten der Materialien für mehrere Baukästen.

Zwischenzusammenfassung 2.2

Mit dem Ziel, die nationale Diskussion in der fachdidaktischen Forschung zum objektorientierten Modellieren (vgl. Abschnitt 2.3.1.1) mit der tatsächlichen Situation im Informatikunterricht in Relation setzen zu können, wurden hier publizierte Unterrichtserfahrungen, -beispiele und -empfehlungen aus der Schulpraxis vorgestellt. Es zeigt sich, dass in vielen Arbeiten *objektorientiertes Programmieren* in den Vordergrund gestellt wird (Baumann 1990, Modrow 1992; Hermes 1996a, 1996b; Füller 1999, LSW 1999[48]). *Objektorientiertes Modellieren* wird nur in den Arbeiten von Penon / Spolwig sowie Hirsch et al. (2000) betont. Die ersten didaktisch-methodischen Arbeiten zur Objektorientierung befassten sich mit dem *Wechsel vom imperativen zum objektorientierten Programmieren* (Modrow 1992, Spolwig 1995, Baumann 1996; Hermes 1996a, 1996b). Auf vorhandene Kenntnisse insbesondere der Lehrpersonen zum imperativen Programmieren wurde, soweit wie möglich, aufgebaut und zunächst *objektbasierte* Programmierung mit imperativen Sprachen (und deren objektorientierten Erweiterungen) propagiert. Die neuen Konzepte und damit die neuen Denkweisen wurden am Anfang zurück- und lediglich als terminologische Änderungen dargestellt (Modrow 1992, Baumann 1996), um Lehrenden den Umstieg zu erleichtern. Diese Vorgehensweise führte allerdings oft nicht zum Erfolg. Spolwig (1995) empfahl hingegen einen objektbasierten Einstieg bei starker Betonung der Konzepte in OOA und OOE und berichtet von positiven Ergebnissen, besonders wenn zuvor bei den Lernenden ein intuitiver Objektbegriff über die objektorientierte Analyse von Standardanwendungen ausgebildet worden ist. Hubwieser (2000b) empfiehlt die objektorientierte Analyse von Standardanwendungen bereits für die Jgst. 6 zur Entwicklung einer soliden Fachsprache. *Die Unterrichtseignung von professionellen Programmiersprachen und Entwicklungsumgebungen* und daraus resultierende Konsequenzen für den Unterricht werden diskutiert bei Penon / Spolwig (1998), Spolwig (1999, 2000) und Füller (1999). Durch die zu starke Ausrichtung des Unterrichts auf die Erfordernisse der jeweiligen Implementierungssprache und -umgebung wurden Entwurfskonzepte unzureichend erlernt, worin ein Kernproblem des Informatikunterrichts besteht. Nach Penon und Spolwig (1998) zeigt sich dies auch in dem Angebot an *Lehrerfortbildungen*, die oft eher Sprachen als Konzepte zum Gegenstand haben. Als eine Konsequenz der oft ambivalenten Einschätzung professioneller Programmiersprachen für den Unterricht wurde die Klassenbibliothek sowie das zugehörige Einsatzkonzept „*Von Stiften und Mäusen*" (LSW 1999) entwickelt, das die oft geforderte integrative Behandlung von Algorithmen und Datenstrukturen in einem ansonsten durchgängig objektorientierten Ansatz sichert. Dennoch gibt es insbesondere aus dem auf einfache Grafikanwendungen eingeschränkten Beispielrahmen resultierende Vorbehalte. Der *Einsatz von (Lego-)Robotern bei der Vermittlung von Konzepten der Objektorientierung* ist für die Lernenden offenbar sehr motivierend (Hirsch et al. 2000, Dietzel / Rinkens 2001), führt aber auch zu einer Einschränkung des Beispielrahmens sowie zu hohen Kosten und organisatorischem Mehraufwand für Schulen.

---

[48] In den neueren Arbeiten auf der Webseite http://www.learn-line.nrw.de/angebote/oop/ (aufgerufen am 12.11.03) lässt sich eine im Vergleich zum im (LSW 1999) publizierten Konzept stärkere Betonung der Modellierung erkennen.

Es zeigt sich der hohe *Bedarf an Konzepten und geeigneten Beispielen für den Informatikunterricht*. Auch Lehrende müssen in Fortbildungsmaßnahmen an die neuen Konzepte herangeführt werden.

2.3.2 Internationale Arbeiten

Wie im Abschnitt 2.1 bereits dargelegt, beziehen sich internationale Arbeiten zum objektorientierten Modellieren im Bildungskontext im Wesentlichen auf den Bereich der Hochschulbildung. Nachfolgend werden zunächst exemplarisch zwei Ausnahmen betrachtet.

Objektorientierung in der Sekundarstufe

Eberle (1996) geht in seiner Habilitationsschrift zur Didaktik der Informatik[49], einer sehr umfangreichen Studie, nicht näher auf OOM ein, liefert aber Vorarbeiten zum informatischen Modellieren. Eberle benennt objektorientierte Techniken unter „Trends bei der Weiterentwicklung der Informations- und Kommunikationstechnologien" (ebd., 70f und 427), konzentriert sich bei Diskussion und Analyse von Problemlösungsmethoden und Vorgehensweisen der Software-Entwicklung aber auf den algorithmenorientierten Ansatz (ebd., 88ff). Dennoch identifiziert er Probleme und nennt Lösungen, die auch auf das objektorientierte Paradigma und dessen unterrichtliche Umsetzung übertragbar sind. Zu starke Konzentration auf eine Programmiersprachensyntax sollte durch Modellierung mit grafischen Strukturierungshilfsmitteln überwunden werden (ebd., 329ff). Bei zu informellen Darstellungsformen sieht er das Problem, dass die Unmissverständlichkeit der Lösungsbeschreibungen verloren geht, die sonst durch formale Sprachen geleistet wird. Dies wird Schwierigkeiten bei der Umsetzung von Modellen zur Folge haben. Eberle gelingt der wichtige Nachweis, dass ein spezifischer Transfer von Kompetenzen zur Algorithmik erfolgt und das Fach Informatik damit einen Beitrag zur allgemeinen Problemlösefähigkeit leistet:

> „Beim Programmieren sind die Problemlöse- und Denkfähigkeiten an die Strukturen der Programmiersprache gebunden. Diese wiederum sind nicht auf die Problemlösestrukturen aller Probleme generalisierbar. Während wir an anderer Stelle [...] vor allem das erste negative Ergebnis betont haben, gilt es auch, deswegen nicht einfach den formalen Bildungswert dieser Inhalte in Frage zu stellen, sondern in positiver Weise den gefundenen spezifischen Transfer zu betonen: Demnach ist Programmieren / Algorithmik geeignet, a) spezifische Problemlösefähigkeiten für andere Bereiche herauszubilden und b) wie in anderen Fächern auch einen aufgrund des jetzigen Forschungsstandes nicht größeren, aber auch nicht kleineren Beitrag zur Entwicklung allgemeiner Problemlösefähigkeiten zu leisten." (ebd., 212)

Eberle leistete damit Vorarbeiten zum informatischen Modellieren im Unterricht, ohne näher auf OOM einzugehen.

An der ETH Zürich wurde auf der Basis von Vorarbeiten zu „Karel the Robot" (vgl. 2.3.1.2, S. 30 und Pattis 1981) eine „Spielumgebung" zur Einführung in die Programmierung entwickelt, in der ein Marienkäfer (Kara) durch eine aus Quadraten zusammengesetzte, rechteckige Welt navigiert werden kann, die durch feste Hindernisse (Baumstümpfe) und bewegliche Hindernisse (Pilze) begrenzt wird und in der es aufsammel- und ablegbare Güter (Kleeblätter) gibt. Gesteuert wird Kara durch endliche Automaten (***Nievergelt*** 1999). Als wesentliche Motivation einer Einführung in die Programmierung wird begründet, dass diese zum allgemeinen Gedankengut einer modernen Industrie und Dienstleistungsgesellschaft gehört.

> „Nicht weil das Programmieren zum Alltag vieler Arbeitskräfte gehören würde, sondern weil es die beste, vielleicht sogar die einzige Art ist, wie man sich ein kompetentes Urteil darüber erlauben kann, was Computer im Prinzip können oder nicht." (ebd., 374)

Im Abschnitt 2.2 wurde ausführlich dargelegt, dass und warum in der informatischen Bildung

[49] Hierbei handelt es sich um die erste deutschsprachige Habilitationsschrift zur Didaktik der Informatik.

in Deutschland heute informatisches Modellieren betont wird. Nach Auffassung des Autors ergeben sich zwei wesentliche Kritikpunkte am Kara-System und dessen Verwendung:

- *Wahl der Aufgabenklassen*: In einer abgeschlossenen Roboterwelt stellt es eine natürliche Aufgabe dar, diese mit einem Roboter zu erkunden. Handelt es sich um einen Roboter zum Transport von Gütern innerhalb eines abgeschlossenen Lagers, ist auch das Aufsammeln und Umlagern von Gütern eine nahe liegende Aufgabe. Der Marienkäfer Kara wird nun aber auch dazu verwendet, mittels Kleeblättern, die er auslegt und einsammelt, und der Umgebung als „externem Gedächtnis" binäre Pascal-Dreiecke zu zeichnen oder das „Türme von Hanoi"-Problem zu lösen (Reichert et al. 2000), Aufgaben und Tätigkeiten, die im Zusammenhang mit einem Marienkäfer etwas ungewöhnlich anmuten.
- *Imperatives Programmieren in der Syntax von Java:* In der Java-Variante (JavaKara) wird der Marienkäfer anstatt mit endlichen Automaten mittels Java-Programmen gesteuert. Problematisch ist neben der Wahl der Aufgabenklasse, dass das dargestellte Beispielprogramm zu den „Türmen von Hanoi" (ebd., 314) imperative Programmierung in der Syntax von Java darstellt. Die Autoren merken dies selbst an:

  > „Die Beispiele decken primär die nicht objektorientierten Eigenschaften von Java ab, bieten aber auch einen kleinen Einblick in die Objektorientierung." (ebd., 314)

Für einen Einstieg in das objektorientierte Modellieren erscheint JavaKara daher aufgrund anderer Schwerpunkte wenig geeignet.

Objektorientierung in der Hochschulbildung

Kölling und ***Rosenberg*** entwickelten in ihren Arbeiten seit 1995 eine im Bildungskontext inzwischen sehr populäre objektorientierte Entwicklungsumgebung namens „BlueJ" (Kölling / Rosenberg 1996). Dieser gingen Arbeiten an einer objektorientierten Ausbildungssprache namens „Blue" (Kölling et al. 1995) und gleichnamiger Entwicklungsumgebung voraus. Bei BlueJ handelt es sich um eine Java unterstützende Variante. Ausgangspunkt dieser Entwicklung war, dass bestehende Software-Entwicklungsumgebungen das objektorientierte Paradigma unzureichend unterstützen. Kölling und Rosenberg sehen darin eine Ursache der Schwierigkeiten, die bei der Vermittlung objektorientierter Konzepte beobachtet werden:

> „It is our convention that the lack of truly object-oriented development environments has created major difficulties for teaching object-oriented technology." (Kölling / Rosenberg 1996, 83)

Die Unzulänglichkeiten bestehender Umgebungen unterteilen sie in zwei Bereiche: die Unterstützung des objektorientierten Paradigmas und die unzureichende Strukturvisualisierung bzw. fehlende Strukturmanipulationsmöglichkeiten (ebd., 84). Zur fehlenden Unterstützung des objektorientierten Paradigmas schreiben sie:

> „When these environments were adapted to object-oriented languages, source files were replaced by class descriptions. Typically more source files exist in the object-oriented environment than in the procedural form. In addition, these files have more relationships with each other, e.g. usage and inheritance relations. Tools have been added to the environment to manage these class files and some of their relationships. The general paradigm of the environment, however, has not changed. The environment is still used to build an application with exactly one entry point that can be compiled and executed only after all its parts have been completed. This is the program/procedure-oriented paradigm. The object-oriented paradigm, used in the programming language, has not been adopted in the environment. The object-oriented paradigm is based on the idea that objects exist independently, and that operations can be executed on them. Consequently, a user in a true object-oriented environment should be able to interactively create objects of any available class, manipulate these objects and call their interface routines. The composition of objects by the user must be possible […]. Any individual class […] can be tested independently as soon as it is completed." (Kölling / Rosenberg 1996, 85)

Ferner beklagen sie, dass Software-Entwicklungsumgebungen zwar grafische Werkzeuge zur Manipulation von GUIs bereitstellen (GUI-Builder), dass für die Manipulation der inneren Struktur von Software (des Modells) solche Möglichkeiten aber nicht vorgesehen werden. Als Ausnahme benennen sie das Smalltalk-System, in dem aber Klassenstrukturen nicht visualisiert würden. Als weiteres Problem benennen Kölling / Rosenberg (2000) den eigentlichen Anwendungszweck dieser Software:

> „The environments are designed for professional programmers. They are too complex and have a steep learning curve. Thus valuable teaching time is spent teaching the students how to use the environment and this detracts from the principles of programming." (Kölling / Rosenberg 2000, 429)

Mit BlueJ stellen sie eine Umgebung bereit, die die von ihnen beschriebenen Mängel vermeidet. Lernende haben darin die Möglichkeit, direkt zur Laufzeit mit Objekten zu interagieren, Methoden aufzurufen, Attributwerte zu verfolgen und so Klassen ohne zusätzlichen Testcode interaktiv zu testen und zu debuggen (Rosenberg / Kölling 1997). Nachteilig ist, dass als grafische Darstellung nur Klassendiagramme angeboten werden und die Laufzeitobjekte am unteren Bildschirmrand aufgelistet werden, nicht aber Beziehungen zwischen ihnen visualisiert werden. Kölling arbeitet inzwischen auch an einem Werkzeug zur Visualisierung von Nebenläufigkeitsaspekten in objektorientierten Programmen (Exton / Kölling 2000).

Woodman et al. (1997) stellen eine Lernumgebung für Studierende für objektorientierte Konzepte („the object shop") vor, die in Kooperation mit der BBC entstand. Zielgruppe sind Anfängerinnen und Anfänger der Informatik ohne Programmierkenntnisse. Es wird angestrebt, dass die Studierenden mit dieser multimedialen Umgebung, die auf CD-ROM bereitgestellt wird, zunächst objektorientierte Konzepte erlernen, bevor sie sich dann Programmierungskenntnisse aneignen. Die Lernumgebung enthält in 3D-Darstellung ein virtuelles, mehrstöckiges Kaufhaus. Woodman et al. erwarten, mit diesem Szenario männliche und weibliche Studierende anzusprechen. Die Studierenden erhalten in der Umgebung den Auftrag, verschiedene Einkäufe zu erledigen. Dazu müssen sie mit verschiedenen Objekten (z.B. Kreditkarte, Kalender, Einkaufsartikel) interagieren und deren Methoden über den jeweils zugehörigen „Objektcontroller" aktivieren oder Eigenschaften abrufen. Steigende Stockwerkzahlen korrespondieren mit wachsender Formalisierung und Detaillierung der Darstellung der jeweiligen Objektcontroller sowie der Bereitstellung zusätzlicher Werkzeuge. So können Studierende im vierten Stockwerk beispielsweise die Klasse eines Objekts inspizieren. Jede der Aktionen in der Umgebung bringt virtuelles Guthaben, das für die Erledigung der Einkäufe benötigt wird. Vermittelt werden hier allerdings nur objektorientierte Basiskonzepte, Modellierung ist nicht der Gegenstand. Positiv ist der anschauliche Bezug zwischen formalen Konzepten und realitätsnahen Darstellungen.

Holland et al. (1997) berichten von typischen Missverständnissen von Lernenden zu objektorientierten Basiskonzepten, die sie in eigener Fernlehre („distance education") identifizierten. Diese ordnen sie folgenden Bereichen zu (Holland et al. 1997, 132f):

- „Avoiding object/variable conflation
- Objects are not simple records
- Work in methods is not all done by assignment
- Object/class conflation
- Identity/attribute confusion
- Conflation of textual representation of objects and references to objects"

Anfängerbeispiele enthielten oft Klassen mit nur einer Instanzvariable. Es bestehe daher die Gefahr, dass Lernende dies generalisierten und zu dem Schluss kämen, Objekte seien eine Art „Wrapper" für Variable. Zu vermeiden sei dies, wenn mindestens zwei Instanzvariablen verwendet würden. Ferner sollten Instanzvariablen verschiedenen Typs verwendet werden (mindestens ein atomarer Typ und eine Klasse), um nicht den falschen Schluss nahe zu legen, Instanzvariablen müssten alle auf Objekte einer einzelnen Klasse zeigen. Um den Datenaspekt von Objekten nicht über zu betonen, sollten Beispiele gewählt werden, bei denen sich die

Auswirkungen von Methoden objektzustandsabhängig signifikant veränderten. Ansonsten bestehe die Gefahr, Objekte für einfache Datenfelder zu halten. Anfängerbeispiele zeigten ferner die Tendenz, in den Methoden zu viele Zuweisungsoperationen und zu wenig Nachrichtenaustausch zu verwenden. Dies könne zu einer Überbetonung imperativen Programmierstils führen. Wenn von jeder Klasse nur ein Objekt erzeugt werde, bestehe die Gefahr, Klasse und Objekt zu verwechseln. Es sollten daher immer mindestens zwei Objekte erzeugt werden. Bei Studierenden mit Vorkenntnissen zu Datenbanken beobachteten sie eine Verwechselung von „name"-Instanzvariablen mit der Identität des Objekts.
Die beobachteten Missverständnisse traten sowohl bei fortgeschrittenen Studierenden als auch bei einem Testdurchlauf eines Kurses für Studienanfänger auf. Holland et al. betonen, dass es sich um persönliche Erfahrungen handele, ohne formal beweisbaren Zusammenhang zwischen Lehrpraktiken und beobachteten Missverständnissen. Dennoch ergeben sich daraus eine Reihe von Gestaltungshinweisen für Anfängerbeispiele und Übungen.
Dershem und ***Vanderhyde*** (1998) entwickelten ein auf den Java-Reflection-Klassen[50] basierendes Werkzeug namens „ObjectVisualizer", um Benutzern die Interaktion mit und die Visualisierung von Java-Objekten zu ermöglichen. Im Rahmen einer Einführung in objektorientierte Konzepte können Studierende mit dem ObjectVisualizer Objekte instanziieren, diese in der Klassenumgebung „bewegen", Methoden aktivieren und Veränderungen der Objektzustände beobachten. Durch Methodenaufruf neu erzeugte Objekte werden in der Klassenumgebung dargestellt und können per „drag and drop" einer Methode als Parameter übergeben werden. Die Autoren betonen, dass dieses Werkzeug gut für Lehr-Lern-Prozesse geeignet sei. Studierende könnten damit selbst entwickelte Klassen testen und debuggen, aber auch externe Klassen (z.B. Java-Standard-Klassen) erkunden, ebenso wie solche, deren class-Dateien zwar verfügbar seien, aber nicht deren Quellen. Der Ansatz erinnert stark an die Arbeiten von Kölling / Rosenberg zu BlueJ (vgl. S. 33). Im Vergleich zu BlueJ bietet der ObjectVisualizer aber deutlich eingeschränkte Visualisierung (z.B. fehlt die Klassenstruktur).
Seffah et al. (1999) berichten von Schwierigkeiten von Studierenden, objektorientierte Basiskonzepte richtig zu verstehen. Es falle Studierenden zwar leicht, konzeptionell zwischen Vererbung, Aggregation und Assoziation zu unterscheiden. Studierende empfänden es aber als sehr schwierig, diese Relationsarten korrekt in Problemlösungen zu integrieren. Ferner empfehlen sie, Entwurfstechniken und Programmierung verzahnt zu vermitteln, da:

> „Teaching programming before design will lead to bad design practices, which would require effort to unlearn. Teaching design before programming could result in a sterile course where students are unable to fully apply what they learn." (Seffah et al. 1999, 71)

Es zeigt sich auch für diese Zielgruppe der große Bedarf an Beispielen zum Üben und Vertiefen der Konzepte.
Mitchell (2000) berichtet von Problemen, die Fakultätsmitarbeiter beim Wechsel von der strukturierten Programmierung hin zum objektorientierten Programmieren hatten. Seiner Einschätzung nach kam es häufig zu einer „ugly [...] transition" (Mitchell 2000, 98). Durch den häufigeren Wechsel der Ausbildungssprache erlangten die Lehrenden nicht genügend Sicherheit, um wirksame Vermittlungsstrategien zu entwickeln. Im Fall von Pascal und Java führte dies dazu, dass die Pascal-Beispiele nach Java übersetzt wurden, die Lehrmethodik aber unverändert blieb. Die Vorteile des Wechsels in das objektorientierte Paradigma gingen so verloren.
Raner (2000) entwickelte eine Modellierungssprache (Object Visualization and Annotation Language – OVAL) zur Visualisierung der Schlüsselideen der Objektorientierung, da er die UML für Bildungsprozesse für ungeeignet hält:

[50] verfügbar in der Java-Distribution von SUN (URL: http://java.sun.com/ (aufgerufen am 12.11.03)) in dem Paket java.lang.reflect

> „Several languages and notations have been developed for the visual presentation of object-oriented ideas and designs […]. Such languages or notations are an excellent means of communication and documentation amongst experts. However, for novice trainings they are not very suitable. Instead they raise additional difficulties: not only a large number of new ideas and a new way of thinking have to be learned, but also a highly non-intuitive graphic notation to present these ideas." (Raner 2000, 45)

Signifikante Bedeutungsunterschiede werden teilweise durch unbedeutende grafische Veränderungen dargestellt. Als Beispiel nennt er die Repräsentation von Klassen und Objekten:

> „Both rectangles look basically the same, however, one represents the concept of a class and the other one presents the concept of an object. The differences between the two concepts are only conveyed by the lettering of the boxes and typographic distinctions like underlining. " (ebd., 45)

In OVAL werden Konzepte so veranschaulicht, dass die Bedeutung aus der Darstellung deduziert werden kann:

> „In contrast to the UML representation, classes and objects do not look the same in OVAL. Classes have a sharp inner boundary, whereas objects have a sharp outer boundary. This visualizes the idea of a class as a template or stencil for its individual objects." (ebd., 46)

Auf ähnliche Weise werden Relationen visualisiert. Allerdings werden von OVAL nur Klassendiagramme unterstützt. Raner berichtet von positiven Erfahrungen in der Anfängerausbildung zur Vermittlung der Grundkonzepte. Problematisch für Studierende könnte sein, dass es sich bei OVAL nicht um eine standardisierte Sprache handelt. Für die schulische Zielgruppe ist dieser Aspekt sekundär.

Box und ***Whitelaw*** (2000) begründen Schwierigkeiten der Studierenden mit objektorientierter Abstraktion über den Konstruktivismus:

> „The theory of constructivism asserts that new knowledge is a development from the old knowledge [...]. The means of concept enrichment are the formation of sub-categories (differentiation), the reduction of an entity into parts (division), association of previously unconnected ideas, the aggregation of an entity from its parts, and the abstraction of a concept from a number of concrete examples. […] Division and differentiation are the easiest concept modifications as they do not involve the global change of the concept structure. Association requires the linking of two concepts. As this involves finding appropriate concepts before the linking can take place, it is intermediate in difficulty. Aggregation requires the grouping of entities. Like association, this involves finding appropriate concepts before linking can take place so it is more difficult than association. […] Abstraction is the really difficult learning action. A significant part of abstraction is the decision as to which entities are to be grouped together and which attributes are to be ignored or parameterised." (Box / Whitelaw 2000, 12f)

Sie empfehlen die CRC-Kartenmethode nach Beck und Cunningham (1989) insbesondere für die Ableitung von Klassen aus lebensweltlichen Objekten. Die CRC-Kartenmethode ist im Bildungskontext zum objektorientierten Modellieren weit verbreitet. Sie fördert kooperative Lehr-Lern-Prozesse und ihr Ergebnis stellt einen nur wenig formalisierten Zwischenschritt auf dem Weg von einer Problembeschreibung zu einem Klassendiagramm dar. Box und Whitelaw sehen allerdings auch einige lernpsychologische Nachteile der CRC-Kartenmethode, z.B.:

> „In the CRC methodology, implicit links are made and broken as cards are moved around the desk. If a student is having difficulty following the process, then the fact that the links are implicit prevents the student from reworking the problem more slowly to understand the methodology." (Box / Whitelaw 2000, 14)

Ein CRC-Kartenmodell kann ebenfalls im Team mit der Methode des Objekt-Rollenspiels (Bellin / Simone 1997) anhand einer Menge von Anwendungsfällen „durchgespielt" und überprüft werden. Die große Stärke der CRC-Kartenmethode und des Objekt-Rollenspiels liegt in der Aktivität der Lernenden und deren Kooperation. Für die Einzelarbeit sind beide Methoden nur bedingt bzw. nicht geeignet. Außerdem setzen sie Grundkenntnisse zu den objektorientierten Konzepten voraus.

Da sich auch in der Hochschulausbildung Vorbehalte hinsichtlich der Eignung professioneller Programmiersprachen, wie Java oder C++, für die Anfängerausbildung ergaben, wurden auch hier verschiedene Software-Bibliotheken entwickelt, um den Einstieg zu erleichtern. Dies korrespondiert mit dem „Stifte und Mäuse"-Ansatz an nordrheinwestfälischen Schulen (vgl. S. 29). Publizierte Beispiele beziehen sich z.B. auf die Konstruktion und Manipulation von Artificial-Life-Welten (Pattis 1997), Grafikanwendungen (Bruce et al. 2001) und GUI-Erstellung (Rasala et al. 2001). Während es sich hierbei um ergänzende Bibliotheken zu professionellen Sprachen handelt, geht der Ansatz von Roberts (2001) in eine andere Richtung. Er entwickelte eine didaktisch reduzierte Java-Version mit nur 17 Klassen im Kern.
Weitere Beiträge thematisieren z.B. Erfahrungen aus einführenden Programmierungsveranstaltungen, die von prozeduraler auf objektorientierte Programmierung umgestellt wurden (z.B. Woodman et al. 1996, McCracken 1999, Hadjerrouit 1999), Software-Mustern in der Anfängerausbildung (Wallingford 1996, Della / Clark 2000) oder verbreitete Fehlwahrnehmungen zur Objektorientierung im Bildungskontext (Lewis 2000).

Zwischenzusammenfassung 2.3
Vorgestellt wurden verschiedene Lernhilfen für die universitäre Informatikausbildung zur Objektorientierung. Es handelt sich hierbei um *Lernumgebungen für objektorientierte Basiskonzepte* („the object shop", Woodman et al. 1997) sowie um Werkzeuge zur Interaktion mit Klassen und Objekten zur Laufzeit mit unterschiedlich stark ausgeprägten Visualisierungsmöglichkeiten („ObjectVisualizer", Dershem / Vanderhyde 1998; bzw. „BlueJ", Kölling / Rosenberg 1996). BlueJ versteht sich außerdem auch als studierendentaugliche, objektorientierte *Entwicklungsumgebung*. Für den schulischen Einsatz wurden solche Werkzeuge bislang nicht beschrieben. Nachteilig bei allen oben genannten Werkzeugen ist, dass sie sich jeweils nur auf Teilaspekte bzw. einzelne Sichten eines Modells konzentrieren, Lernenden so aber kein Gesamtbild vermitteln können. Aufgrund von Vorbehalten gegenüber der UML wurde eine eigene *Modellierungssprache für den Bildungskontext* entwickelt („OVAL", Raner 2000). Obwohl diese im Bereich der Klassendiagramme eine intuitivere Darstellung zeigt, wurde der wichtige Bereich der dynamischen Modellierung leider völlig ausgeklammert. Es wurden *Missverständnisse von Studierenden beim Erlernen objektorientierter Konzepte* dargestellt (Holland et al. 1997). Die daraus gewonnenen Erkenntnisse sind beim Entwurf von Lehr-Lern-Materialien und Konzepten zu berücksichtigen. Auch auf Hochschulebene zeigten sich *Probleme mit dem Paradigmenwechsel vom imperativen zum objektorientierten Problemlösen* (Mitchell 2000). Um bei der Betonung der Modellierung Lernende und Lehrende stärker zu unterstützen, sind geeignete Lehr-Lern-Materialien erforderlich.

2.3.3 Kritische Positionen zur Objektorientierung

Nachdem bislang Konzepte, die Vorzüge eines objektorientierten Ansatzes und ausgewählte Problembereiche betont wurden, werden nun exemplarisch zwei kritische Ansichten zur Objektorientierung vorgestellt.
Böszörményi (1998) argumentiert am Beispiel der Programmiersprache Java gegen einen Beginn der Programmierausbildung an Universitäten nach dem objektorientierten Ansatz. Als wesentliches, programmiersprachenunabhängiges Argument nennt er, dass das Konzept der Modularisierung fundamentaler sei als das der Objektorientierung:

> „First for many purposes it [object-oriented development] is definitely the wrong approach, and, e.g., the concept of *modularization* is much more fundamental than that of object-orientation. I don't know any software of non-trivial size where modularization was not indispensable to handling complexity, whereas I know many cases where object-orientation does not help much. Of course, object-orientation includes modularization, but this is what makes modularization [...] a simpler and more basic concept." (ebd., 142)

Im Sinne der Strukturierung von Lehr-Lern-Prozessen nach dem didaktischen Prinzip „Vom Einfachen zum Komplexen“ ist Böszörményi uneingeschränkt zuzustimmen. Die Diskussion der Frage, ob oder ob nicht die Informatikausbildung an Schulen bzw. Hochschulen mit einem Einstieg in die Objektorientierung begonnen oder ob zunächst ein imperatives Fundament gelegt werden sollte, wurde in der Literatur intensiv geführt. Da beide Positionen gut begründet wurden, ist eine abschließende Beantwortung dieser Frage weiterhin offen (wenn sich diese Frage überhaupt abschließend beantworten lässt) und soll aus diesem Grund auch hier nicht um einen weiteren Beitrag ergänzt werden. Im Hinblick auf die im Rahmen dieser Arbeit verfolgte Betonung der Vermittlung von Fachkonzepten aus dem Themenbereich des informatischen Modellierens bei Zurückdrängung der Dominanz von Programmiersprachenkonstrukten spricht einiges für einen objektorientierten Einstieg. Das Konzept der Modularisierung ist tendenziell implementierungsnäher als das Konzept der Objektorientierung, welches in allen Phasen der Entwicklung angewandt werden kann. Objektorientierung stellt eine spezielle, am Denken der Menschen orientierte, Sichtweise auf Realitätsausschnitte dar, welche im Rahmen der Modellierung dementsprechend abstrahiert und strukturiert werden. Während sich das Konzept der Objektorientierung anhand von beliebigen natürlichen Objekten, ihren Eigenschaften und Relationen prinzipiell anwenden lässt, ist dies im Fall der Modularisierung deutlich eingeschränkter möglich. Modularisierung lässt sich zwar prinzipiell auch in der Lebenswelt beobachten, indem z.B. technische Geräte über eine Benutzungsschnittstelle bedient werden können, ohne etwas über ihr Innenleben wissen zu müssen. Die Vorteile zeigen sich dort, wo es darum geht, ein Innenleben (eine Implementierung) nach außen hin zu verbergen, so dass dieses ausgetauscht werden kann bei Erhalt der Schnittstelleneigenschaften. Diese Sichtweise weist einen hohen Entwurfs- und Implementierungsbezug auf. Deshalb könnte einiges dafür sprechen, imperativ zu beginnen, wenn es das Ziel ist, Programmierung zu erlernen (vgl. Argumentation Böszörményi) und objektorientiert, wenn Modellierungstechniken im Vordergrund stehen.

Die Kritik von ***Broy*** und ***Siedersleben*** (2002) erfolgt nicht unter pädagogischen Gesichtspunkten, sondern vor dem Hintergrund von Problemlösungseigenschaften:

> „Objektorientierung löst trotz ihrer Beliebtheit und Verbreitung längst nicht alle Probleme, und manche hat sie erst geschaffen.“ (Broy / Siedersleben 2002, 4)

Als ein wesentliches, durch Objektorientierung neu geschaffenes Problem nennen sie die Verletzung des Geheimnisprinzips bei der Vererbung (im Sinne von Implementationsvererbung). Aufgrund verschiedener Kritiken kommen die Autoren zu dem Schluss:

> „Die Objektorientierung unterstützt uns nicht ausreichend bei der Erstellung großer verteilter Softwaresysteme.“ (ebd., 5)

Sie fordern deshalb eine Weiterentwicklung. Die Erstellung großer verteilter Software-Systeme ist andererseits kein gegenwärtiges Ziel von allgemein bildendem Informatikunterricht. Ferner kann nicht davon ausgegangen werden, dass die Entwicklung in der Software-Technik mit der Etablierung der Objektorientierung abgeschlossen ist. So wie Weiterentwicklungen im Bereich der imperativen Programmierung schließlich zur objektorientierten Programmierung geführt haben, wird sehr wahrscheinlich der Zeitpunkt kommen, an dem auch die Objektorientierung weiterentwickelt und dann vielleicht auch anders benannt wird. Für die informatische Bildung in Schulen wird dann erneut die Frage zu beantworten sein, welche der dann neuen informatischen Fachkonzepte mit den zu diesem Zeitpunkt gültigen Zielen des Informatikunterrichts übereinstimmen.

2.4 Fazit

2.4.1 Zusammenfassung

In den vergangenen Abschnitten wurde das Themenfeld *objektorientiertes Modellieren im Bildungskontext* in verschiedenen Teilaspekten erörtert.
Im Abschnitt 2.2 wurde der gewachsene Stellenwert der informatischen Modellierung aus fachdidaktischer Sicht dargelegt. Dies stärkt die Begründung der Thematisierung objektorientierten Modellierens im Informatikunterricht, wenngleich von verschiedenen Autoren (Humbert 2001b, Thomas 2002b) und in Bildungsempfehlungen (Breier et al. 2000b) gefordert wird, eine Verkürzung des Unterrichts auf eine Art der Modellierung zu vermeiden. Diese eher gesamtcurriculare Sicht wird auch vom Autor dieser Arbeit geteilt und steht nicht im Widerspruch zur Entwicklung eines Konzeptes für objektorientiertes Modellieren.
Im Abschnitt 2.3.1 wurden nationale Arbeiten zum objektorientierten Modellieren diskutiert unterteilt nach „Empfehlungen, Konzepten und Ergebnissen“ (Abschnitt 2.3.1.1) und „Unterrichtserfahrungen und -beispielen“ (Abschnitt 2.3.1.2). Im Abschnitt 2.3.1.1 wurde die Sicht auf dieses Themengebiet anhand von fachdidaktischen Forschungspublikationen erörtert (vgl. Zwischenzusammenfassung 2.1 auf S. 22). Es wurden Argumente für eine Einbeziehung von OOM in Lehr-Lern-Prozesse der Sek. I und Sek. II identifiziert und verschiedene Unterrichtskonzeptionen vorgestellt. Diese konzentrierten sich auf die objektorientierte Analyse von Standardanwendungen in der Sek. I (Hubwieser 2000b) sowie die unterrichtliche Behandlung des Software-Entwicklungsprozesses (Husch 1993, Hampel et al. 1999). Verschiedene Strukturierungsmöglichkeiten für den Informatikunterricht zum OOM im Sinne von didaktischen Linien wurden vorgestellt (Crutzen / Hein 1995) sowie das Problem fehlender unterrichtsgeeigneter Software-Werkzeuge thematisiert (Husch 1993, Schwill 1995, Hampel et al. 1999). Diese Sicht von Fachdidaktikern auf das Themenfeld wurde im Abschnitt 2.3.1.2 mit der schulpraktischen Sicht kontrastiert (vgl. Zwischenzusammenfassung 2.2 auf S. 31). Es zeigte sich, dass in vielen Arbeiten zwar Aspekte der objektorientierten Programmierung betont werden (Baumann 1990, Modrow 1992; Hermes 1996a, 1996b; Füller 1999, LSW 1999), aber objektorientiertes Modellieren selten diskutiert wird (Arbeiten von Penon / Spolwig; Hirsch et al. 2000). Eine insbesondere für Informatiklehrende bedeutende Fragestellung waren Vorgehensweisen beim Wechsel von der imperativen zur objektorientierten Programmierung in der Schulpraxis (Modrow 1992, Spolwig 1995, Baumann 1996; Hermes 1996a, 1996b). Aufgrund eines Mangels an unterrichtsgeeigneten Software-Werkzeugen wurde die Unterrichtseignung von professionellen Programmiersprachen und Entwicklungsumgebungen und daraus resultierende Konsequenzen für den Unterricht diskutiert (Penon / Spolwig 1998; Spolwig 1999, 2000; Füller 1999). Als ein gut ausgearbeitetes Konzept für das Erlernen von OOP wurde „Von Stiften und Mäusen“ (LSW 1999) vorgestellt, gegenüber dem es aber aus dem auf einfache Grafikanwendungen eingeschränkten Beispielrahmen resultierende Vorbehalte gibt. Obwohl sich zeigte, dass der Einsatz von (Lego-)Robotern bei der Einführung von Informatikkonzepten für Lernende motivierend ist (Hirsch et al. 2000, Dietzel / Rinkens 2001), stellt auch dieser Ansatz aus Kosten- und organisatorischen Gründen noch keine für alle Schulen geeignete Variante dar. Ferner erfolgt hierbei eine Beschränkung auf eine spezielle Beispielklasse.
Im Abschnitt 2.3.2 wurde die Sicht dann auf internationale Arbeiten erweitert (vgl. Zwischenzusammenfassung 2.3 auf S. 37). Da die informatische Bildung an Schulen im Bereich des objektorientierten Modellierens in diesem Kontext kaum diskutiert wird, wurden hier insbesondere Arbeiten aus dem Bereich der Informatikanfängerausbildung an Hochschulen analysiert. Ein Schwerpunkt hierbei waren verschiedene Lernhilfen für Studierende in Form von Lernumgebungen für Basiskonzepte (Woodman et al. 1997), Werkzeugen zur Interaktion mit Klassen und Objekten zur Laufzeit (Kölling / Rosenberg 1996, Dershem / Vanderhyde 1998)

und anschaulichen Modellierungssprachen (Raner 2000). Es handelt sich hierbei um Werkzeuge zur Einführung in Basiskonzepte, sowie zur Interaktion mit Klassen und Objekten, die sich auf statische Teilaspekte der Objektorientierung konzentrieren. Ein einführender Überblick über statische und dynamische Aspekte, der verschiedene Modellsichten kombiniert, ist damit nicht möglich. Es zeigte sich, dass der Paradigmenwechsel vom imperativen zum objektorientierten Problemlösen auch an Hochschulen Schwierigkeiten bereitete (Mitchell 2000). Holland et al. (1997) beschrieben typische Missverständnisse von Studierenden zu objektorientierten Konzepten und nannten Strategien zur Vermeidung.
Aus diesen drei Sichten auf das Themenfeld ergeben sich eine Reihe von Schlussfolgerungen für die Entwicklung eines Bildungskonzepts zum objektorientierten Modellieren.

2.4.2 Schlussfolgerungen

Argumente für eine Einbeziehung von OOM in den Informatikunterricht

Argumente für eine Einbeziehung von OOM in den Informatikunterricht wurden bereits im Abschnitt 2.3.1.1 (vgl. Zwischenzusammenfassung 2.1 auf S. 22) geliefert. Auf der Basis der vorliegenden Analyse des fachdidaktischen Standes zum objektorientierten Modellieren lassen sich diese Argumente um ein weiteres, gewichtiges Argument ergänzen. Die Objektorientierung ist zu einem fundamentalen Gegenstandsbereich der Informatik im Sinne der *fundamentalen Ideen* von Schwill (1993a, 1997) geworden und sollte deshalb in allgemein bildendem Informatikunterricht einen festen Stellenwert haben. Dies lässt sich belegen, wenn man die vier Fundamentalitätskriterien von Schwill (s. S. 11) auf die Objektorientierung anwendet.

- *Horizontalkriterium*
  Das Horizontalkriterium ist erfüllt, weil objektorientierte Sichtweisen in verschiedenen informatischen Teilgebieten etabliert sind, wie z.B. in der Software-Technik, bei Datenbanken, bei Rechnernetzen und verteilten Systemen und bei der Beschreibung von parallelen Systemen.

- *Vertikalkriterium*
  Die Vermittelbarkeit von Gegenständen aus dem Bereich der Objektorientierung auf verschiedenen Bildungsniveaus zeigte sich im Abschnitt 2.3, wo Ausbildungserfahrungen aus den Sekundarstufen I und II sowie aus der Hochschulbildung analysiert wurden. Schwill selbst untersuchte (2001) die informatischen Fähigkeiten von Kindern und stellte die Frage, „ab wann [...] man mit Kindern Informatik machen [kann]?“. Er kam zu dem Schluss, dass bestimmte Fähigkeiten bereits im Grundschulalter vorhanden und vermittelbar sind, wie z.B. die Hierarchisierung, die eine wesentliche Grundlage der Objektorientierung darstellt. Aspekte der Objektorientierung sind also in Ansätzen prinzipiell bereits im Grundschulalter vermittelbar (vgl. Abschnitt 3.8).

- *Zeitkriterium*
  Die zeitlichen Anfänge der Objektorientierung reichen bis in die sechziger Jahre zurück[51]. Auch wenn ihre große Popularität erst in den neunziger Jahren einsetzte, kann das Zeitkriterium (Bedeutung über einen längeren Zeitraum) somit als erfüllt angesehen werden.

[51] Als erste Sprache, in der bereits Objekte, Klassen und Vererbung genutzt werden konnten, gilt heute das bereits 1966 entwickelte Simula (Dahl / Nygaard 1966). Dort wurden die objektorientierten Konzepte aber noch anders genannt. Als erste objektorientierte Programmierumgebung gilt das Smalltalk-72-System (Goldberg / Kay 1976).

- *Sinnkriterium*
  Das Sinnkriterium schließlich ist ebenfalls erfüllt, da sich die Objektorientierung gut als Erklärungsmodell für informatische Erscheinungen eignet (Knapp / Fischer 1998, Füller 1999, Spolwig 1999).

In wissenschaftspropädeutischem Informatikunterricht der Sek. II müssen Konzepte aus dem Bereich der Objektorientierung deshalb thematisiert werden.

Ursachen für den unzureichenden Transfer in die Schulpraxis

Es zeigt sich eine deutliche Differenz zwischen den Gegenständen fachdidaktischer Forschung zur Objektorientierung und schulpraktischen Erwägungen. Während sich die fachdidaktische Forschung mit der Begründung von Inhalten und der Entwicklung von Konzepten zum informatischen und hier speziell zum objektorientierten Modellieren befasst, werden in Arbeiten von Schulpraktikern zu großen Teilen unterrichtshandwerkliche Fragen zum OOP diskutiert. Es fehlt an geeigneten Schnittstellenkonzepten, die dazu beitragen, die Gegenstände fachdidaktischer Forschung besser in die Schulpraxis zu transferieren. Hierbei lassen sich um 2001 folgende Hauptproblemfelder identifizieren:

1. *Fehlende Konzepte für die Vermittlung von Basiskonzepten zum OOM*
   Beschriebene Unterrichtskonzepte verwenden die objektorientierte Analyse von Standardanwendungen (Hubwieser 2000b) zur Vorbereitung einer „sauberen“ Fachsprache oder betonen die Vermittlung von Systemgestaltung (Husch 1993, Hampel et al. 1999). In Teilen erfolgt eine Vermittlung von Basiskonzepten zum OOM im Konzept „Von Stiften und Mäusen (SUM)“ (LSW 1999)[52] sowie den Arbeiten zur Informatikvermittlung mittels Lego-Robotern (Hirsch et al. 2000, Dietzel / Rinkens 2001). Das SUM-Konzept stellte in der Version von 1999 aber besonders die OOP in den Vordergrund.
2. *Fehlende Unterrichts- und Übungsbeispiele zum OOM*
   Es finden sich nur vereinzelt Publikationen zu Unterrichts- und Übungsbeispielen (vgl. Abschnitt 2.3.1.2) zur Objektorientierung. Oft sind dies lediglich gut dokumentierte Lösungen zu objektorientierten Programmieraufgaben und Einzelbausteine, denen die Einordnung in ein Unterrichtskonzept zum OOM fehlt. Der Unterricht wird häufig projektorientiert organisiert. Verschiedene Autoren beschreiben Schwierigkeiten mit der „richtigen“ Projektgröße. Um die Vorteile der Objektorientierung im Unterricht richtig ausnutzen zu können, dürfen die Projekte nicht zu klein sein (Husch 1993, Hampel et al. 1999). Objektorientierte Projekte binden dann aber viel Unterrichtszeit aufgrund der spezifischen Weise der Strukturierung (Penon / Spolwig 1998). Es besteht die Gefahr, dass das Interesse der Lernenden im Projektverlauf nachlässt. Solche Projekte setzen fundierte Kenntnisse zum OOM voraus oder erfordern projektbegleitend deren Erwerb. Wie dies geschieht, wird selten beschrieben. Aufgrund von Schwierigkeiten mit den Kategorien der „neuen Informatikdidaktik“, Komplexität, Abstraktion, Modellbildung, Modularisierung und Klassenbildung, sind nach der Einschätzung von Spolwig (2000) hierzu nur wenige Unterrichtsbeispiele publiziert. Lehrerfortbildungen konzentrieren sich auf objektorientierte Programmiersprachen und Programmierumgebungen und vernachlässigen die Vermittlung von Konzepten zum objektorientierten Modellieren (Penon / Spolwig 1998). Es besteht also ein Bedarf an Unterrichts- und Übungsbeispielen, auch im Bereich der Hochschulausbildung (Seffah et al. 1999). Gestaltungshinweise lassen sich aus (Crutzen / Hein 1995, Holland et al. 1997) ableiten.

[52] insbesondere die neueren Arbeiten (vgl. 2.3.1.2, S. 29)

3. *Fehlende unterrichtsgeeignete Informatiksystemunterstützung beim Erlernen der Basiskonzepte des OOM*
   Aufgrund des Mangels an unterrichtsgeeigneten Werkzeugen kommen professionelle, komplexe Software-Entwicklungsumgebungen zum Einsatz. Deren Einsatz bindet viel Unterrichtszeit, die für die Vermittlung von Konzepten verloren geht (Husch 1993, Füller 1999, Hampel et al. 1999). Arbeiten aus der Schulpraxis thematisieren Aspekte, wie professionelle Werkzeuge und Sprachen für den Unterricht genutzt werden können (Penon / Spolwig 1998; Füller 1999; Spolwig 1999, 2000). Gefordert werden Umgebungen, mit denen Lernende „interaktiv und spielerisch explorativ Objekte auf ihre Funktionen analysieren und sie mit zunehmendem Niveau manipulieren, kombinieren, rekonfigurieren, evolutionär erweitern und schließlich neu entwickeln können“ (Schwill 1995). Werkzeuge, die für die Hochschulausbildung beschrieben wurden, können diesen Bedarf nur zu einem kleinen Teil decken, da sie sich vorwiegend auf statische Modellaspekte konzentrieren und einen Sichtenwechsel bei der Modellbetrachtung nicht unterstützen (Kölling / Rosenberg 1996, Woodman et al. 1997, Dershem / Vanderhyde 1998, Raner 2000). Konzepte zum Einsatz solcher Werkzeuge im Unterricht wurden bislang nicht beschrieben. In der Einbeziehung von Werkzeugen zur Exploration von Basiskonzepten des OOM in den Informatikunterricht besteht die Möglichkeit, die starke Orientierung auf Programmiersprache und -umgebung in der Anfangsphase zu überwinden und neue Formen der Handlungsorientierung im Unterricht zu etablieren. Für „Wirkprinzipien von Informatiksystemen“ konnte bereits in der Vorbereitungsphase dieser Arbeit beispielhaft gezeigt werden, dass ein experimentierender, handlungsorientierter Zugang möglich ist, der auf Programmierung verzichtet (Steinkamp 1999). Schülerinnen und Schüler können dort verschiedene Komponenten miteinander verknüpfen und deren dynamisches Verhalten beobachten, Vorhersagen über zukünftiges Systemverhalten treffen und Fehlersituationen analysieren.
4. *Fehlende Empfehlungen zur Strukturierung des Lehr-Lern-Prozesses zum OOM*
   Einzelne Aspekte der Strukturierung des Lehr-Lern-Prozesses werden in verschiedenen Arbeiten thematisiert (vgl. Abschnitt 2.3.1). Von Crutzen und Hein (1995) wurden Metastrukturen in Form von didaktischen Linien beschrieben (s. S. 16). Eine Gesamtsicht auf das Themenfeld OOM, die Lernenden und Lehrenden Orientierung gibt, fehlt.

Es zeigt sich um 2001 also ein Bedarf für einen fachdidaktischen Verbund bestehend aus Unterrichts- und Übungsbeispielen, unterrichtsgeeigneten Informatiksystemen zur Förderung der Aneignung von Fachkonzepten, Strukturierungshinweisen für den Unterricht und damit verbundenen Einsatzkonzepten. Ein solcher Verbund wird im Kapitel 3 als *didaktisches System für OOM* konzipiert und von, nach Abschluss der Literaturanalyse publizierten, Ergebnissen informatikdidaktischer Forschung zum OOM im Informatikunterricht bis 2003 abgegrenzt (vgl. 3.8). Durch die Begründung und Ausgestaltung eines solchen Verbundes kann die Entwicklung von Bildungsstandards für den Informatikunterricht fachdidaktisch vorbereitet werden.
Im Rahmen eines solchen Verbundes bereitgestellte Materialien müssen für verschiedene Unterrichtsformen geeignet sein, für Gruppen- und Einzelarbeit ebenso wie für Unterrichtsgespräche. Bekannte handlungsorientierte und lernerzentrierte Aktionsformen zum OOM speziell für Bildungsprozesse, wie die CRC-Kartenmethode oder das Objekt-Rollenspiel, eigenen sich nur für die Gruppenarbeit und setzen zudem fundierte Kenntnisse zu objektorientierten Basiskonzepten voraus. Für Lernerfolgskontrollen oder das Üben bzw. Vertiefen einzelner Aspekte des Modellierungsprozesses sind diese Methoden ungeeignet.
Als Multiplikatoren für die Komponenten dieses fachdidaktischen Verbundes kommen insbesondere Informatiklehrende in Betracht. Praktizierende Lehrende müssen über konzeptorientierte Fortbildungen und Publikationen an das Konzept herangeführt und um Gestaltungs-

rückkopplung gebeten werden. Lehramtsstudierende können im Rahmen von Lehrveranstaltungen das erforderliche Wissen erwerben. Exemplarische Studien mit Lernenden liefern Hinweise zur Akzeptanz und zur unterrichtspraktischen Eignung der Komponenten.

2.4.3 Wissenschaftliche Fragestellungen

Aus den in Abschnitt 2.4.2 beschriebenen Schlussfolgerungen ergeben sich folgende Fragestellungen bzw. Forschungsaufgaben:

1. *Welches sind Komponenten eines didaktischen Systems für OOM?*
   Zur Beantwortung dieser Frage ist ein Bildungskonzept mit Informatikbausteinen zu entwickeln.
2. *Wie kann man bei der Entwicklung didaktischer Systeme vorgehen?*
   Von großer Bedeutung ist hierbei die Entwicklung korrespondierender Elemente in der Fachwissenschaft mit dem Ziel der Ausbildung von Studierenden oder im Beruf stehender Informatikerinnen und Informatiker. Insbesondere in den Bereichen Aufgaben und Software für den Lehr-Lern-Prozess gibt es für diese Zielgruppe eine Reihe von Materialien, aus denen Schlussfolgerungen für die Entwicklung korrespondierender Materialien für den Informatikunterricht in der Sek. II gezogen werden können. Die methodische Vorgehensweise bei der Entwicklung des didaktischen Systems ist prozessbegleitend festzuhalten und zu abstrahieren, um für die Gestaltung weiterer didaktischer Systeme einen Leitfaden bereitzustellen.
3. *Wie kann dieses didaktische System in der Lehrerbildung und im Informatikunterricht der Sek. II exemplarisch erprobt werden?*
   Das didaktische System wird prozessbegleitend erprobt mit dem Ziel der Gestaltungsrückkopplung und Gewinnung von Hinweisen zu dessen Akzeptanz. Hierzu werden begleitend zur Entwicklung Lehrerfortbildungen zum didaktischen System durchgeführt. Weiterhin erfolgt eine Einbeziehung in die Informatiklehrerausbildung[53]. Der Einsatz im Informatikunterricht der Sek. II ist exemplarisch zu erproben.

[53] Bedingt durch den Arbeitsortwechsel des Autors der vorliegenden Arbeit erfolgte dies bis zum 31.10.2002 an der Universität Dortmund, danach an der Universität Siegen.

3 Konzeption eines didaktischen Systems für OOM

3.1 Überblick

Das vorliegende Kapitel gibt einen Überblick über die Konzeption eines didaktischen Systems für OOM. Im Abschnitt 3.2 wird dazu das didaktische System definiert und dessen Zielsetzung präzisiert. Die Systemkomponenten Aufgabenklassen, Explorationsmodule und Wissensstrukturen werden in den Abschnitten 3.3 bis 3.5 konkretisiert und im Abschnitt 3.6 miteinander verknüpft. Die einzelnen Komponenten wurden exemplarisch in verschiedenen Bereichen der informatischen Bildung erprobt. Das den Erprobungen im Bereich der Informatiklehrerbildung zugrunde liegende Konzept wird im Abschnitt 3.7 vorgestellt. Im Abschnitt 3.8 erfolgt abschließend eine Abgrenzung des im Rahmen dieser Arbeit entwickelten Konzeptes von anderen Ansätzen zum objektorientierten Modellieren im Informatikunterricht.
Eine umfassende Darstellung zur Entwicklung und zur Erprobung der Systemkomponenten Aufgabenklassen, Explorationsmodule und Wissensstrukturen folgt in den Kapiteln 4 bis 6.

3.2 Zielsetzung und Definition

Fehlende Standards für die informatische Bildung

Für die informatische Bildung in Schulen existieren seit langem zahlreiche informatikdidaktische Publikationen zu anspruchsvollen Unterrichtskonzepten (z.B. 2.3.1) sowie wohlbegründete Gestaltungsempfehlungen (z.B. Brauer et al. 1976, Schulz-Zander et al. 1993, Kommission Informatik 2000[54], Breier et al. 2000b). Da es sich hierbei um, wenngleich von renommierten Informatikdidaktikern erarbeiteten und der GI herausgegebenen, Gestaltungs*empfehlungen* handelt, sind diese für die Neukonzeption bzw. Weiterentwicklung von Informatikrichtlinien und -lehrplänen der Bundesländer nicht bindend. Während sich beispielsweise die Informatikrichtlinien und -lehrpläne von Nordrhein-Westfalen (MSWF 1999) und Schleswig-Holstein (MBWFK 2002) nicht an den Leitlinien des GI-Gesamtkonzeptes für die informatische Bildung in Schulen (vgl. GIFA 1999; Breier et al. 2000a, 2000b) orientieren, ist z.B. in den Bundesländern Thüringen (TK 1999), Bremen (SBW 2001), Mecklenburg-Vorpommern (MBWK 2001), Hessen (HK 2003), Sachsen-Anhalt (KSA 2003) und Hamburg (HHBBS 2003) eine mehr oder weniger ausgeprägte Orientierung festzustellen. Die Folge ist eine große Bandbreite im Lernerfolg der Schülerinnen und Schüler. Es lässt sich ein Trend zu international angelegten, vergleichenden Analysen zu den Schulleistungen von Lernenden (vgl. PISA-Studie: OECD 2001[55]) feststellen, in denen die Informatik bislang noch nicht berücksichtigt wurde, obwohl sie weltweit in vielen Ländern als eigenes Unterrichtsfach in der Sekundarstufe angeboten wird (für den europäischen Raum: Eurydice 2001, 23f). Im Bereich der informatischen Bildung fehlt es noch an anerkannten und empirisch überprüften Bildungsstandards. Mit der Konzeption und Entwicklung einer Sammlung aufeinander abgestimmter Lehr-Lern-Materialien zum objektorientierten Modellieren sowie der Begründung von Vorgehensweisen zu deren Erstellung und Erprobung (vgl. 2.4.2 und 2.4.3) soll auch ein Beitrag zur Vorbereitung von Bildungsstandards für die Informatik geleistet werden.

Motivation für Komponenten des didaktischen Systems

Dargelegt wurde in Bezug auf das Themenfeld OOM im Informatikunterricht ein Bedarf an unterrichtsgeeigneten Aufgaben, Informatiksystemen zur Unterstützung des Aneignungspro-

[54] vgl. Fußnote 22 auf S. 13

[55] Die Ergebnisse wurden in der öffentlichen Diskussion in Deutschland als sehr alarmierend bewertet, da die gemessenen Schulleistungen in den Bereichen Lesekompetenz und mathematisch-naturwissenschaftliche Grundbildung jeweils unter dem Durchschnitt aller teilnehmenden Länder lagen.

zesses, Strukturierungsempfehlungen für die Lehr-Lern-Prozesse und damit verbundene Einsatzkonzepte (vgl. 2.4.2). Da sich ähnliche Bedarfssituationen auch in anderen Bereichen der informatischen Bildung identifizieren lassen, z.B. bei „Wirkprinzipien von Informatiksystemen“, werden im Rahmen der vorliegenden Arbeit Vorgehensweisen und Konzepte zur Bewältigung des mit dem identifizierten Bedarf verbundenen Metaproblems entwickelt, angewandt und die Ergebnisse in der informatischen Bildung exemplarisch erprobt.

- *Bedarf an unterrichtsgeeigneten Aufgaben zum OOM*
  → ***Aufgabenklassen-Komponente*** *des didaktischen Systems*
  Um dem Mangel an unterrichtsgeeigneten Aufgaben zum OOM zu begegnen, werden Vorgehensweisen zur Entwicklung und Erprobung einer strukturierten Sammlung von Aufgabenklassen (Gestaltungsmittel für konkrete Aufgaben) für den Unterricht entwickelt, begründet und im Themenfeld OOM angewandt. Begründet wird, wie sich mittels Aufgabenklassen konkrete Aufgaben für unterschiedliche Niveaustufungen gestalten lassen.
- *Bedarf an Informatiksystemen zur Unterstützung des Aneignungsprozesses*
  → ***Explorationsmodul-Komponente*** *des didaktischen Systems*
  Entwickelt, begründet und bei der Implementierung von Prototypen angewandt werden Vorgehensweisen zur Gestaltung von Software zur Exploration von informatischen Fachkonzepten (so genannte Explorationsmodule). Ein Konzept für das Lernen mit Explorationsmodulen wird entwickelt, begründet und exemplarisch in der informatischen Bildung erprobt.
- *Bedarf an Strukturierungsempfehlungen für Lehr-Lern-Prozesse*
  → ***Wissensstrukturen-Komponente*** *des didaktischen Systems*
  Begründet und exemplarisch erprobt werden graphbasierte Darstellungsmittel zur Repräsentation der Struktur von Lehr-Lern-Prozessen mit dem Ziel, Strukturierungserfahrungen besser explizieren, kommunizieren, analysieren und diskutieren zu können.

Die genannten Komponenten werden im Weiteren verknüpft zu einem didaktischen System.

Didaktisches System

Informatikdidaktisch begründeter, offener Verbund von Komponenten des Lehr-Lern-Prozesses

Unter einem *didaktischen System* wird im Rahmen der vorliegenden Arbeit ein informatikdidaktisch begründeter, offener Verbund bestehend aus

- traditionellen Komponenten des Lehr-Lern-Prozesses, wie z.B. *Aufgabenklassen* (vgl. 3.3) und *Explorationsmodulen* (vgl. 3.4), und
- neuen, noch ungewohnten Komponenten, wie z.B. Graphen für die Repräsentation fachlicher und fachdidaktischer Erarbeitungsstrukturen (hier als *Wissensstrukturen* bezeichnet, vgl. 3.5)

verstanden (vgl. Brinda 2000a, Brinda / Schubert 2001). Da

- neue Erkenntnisse, z.B. aus
  - der Pädagogik,
  - der Didaktik der Informatik,
  - der Praxis des Informatikunterrichts und bzw. oder
  - der Fachwissenschaft Informatik,

verknüpft mit

- Gestaltungserfordernissen

- o für andere Themenbereiche innerhalb der Informatik oder auch darüber hinaus und bzw. oder
- o andere Zielgruppen als Lernende im Informatikunterricht der Sek. II

dazu führen (können), dass die hier dargestellten Systemkomponenten verfeinert, spezialisiert und bzw. oder neue Komponenten aufgrund der Erkenntnisse eingeführt und mit den vorhandenen in Relation gesetzt werden müssen, wird das didaktische System explizit als *offener* (d.h. erweiterbarer) Verbund konzipiert. In Analogie zur Software-Entwicklung bietet sich eine kooperative Anwendung und Weiterentwicklung solcher Systemkerne an, da ihre Attraktivität mit dem Variantenreichtum wächst.

Verknüpfung der Komponenten mit Vorgehensweisen ihrer Entwicklung und Erprobung

Für jede Komponente eines didaktischen Systems werden verknüpft (vgl. 2.4.3):

- konkrete Ausprägungen der Komponenten,
- Vorgehensweisen der Entwicklung,
- Vorgehensweisen und Ergebnisse der Erprobung.

Ziel hierbei ist es, die Übertragbarkeit auf andere Themenbereiche oder andere Zielgruppen bestmöglich zu unterstützen.

Didaktische Funktionen der Komponenten

Durch die Verknüpfung der Komponenten mit Vorgehensweisen ihrer Entwicklung und Erprobung erfüllen diese jeweils insbesondere die folgenden, didaktischen Funktionen:

- Gestaltungsmittel für Lehr-Lern-Prozesse,
- Anwendung in Lehr-Lern-Prozessen,
- Förderung der fachdidaktischen Kommunikation und Diskussion zu Lehr-Lern-Prozessen.

Diese Zusammenhänge werden in den Abschnitten 3.3.3, 3.4.3 und 3.5.3 für die Komponenten Aufgabenklassen, Explorationsmodule und Wissensstrukturen konkretisiert.

Verknüpfung der didaktischen Funktionen der Komponenten mit wesentlichen Akteuren in Lehr-Lern-Prozessen

Die didaktischen Funktionen adressieren in unterschiedlicher Weise wesentliche Akteure in Lehr-Lern-Prozessen (vgl. Tabelle 6).

Didaktische Funktion	**Lernende**	**Lehrende / Lehramts-studierende**	**Entwickler / innen**
1. Gestaltungsmittel für Lehr-Lern-Prozesse	●	●●	●
2. Anwendung in Lehr-Lern-Prozessen	●●	●	–
3. Förderung der fachdidaktischen Kommunikation und Diskussion zu Lehr-Lern-Prozessen	–	●●	●●

Tabelle 6: Didaktische Funktionen der Komponenten des didaktischen Systems im Hinblick auf wesentliche Akteure in Lehr-Lern-Prozessen

Eine größere Anzahl von Punkten in einer Tabellenzelle markiert dabei eine stärkere Verknüpfung zwischen der didaktischen Funktion und der jeweiligen Gruppe von Akteuren.

Nachfolgend werden die Systemkomponenten Aufgabenklassen, Explorationsmodule und Wissensstrukturen präzisiert, die Vorgehensweisen ihrer Entwicklung und Erprobung skizziert und die didaktischen Funktionen im Hinblick auf die jeweilige Komponente konkretisiert. Eine ausführliche Darstellung zu den Komponenten erfolgt in den Kapiteln 4, 5 und 6 der vorliegenden Arbeit.

3.3 Aufgabenklassen

3.3.1 Didaktische Funktionen von Aufgaben

Zur Gestaltung eines für die Lernenden erfolgreichen Lehr-Lern-Prozesses sind an verschiedenen Stellen geeignete Aufgaben erforderlich. Der Aufgabenbegriff wird sehr unterschiedlich verwendet. Einerseits wird darunter das Unterrichtsthema als „die vom Lehrer vorgegebene oder zwischen dem Lehrer und den Schülern vereinbarte, für einen begrenzten Zeitraum gestellte Lernaufgabe“ verstanden (Meyer 1994, 83). Andererseits spielen im Unterricht Arbeitsaufträge, die auch als Aufgaben interpretiert werden können, eine zentrale Rolle. Im Zusammenhang mit problemorientiertem Unterricht werden Probleme als nicht routinemäßig lösbare Aufgaben verstanden. Arbeitsaufträge werden häufig in Form von Arbeitsblättern dargeboten. Darunter wird „ein didaktisch strukturierter, schriftlich, rechnerisch oder bildnerisch zu lösender Arbeitsauftrag“ verstanden (Meyer 1987, 307). Je nachdem, an welcher Stelle sie im Unterrichtsprozess eingesetzt werden, können Arbeitsblätter (und damit die auf ihnen enthaltenen Aufgaben) unterschiedliche didaktische Funktionen erfüllen (ebd., 308):

- zu Beginn der Beschäftigung mit neuem Thema: erstes *Problembewusstsein* schaffen,
- in Verarbeitungsphasen: Lernende *arbeiten* individuell oder partnerschaftlich den neuen *Lehrstoff auf,*
- in der Mitte / am Ende eines Unterrichtsabschnitts: *Vertiefung*, *Übung*[56], vorstrukturierte Hausaufgabe.

Im Hinblick auf Aufgaben lassen sich zwei Funktionen ergänzen:

- *Beispiel* (z.B. Einführungsbeispiel, Anwendungsbeispiel),
- *Leistungsüberprüfung* (z.B. Klausuraufgabe).

Die Arbeitsaufträge lassen sich differenzieren in:

- *geschlossene Arbeitsaufträge*, bei denen Weg und Ziel vorstrukturiert sind,
- *offene Arbeitsaufträge*, bei denen das Ziel vorgegeben, der Weg aber von den Lernenden selbstständig festgelegt werden muss, und
- *freie Arbeitsaufträge*, bei denen die Lernenden Ziel und Weg selbstständig wählen.

Orthogonal dazu ist die Unterscheidung von themendifferenzierten und themengleichen Arbeitsaufträgen (ebd., 257). Der Lernerfolg hängt ganz wesentlich von der Wahl der richtigen Arbeitsaufträge bzw. Aufgaben ab.

[56] Meyer (1987, 167ff) betont, dass es in der Schule immer schwieriger werde, sinnvoll zu üben. Wichtige Probleme seien ein Widerspruch zwischen Stofffülle und zur Verfügung stehender Unterrichtszeit, die Verknüpfung von Übungen mit einer Leistungsbewertung durch den Lehrenden sowie zunehmende Konzentrations- und Motivationsschwierigkeiten der Lernenden. Andererseits stellt er aber auch fest, dass das Üben nicht zwangsläufig die Einsicht in den Gesamtzusammenhang bzw. die Konzentration und Motivation der Lernenden voraussetzt. Vielmehr könnten Übungsphasen auf beide Aspekte positiv wirken.

3.3.2 Entwicklung und Erprobung von Aufgabenklassen und Aufgaben

Im Abschnitt 2.4.2 wurde im Punkt 2 ein Mangel an interessanten und abwechslungsreichen Aufgaben zum objektorientierten Modellieren im Informatikunterricht festgestellt, die sich dazu eignen, Problembewusstsein zu schaffen, Lehrstoff aufzuarbeiten oder Fachkonzepte zu üben bzw. zu vertiefen, etc. Aufgrund dieser Mangelsituation wurden vom Autor der vorliegenden Arbeit einführende, fachwissenschaftliche Lehrbücher zum objektorientierten Modellieren (Standardwerke), an die die Anforderung gestellt wurde, dass sie Aufgaben enthalten müssen, ausgewählt (Rumbaugh et al. 1993, Balzert 1999) und auf unterrichtsgeeignete Aufgabenstellungen hin analysiert. Da die in diesen Lehrbüchern enthaltenen Aufgaben andere Zielgruppen adressierten als Lernende in der Sekundarstufe II, wurden Kriterien zu deren Auswahl bzw. Transformation für den Informatikunterricht entwickelt (vgl. 4.3.1) und angewandt (vgl. Anhang B). Die so ausgewählten Aufgaben wurden anschließend einem Abstraktionsprozess unterzogen, bei dem insbesondere eine Trennung von konkreten Bezeichnern, Erläuterungstexten und Kontexten erfolgte. Beispielsweise stellt in einer Aufgabe, in der es darum geht, zu einer textuellen Beschreibung einer Schulbibliothek ein Klassendiagramm zu entwickeln, die Schulbibliothek den Kontext dar. Durch Abstraktion von diesem Kontext wird die zugrunde liegende Aufgabenklasse sichtbar, die aus einer Menge von Angaben (z.B. Texte, Diagramme) und mindestens einem sich darauf beziehenden Arbeitsauftrag (z.B. Klassendiagramm erstellen) besteht (vgl. 4.3.2). Die so identifizierten Aufgabenklassen wurden fachlich und fachdidaktisch klassifiziert und auf der Grundlage dieser Klassifikation eine strukturierte Sammlung von Aufgabenklassen in erster und zweiter Fassung entwickelt (vgl. Abschnitt 4.3.3, Anhänge C.1 und C.2). Die zweite Fassung (C.2) berücksichtigt Erkenntnisse aus einer Erprobung von Aufgabenklassen im Informatikunterricht der Jgst. 11 und 12 (vgl. 4.5) sowie der Informatiklehreraus- und -weiterbildung (vgl. 4.6).
Parallel zu dieser Entwicklung entstand ein Konzept für die Gestaltung von Aufgaben auf der Basis von Aufgabenklassen (vgl. 4.4). Dazu wurden die ausgewählten Aufgaben (Anhang B) einer eingehenden Strukturanalyse unterzogen und Merkmale zur Gestaltung von Niveaustufungen identifiziert (vgl. 4.4.2). Des Weiteren wurden die von den ausgewählten Aufgaben im Rahmen des Abstraktionsprozesses (vgl. 4.3.2) getrennten Kontexte gesammelt, zu Kontextklassen abstrahiert und Kriterien für die Auswahl von Kontexten bei der Gestaltung von Aufgaben begründet (vgl. 4.4.3).
Eine zusammenfassende Übersicht über die im Rahmen dieser Arbeit begründete Vorgehensweise bei der Entwicklung und Erprobung von Aufgabenklassen und Aufgaben zum OOM zeigt Abbildung 10 auf S. 107.

3.3.3 Didaktische Funktionen von Aufgabenklassen

Gestaltungsmittel für Lehr-Lern-Prozesse

Für Lehrende dienen Aufgabenklassen im Wesentlichen beim Entwurf von Lehr-Lern-Prozessen als Gestaltungsmittel für Aufgaben zur Erfüllung verschiedener didaktischer Funktionen (vgl. 3.3.1). Der größte Gewinn wird hierbei für Lehramtsstudierende und im hier betrachteten Thema objektorientiertes Modellieren wenig erfahrene Lehrende erwartet (vgl. auch 4.2).
Da die Mehrzahl der Aufgabenklassen zu einer größeren Anzahl an erprobten Aufgaben korrespondiert, bündeln sie Bildungserfahrungen zu einem kleinen Bereich der Informatik und machen diese zur Wiederverwendung verfügbar. Durch die Berücksichtigung von in Aufgabenklassen enthaltenen Bildungserfahrungen bei der Gestaltung von Unterricht erhalten Lernende eine implizite Zusage auf wahrscheinlichen Bildungserfolg. In anderen Unterrichtsfächern wurden bereits viele Generationen mit wiederkehrenden Aufgabenklassen erfolgreich ausgebildet, so dass das dort in den jeweiligen Aufgabenklassen liegende, heuristische Wissen

bereits viel zuverlässiger ist als in der informatischen Bildung, in der noch kein Konsens zu den Aufgabenklassen existiert, die in einer Jahrgangstufe verpflichtend bzw. ergänzend sind.
Durch die Einordnung der Aufgabenklassen in eine strukturierte Sammlung (vgl. 4.3.3), die fachliche und fachdidaktische Ordnungskriterien berücksichtigt, sowie durch die Entwicklung einer Methodik zur Gestaltung von Aufgaben aus Aufgabenklassen für unterschiedliche Niveaustufungen (vgl. 4.4), wird die Entwicklung von lernerangemessenen Aufgabenstellungen, auch im Hinblick auf die geringe, zur Verfügung stehende, Bildungszeit, erleichtert. Das trägt für Lehrende zur Komplexitätsreduzierung im Hinblick auf die Unterrichtsgestaltung bei. Die Gefahr, dass Lernende an komplexeren Aufgabenklassen scheitern, weil wichtige Basisfähigkeiten übersehen wurden, nimmt ab. Informatiklehramtsstudierende an der Universität Dortmund wurden dazu zwischen 2000 und 2002 bereits im Rahmen ihrer universitären Informatikdidaktikausbildung sowohl in die Identifikation von Aufgabenklassen aus Aufgaben (gestaltende Rolle) als auch in die Gestaltung von Aufgaben zu Aufgabenklassen (anwendende Rolle) mit einbezogen (vgl. 4.5, 4.6).
Durch die Bereitstellung und Verteilung einer strukturierten Sammlung von Aufgabenklassen sollen ferner Lehrenden und Lehramtsstudierenden Anregungen für die Erprobung neuer Aufgabenstellungen im Unterricht gegeben werden. Dies fördert einerseits den Variantenreichtum, andererseits können Lehrende dazu ermutigt werden, durch sie bislang unerprobte Aufgabenstellungen in ihren Unterricht einzubeziehen.

Anwendung in Lehr-Lern-Prozessen

Die Lernenden sollen den Abstraktionsprozess von konkreten Aufgaben zu einer Aufgabenklasse nicht nur nachvollziehen, sondern selbst organisieren können. Diese Fähigkeit hilft ihnen bei der Bewältigung von kognitiven Problemen mit einer Lösungsidee. Durch die unterrichtliche Zuordnung von Aufgaben zu Aufgabenklassen anhand der Identifikation von Merkmalen bereits gelöster Aufgaben und durch die Verknüpfung von Aufgabenklassen mit im Unterricht erarbeiteten Lösungsstrategien wird von den Lernenden selbstständig eine Lernhilfe für die Bearbeitung ähnlicher Aufgaben gestaltet. Dies fördert ihre Selbstständigkeit (vgl. Meyer 1994, 151). Dazu abstrahieren die Lernenden die in der jeweiligen Aufgabe gegebenen Angaben zu Daten, verknüpfen diese mit dem bekannten Strukturkonzept und lösen die Aufgabe durch Analogieschluss, indem sie zuvor erarbeitete Lösungen wieder entdecken. Diese Fähigkeit hilft ihnen, in der Fülle von Einzelheiten zum Wesentlichen vorzudringen.

Förderung der fachdidaktischen Kommunikation und Diskussion über Lehr-Lern-Prozesse

Die Bereitstellung von Aufgabenklassen kann die fachdidaktische Kommunikation und Diskussion fördern und anregen. Unterstützt werden kann dies durch geeignete Visualisierungshilfsmittel, wie z.B. dargestellt in Abbildung 9 auf S. 79. Informatiklehrende sowie andere, an der Entwicklung von Informatikdidaktik beteiligte Personen, können damit Aufgabenstellungen erörtern, Varianten entwickeln, Teilaufgaben verknüpfen und Lösungswege strukturieren.

3.4 Explorationsmodule

3.4.1 Grundprinzipien des OOM mit Lern-Software entdecken

Die Analyse von publizierten Unterrichtsbeispielen zur Objektorientierung zeigte, dass Handlungsorientierung und informatisches Modellieren in der Sek. II noch zu stark mit einer Programmiersprache verknüpft sind (vgl. 2.3.1.2). Um erfolgreich modellieren und eigene Lösungen gestalten und bewerten zu können, müssen die Lernenden viele programmiersprachliche Details erlernen und komplexe Software-Entwicklungsumgebungen anwenden. Diese Zeit

geht verloren für so wichtige Themen, wie Wirkprinzipien von Informatiksystemen. Die gewünschte Bildungsqualität wird durch diese Art von Medieneinsatz nur auf Umwegen oder überhaupt nicht erreicht. Guter und für Lernende interessanter Informatikunterricht ist möglich ohne diese enge Bindung an eine spezielle Implementierungssprache, wenn im Unterricht Analyse und Entwurf von Lösungen stärker betont und durch lernergerechte Informatiksysteme unterstützt werden. Im Rahmen dieser Arbeit werden solche Werkzeuge konzipiert. Ziel ist dabei nicht die Gestaltung von didaktisch reduzierten Versionen von Software-Entwicklungsumgebungen oder von E-Learning-Materialien in Form von multimedialen Präsentationen, durch die Lernende ziellos hindurch navigieren können, sondern die Entwicklung von Software zur Anregung des entdeckenden Lernens objektorientierter Fachkonzepte, im Rahmen dieser Arbeit als *Explorationsmodule* bezeichnet. Diese Software wird vollständig vom Lernenden gesteuert. Erforderlich für die zielgerichtete Interaktion mit dem Lernangebot ist ein Explorationswunsch oder eine fachlich fundierte Hypothese zum Lernmaterial. Der Explorationswunsch oder ein Erkundungsziel führen zur Erforschung des Systems. Bei der Erkundung bilden die Lernenden Hypothesen über Zusammenhänge, die sie anhand von Experimenten mit der Software bestätigen oder widerlegen. Ungewünschte Zielzustände führen zur Modifikation von Hypothesen (vgl. 5.3.1). Keinesfalls ist es das Ziel, solche Handlungsszenarios zu programmieren. Zur Begleitung und zur Anregung der gewünschten fachlichen Entdeckungen benötigen Lernende aber geeignete Lernhilfen. Durch vielfältige Interaktionsmöglichkeiten (vgl. Kerres 2001) und Visualisierung der Handlungskonsequenzen in verschiedenen Sichten (vgl. Anderson 2001, 267) auf den Explorationsgegenstand werden sie zum aktiven Lernen angeregt. Im Sinne des konstruktivistischen Lernens (vgl. Vygotsky 1978) sollen die Lernenden durch die Interaktion mit der Software dazu stimuliert werden, mentale Modelle der objektorientierten Gestaltung von Software zu konstruieren und schrittweise zu verfeinern (vgl. Kerres 2001, 74ff). Der besondere Wert solcher Lernangebote in der Anfangsphase begründet sich ferner darin, dass die Lernenden zu diesem Zeitpunkt ihrer Informatikausbildung noch über keine eigene Entwurfskompetenz verfügen. Mit dem Konzept der Explorationsmodule werden handlungsorientierte Zugänge zu Wissensbereichen eröffnet, in denen sich das konstruktivistische Lernen bislang zu stark auf die Programmierung konzentrierte.

3.4.2 Entwicklung und Erprobung von Explorationsmodulen

Im Punkt 3 von Abschnitt 2.4.2 wurde festgestellt, dass es an unterrichtsgeeigneten Informatiksystemen zur Unterstützung des Erlernens objektorientierter Basiskonzepte fehlt. Um handlungsorientierten Unterricht zu ermöglichen, kommen oft bereits frühzeitig professionelle Software-Entwicklungsumgebungen zum Einsatz, die weder für die Zielgruppe der Lernenden im Informatikunterricht der Sek. II noch für die Zielsetzung der Unterstützung der Aneignung objektorientierter Fachkonzepte konzipiert wurden (vgl. 5.3.2). Dies ist oft zu beobachten bei Unterricht, der das Erlernen einer Programmiersprache in den Mittelpunkt rückt.
Zur Förderung der Handlungsorientierung im Informatikunterricht zum informatischen Modellieren bei Fokussierung der zentralen informatischen Strukturierungskonzepte wurde ein auf explorativem Lernen mit Informatiksystemen basierendes Konzept für die Gestaltung und die unterrichtliche Einbeziehung von Software zur Förderung der Exploration von objektorientierten Fachkonzepten entwickelt. Dazu wurden zunächst Formen, Grundmuster, Charakteristika sowie Strategien des explorativen Lernens analysiert (vgl. 5.3.1.1). Zur Entwicklung eines Gestaltungskonzepts für Explorationsmodule wurden darauf aufbauend Ergebnisse zum explorativen Lernen mit Informatiksystemen ausgewertet (vgl. 5.3.1.2). In diesen Kontext sind erste eigene Arbeiten mit einem Diplomanden zur Übertragung des Experimentbegriffs des Bildungsbereiches aus den Naturwissenschaften in die Informatik sowie die Analyse von Erkenntnissen zur Gestaltung explorationsfreundlicher Anwendungssoftware und zur

Gestaltung von explorationsfreundlichen Lernangeboten einzuordnen. Den zweiten wesentlichen Grundbaustein zur Entwicklung eines Gestaltungskonzepts für Explorationsmodule bildete eine fachdidaktische Analyse von ausgewählten professionellen Werkzeugen für objektorientiertes Modellieren im Hinblick auf Unterrichtseignung, erkenntnisfördernde Funktionen sowie potentielle Erkenntnisbarrieren (vgl. 5.3.2). Auf dieser Basis wurde ein Gestaltungskonzept für Explorationsmodule zum objektorientierten Modellieren entwickelt und begründet. Dieses betont vielfältige, interaktive und synchronisierte Sichten auf den Explorationsgegenstand (vgl. 5.3.3). Verzahnt mit der Entwicklung des Gestaltungskonzepts entstanden verschiedene Prototypen, die Konzeptausschnitte implementieren und zu dessen Verfeinerung und Weiterentwicklung beitrugen (vgl. 5.3.4). Im Rahmen einer vom Autor betreuten Diplomarbeit wurde ein Architekturkonzept abgeleitet zur Vereinfachung der Gestaltung von Software zur Exploration im Bildungskontext (vgl. 5.3.5).
Ebenfalls verzahnt mit Konzeption und Gestaltung der Explorationsmodule wurde ein Konzept für das Lernen mit Explorationsmodulen im Informatikunterricht entworfen (vgl. 5.4), da die Qualität von Unterrichtsmedien grundsätzlich nur im Zusammenhang mit Lehr-Lern-Prozessen bewertet werden kann (Kerres 2001, 23). Elemente dieses Lernkonzepts wurden exemplarisch im Informatikunterricht, in Studienwerbeveranstaltungen, in der Informatiklehreraus- und -weiterbildung sowie im Informatikstudium erprobt. Es zeigte sich eine breite Akzeptanz (vgl. 5.5).
Eine zusammenfassende Übersicht über die im Rahmen dieser Arbeit begründete Vorgehensweise bei der Entwicklung und Erprobung von Explorationsmodulen zum objektorientierten Modellieren zeigt Abbildung 22 auf S. 169.

3.4.3 Didaktische Funktionen von Explorationsmodulen

Gestaltungsmittel für Lehr-Lern-Prozesse

Explorationsmodule stellen Unterrichtsmittel zur Ausgestaltung von explorativen, handlungsorientierten Lernphasen im Informatikunterricht dar (vgl. 5.2). Exploratives Lernen erfordert vom Lehrenden eine besonders intensive Unterrichtsvorbereitung, da individuelle Entdeckungsprozesse der Lernenden mit vielfältigen potenziellen unterrichtlichen Verlaufsstrukturen korrespondieren (vgl. Kerres 2001, 219ff). Insbesondere in offenen Explorationsumgebungen (z.B. Internet, vgl. 5.4.2) ist damit ein hoher Vorbereitungsaufwand für den Lehrenden verbunden. Exploratives Lernen führt aber nur dann zuverlässig zu Lernerfolgen, wenn neben einem definierten fachlichen Fundament und Lernmotivation auch erste Erfahrungen mit selbstorganisiertem Lernen vorliegen (vgl. 5.3.1.1). Deshalb werden für den Informatikanfangsunterricht geführte Explorationsprozesse in geschlossenen Explorationsumgebungen empfohlen. Die Führung kann einerseits erfolgen durch Leitfragen (vgl. 5.4.2). Andererseits kann durch die Auswahl und Kombination von Explorationsmodulen für den Unterricht (Exploration in geschlossenen Explorationsumgebungen) das Erkenntnisinteresse der Lernenden auf die für den jeweiligen Lernprozess relevanten Aspekte gelenkt werden. Damit wird Lernenden ein Handlungsrahmen vorgegeben. Dies ist Vor- und Nachteil zugleich. Der Lehrende wird entlastet, andererseits sind nur die Entdeckungen möglich, die vom Entwickler der Software vorgesehen wurden.

Anwendung in Lehr-Lern-Prozessen

Durch die doppelte Führung (s.o.) können Lernende an die fachlichen Zusammenhänge und an die Methodik explorativen Lernens herangeführt werden. Wenn sie entsprechend lernkompetent sind, ihre individuellen Lernprozesse zielorientiert zu organisieren, können geschlossene Explorationsumgebungen schrittweise geöffnet und von einer weiteren Vorstrukturierung des Explorationsprozesses durch Leitfragen des Lehrenden abgesehen werden.

Explorationsmodule stellen den zu explorierenden Fachgegenstand in vielfältigen, interaktiven und synchronisierten Sichten dar. Die Steuerung erfolgt vollständig durch den jeweiligen Lernenden. Ziele und bzw. oder Wege des Lernens sind nicht von außen vorgegeben, Leitfragen dienen lediglich als Orientierungshilfen. Diese Freiheitsgrade stellen eine notwendige Voraussetzung für exploratives Lernen dar (Kerres 2001, 217). Exploratives Lernen wird als aktives, forschendes und lernerzentriertes Lernen charakterisiert (vgl. 5.3.1.1). Interaktionen eines Lernenden mit einer Sicht eines Explorationsmoduls führen zu Zustandsänderungen aller davon abhängigen Sichten. Der Lernende wird damit dazu angeregt, Hypothesen über den Lerngegenstand zu bilden und diese in Experimenten (Interaktionen mit dem System) zu überprüfen (Paul 1994). Der informatische Fachgegenstand wird in den Mittelpunkt gerückt. Das Erlernen für einen Erkenntnisprozess sekundärer Details kann so vermieden werden (Steinkamp 1999). Die Exploration erfolgt vollständig eingebettet in traditionelle Lehr-Lern-Prozesse (vgl. 5.4). *Keinesfalls ist es das Ziel, diese zu ersetzen*. Lernen mit Explorationsmodulen stellt eine interessante Zugangsalternative zu fachlichen Inhalten in geeigneten Phasen des Lehr-Lern-Prozesses dar.

Förderung der fachdidaktischen Kommunikation und Diskussion zu Lehr-Lern-Prozessen

Durch die Verknüpfung von Explorationsmodulen mit Erprobungsergebnissen kann die fachdidaktische Kommunikation und Diskussion angeregt werden, indem daraus beispielsweise Ideen für neue Explorationsmodule zum objektorientierten Modellieren oder anderen Themen der Informatik abgeleitet und diskutiert werden können.

3.5 Wissensstrukturen

3.5.1 Repräsentation von Wissensstrukturen

Lehrbücher, Kurse, Unterrichtsentwürfe etc. enthalten eine Menge an implizitem heuristischen fachdidaktischen Wissen zur Gestaltung von Lehr-Lern-Prozessen, das sich entweder in der Lehrerfahrung der Autoren zum jeweiligen Gegenstand manifestiert oder durch diese per Analogieschluss aus anderen Teilgebieten und aufgrund fachdidaktischer Analyse des Gegenstands übertragen wurde. Da im Unterricht die Entwicklung von Kompetenzen im Vordergrund steht, ist die sachlogische Struktur des Fachgegenstands allein ungeeignet zur Strukturierung des Unterrichts (vgl. Kerres 2001, 148ff). Aufgrund der Struktur der Fachkonzepte wird aber deutlich, welche Elemente als Vorkenntnisse für andere erforderlich oder hilfreich sind. Dieses fachdidaktische Wissen bleibt in der Regel verborgen, sofern es nicht von den jeweiligen Autoren expliziert wird, wie dies z.B. bei der Begründung von Unterrichtsentwürfen der Fall ist. Unterrichtsentwürfe werden aber in der Regel nicht veröffentlicht, sondern sind interne Dokumente zur Dokumentation für den Lehrenden oder Diskussionsgrundlage für Unterrichtshospitationen. Das enthaltene fachdidaktische Wissen bezieht sich beispielsweise auf erfolgreiche Erarbeitungsreihenfolgen von Fachkonzepten, auf Unterrichtsmethoden, auf Lernschwierigkeiten etc. und stellt damit Wissen über das Wissen, so genanntes Metawissen dar. Fachdidaktische Publikationen analysieren fachspezifische Lehr-Lern-Prozesse in ihren Erfolgen und Problemen (vgl. 2.3). Wissen zur Sequenzierung von Fachkonzepten wird beispielsweise dazu verwendet, Lernpfade durch E-Learning-Module automatisch zu generieren (vgl. 6.2.4). Dieses Wissen bleibt in der Regel auch in solchen Lernangeboten für die Anwenderin oder den Anwender verborgen, sofern nicht ein entsprechender Zugang zu solchen Entscheidungsstrukturen gegeben ist. Technisch realisiert werden solche Strukturen beispielsweise durch die Spezifikation von elektronischen Lerneinheiten mittels formalisierten Metadaten, z.B. Learning Object Metadata – LOM (vgl. IEEE 2002). Solche Metadaten fördern die au-

tomatische Anordnung und die Weiterverwendung von Lerneinheiten in anderen Kontexten. Durch die maschinelle Repräsentation lässt sich aus diesem formalisierten Wissen neues Wissen mittels Informatiksystemen ableiten. Repräsentationsformen, die eine solche automatische Auswertung erlauben, sind aber nicht zwangsläufig gleichermaßen geeignet für fachdidaktische Analysen und Diskussionsprozesse. Das Ziel ist es hier, durch die Bereitstellung einer Darstellungsform zur Diskussion solcher Strukturen von Fach- und Fachdidaktikwissen einen Beitrag zur Verbesserung der Analysemöglichkeiten von Bildungsprozessen zu leisten. Eine solche Repräsentation fachdidaktischen Wissens erleichtert die Kommunikation zum Stand der Didaktik der Informatik und zu den Bildungsergebnissen und fördert deren Vergleichbarkeit.
Zur Explikation und zur Strukturierung des informatikdidaktischen Fachwissens wurden verschiedene Wissensrepräsentationsformen aus Informatik und Mathematik anhand von zu diesem Zweck entwickelten Kriterien analysiert (vgl. 6.4.2). Einige dieser Formalisierungen eignen sich hervorragend für die Dokumentation und Präzisierung menschlicher Vorgehensweisen. Analysiert wurden im Einzelnen Begriffsnetze (concept maps), Semantische Netze, Objekt- und Klassendiagramme sowie Und-Oder-Graphen. Prinzipiell eignen sich diese alle, um das genannte fachdidaktische Wissen zu modellieren. Verschiedene Fachkonzepte lassen sich mit einer *ist-erforderlich-für-* bzw. einer *ist-hilfreich-für-Relation* verknüpfen. Da es alternative Möglichkeiten gibt, resultiert daraus eine Vielzahl von Varianten, Wissen und Können schrittweise zu entwickeln. Probleme können dabei z.B. auftreten, wenn es in der Struktur zu Sprüngen im Abstraktionsniveau kommt, wie z.B. bei der Konstruktion von Klassenhierarchien zu in Aufgabenstellungen beschriebenen Realitätsausschnitten zu beobachten ist (vgl. 6.2.3). Begriffsnetze ermöglichen zwar beliebige Beziehungen, die boolesche Verknüpfung ist aber nicht bzw. nur auf Umwegen darstellbar. Semantische Netze eröffnen den größten Darstellungsspielraum, was allerdings stark zu Lasten der Übersichtlichkeit geht. Die Erweiterung von Und-Oder-Bäumen zu gerichteten, azyklischen Und-Oder-Graphen stellt den besten Kompromiss bzgl. der gegebenen Anforderungen dar (vgl. Brinda 2000a). Darin repräsentieren die Knoten Lerneinheiten (elementare oder komplexe Fachkonzepte oder -methoden). Dessen Kanten repräsentieren Vorkenntnisstrukturen (ist-erforderlich-für- bzw. ist-hilfreich-für-Relation), anhand derer ersichtlich wird, wie komplizierte Informatikkonzepte auf einfachen aufbauen. Diese Strukturierung lässt die Schwerpunkte und Etappen des Lehr-Lern-Prozesses erkennen. Obligatorische Bestandteile, die Lernende verstanden haben müssen, bevor sie ein komplexeres Konzept verstehen können, werden in diesem Graphen durch ein „Und“ markiert. Diese Bestandteile sind nicht wählbar. Durch die Graphdarstellung wird für eine Menge von obligatorischen Teilkonzepten keine Reihenfolge empfohlen, da sie nach dem Verständnis wichtiger Basiskonzepte unabhängig voneinander erarbeitet werden können. Durch diesen Verzicht auf eine Sequenzierung lassen sich vielfältige Varianten modellieren. Alle möglichen Lernpfade stellen überprüfenswerte informatikdidaktische Varianten dar und werden im Graphen durch „Oder“ markiert. Keine Variante sollte ohne Begründung ausgeschlossen werden (vgl. Beispiele in Abbildung 26 auf S. 185).

3.5.2 Entwicklung von Wissensstrukturen

Im Punkt 4 von Abschnitt 2.4.2 wurde ein Mangel an Strukturierungsempfehlungen für Lehr-Lern-Prozesse zum objektorientierten Modellieren festgestellt. Speziell eine Gesamtsicht auf das Themenfeld objektorientiertes Modellieren, die Lehrenden und Lernenden Orientierung gibt, fehlt.
Ausgehend von dem in Abschnitt 2.4.2 betonten Bedarf an expliziten Strukturierungsempfehlungen für Lehr-Lern-Prozesse im Bereich des objektorientierten Modellierens bildete den Ausgangspunkt eine vertiefende Analyse fachwissenschaftlicher Erarbeitungsstrukturen (vgl.

6.2.2). Analysiert wurden fachwissenschaftliche Lehrbücher (Standardwerke: Wirfs-Brock et al. 1990, Jacobson et al. 1992, Rumbaugh et al. 1993, Booch 1994, Meyer 1997) im Hinblick auf die Begriffsbildung zu objektorientierten Basiskonzepten über Metaphern sowie zu Varianten von Erarbeitungsreihenfolgen. Ergänzt wurde dies durch die Analyse fachdidaktischer Problembereiche, z.B. Vermeidung von Sprüngen im Abstraktionsniveau sowie Verknüpfung von Algorithmik und Objektkommunikation bei der grafischen Modellierung (vgl. 6.2.3). Auf dieser Basis wurden der theoretische Hintergrund zu Wissensrepräsentationsformen, insbesondere Mapping-Techniken, betrachtet (vgl. 6.3) und verschiedene didaktische Funktionen einer grafischen Repräsentation von Erarbeitungs- und Vorkenntnisstrukturen von Fachkonzepten (in der vorliegenden Arbeit als Wissensstrukturen bezeichnet) begründet (vgl. 6.4.1). Die Grundlagen aus den Bereichen Fachwissenschaft, Fachdidaktik und Wissensrepräsentation führten zur Spezifikation von Anforderungen an eine für schulische Lehr-Lern-Prozesse geeignete Darstellungsform (vgl. 6.4.2). Anschließend wurden verschiedene Darstellungsformen aus Informatik, Psychologie und Pädagogik (Begriffsnetze, Klassen- und Objektdiagramme, Semantische Netze, Und-Oder-Graphen) im Hinblick auf die Erfüllung der Anforderungen untersucht (vgl. 6.4.3). Es zeigte sich, dass Und-Oder-Graphen von den betrachteten Darstellungsformen am besten geeignet sind, die spezifizierten Anforderungen zu erfüllen.
Zusammen mit einer studentischen Projektgruppe wurde die Repräsentation von Wissensstrukturen unter Verwendung einer Landkartenmetapher erprobt. Dabei entstand ein Vorgehensmodell (vgl. 6.5), welches Möglichkeiten und Grenzen der Landkartenmetapher offenbarte.
Eine zusammenfassende Übersicht über die im Rahmen dieser Arbeit begründete Vorgehensweise bei der Entwicklung von Wissensstrukturen zum objektorientierten Modellieren zeigt Abbildung 27 auf S. 190.

3.5.3 Didaktische Funktionen von Wissensstrukturen

Gestaltungsmittel für Lehr-Lern-Prozesse

Speziell für Lehramtsstudierende und im Themenfeld objektorientiertes Modellieren noch wenig erfahrene Lehrende dient die grafische Repräsentation von fachlichen Erarbeitungsstrukturen als kompakte und ausdrucksstarke Veranschaulichung der Zusammenhänge (vgl. 6.4.1). Das erleichtert die Orientierung. Ersichtlich wird, wo an Vorwissen einer Lerngruppe angeknüpft werden kann, in welchen Reihenfolgen Aneignungsprozesse möglich sind, welche Alternativen es dabei gibt und wie komplexe Fachkonzepte schrittweise auf einfacheren Fachkonzepten aufbauen. Durch solche Repräsentation wird auch die Weitergabe von fachdidaktischem Wissen an nachfolgende Generationen von Lehrenden gefördert (vgl. auch Quibeldey-Cirkel 1998).

Anwendung in Lehr-Lern-Prozessen

Lernende wenden grafische Repräsentationen der fachlichen Erarbeitungsstrukturen an bei der Organisation von Selbststudienphasen sowie zur Vorbereitung, Nachbereitung und Wiederholung von Unterricht.
Der erreichte Bildungsstand wird für Lernende sowie Lehrende anschaulicher, indem in der grafischen Repräsentation Elemente abgebildet werden, die sich mit Lernerfolgskontrollen messen und bewerten lassen. Die grafische Repräsentation kann damit als Messinstrument des erreichten Bildungsstands angewendet werden.

Förderung der fachdidaktischen Kommunikation und Diskussion zu Lehr-Lern-Prozessen

Durch die grafische Repräsentation von Erarbeitungsstrukturen von Fachwissen können die

Analysemöglichkeiten zu Bildungsprozessen verbessert werden. Explizierte Erarbeitungsstrukturen erleichtern Diskussionen zum Stand der Fachdidaktik Informatik. Eine einheitliche Darstellungsform erhöht die Vergleichbarkeit von Lehr-Lern-Prozessen und Bildungsergebnissen. Durch die Repräsentation kann die Kooperation beim gemeinsamen Weiterentwickeln informatischer Bildung gefördert werden, indem anhand der Darstellung z.B. fachliche und lehrmethodische Fehler konkreter Lehr-Lern-Prozesse diskutiert werden können.

3.6 Verknüpfung der Komponenten

Die Wissensstrukturen stehen in engem Verbund mit den übrigen Komponenten des didaktischen Systems. Eine oder mehrere Aufgabenklassen gehören zu einer Lerneinheit in der Wissensstruktur (ein Fachkonzept, eine Menge von Fachkonzepten). Wenn Lernende typische Vertreter dieser Aufgabenklassen lösen können, wurde das Bildungsziel erreicht, das mit einem oder einer Menge von Knoten im Und-Oder-Graph verbunden ist. Nachbarknoten enthalten die nächsten Bildungsziele. Jedes Explorationsmodul gehört zu einer Lerneinheit und unterstützt die Erkundung der Wissensstruktur durch den Lernenden. Wissensstruktur, Aufgabenklassen und Explorationsmodule stehen somit in enger Korrelation (vgl. Abbildung 1). Jedes Fachkonzept (elementar, komplex) ist verknüpft mit einer beliebigen Anzahl an Aufgabenklassen und Explorationsmodulen. Umgekehrt korrespondieren jede Aufgabenklasse und jedes Explorationsmodul zu mindestens einem Fachkonzept. Eine Aufgabenklasse kann Gestaltungsvorlage für beliebig viele Explorationsmodule sein. In einem Explorationsmodul können beliebig viele Aufgabenklassen exploriert werden. Abbildung 2 zeigt die exemplarische Verknüpfung von Fachkonzepten F_i mit Explorationsmodulen $E_{i,j}$ und Aufgabenklassen $A_{i,k}$ in einem Und-Oder-Graphen. Ersichtlich wird, dass jedem Knoten sowie beliebigen Teilgraphen jeweils eine Menge von Aufgabenklassen und Explorationsmodulen zugeordnet werden kann. Eine exemplarische Verknüpfung von konkreten OOM-Fachkonzepten mit im Rahmen dieser Arbeit entwickelter Lernsoftware zur Exploration und mit konkreten Aufgabenklassen zeigt Abbildung 26 auf S. 185. Komplexität der gewählten Aufgabenklassen und Graphen ist mit definierten Bildungsstandards gut vergleichbar. Insofern lässt sich bereits auf der Basis solcher Wissensstrukturen die Qualität informatischer Bildung diskutieren. Damit bilden sie auch einen interessanten Ansatz für einen Soll-Ist-Vergleich zwischen Schulen und Ländern.

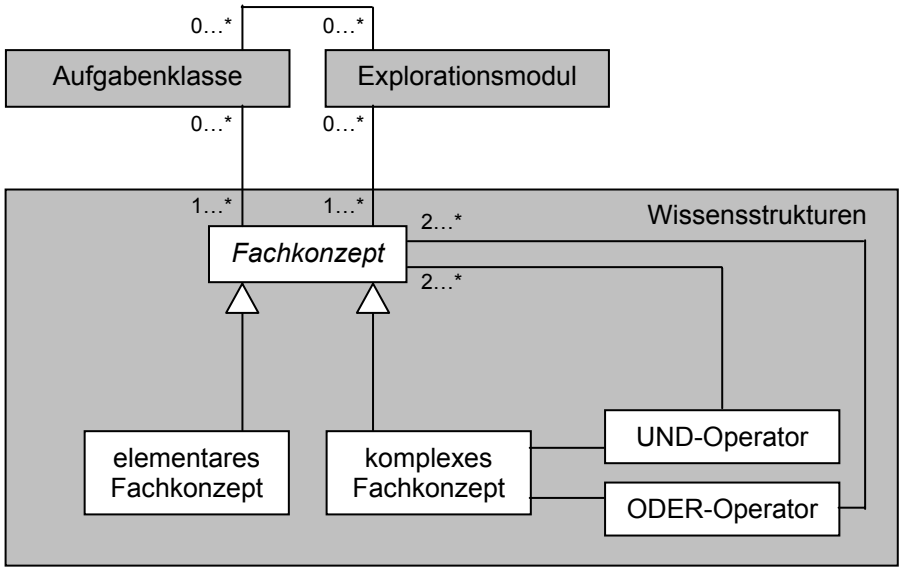


Abbildung 1: Verknüpfung der Komponenten des didaktischen Systems mit der Wissensstruktur als Metamodell

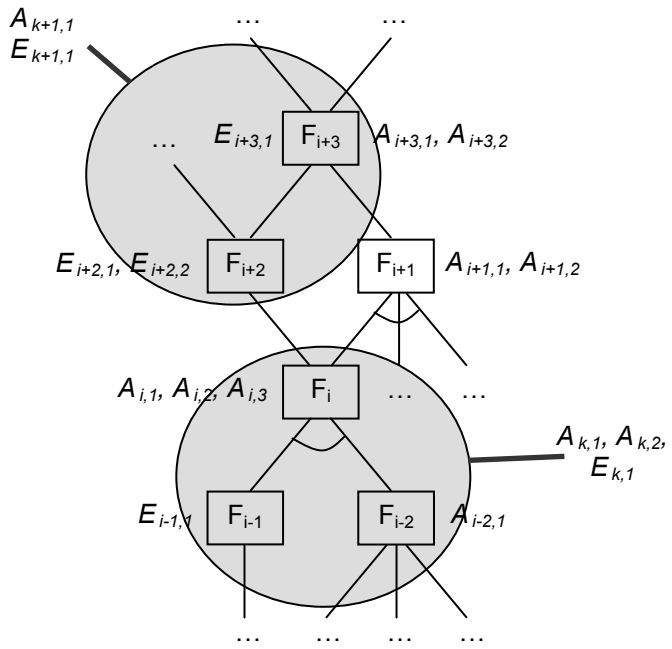


Abbildung 2: Verknüpfung von Aufgabenklassen und Explorationsmodulen mit Wissensstrukturen

3.7 Konzept für die Lehrerbildung

Für die Einbeziehung des didaktischen Systems in die Informatiklehrerausbildung wurde ein Stufenkonzept entwickelt (vgl. Brinda / Schubert 2002b) und exemplarisch in der Ausbildung von Lehramtsstudierenden an der Universität Dortmund sowie in vom Autor der Arbeit durchgeführten Lehrerfortbildungen erprobt (vgl. 4.6, 5.5.3). Nachfolgend wird eine verfeinerte und ergänzte Fassung dieses Konzepts präsentiert.

Informatiklehramtsstudium

Stufe I: Lehrveranstaltungen

In Lehrveranstaltungen zur „Didaktik der Informatik“ in Grund- und Hauptstudium lernen die Studierenden die Konzeption des didaktischen Systems (Brinda 2000a, Brinda / Schubert 2001, 2002a) kennen. In, die Vorlesungen begleitenden, Übungen wird die Anwendung und Gestaltung der Komponenten eines didaktischen Systems exemplarisch nachvollzogen. In Bezug auf die Komponenten nehmen die Studierenden zum einen die Rolle des Lernenden ein, indem sie die Anwendung aus Schülersicht erproben. Zum anderen nehmen sie die Rolle des Lehrenden ein, indem sie einerseits die Anwendung der Komponente bei der Gestaltung von Unterricht und andererseits die Gestaltung und Weiterentwicklung der Komponente erproben (vgl. Tabelle 7 und Abschnitte 4.6, 5.5.3, 6.5).

	Rolle des Lernenden (anwendend)	**Rolle des Lehrenden (anwendend)**	**Rolle des Lehrenden (gestaltend)**
Aufgabenklassen	• Aufgaben zum OOM bearbeiten • Lösungsstrategie der Aufgabe abstrahieren • vergleichbare Aufgabe mit der Lösungsstrategie verknüpfen	• unter Verwendung des Aufgabenklassen-Konzepts Aufgaben für unterschiedliche Anforderungsniveaus entwerfen	• Übungsaufgaben analysieren und Aufgabenklassen, Kontextklassen und Möglichkeiten der Gestaltung von Niveaustufungen identifizieren

	Rolle des Lernenden (anwendend)	Rolle des Lehrenden (anwendend)	Rolle des Lehrenden (gestaltend)
Explorationsmodule	• Explorationsmodule anwenden und Lernprozesse reflektieren	• gegebene Explorationsmodule im Hinblick auf ihre unterrichtlichen Einsatzmöglichkeiten analysieren • Unterricht unter Verwendung von Explorationsmodulen planen	• eigene Explorationsmodule entwerfen
Wissensstrukturen	• bestehende Wissensstrukturen analysieren • Selbstlernphasen unter Verwendung gegebener Wissensstrukturen organisieren	• Unterricht unter Verwendung gegebener Wissensstrukturen planen • fachliche Verlaufsstrukturen des Unterrichts visualisieren und diskutieren	• bestehende Wissensstrukturen erweitern • eigene Wissensstrukturen konstruieren

Tabelle 7: Wechselnde Rollen von Studierenden in der Informatiklehrerausbildung

Stufe II: Unterrichtspraktika

Unterrichtspraktika sind eine verpflichtende[57], aber auch sehr wertvolle Studienkomponente für Informatik-Lehramtsstudierende. Ein Semester lang muss jeder Lehramtsstudierende mehrere Informatik-Doppelstunden vorbereiten, durchführen und reflektieren zusammen mit einem Universitäts- und einem Schulmentor. Die anderen Lehramtsstudierenden hospitieren den Unterricht, den die Kommilitonen halten, und tragen später zur Reflexion bei. In Vorbereitung auf solche Unterrichtspraktika verwenden die Lehramtsstudierenden Wissensstrukturen zur Planung der Lehr-Lern-Prozesse. Sie entwickeln Aufgabenklassen, diskutieren geeignete Beispielkontexte und leiten konkrete Übungsaufgaben aus beidem ab. In den Stunden werden die Lösungsstrategien der Lernenden anschließend expliziert, strukturiert und kombiniert mit den abstrakten Aufgabenklassen, damit die Lernenden diese für die Bearbeitung ähnlicher Aufgaben in Zukunft verwenden können.

Die Studierenden analysieren die Akzeptanz und den unterrichtlichen Erfolg der Explorationsmodule. Dafür werden sie eingeführt in die Evaluation von Bildungsprozessen mittels Videoanalyse (vgl. Magenheim / Schubert 2000) und die damit verbundenen Schwierigkeiten. Es ist ziemlich schwierig, die Lernprozesse von Schülerinnen und Schülern auf einem Videoband zu identifizieren. Weiterhin kann schon die bloße Anwesenheit einer Videokamera im Unterricht das Verhalten der Lernenden ändern und somit das Bild verfälschen. Die Ergebnisse der Analyse von Informatikunterrichtsvideos können verbessert werden durch die Verwendung von Logfiles, in denen die Interaktionshistorie der Lernenden erfasst wird (vgl. Freudenreich / Schulte 2002). Die auswertende Person sieht sich dann aber einer derart großen Datenmenge gegenüber, die, wenn überhaupt, nur mit großem zeitlichem Aufwand auszuwerten ist.

Stufe III: Projekte

Um den Studierenden einen tieferen Einblick in die Gestaltung von Lern-Software zu ermöglichen, gestalten sie ein eigenes Explorationsmodul in einem Projekt. An der Universität Dortmund ist die Projektarbeit eine verpflichtende Studienkomponente im gewählten Vertiefungsgebiet, in der Lehramtsstudierende ein größeres Problem mit Bezug zur Informatikausbildung über ein Semester selbstständig bearbeiten. Die Studierenden analysieren dazu verfügbare Explorationsmodule und entwerfen und implementieren ein neues für ein selbst gewähltes Informatikfeld. Dadurch erhalten sie ein tieferes Verständnis für die Gestaltung von

[57] an den Universitäten Dortmund und Siegen

Lern-Software und schärfen ihre Fach- und Methodenkompetenzen beim objektorientierten Problemlösen (vgl. 5.5.4).

Stufe IV: Forschungsauftrag (Staatsexamensarbeit)
In Staatsarbeiten arbeiten die Lehramtsstudierenden an eigenen, begrenzten Forschungsaufgaben. Zum Beispiel können sie eine kleine Sequenz von Unterrichtsstunden mit einem Mentor der Universität und einem Mentor der Schule (team-teaching) unter Verwendung der „Aufgabenklassen" vorbereiten. Anschließend dokumentieren die Lehramtsstudierenden die vorbereiteten Stunden während der Schulmentor diese nach dem erarbeiteten Konzept durchführt. Diese Dokumente, ebenso wie die Lösungen der Lernenden zu vorbereiteten Übungen, werden analysiert auf typische Fehler, Hinweise auf die Motivation der Lernenden etc. (vgl. Ortmann 2002). Damit lässt sich ein tieferes Verständnis des Lehr-Lern-Prozesses erreichen.

Referendariat und Lehrerfortbildungen

Im Bereich der zweiten und der dritten Ausbildungsphase von Informatik-Lehrenden (Referendariat, Lehrerfortbildungen) können, je nach zur Verfügung stehender Ausbildungszeit, Elemente dieser Konzeption analog umgesetzt werden (vgl. 4.6.2, 5.5.3.2).

3.8 Abgrenzung von anderen Ansätzen

Im Weiteren erfolgt eine Abgrenzung des hier dargestellten Ansatzes von anderen Arbeiten zum objektorientierten Modellieren im Informatikunterricht, die während dessen Entwicklung veröffentlicht wurden.

Ab welcher Alterstufe können Informatikinhalte prinzipiell vermittelt werden?

Schwill (2001) geht der Frage nach, „ab wann [...] man mit Kindern Informatik machen [kann]". Grundlage ist sein eigener Ansatz der fundamentalen Ideen der Informatik:

> „Als ein zentrales Merkmal wurde dazu von fundamentalen Ideen das sog. Vertikalkriterium gefordert: ‚Eine **fundamentale Idee** (bezgl. einer Wissenschaft) ist ein Denk-, Handlungs-, Beschreibungs- oder Erklärungsschema, das ... auf jedem intellektuellen Niveau aufgezeigt und vermittelt werden kann (**Vertikalkriterium**) ...'" (ebd., 14; Hervorhebungen im Original)

Er zeigt dies für Kinder im Grundschulalter unter anderem am Beispiel der strukturierten Zerlegung, einer seiner „Masterideen". Die Fähigkeit zur strukturierten Zerlegung ist eine Grundvoraussetzung für objektorientiertes Modellieren. Schwill wertet im Hinblick auf die strukturierte Zerlegung psychologische Studien aus, bei denen Kindern ein Bild gezeigt wurde, in welchem aus zwei Giraffen ein Herz geformt ist. Die Aufgabe bestand darin, Giraffen und Herz zu identifizieren und den Zusammenhang herzustellen. Er kommt zu folgenden Schlussfolgerungen:

> „Wir können also aus den Versuchen ableiten, dass die wichtigste kognitive Voraussetzung für ein Verständnis der strukturierten Zerlegung, nämlich die Wahrnehmung eines Objekts als strukturierte Summe seiner Teile schon im Grundschulalter weitgehend vorliegt." (ebd., 25)

Bei Versuchen zur Nachbildung hierarchischer bzw. baumartiger Objekte mit je 10 Kindern der Altersstufen 3, 4, 5, 6, 7, 9 und 11 am Beispiel eines Mobilés kommt er zu folgenden Schlussfolgerungen:

> „Eine scharfe Altersgrenze bildet die Altersstufe 6 Jahre. Nur ein Kind ab dem 6. Lebensjahr kann das Mobilé *nicht* exakt kopieren [...]" (ebd., 25; Hervorhebung im Original)

> „[...] wenige 4-jährige [können] bereits Original und Kopie vergleichen, wahrnehmen, dass beide verschieden sind, und erkennen, dass ihr Modell einer Teilstruktur des Originals entspricht oder nicht entspricht. Diese Gruppe ist also bereits eingeschränkt in der Lage, hierarchische Strukturen

> kognitiv zu erfassen, zu analysieren und in Bestandteile zu zerlegen, nicht jedoch dazu, diese Fähigkeit konstruktiv operational umzusetzen.“ (ebd., 26)

Bei Untersuchungen zu den planerischen Fähigkeiten von Kindern kommt er bezüglich der Planungsbreite (Anzahl der Teilziele, in die ein Ziel maximal zerlegt werden kann; Modularisierung) zu dem Schluss, dass „spätestens bei Grundschulkindern von einer schon weitreichenden Fähigkeit zur Modularisierung in bis zu sechs verschiedene ‚Komponenten“ ausgegangen werden“ kann (ebd., 28). Bezüglich der Planungstiefe (Anzahl der Hierarchieebenen bei fortlaufender weiterer Zerlegung der Teilziele in Unterziele, Hierarchisierung) stellt er fest:

> „Folglich können wir auch hier [...] davon ausgehen, dass die Kinder im Grundschulalter Baumstrukturen, wie sie durch die hierarchische Modularisierung entstehen, der Höhe 3 mit bis zu 6 Teilbäumen je Knoten, konstruieren, kognitiv erfassen und bewältigen können“ (ebd. 29)

Es zeigt sich also, dass bereits im Grundschulalter kognitive und planerische Fähigkeiten bei Kindern vorliegen, die wesentliche Voraussetzungen für die Beschäftigung mit informatischen Inhalten darstellen. An dieser Stelle soll nun keine Diskussion darüber geführt werden, ob oder ob nicht Informatikunterricht bereits in der Grundschule möglich ist bzw. angestrebt werden sollte, da diese Zielgruppe nicht im Fokus der vorliegenden Arbeit liegt.

Sekundarstufe I

Voß (2003) befasst sich mit der objektorientierten Modellierung von Software zur Textgestaltung. Erprobt wurde dies mit Schülerinnen und Schülern der 10. Jahrgangsstufe einer Realschule im Fach „Textverarbeitung“ (nicht: Informatik). Ziel war es, Fähigkeiten der Textverarbeitung unabhängig von der eingesetzten Software im Sinne der Berufsvorbereitung zu vermitteln. Grundlage für die Behandlung der Textverarbeitung war ein Klassenmodell eines Textdokuments (ebd., 212). Unterrichtet wurden zwei Lerngruppen im Bereich der Textverarbeitung (zwei Doppelstunden), die eine nur anhand der Informationen der Hilfedokumente der Software, die andere unter Verwendung objektorientierter Denkweisen. Alle Arbeitsaufträge wurden entsprechend formuliert. Voß beobachtete deutliche Unterschiede in den Schüleraktivitäten. Während sich die OOM-Gruppe aktiv beteiligte, zeigte die andere Gruppe deutliche Motivationsschwierigkeiten. Ferner zeigte sich, dass der OOM-Gruppe die Bearbeitung der gestellten Aufgaben insgesamt besser gelang als der Vergleichsgruppe.
Die Sekundarstufe I ist in der vorliegenden Arbeit nicht die primär adressierte Zielgruppe. Im Abschnitt 5.3.4.3 wird aber begründet, wie im Rahmen dieser Arbeit entwickelte Software auch solche Konzepte fördern kann. Im Rahmen der vorliegenden Arbeit wird ein Konzept gestaltet, um die Vermittlung und die Aneignung objektorientierter Fachkonzepte anzuregen. Voß hingegen verwendete ausgewählte Modellierungstechniken der Objektorientierung als Medium für die Vermittlung der Anwendungsstrategie von Textverarbeitungssoftware.

Sekundarstufe II

Unterrichtserfahrungen

Moll (2002) stellt die Ergebnisse seiner zweiten Staatsexamensarbeit dar. Auf der Basis von didaktischen Überlegungen zur Modellierung im Informatikunterricht und der objektorientierten Methode in der Schule sowie der Analyse von verschiedenen Problembereichen des Anfangsunterrichts („Anwendungsproblem analysieren“, „objektorientierte Designprinzipien kennen lernen“, „Pascal lernen“) begründete er den Bedarf für den Einsatz von Modellierungswerkzeugen im Unterricht. Er bewertete verschiedene Werkzeuge im Hinblick auf den unterrichtlichen Einsatz und entwickelte schließlich auf der Basis von Vorarbeiten der Schülerakademie Mathematik-Informatik Münster ein eigenes kleines Werkzeug „UMLed“. Es zeigt sich der Bedarf für unterrichtsgeeignete Software-Werkzeuge. Lehrende verwenden entweder professionelle Werkzeuge (vgl. 5.3.2) oder behelfen sich, wie hier, durch Eigenentwicklung bzw. Systemanpassung.

Moll begründete und erprobte ein Unterrichtskonzept für die Jahrgangsstufe 11. Den Einstieg bildete eine mehrwöchige bzw. mehrmonatige Phase zur Vermittlung der Grundkonzepte der Objektorientierung auf der Basis des Konzepts „Von Stiften und Mäusen“ (vgl. S. 29). Das daran anschließende Modellierungsprojekt ist wie folgt strukturiert (Moll 2002, 49):

> „1. Anwendungskontext erkunden, Erstellung eines CRC-Modells (oder Auseinandersetzung mit gegebenem Modell) und Simulation in einem Objektspiel
> 2. Übertragung dieses Modells in ein UML-Diagramm (oder mehrere) mit einem CASE-Tool inkl. Dokumentation
> 3. Entwurf einer Benutzungsoberfläche und Beschreibung der Programmfunktion
> 4. Erweiterung des Modells um (technisch) notwendige Klassen, Attribute, Methoden, Beziehungen mit dem CASE-Tool
> 5. Codeerzeugung mit dem Tool und Implementation des Modells“

Moll stellt ausführlich seine Erfahrungen und Schwierigkeiten mit diesem Konzept dar. Aufgrund von verschiedenen Lernschwierigkeiten sieht er einen Bedarf für die Ergänzung des Projekts durch begleitende Aufgaben (ebd., 51):

> „Weitere Übungsaufgaben zu den neuen Konzepten aus den einzelnen Gruppenarbeiten müssen das Projekt ergänzen.“

Ziel des im Rahmen der vorliegenden Arbeit präsentierten Konzepts ist es, traditionellen Unterricht an verschiedenen Stellen zu bereichern. Durch die Komponente „Aufgabenklassen“ wird hier ein Reichtum an möglichen Aufgabenstellungen eröffnet, die leicht in eine konkrete Lernsituation integriert werden können.

Diethelm et al. (2002, 2003) berichten von Unterrichtserfahrungen mit „Story Driven Modeling“, einem Vorgehensmodell aus der Software-Technik zur Entwicklung objektorientierter Programme (vgl. Zündorf 2001). In (Diethelm et al. 2002) wird eine Strategie zur systematischen Modellierung eines abstrakten Problems am Beispiel des „Türme von Hanoi“-Problems vorgestellt, die in einem Grundkurs der Jgst. 12 erprobt wurde. Die Grundlage bildet ein handlungsorientiertes Objektspiel. Die Arbeitsschritte des Objektspiels und entstehende Objektstrukturen werden detailliert mit „Story Boards“ dokumentiert. Hierbei handelt es sich um eine UML-Ergänzung, mit welcher Objektstrukturen und deren zeitliche Änderungen beschrieben werden können. Problematisch ist, dass die Syntax dieser Diagramme nichttrivial ist und sich nicht intuitiv erschließt. Auf der Basis dieser Diagramme wird anschließend ein Programm abgeleitet. Dieser Prozess wird sehr detailliert beschrieben, die Aktivitäten der Lernenden hierbei bleiben aber offen.
Ein zweites Beispiel wird in (Diethelm et al. 2003) vorgestellt, wo eine Verknüpfung mit den Lego Mindstorms Robotern erfolgt (vgl. 2.3.1.2, S. 30). Erprobt wurde dies in einem jahrgangsstufenübergreifenden Informatikgrundkurs der Stufen 12 und 13. Eingesetzt wurde die Entwicklungsumgebung Fujaba[58], die „Story Boards“ unterstützt. Ziel war es, den Lernenden „Grundzüge der Software-Technik zu vermitteln und ihnen ein Gefühl dafür zu geben, wie Software in der Wirtschaft entwickelt wird“ (ebd., 230). Aus diesem Grunde erfolgte eine Organisation in Form eines „Industriepraktikums“. Die Schüler erhielten die Aufgabe: „Entwirf ein Programm, das einen Mindstorms-Roboter so steuert, dass er das Problem ‚Türme von Hanoi mit 4 Scheiben’ löst.“ (ebd., 230). Zusätzlich zum in (Diethelm et al. 2002) beschriebenen Vorgehen mussten die Lernenden die Steuerung der Roboter bewältigen. Die Autoren berichten von positiven Erfahrungen.
Von Diethelm et al. wurden Lerngruppen mit Vorwissen zur objektorientierten Modellierung adressiert. Im Hinblick auf die Anwendung des Konzeptes bei Informatikanfängerinnen und -anfängern wurden noch keine Erfahrungen publiziert.
Sowohl bei Moll (2002) als auch bei Diethelm et al. (2002, 2003) ist die Vermittlung von ob-

[58] URL: http://www.fujaba.de/ (aufgerufen am 09.12.2003)

jektorientierter *Systemgestaltung* das Ziel. In der vorliegenden Arbeit geht es um Strategien zur Vermittlung von objektorientierten Basiskonzepten, die die Voraussetzung für die Vermittlung von Systemgestaltung darstellen.

Vorteile objektorientierter Sprachen

Modrow (2003) verfasste seine Dissertationsschrift zum Thema „Pragmatischer Konstruktivismus und Fundamentale Ideen als Leitlinien der Curriculumsentwicklung". Modrow betont den Stellenwert der Programmierung im Informatikunterricht. Aspekte der Objektorientierung werden in seiner Arbeit nur am Rande thematisiert. In diesem Zusammenhang stellt Modrow die Vorteile objektorientierter Sprachen heraus (ebd., 61):

> „Der Hauptvorteil liegt m. E. darin, dass im Zusammenspiel mit integrierten Entwicklungsumgebungen die Bedeutung der ‚Benutzerschnittstelle', also der Programmoberfläche, auf das angemessene geringe Maß heruntergestuft wurde. Oberflächen werden schnell ‚zusammengeklickt', und dann bleibt Zeit für die eigentliche Problemlösung. Sprachen wie Delphi, Java (mit IDE) und andere ersetzen endlose Folgen von Read-Write-Anweisungen oder entsprechende Grafikbefehlsfolgen, die Oberflächen erzeugen sollen, durch einige wenige Ressourcendefinitionen, die z.B. bei Delphi im Programmtext kaum noch zu finden sind. Sie schaffen Raum für die Implementierung abstrakter Konzepte, von Datenstrukturen und anspruchsvollen Algorithmen, deren Ergebnisse nur noch an Oberflächenelemente gekoppelt werden müssen. OOP ermöglicht so wirklich inhaltsreicheren Informatikunterricht, stellt eine qualitative Verbesserung dar – und das nicht so sehr durch die neuen Sprachkonzepte, sondern durch die verbesserten Entwicklungswerkzeuge."

Die Gefahr der Verkürzung des Software-Entwicklungsprozesses durch das „Zusammenklicken von Oberflächen" und durch die Verknüpfung von Oberflächen mit Ereignisroutinen wurde bereits von Penon und Spolwig (1998) ausführlich thematisiert (vgl. S. 25 in der vorliegenden Arbeit). Insbesondere die letzte Aussage des Zitats wird vom Autor der vorliegenden Arbeit nicht geteilt. Komplexe Entwicklungsumgebungen, das Zusammenspiel von Editor, Compiler, GUI-Builder, Debugger etc., stellen eine erhebliche kognitive Anforderung an Lernende dar (vgl. 5.3.2). Es wird viel Unterrichtszeit durch das Erlernen der Handhabung der Software gebunden, bevor Lernende damit erfolgreich informatische Probleme bearbeiten können. Ferner ist es weniger die konkrete Realisierung von Fachkonzepten aus dem Bereich der Objektorientierung in einer gegebenen Programmiersprache, die günstig für Lernende ist, sondern die am Denken der Menschen orientierte Sichtweise, die Durchgängigkeit der Methode in allen Phasen der Problemlösung etc. (vgl. Abschnitt 2.3, S. 22).

Theoriegeleitete Entwicklung und Evaluation eines Unterrichtskonzepts zur Objektorientierung in der Sekundarstufe II

Schulte entwickelte bis 2003[59] ein Unterrichtskonzept zur Objektorientierung in der Sekundarstufe II und führte eine eingehende Evaluation durch.

Schulte untersuchte die Fragestellung, ob Implementationswissen im Zusammenhang mit der Thematisierung der eigentlichen Unterrichtsinhalte, Fachkonzepten aus dem Bereich der Objektorientierung, erlernbar ist oder ob dazu die üblichen Sprachkurse, in denen die grundlegenden Sprachstrukturen vermittelt werden, notwendig sind. Zur Prüfung dieser Frage entwickelte er ein empirisches Unterrichtsszenario. Auf der Basis von Lehr-Lern-Konzepten zur Objektorientierung im Anfangsunterricht sowie Ergebnissen fachdidaktischer und lernpsychologischer Ansätze entwickelte er ein Unterrichtskonzept, welches den systemorientierten und den informationszentrierten Ansatz der Informatikdidaktik verbindet und in der unterrichtsmethodischen Ausgestaltung dem Ansatz des Cognitive Apprenticeship folgt (Schulte 2001). Dabei wurden Entwicklungswerkzeuge (Fujaba, Dobs) als Lernmedien in die Unterrichtsmethodik integriert und ein Phasenmodell des Unterrichtsablaufs entwickelt (Schulte / Block 2002). Einen Schwerpunkt von Schultes Arbeit bildete die Entwicklung und Erprobung von

[59] Die zugehörige Dissertationsschrift wurde im Oktober 2003 eingereicht.

empirischen Untersuchungsmethoden und -instrumenten zur Evaluation des Konzepts (Freudenreich / Schulte 2002).
Im Ergebnis zeigte sich, dass es den einzelnen Schülergruppen mit dem Konzept gelang, ein Projekt zu modellieren und zu implementieren, sowie Grundkonzepte der Objektorientierung, Modellierungskenntnisse und ein differenziertes Bild von Software-Entwicklung zu erwerben (Schulte / Niere 2002).
Schulte untersuchte eine andere Fragestellung als der Autor der vorliegenden Arbeit (vgl. 2.4.3). Ziel dieser Arbeit war nicht die Entwicklung und systematische Evaluation eines konkreten Unterrichtskonzeptes zur Objektorientierung unter definierten Laborbedingungen, sondern die Gestaltung von Lehr-Lern-Materialien sowohl für Lehrende als auch für Lernende verknüpft mit Vorgehensweisen zu deren Erstellung und Erprobung zur Bereicherung und Ergänzung von „realem" Informatikunterricht im Sinne der im Abschnitt 2.4.2 genannten Problembereiche. Auch in Schultes Konzeption gibt es einen Bedarf für geeignete Aufgabenstellungen und andere Lernangebote, um neu eingeführte Fachkonzepte zu üben oder zu vertiefen. Insofern ergänzen sich die Ergebnisse Schultes und des Autors der vorliegenden Arbeit gegenseitig.

Informatik-Lernlabor

Magenheim (2001, 2003) beschreibt die Konzeption eines „Informatik-Lernlabors", welches eine Weiterentwicklung des in (Hampel et al. 1999, vgl. 2.3.1.1) vorgestellten Ansatzes darstellt. Die verzahnten konstruktiven und dekonstruktiven Unterrichtsphasen werden darin im Sinne des „Blended Learning" angereichert um verschiedene multimediale Dokumente, Lernmaterialien und Werkzeuge. Zwei der vier beschriebenen Inhaltsmodule („Hochregallager", „Schulkiosk") sind in der Entwicklung soweit fortgeschritten, dass ein erster Einsatz im Schulunterricht bzw. in Lehrveranstaltungen der Hochschule möglich ist. Zu den Inhaltsmodulen werden multimediale Dokumente und Lernobjekte bereitgestellt (allgemeine Dokumente, inhaltsmodulbezogene Dokumente und Lernobjekte, wie z.B. Animation zum Hochregallager, Videosequenzen von realen Arbeitsabläufen etc.), Software-Tools (UML-Editor, grafischer Debugger, etc.) und spezifische Tools (z.B. virtuelle Simulationsumgebung für Lego-Mindstorms-Hardware im Falle des Hochregallagers), „Guided Tours" durch die multimedialen Lernmaterialien, didaktische Software zur Unterstützung der Dekonstruktion sowie eine Groupware mit elementaren Funktionen einer Lernplattform (Lern- und Contentmanagement), über welche die Materialien zur Verfügung gestellt werden.
Die Arbeitsphasen der Lernenden orientieren sich am traditionellen Projektunterricht, welcher ergänzt wurde um Dekonstruktions- und Transferphasen sowie das Konzept des „Blended Learning". Daraus ergeben sich für Magenheim methodische Schlussfolgerungen (Magenheim 2003, 22f):

> „Lernen im Informatik Lernlabor in dem hier vorgestellten Umfang eignet sich nicht für den Anfangsunterricht, es sei denn, man möchte gezielt einzelne informatische Konzepte erkunden und nutzt entsprechende Lernobjekte oder Views auf die vorliegende Software. Effizientere Lernergebnisse sind zu erwarten, wenn die Mitglieder der Lerngruppe bereits über fundamentale Vorkenntnisse in der Modellierung und in der Generierung von Sourcecode verfügen. Das Konzept [...] wird gegenwärtig mit entsprechend gesteigertem Anforderungsniveau auch in der Hochschullehre zur Softwaretechnik eingesetzt und evaluiert."

Das beschriebene Konzept richtet sich vorwiegend an fortgeschrittene Lerner. Das im Rahmen der vorliegenden Arbeit entwickelte Konzept adressiert hingegen im Wesentlichen Lernende im Anfangsunterricht sowie im Bereich objektorientiertes Modellieren noch wenig erfahrene Lehrende und Lehramtsstudierende.

3.9 Zum weiteren Aufbau der Arbeit

In den nachfolgenden Kapiteln werden die hier eingeführten Komponenten des didaktischen Systems „Aufgabenklassen“ (vgl. Kapitel 4), „Explorationsmodule“ (vgl. Kapitel 5) und „Wissensstrukturen“ (vgl. 6) weiter ausdifferenziert sowie die Vorgehensweise ihrer Entwicklung und erste Ergebnisse ihrer Erprobung in der informatischen Bildung präsentiert. Eine detailliertere Übersicht über den Inhalt dieser Kapitel liefert Abschnitt 1.4.

4 Aufgabenklassen

4.1 Überblick

Die inhaltliche Schwerpunktverschiebung vom Programmieren zum Modellieren (vgl. Abschnitt 2.2) muss im gesamten Informatikunterricht umgesetzt werden. Damit werden neue Klassen von Aufgaben erforderlich, anhand derer sich die neuen Fachkonzepte erlernen, üben und vertiefen lassen[60]. In fachdidaktischen Publikationen zum Informatikunterricht findet sich ein großer Reichtum an Unterrichts- und Übungsbeispielen zum objektorientierten Programmieren. Vergleichbare Lehr-Lern-Materialien zum objektorientierten Modellieren sind noch selten (vgl. Abschnitt 2.3.1). Publiziert wurden z.B. mittels objektorientierten Vorgehensmodellen bearbeitete Unterrichtsprojekte, aus denen aber nicht genug ersichtlich wird, wie die Lernenden verschiedene Modellierungstechniken einzeln oder kombiniert anwendeten, festigten oder wichtige Modellierungsschritte übten oder vertieften. Dies ist aber erforderlich für ein solides und gut vernetztes Grundwissen, das als Voraussetzung für wissenschaftspropädeutisches Lernen in der Sekundarstufe II benötigt wird. Einen reichen Vorrat an solchen Aufgaben zu objektorientierten Modellen, zum Üben und Vertiefen statischer bzw. dynamischer Modellierungstechniken etc., findet man hingegen in ausgewählten einführenden fachwissenschaftlichen Lehrbüchern zum objektorientierten Modellieren. Betrachtet wurden hierzu internationale und nationale Standardwerke, um eine breite Akzeptanz der Darstellung der objektorientierten Fachkonzepte abzusichern. Die Bücher von Rumbaugh et al. (1993) und Balzert (1999) wurden aufgrund des in ihnen vorhandenen Aufgabenreichtums für die Analyse ausgewählt. Trotz des bereits länger zurück liegenden Publikationsjahrs des Buchs von Rumbaugh et al. (1993) sind gerade in diesem Werk besonders viele anfängergeeignete Aufgaben zum statischen Modell zu finden, die auch heute noch genauso einzusetzen sind. Da sich die in diesen Quellen vorhandenen Übungsaufgaben an eine andere Zielgruppe richten, nämlich Informatikstudierende, Informatikerinnen und Informatiker, muss eine begründete Auswahl bzw. Modifikation dieser Aufgaben für den Informatikunterricht erfolgen (vgl. Abschnitt 4.3.1). Um das Problem der fehlenden Unterrichts- und Übungsbeispiele nachhaltig zu lösen, wurden die ausgewählten Aufgaben zu so genannten *Aufgabenklassen* abstrahiert (vgl. 4.3.2), für einen systematischen, lernzielorientierten Zugang in einer Sammlung strukturiert (vgl. 4.3.3) und Elemente einer Methodik zur Gestaltung von Aufgaben zum objektorientierten Modellieren im Informatikunterricht bereit gestellt (vgl. 4.4). Publiziert wurde dieser Prozess zunächst für Aufgabenklassen zum statischen Systemmodell, vgl. (Brinda 2000b) und (Brinda / Schubert 2001). Die dort beschriebenen Ergebnisse stellten die Basis für empirische Fallstudien zur Einbettung einerseits in den Informatikunterricht (vgl. 4.5 und Brinda / Ortmann 2002, Ortmann 2002) und andererseits in die Informatiklehrerbildung dar (vgl. 4.6). Die daraus gewonnenen Erkenntnisse zur Gestaltung von Aufgaben mittels Aufgabenklassen, zu deren unterrichtlicher Einbettung sowie typische Strategien und Fehler bei der Bearbeitung führten zu einer Verfeinerung und Weiterentwicklung des Aufgabenklassen-Konzepts. Eine ausführliche Zusammenfassung mit grafischem Überblick über den Entwicklungs- und Erprobungsprozess (vgl. Abbildung 10 auf S. 107), Fazit und offene Fragen bilden den Abschluss im Abschnitt 4.7.

Zu beachten ist, dass der Prozess zur Entwicklung einer strukturierten Sammlung von Aufgabenklassen zum OOM (vgl. 4.3) hierbei zweimal durchlaufen wurde. Im Rahmen der Erprobung wurde die Menge der zu analysierenden Aufgaben zunächst auf das statische Modell beschränkt. Die empirischen Erkundungsstudien hierzu zeigten als ein Resultat einen Bedarf an Aufgabenklassen auch zum dynamischen Modell (vgl. 4.5.2). Als logische Konsequenz hieraus wurden damit auch Aufgabenklassen erforderlich, anhand derer sich der Prozess der

[60] Die verschiedenen fachdidaktischen Funktionen von Aufgaben wurden im Abschnitt 3.3.1 erörtert.

Verknüpfung beider Modelle üben und vertiefen lässt. Der daraus resultierende Bedarf an weiteren Aufgabenklassen bewog den Autor dazu, die Analyse auf alle Übungsaufgaben in den ausgewählten Lehrbüchern auszudehnen. Im Sinne einer möglichst redundanzarmen Gesamtstruktur der vorliegenden Arbeit, erfolgt im Abschnitt 4.3 keine chronologische, sondern eine an den Schritten des Entwicklungsprozesses der strukturierten Sammlung der Aufgabenklassen orientierte Darstellung. Die Ergebnisse vor und nach den Erkundungsstudien werden für jeden Entwicklungsschritt gegenüber gestellt und erfolgte Verfeinerungen, Ergänzungen und Änderungen entsprechend dargestellt und begründet.
Anzumerken ist weiterhin, dass sich die hier dargestellten Ergebnisse zu Aufgabenklassen auf die analysierten Lehrbücher beziehen. Es wird nicht der Anspruch erhoben, dass keine weiteren Aufgabenklassen sinnvoll oder denkbar sind. Ähnliches gilt für die Aussagen zu Kontextklassen (vgl. 4.4.3).

4.2 Stellenwert der Aufgabenklassen im didaktischen System

Der Stellenwert der Aufgabenklassen lässt sich durch die Beschreibung der erwarteten Vorteile für wesentliche Akteure im Kontext des didaktischen Systems erfassen (vgl. auch 3.3.3). Diese Akteure sind

- *Lehrende*,
- *Lehramtsstudierende*,
- *Lernende* und
- *Entwicklerinnen und Entwickler* des didaktischen Systems.

Im Hinblick auf *Lehrende* und *Lehramtsstudierende* ist das primäre Ziel der Identifikation und Strukturierung von Aufgabenklassen die Bereitstellung eines Gestaltungsmittels für Übungsaufgaben zum OOM unterschiedlichen Schwierigkeitsgrades. Erwartet wird hierdurch eine Qualitätsverbesserung und Vereinfachung hinsichtlich

- der *zielgerichteten Auswahl*,
- der *Gestaltung* und
- des *Variantenreichtums*

von Übungsaufgaben zum OOM im Informatikunterricht. Insbesondere für Lehramtsstudierende und mit OOM wenig erfahrene Lehrende kann hierdurch die Auswahl von schülergemäßen Übungsaufgaben zu Fachkonzepten vereinfacht werden (vgl. 4.3). Durch die Bereitstellung von Aufgabenklassen und durch deren Verknüpfung mit Beispielaufgaben und Musterlösungen (vgl. Anhang D.5 und Brinda 2000b) sowie durch die Begründung von Auswahlkriterien für Aufgabenkontexte und Empfehlungen für die Stufung der Komplexität und des Schwierigkeitsgrades ist die Vereinfachung der Gestaltung von Übungsaufgaben zum OOM für die jeweilige Lerngruppe möglich (vgl. 4.4). Ferner sollen durch eine strukturierte Sammlung von Aufgabenklassen Lehrenden und Lehramtsstudierenden Anregungen für die Erprobung neuer Aufgabenstellungen im Unterricht gegeben werden. Dies fördert einerseits den Variantenreichtum, andererseits können Lehrende dazu ermutigt werden, Aufgabenstellungen zur objektorientierten Modellierung zu erproben. Die Betonung der Modellierung und damit ein Ziel aktueller Informatikbildungsempfehlungen wird damit unterstützt (vgl. 2.2). Langfristig ist hierdurch eine Qualitätsverbesserung des Informatikunterrichts zu erzielen.
Im Rahmen ihres Studiums nehmen Informatiklehramtsstudierende abwechselnd die Rolle des Gestalters und des Anwenders im didaktischen System ein. In begleitenden Übungen zu Lehrveranstaltungen zur Didaktik der Informatik sowie in schulpraktischen Studienkomponenten erproben sie einerseits die Identifikation von Aufgabenklassen aus Übungsaufgaben (gestaltende Rolle) und andererseits die Gestaltung von Übungsaufgaben unter Anwendung

von Aufgabenklassen (anwendende Rolle) (vgl. 3.7, 4.5, 4.6). Dadurch wird eine Verbesserung ihres Verständnisses für die Struktur und die Gestaltung von Übungsaufgaben und damit eine Erleichterung des Einstiegs in das Lehramt erwartet. Ferner erfolgt hierdurch eine Weiterentwicklung und Pflege der Aufgabenklassen-Komponente im didaktischen System.
Für *Lernende* sind die Aufgabenklassen zum OOM erstens von Bedeutung in verschiedenen Phasen des Unterrichts (z.B. in der Sicherungsphase, vgl. 3.3.3), in denen sie die Anwendung und Verknüpfung von theoretischen Konzepten der objektorientierten Modellierung üben können, ohne den Kontext eines (großen) Software-Projektes berücksichtigen zu müssen. Das Lernen aus Beispielen und Fehlern wird gefördert und praktische Modellierungskompetenzen der Lernenden werden entwickelt. Hierbei sind es allerdings vorwiegend die aus Aufgabenklassen entwickelten Aufgaben (vgl. Ausführungen zu Lehrende), die für Lernende von Bedeutung sind. Zweitens können Aufgabenklassen Lernenden dann helfen, wenn es darum geht, ein kognitives Problem mit einer Lösungsidee zu bewältigen. Durch die Zuordnung von Aufgaben zu Aufgabenklassen anhand der Identifikation von Merkmalen bereits gelöster Aufgaben im Unterricht und durch die Verknüpfung von Aufgabenklassen mit im Unterricht erarbeiteten Lösungsstrategien wird von den Lernenden selbstständig eine Lernhilfe für die Bearbeitung ähnlicher Aufgaben gestaltet. Dazu abstrahieren die Lernenden die in der jeweiligen Aufgabe gegebenen Angaben zu Daten, verknüpfen die Daten mit dem bekannten Strukturkonzept und lösen die Aufgabe durch Analogieschluss, indem sie zuvor erarbeitete Lösungen wieder entdecken. Das trägt zur Entwicklung theoretischer Kompetenzen bei.
Entwicklerinnen und Entwickler von Komponenten des didaktischen Systems schließlich finden in der strukturierten Sammlung der Aufgabenklassen einen Vorrat an Ideen für den Inhaltsbereich von Explorationsmodulen, mit denen sich objektorientierte Basiskonzepte im Sinne entdeckenden Lernens handelnd erkunden lassen (vgl. Kapitel 5).
Viele der genannten Zieldimensionen korrespondieren mit dem Entwurfsmustergedanken, der zuerst in der Architektur bekannt wurde (Alexander et al. 1977) und später auch in der Informatik beim Entwurf von Software erfolgreich aufgegriffen wurde (Gamma et al. 1996). In der internationalen Computer Science Education Community wurde und wird ein Verzeichnis von „pedagogical patterns“ diskutiert (Quibeldey-Cirkel 1998, Manns et al. 2000), in dem spezielle Informatikzusammenhänge aus dem Bereich der Objektorientierung mit bewährten Bildungsstrategien verknüpft werden. Das Ziel wird, wie folgt, charakterisiert:

> „While many good pedagogical ideas are presented at OO conferences and published in proceedings and journals each year, very little has been done to collate the effective practices of many OO educators into one publication. The purpose of the pedagogical patterns project is [to] do just that.“ (Manns et al. 2000)

Dabei besteht aber die Gefahr der Schematisierung von sozialen Lehr-Lern-Prozessen. Bei der im Weiteren beschriebenen Entwicklung und Erprobung einer strukturierten Sammlung von Aufgabenklassen war und ist es keinesfalls das Ziel, kreative Prozesse der Unterrichtsgestaltung durch schematische Darstellungen einzuengen oder bewährte Methoden des Informatikunterrichts zu ersetzen. Vielmehr soll dazu beigetragen werden, Fachkonzepte aus dem Bereich des objektorientierten Modellierens fachdidaktisch leichter zugänglich zu machen, ihre Verbreitung zu fördern und damit Informatikunterricht insgesamt zu verbessern.

4.3 Entwicklung einer strukturierten Sammlung von Aufgabenklassen

4.3.1 Auswahl von Übungsaufgaben aus Lehrbüchern

Da es sich bei den Adressaten der Übungsaufgaben in fachwissenschaftlichen Standardwerken zum OOM um Informatikstudierende, Informatikerinnen und Informatiker handelt, im

Allgemeinen aber nicht um Lernende in allgemein bildendem Informatikunterricht, ist ein Maßstab in Form von Auswahlkriterien erforderlich, anhand dem geeignete Aufgabenstellungen für den Informatikunterricht ausgewählt bzw. transformiert werden können. Als Maßstab werden hier folgende Kriterien verwendet:

- *Fachkonzepte des Informatikunterrichts*
  Verschiedene informatikdidaktische Arbeiten lieferten Kriterien für die Auswahl von Fachkonzepten für den Informatikunterricht (z.B. Schwill 1993, Hubwieser 2000b). Es werden nur solche Aufgaben in der strukturierten Sammlung berücksichtigt, die sich auf die für den Informatikunterricht ausgewählten Fachkonzepte beziehen.

- *Betonung der Modellierung*
  Da es im Rahmen dieser Arbeit das Ziel ist, neue Aufgabentypen zum objektorientierten Modellieren für den Informatikunterricht verfügbar zu machen, erfolgt eine Konzentration auf Aufgaben, die die Modellierung betonen (vgl. 2.2). Dieses Kriterium kann im Bedarfsfall durch andere, z.B. auch verfeinerte, Schwerpunktsetzungen ausgetauscht werden, wie Analyse, Entwurf oder Programmierung.

- *Sprachenunabhängigkeit*
  Syntaxdetails einer (Modellierungs- oder Programmier-)Sprache können durchaus sinnvolle Aspekte einer Berufsvorbereitung sein, keinesfalls aber explizite Unterrichtsinhalte einer (vertieften) informatischen Allgemeinbildung (Hubwieser / Broy 1997a, 44). Es werden daher Aufgaben ausgewählt, zu deren Bearbeitung keine spezielle Modellierungs- oder Programmiersprache erforderlich ist. Bei Aufgaben, die dieses Kriterium verletzen, wird überprüft, ob die Sprachenabhängigkeit vermieden werden kann. Im positiven Fall wird die Aufgabe ausgewählt und eine entsprechende Überarbeitung vorgenommen. Damit ist auch gewährleistet, dass die Aufgabe leicht an die jeweils im Unterricht verwendeten Sprachen angepasst werden kann.

Das vierte Kriterium wurde nur beim ersten Durchgang der Entwicklung einer strukturierten Sammlung von Aufgabenklassen eingesetzt und dann verworfen (zur Begründung s. Ergebnisse). Deshalb ist es hier in Klammern dargestellt.

- *(Komplexität*
  Für den Informatikunterricht werden Aufgaben sehr unterschiedlicher Komplexität benötigt, je nach spezifischer Zielsetzung. Die Bandbreite reicht von sehr einfachen und wenig komplexen Aufgaben zum Üben und Vertiefen einzelner Modellierungsschritte (z.B. als Bestandteil eines im Unterricht zu bearbeitenden Arbeitsblattes) über die Beschreibung, Analyse, Modifikation von Modellen bis hin zur selbstständigen Gestaltung komplexer Modelle (z.B. im Rahmen eines Einzel- oder Gruppenprojektes). Die Komplexität der Beispiele muss den intellektuellen Anforderungen der Lernenden folgen. Bezogen auf ein oder mehrere gegebene Lernziele liegt die für den jeweiligen Lernenden optimale Komplexität einer Aufgabe innerhalb eines Intervalls:

  - *Mindestgröße*
    Wenn es das Ziel ist, anhand einer Aufgabe bestimmte neue fachliche Zusammenhänge zu entfalten, so ist eine bestimmte Mindestgröße erforderlich, die die neuen Inhalte wirklich notwendig macht. Um beispielsweise ein „Hello-world“-Programm zu erstellen, werden keine Kenntnisse über objektorientierte Analyse- und Entwurfsmethoden benötigt.

  - *Maximalgröße*
    Eine sehr umfangreiche Aufgabe führt dazu, dass viel Unterrichtszeit gebunden wird. Die zu modellierenden Zusammenhänge werden dabei unter Umständen sehr viel-

schichtig. Dadurch verursachte Misserfolgserlebnisse können zum Verlust der Motivation bei den Lernenden führen, vgl. (Meyer 1994, 184f))

Übungsaufgaben, die alle der o.g. Kriterien erfüllen, werden ausgewählt. Bei Aufgaben, die einzelne Kriterien verletzen, wird überprüft, ob die Aufgabe so modifiziert werden kann, dass sie den Kriterien genügt. Im positiven Fall wird die Aufgabe ausgewählt.

Ergebnisse

Zur Erprobung des Konzepts wurde der Kriterienkatalog zunächst angewandt auf Übungsaufgaben zum statischen Systemmodell. Das statische Systemmodell wurde als Inhaltsbereich ausgewählt, da es im Modellierungsprozess zentral ist und zumindest in Grundzügen vor dem dynamischen Systemmodell entwickelt wird. Mit dieser Auswahl (und daraus resultierenden Veröffentlichungen) wurde ein direkterer Nutzen für Lehrende und Lernende erwartet. Die Festlegung des Inhaltsbereiches führte zu einem zusätzlichen, inhaltlichen Selektionskriterium (Aufgaben zum statischen Systemmodell). Ausgewählt wurden dabei 55 Übungsaufgaben aus den Kapitel 1 bis 4 aus Rumbaugh et al. (1993) und 6 Übungsaufgaben aus den Kapiteln 2 und 3 aus Balzert (1999). Die Ergebnisse dieses Auswahlprozesses sind dokumentiert im Anhang B in Tabelle 42 auf S. 204 und Tabelle 43 auf S. 205. Hierbei zeigte sich, dass das vierte Kriterium (Komplexität) an dieser Stelle ungünstig verortet ist. Die Entscheidung bzgl. des Schwierigkeitsgrades und der Komplexität einer Aufgabenstellung ist erst sinnvoll auf der Basis einer konkreten Unterrichtshistorie einer Lerngruppe zu treffen, aus der sich Schlussfolgerungen über deren Kenntnisstand ableiten lassen. Aus diesem Grund wurde dieses Kriterium beim zweiten Durchgang nicht weiter berücksichtigt. Anstatt dessen wurden die ausgewählten Übungsaufgaben analysiert und Empfehlungen für die Anpassung von Schwierigkeitsgrad und Komplexität bei der Aufgabengestaltung abgeleitet (vgl. 4.4.2).
Die Auswertung der empirischen Erkundungsstudien zeigte als ein Resultat einen Bedarf an Aufgabenklassen auch zum dynamischen Systemmodell (vgl. Abschnitt 4.5.2, Zwischenresümee 4.2) in dessen Folge sich der Autor der vorliegenden Arbeit dazu entschied, die Analyse auf alle Übungsaufgaben in den ausgewählten Lehrbüchern auszudehnen (vgl. 4.1). Ausgewählt wurden dabei 142 von insgesamt 256 Übungsaufgaben aus Rumbaugh et al. (1993) und 46 von insgesamt 70 Übungsaufgaben aus Balzert (1999). Auch diese Ergebnisse sind in Tabelle 42 bzw. Tabelle 43 (s.o.) dokumentiert.
Nicht ausgewählt wurden Aufgaben insbesondere aus folgenden Themenbereichen:

- Aufgaben, die Vorwissen als Software-Entwickler/in erforderten,
- verschachtelte und parallelisierte Zustandsdiagramme,
- spezielle Entwurfmuster,

} *verletztes Kriterium:* Fachkonzepte des Informatikunterrichts

- Datenflusspläne,
- Implementierungsaspekte verteilter objektorientierter Anwendungen,
- Algorithmen, Pseudocode und Programmiersprachen,
- Grundlagen zu Menüs und Dialogfenstern,
- formale Inspektion von Software,
- Implementierungsaspekte (objekt-)relationaler und objektorientierter Datenbanken,

} *verletztes Kriterium:* Betonung der Modellierung

- spezielle Notationsdetails der UML.

} *verletztes Kriterium:* Sprachenunabhängigkeit

Bei den Aufgaben zu Datenflussplänen (nur in Rumbaugh et al. 1993) wurde geprüft, ob diese in Aufgaben zu anderen Modellierungstechniken für funktionale Abhängigkeiten in objektorientierten Modellen (z.B. Aktivitätsdiagramme) transformierbar sind. Im positiven Fall wurde die Transformation durchgeführt und die jeweilige Aufgabe ausgewählt.

4.3.2 Abstraktion von Übungsaufgaben zu Aufgabenklassen

Abstraktionsprozess

Das Ziel des Abstraktionsschrittes ist es, die jeweils ausgewählten Aufgaben (vgl. 4.3.1) zu strukturierten Aufgabenvorlagen zu verdichten. Dazu werden alle Aufgaben von konkreten Erläuterungstexten, Kontexten und Bezeichnern abstrahiert, soweit in ihnen vorhanden. Ein Kontext definiert dabei den Realitätsausschnitt, auf den sich die Aufgabe bezieht, also z.B. ein konkretes, zu analysierendes oder zu modellierendes, Informatiksystem. Die Kontexte werden gesammelt und zu Kontextklassen verdichtet (vgl. 4.4.3). Durch diesen Abstraktionsprozess werden Übungsaufgaben zu strukturierten Aufgabenvorlagen, die den jeweiligen Fachkern beinhalten. Eine Ausnahme stellen Wissens- und Verständnisfragen dar. Diese werden, soweit wie möglich, von spezifischen Fachkonzepten abstrahiert.
Jede dieser strukturierten Aufgabenvorlagen gehört zu einer Aufgabenklasse. Jede Aufgabenklasse besteht aus einer Menge von Angaben (z.B. Text, Diagramm), im Weiteren markiert durch das Zeichen „!“, und einem darauf basierenden Arbeitsauftrag, markiert durch das Zeichen „?“. Abbildung 3 zeigt ein Beispiel einer Aufgabenklasse.

!	Liste von Klassen, Attributen und Operationen mit kurzer Beschreibung
?	Zuordnung von Attributen und Operationen zu Klassen

Abbildung 3: Beispiel einer Aufgabenklasse

Bei der Abstraktion von Aufgaben zu Aufgabenklassen erfolgt eine partielle Normalisierung. Darunter werden hier die Arbeitsschritte verstanden, die nachfolgend dargestellt sind. Alle Formulierungen zu Angaben oder Arbeitsaufträgen werden, soweit wie möglich, im Singular verfasst. Ferner werden konkrete Anzahlen in Aufgabenstellungen vermieden und zu allgemeinen Auflistungen generalisiert (vgl. Abbildung 3). Ziel dieser partiellen Normalisierung ist es, die Zuordnung von verschiedenen Aufgabenvorlagen zu einer Aufgabenklasse zu erleichtern. Zudem erfolgt eine terminologische Anpassung, die aus unterschiedlichen Begriffsdefinitionen in den beiden Lehrbüchern resultiert. Zur Verdeutlichung werden nachfolgend Definitionen wichtiger Fachtermini in den beiden Lehrbüchern gegenüber gestellt und eine Begriffsentscheidung für die Aufgabenklassen getroffen.
Rumbaugh et al. definieren die Begriffe Objekt-, Klassen- und Instanzendiagramm, wie folgt:

> „*Objektdiagramme* stellen eine formale grafische Notation für die Modellierung von Objekten, Klassen und ihren Relationen zueinander bereit. [...] Es gibt zwei Arten von Objektdiagrammen: Klassendiagramme und Instanzendiagramme.“ (Rumbaugh et al. 1993, 29; Hervorhebung im Original)
>
> „Ein *Klassendiagramm* ist ein Schema, Muster oder Template (Schablone) zur Beschreibung vieler möglicher Dateninstanzen. Ein Klassendiagramm beschreibt Objektklassen.“ (ebd., 29; Hervorhebung im Original)
>
> „Ein *Instanzendiagramm* beschreibt, wie eine bestimmte Menge von Objekten zueinander in Relation stehen. Ein Instanzendiagramm beschreibt Objektinstanzen. [...] Ein gegebenes Klassendiagramm entspricht einer unendlichen Menge von Instanzendiagrammen.“ (ebd., 29; Hervorhebung im Original)

Balzert hat ein anderes Verständnis des Begriffs Objektdiagramm:

> „Das Objektdiagramm stellt Objekte und ihre Verbindungen zueinander dar. Objektdiagramme werden im allgemeinen dazu verwendet, um einen Ausschnitt des Systems zu einem bestimmten Zeitpunkt zu modellieren." (Balzert 1999, 545)

Dies entspricht der Definition des Instanzendiagramms bei Rumbaugh et al. Sie erlauben Klassen und Objekte im Objektdiagramm, verwenden Objektdiagramme aber überwiegend als Klassendiagramm. Bei der Abstraktion werden deshalb die Begriffe Instanzen- und Objektdiagramm (bei Rumbaugh et al.) ersetzt durch Objekt- und Klassendiagramm, zumal die Darstellung von Instanziierungsbeziehungen zwischen Objekten und Klassen im Klassendiagramm ebenfalls möglich ist (Oestereich 1998, 48). Die Begriffe Szenario und Ereignispfad definieren Rumbaugh et al. wie folgt:

> „Ein *Szenario* ist eine Folge von Ereignissen, die bei einer ganz bestimmten Ausführung eines Systems auftritt." (Rumbaugh et al. 1993, 106; Hervorhebung im Original)

> „Sowohl die Ereignisreihenfolge als auch die Objekte, die Ereignisse austauschen, können in einem erweiterten Szenario, einem sogenannten *Ereignispfad*-Diagramm, dargestellt werden." (ebd., 106; Hervorhebung im Original)

Dieses Ereignispfad-Diagramm stellt einen Vorläufer des Sequenzdiagramms dar. Balzert definiert Szenario wie folgt:

> „Ein Szenario ist eine Sequenz von Verarbeitungsschritten, die unter bestimmten Bedingungen auszuführen sind. [...] Ein Geschäftsprozeß wird durch eine Kollektion von Szenarios dokumentiert. Szenarios werden mit Hilfe von Sequenz- oder Kollaborationsdiagrammen dokumentiert." (Balzert 1999, 552)

In der UML wird der Begriff Interaktionsdiagramm als Oberbegriff über Sequenz- und Kollaborationsdiagramm benutzt (Balzert 1999, 541). Anstelle der Begriffe Ereignispfad-, Sequenz- und Kollaborationsdiagramm wird deshalb in den Aufgabenklassen der Begriff Interaktionsdiagramm verwendet.
Die Methode von Rumbaugh et al. enthält noch Repräsentationsformen der strukturierten Programmierung[61]:

> „Das funktionale Modell beschreibt diejenigen Aspekte eines Systems, die mit der Transformation von Werten zu tun haben – Funktionen, Abbildungen, Einschränkungen und funktionale Abhängigkeiten. Das funktionale Modell erfaßt, was ein System tut, unabhängig davon, wie oder wann es getan wird. Das funktionale Modell wird durch Datenflussdiagramme repräsentiert." (Rumbaugh et al. 1993, 22).

Die Transformation von Werten entspricht der imperativen Sichtweise, nicht der objektorientierten. Dennoch bestehen auch in objektorientierten Modellen funktionale Abhängigkeiten. Zur ihrer Darstellung werden Aktivitätsdiagramme verwendet:

> „Ein Aktivitätsdiagramm ist der Sonderfall eines Zustandsdiagramms, bei dem – fast – alle Zustände mit einer Verarbeitung verbunden sind. Ein Zustand wird verlassen, wenn die Verarbeitung beendet ist. Außerdem ist es möglich, eine Verzweigung des Kontrollflusses zu spezifizieren und zu beschreiben, ob die Verarbeitungsschritte in festgelegter oder beliebiger Reihenfolge ausgeführt werden können." (Balzert 1999, 534)

Deshalb wird in den Aufgabenklassen anstelle des Datenflussdiagramms das Aktivitätsdiagramm verwendet, wenngleich es Bedeutungsunterschiede gibt. Die Notwendigkeit der Offenlegung von Bedeutungsunterschieden gleich benannter Fachkonzepte in verschiedenen Publikationen, wie dargestellt, ist typisch für die Informatik, wenngleich auch in anderen Fachbereichen (z.B. in den Erziehungswissenschaften) unterschiedliche Begriffsverständnisse für Konzepte anzutreffen sind. Für Informatiklehrende stellt die Identifikation solcher Bedeu-

[61] Zu beachten ist hierbei das Publikationsjahr 1993. Es existierten damals verschiedene objektorientierte Vorgehensmodelle, die noch mehr oder weniger starke Bezüge zur strukturierten Programmierung beinhalteten. Die Methoden und der Begriffsapparat wurden erst mit der Entwicklung der UML seit Mitte der 90er Jahre stärker vereinheitlicht.

tungsunterschiede daher eine wichtige Kompetenz dar, um für Lernende möglichst widerspruchsfreie Fachbegriffsysteme konstruieren zu können.

Bildung von Aufgabenklassen

Verschiedene, so abstrahierte Übungsaufgaben können zur selben Aufgabenklasse gehören. Existiert für eine abstrahierte Übungsaufgabe noch keine passende Aufgabenklasse, so wird eine neue angelegt. Werden darin die gegebenen Angaben mit genau einem Arbeitsauftrag verknüpft, so handelt es sich um eine *elementare Aufgabenklasse*. Abbildung 3 zeigte hierfür ein Beispiel. Beziehen sich zwei Aufgabenklassen auf typgleiche Angaben, unterscheiden sich aber in ihren Arbeitsaufträgen, so werden sie als Varianten aufgefasst. Hierfür wird die Kurzschreibweise in Abbildung 4 verwendet.

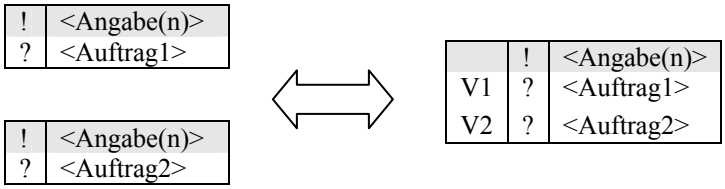


Abbildung 4: Bildung von Varianten zu Aufgabenklassen

Ist das Ergebnis einer ersten Aufgabenklasse typgleich zur Angabe in einer zweiten Aufgabenklasse, so wird die zweite als Ergänzung der ersten aufgefasst. Hierfür wird die Kurzschreibweise aus Abbildung 5 verwendet.

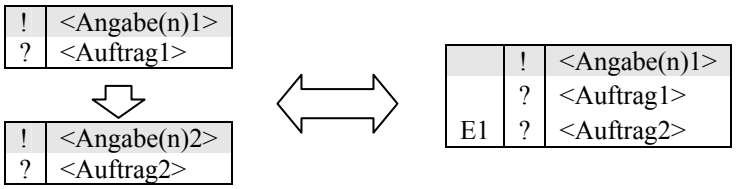


Abbildung 5: Bildung von Ergänzungen zu Aufgabenklassen

Kombinationen von Varianten und Ergänzungen sind möglich. Aufgabenklassen mit mehr als einem Arbeitsauftrag werden als *komplexe Aufgabenklassen* bezeichnet. Werden bei einer komplexen Aufgabenklasse für einen Teilauftrag zusätzliche Angaben erforderlich, so wird dies dargestellt, wie Abbildung 6 zeigt.

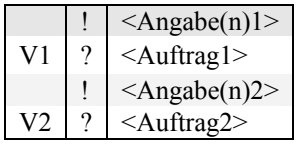


Abbildung 6: Komplexe Aufgabenklasse mit zusätzlichen Angaben

Die zusätzlichen Angaben (3. Zeile, hier hellgrau unterlegt) beziehen sich nur auf den direkt folgenden Auftrag. Eine komplexe Aufgabenklasse mit *n* Arbeitsaufträgen korrespondiert zu maximal *n* verschiedenen elementaren Aufgabenklassen. Mit der Kurzschreibweise soll die Gestaltung komplexer Übungsaufgaben mit vorstrukturierten Lösungswegen vereinfacht werden, indem Strukturen potentieller Varianten und Ergänzungen direkt in den Aufgabenklassen abgebildet werden (vgl. 4.5.2, Zwischenresümee 4.2).

Ergebnisse

Bei der Entwicklung der Aufgabenklassen zum statischen Systemmodell wurden 17 elementare und 5 komplexe Aufgabenklassen (vgl. Anhang C.1) gebildet. Innerhalb der komplexen Aufgabenklassen wurde hierbei zunächst nicht zwischen Varianten und Ergänzungen unterschieden. Der Arbeitsauftrag bestand jeweils aus einer Sequenz von Teilaufträgen, von denen eine Auswahl in einer konkreten Aufgabe berücksichtigt werden konnte. Um die Gestaltung komplexer Aufgaben mit vorstrukturierten Lösungswegen mittels Aufgabenklassen zu vereinfachen, wurde die im vergangenen Abschnitt eingeführte Kurzschreibweise entwickelt und angewandt bei der Abstraktion der im Abschnitt 4.3.1 ausgewählten Teilmenge aller Übungsaufgaben. Hierbei entstanden 31 elementare und 28 komplexe Aufgabenklassen, die im Anhang C.2 dokumentiert sind.

4.3.3 Strukturierung der Aufgabenklassen

Ziel der Strukturierung ist es, die identifizierten Aufgabenklassen (vgl. 4.3.2) so zu ordnen, dass für Lernende und Lehrende eine zielgerichtete Auswahl für gegebene Lehr-Lern-Situationen vereinfacht wird. Zur Strukturierung werden die identifizierten Aufgabenklassen hinsichtlich drei Eigenschaftsdimensionen analysiert und klassifiziert. Diese Dimensionen sind

- der *Fachkern,*
- der *Gegenstand* und
- der *Aufgabentyp*

einer Aufgabenklasse. Die genannte Reihenfolge der Dimensionen spiegelt die vermutete Reihenfolge wider, in der eine Auswahl von Aufgabenklassen zur Konstruktion einer Übungsaufgabe erfolgt. Zur Bestimmung der zugehörigen Kategorien wurden die identifizierten Aufgabenklassen hinsichtlich dieser drei Dimensionen analysiert.

Fachkern

Die zentralen Kategorien bilden das statische und das dynamische Modell sowie deren Verknüpfung. Unterkategorien verbinden Aufgabenklassen mit zentralen Tätigkeiten oder Artefakten im Kontext der übergeordneten Kategorie. Folgende Kategorien und Unterkategorien wurden abgeleitet:

- *Statisches Modell*
  - Objekte und Objektstrukturen,
  - Klassen und Klassenstrukturen,
  - Verknüpfung von Objekt- und Klassenstrukturen.
- *Dynamisches Modell*
  - Anwendungsfälle,
  - Funktionale Abhängigkeiten,
  - Objektinteraktion,
  - Objektzustände,
  - Verknüpfung von Objektinteraktion und -zuständen.
- *Verknüpfung des statischen und dynamischen Modells*
  - Anforderungsanalyse,
  - Verknüpfung von Teilmodellen,
  - Analyse- und Entwurfsmuster,
  - Prototyp einer Benutzungsoberfläche.

Bei der Analyse der Aufgabenklassen zum statischen Modell wurde diese Dimension zunächst nicht berücksichtigt. Mit der Berücksichtigung von Aufgabenklassen aus anderen Teilbereichen wurde eine Klassifizierung hinsichtlich des Fachkerns erforderlich. Da sich als ein Resultat der empirischen Untersuchung ein Bedarf an Aufgabenklassen zur Festigung des Unterschiedes zwischen Objekt- und Klassenkonzept zeigte (vgl. 4.5.2, Zwischenresümee 4.2), wurden an dieser Stelle Unterkategorien eingeführt, mit denen Aufgabenklassen zu wichtigen Fachkonzepten expliziter zur Auswahl angeboten werden können.
Bei der Gestaltung von Übungsaufgaben für den Kurs, in dem die Probleme mit dem Objekt- und Klassenkonzept auftraten (vgl. 4.5.1, Kurs A-11), wurden keine Aufgaben aus dem Bereich der Verknüpfung von Objekt- und Klassenstrukturen berücksichtigt, obwohl die verwendete Aufgabenklassensammlung (vgl. Anhang C.1) auch Aufgabenklassen hierzu bereit stellte. Mögliche Ursachen hierfür liegen in der Struktur der Aufgabenklassen, die deswegen hinsichtlich des Fachkerns verfeinert wurde, und möglicherweise darin, dass die aufgetretenen Probleme im Vorfeld nicht genug bekannt waren.
Einen Sonderfall stellen Aufgabenklassen aus dem Bereich der Metamodellierung dar, bei denen objektorientierte Modelle z.B. objektorientierter Modellierungstechniken erstellt werden. Eine Zuordnung von entsprechenden Aufgabenklassen zum statischen oder dynamischen Modell kann abhängig davon erfolgen, ob die bei der Modellierung angewandten Techniken dem statischen oder dem dynamischen Modell zuzuordnen sind (vgl. C.2.6).

Gegenstände

Hierbei lassen sich folgende Varianten unterscheiden: Aufgabenklassen zu

- *Fachkonzepten der Objektorientierung (OO-Konzepte),*
- einem oder mehreren *Modellelementen,*
- einem oder mehreren *Modellen.*

Bedingt durch die ausgewählte Aufgabenteilmenge wurden beim statischen Modell zunächst keine Aufgabenklassen zu OO-Konzepten berücksichtigt.

Aufgabentypen

Folgende Aufgabentypen zum OOM wurden bei der Analyse der Aufgabenklassen identifiziert:

- *Wissensfragen,*
- *Verständnisfragen,*
- Beschreibungsaufgaben,
- Zuordnungsaufgaben,
- Spezifikationsaufgaben,
- *Ordnungsaufgaben,*
- *Diskussionsaufgaben,*
- Analyseaufgaben,
- *Vergleichsaufgaben,*
- *Validierungsaufgaben,*
- Identifikationsaufgaben,
- Modifikationsaufgaben,
- *Transformationsaufgaben,*
- Konstruktionsaufgaben.

Die Analyse der Aufgabenklassen zum statischen Systemmodell lieferte zunächst nur die Aufgabentypen, die nicht kursiv gesetzt sind. Tabelle 8 zeigt den Zusammenhang zwischen den Aufgabentypen und den Stufen der Bloom'schen Taxonomie für kognitive Lernziele

(Bloom 1976). Dunklere Einfärbung bzw. mehr Punkte in einer Tabellenzelle repräsentieren dabei eine stärkere Zuordnung des Aufgabentyps zur jeweiligen Lernzielstufe. Bei der Zuordnung von Modellelementen zu Modellen oder Modellen zu bestimmten Angaben (Beispiele: Zeile 3.1 in Tabelle 10 auf S. 76), bei der Spezifikation von Modellelementen und bei der chronologischen oder sachlogischen Ordnung von Modellelementen (beim dynamischen Modell) gibt es geringe Analyseelemente. Bei Diskussions-, Analyse-, Vergleichs- und Validierungsaufgaben steht die Analyse im Vordergrund, das Anwenden ist sekundär.

	1. Wissen	2. Verstehen	3. Anwenden	4. Analysieren	5. Synthetisieren	6. Bewerten
Wissensfragen	●●’					
Verständnisfragen	●●	●●’				
Beschreibungsaufgaben	●●	●●’				
Zuordnungsaufgaben	●●	●●	●●’	●		●
Spezifikationsaufgaben	●●	●●	●●’	●		●
Ordnungsaufgaben	●●	●●	●●’	●		●
Diskussionsaufgaben	●●	●●	●	●●’		●
Analyseaufgaben	●●	●●	●	●●’		
Vergleichsaufgaben	●●	●●	●	●●’		●
Validierungsaufgaben	●●	●●	●	●●’		
Identifikationsaufgaben	●●	●●	●●	●●’		
Modifikationsaufgaben	●●	●●	●●	●●	●●’	●
Transformationsaufgaben	●●	●●	●●	●●	●●’	●
Konstruktionsaufgaben	●●	●●	●●	●●	●●	●●’

Tabelle 8: Verknüpfung der Aufgabentypen mit der Bloom'schen Lernzieltaxonomie

Bei Zuordnungs-, Spezifikations-, Ordnungs-, Diskussions-, Vergleichs-, Modifikations- und Transformationsaufgaben gibt es geringe Variationsmöglichkeiten bei der Lösung. Lernende müssen alternative Lösungsmöglichkeiten bewerten und aufgrund dessen eine Entscheidung treffen. In besonderem Maße gilt dies bei nicht geführten Konstruktionsaufgaben, wo zwischen einer Vielzahl von möglichen Varianten eine Entscheidung zu treffen ist. Es zeigt sich somit, dass Aufgaben zum OOM für alle kognitiven Anforderungsstufen entwickelt werden können. Für die weitere Betrachtung werden die Aufgabentypen jeweils der höchsten Lernzielstufe zugeordnet, zu der eine Zuordnung maximalen Grades besteht (in Tabelle 8 markiert durch ein zusätzliches „’“ in der jeweiligen Tabellenzelle).

Verknüpfung von Gegenständen und Aufgabentypen

Nicht alle der identifizierten Aufgabentypen lassen sich sinnvoll mit OO-Konzepten, Modellelementen bzw. Modellen verknüpfen. Tabelle 9 zeigt sinnhafte Kombinationen.

Nr.	Aufgabentyp	#	I. OO-Konzept(e)	II. Modellelement(e)	III. Modell(e)
1	Wissensfrage	2	2	(3)	(3)
2.1	Verständnisfrage	2	2	(3)	(3)
2.2	Beschreibungsaufgabe	1	2	1	1
3.1	Zuordnungsaufgabe	1	3	1	2
3.2	Spezifikationsaufgabe	1	–	1	3
3.3	Ordnungssaufgabe	2	(3)	2	(3)
4.1	Diskussionsaufgabe	2	3	2	2
4.2	Analyseaufgabe	1	3	1	1
4.3	Vergleichsaufgabe	2	3	2	2

Nr.	Aufgabentyp	#	I. OO-Konzept(e)	II. Modell-element(e)	III. Modell(e)
4.4	Validierungsaufgabe	2	–	3	2
4.5	Identifikationsaufgabe	1	3	1	(3)
5.1	Modifikationsaufgabe	1	(3)	2	1
5.2	Transformationsaufgabe	2	(3)	–	2
6	Konstruktionsaufgabe	1	(3)	–	1

Tabelle 9: Verknüpfung von Gegenständen und Aufgabentypen

In Tabelle 9 korrespondiert die erste Ziffer der *Nr.* mit der zum Aufgabentyp jeweils zugehörigen Stufe der kognitiven Lernziele nach Bloom (1976), die jeweils zweite Ziffer dient der laufenden Nummerierung. In der Spalte „#“ gibt die Ziffer 1 an, dass der Aufgabentyp bereits bei der Analyse der Aufgabenklassen zum statischen Modell identifiziert wurde (1. Durchgang). Die mit der Ziffer 2 markierten Aufgabentypen wurden nach der Analyse aller Aufgabenklassen ergänzt (2. Durchgang). In den mit den Gegenständen korrespondierenden letzten drei Spalten der Tabelle haben die Ziffern 1 und 2 dieselbe Semantik, wie in der Spalte „#“, nur übertragen auf die jeweilige Verknüpfung von Gegenstand und Aufgabentyp. Die mit der Ziffer 3 markierten Verknüpfungen wurden vom Autor der vorliegenden Arbeit durch Analogieschluss ergänzt. Beispiele hierfür waren in den analysierten Quellen nicht enthalten. Eingeklammerte Ziffern zeigen dabei an, dass die jeweilige Verknüpfung zwar prinzipiell möglich ist, aus fachdidaktischen Gründen aber nicht empfohlen wird. Mit einem Strich versehene Verknüpfungen sind nicht definiert oder werden bereits durch andere Verknüpfungen abgedeckt. Ansätze hierzu waren bereits vorhanden bei Crutzen / Hein (1995), vgl. 2.3.1.1, S. 16. Sie empfahlen als eine von vier didaktischen Linien „Sehen – Verstehen – Ändern – Selbertun“ und kamen zu dem Ergebnis, dass objektorientiertes Denken besonders gut zu erlernen sei, wenn entsprechend den von ihnen vorgestellten Linien objektorientierte Modelle geändert, verfeinert, erweitert oder mit anderen Modellen kombiniert werden. Die Stufen der vorgeschlagenen Linie korrespondieren zu Stufen kognitiver Lernziele und das Ändern, Verfeinern, Erweitern und Kombinieren von Modellen korrespondiert zu vorgeschlagenen Aufgabentypen.

Zur Erhöhung der Anschaulichkeit werden die in Tabelle 9 markierten Verknüpfungen in Tabelle 10 mit Beispielen für Aufgaben bzw. Aufgabenklassen untersetzt. Wissens- und Verständnisfragen zu Modellelementen und Modellen sind, wie in Tabelle 10 gezeigt, prinzipiell möglich, stehen hier aber nicht im fachdidaktischen Fokus. Ordnungsaufgaben zu OO-Konzepten bzw. Modellen lassen sich ebenfalls prinzipiell gestalten, wenn man beispielsweise eine „ist geeigneter als“-Relation im Hinblick auf ein gegebenes Kriterium zugrunde legt.

Bei den in Tabelle 10 gezeigten Beispielen sind allerdings zunächst erhebliche Analyse- und Bewertungsleistungen zu erbringen, bevor ein Ordnen in diesem Sinne erfolgen kann, welches dann eigentlich trivial ist. Vom Anforderungsniveau her sind solche Aufgaben also eher den Analyseaufgaben zuzuordnen. Bei verschiedenen Aufgaben zu OO-Konzepten ist ein Bezug zu einem konkreten Modell nahe zu legen, um die für schulische Lehr-Lern-Prozesse so wichtige Anschaulichkeit und Exemplarität sicherzustellen. Auch viele der Aufgaben zu den Modellelementen beziehen ein konkretes Modell als Bezugsrahmen mit ein. Damit zeigt sich, dass eine Zuordnung zu den vorgeschlagenen Verknüpfungen nicht immer völlig trennscharf möglich ist und dass sich in der Praxis Mischformen identifizieren lassen. Bei den Identifikationsaufgaben ist es prinzipiell möglich, dass einer textuellen Beschreibung zugrunde liegende Modell (z.B. eines Klassendiagramms) zu identifizieren, wenn es darin hinreichend genau beschrieben ist. Üblicher ist allerdings die Aufgabenstellung, Modellelemente bzw. Kandidaten für solche in Texten zu identifizieren und auf der Basis dieser, in der Regel unvollständigen, Gestaltungselemente ein Modell zu konstruieren.

Nr.	Aufgabentyp	I. OO-Konzept(e) (abstrakte Konzepte)	II. Modellelement(e) (Bezug zu konkretem Modell)	III. Modell(e) (Bezug zu konkretem Modell)
1	Wissensfrage	Bestandteile einer Klasse nennen	(das Element nennen, das im Modell bestimmten Teilaspekt modelliert)	(in früherer Lerneinheit gemeinsam erstelltes Objektdiagramm zeichnen)
2.1	Verständnisfrage	Unterschied zwischen Klasse und Objekt erläutern	(Unterschied zwischen zwei Modellelementen erläutern)	(erläutern, warum bestimmter Sachverhalt als eigene Klasse modelliert wurde)
2.2	Beschreibungsaufgabe	Vererbungskonzept beschreiben	eine Klasse eines Klassendiagramms beschreiben	Sequenzdiagramm mit eigenen Worten beschreiben
3.1	Zuordnungsaufgabe	Aussagen der Form „ein <x> {hat | ist | kennt} ein <y>“ den Konzepten Assoziation, Aggregation, Vererbung zuordnen	als Liste gegebene Begriffe den Kategorien Klasse, Attribut, Methode etc. zuordnen	gegebenes Klassen- und Sequenzdiagramm gegebenen Fragen zu deren Beantwortung zuordnen
3.2	Spezifikationsaufgabe	–	Attribute / Methoden / Assoziationen in einem Klassendiagramm spezifizieren	informelles Klassendiagramm spezifizieren
3.3	Ordnungssaufgabe	(beurteilen, welche Art der Klassenrelation am besten geeignet ist, einen gegebenen Zusammenhang zwischen zwei Klassen zu modellieren)	textuell beschriebene Prozesse in partielle Ordnung bringen	(beantworten, welches von *n* gegebenen Klassendiagrammen am besten für einen gegebenen Algorithmus zum „Türme von Hanoi“-Problem geeignet ist)
4.1	Diskussionsaufgabe	Diskutieren, welche Vor- und Nachteile die Anwendung eines bestimmten Entwurfsmusters auf eine Klassenstruktur hat	diskutieren, inwieweit das Hinzufügen einer bestimmten Assoziation zu einem Klassendiagramm eine Verbesserung darstellt	diskutieren, inwieweit ein Klassendiagramm zur Erfüllung gegebener Anforderungen geeignet ist
4.2	Analyseaufgabe	Eignung von Assoziation, Aggregation, Komposition, Vererbung im Hinblick auf die Modellierung eines gegebenen Zusammenhangs zwischen zwei Klassen untersuchen	Objekt, Verwendungszweck, Liste von Attributen gegeben; Eignung der Attribute im Hinblick auf den Verwendungszweck untersuchen	beschreiben, wie gegebene Anfragen der Form „finde alle Objekte der Klasse <x>, für die gilt, dass ...“ mittels eines gegebenen Klassendiagramms bearbeitet werden können
4.3	Vergleichsaufgabe	Eigenschaften von Klassenmethode und Methode vergleichen	Klassen im Hinblick auf strukturelle Gemeinsamkeiten untersuchen	zwei unterschiedliche Klassenstrukturen zum gleichen Sachverhalt im Hinblick auf ihre Wiederverwendbarkeit vergleichen
4.4	Validierungsaufgabe	–	prüfen, ob eine Klasse mit einem Sequenzdiagramm konsistent ist	Konsistenz eines Klassen- und eines Sequenzdiagramms prüfen
4.5	Identifikationsaufgabe	Entwurfsmuster in einem Klassendiagramm entdecken	in gegebenem Text Klassenkandidaten identifizieren	(im Text beschriebenes Diagramm identifizieren)
5.1	Modifikationsaufgabe	s. Erläuterung auf der nächsten Seite	einen in einem Text dargestellten Multiplizitätszusammenhang zwischen zwei Klassen an der betroffenen Assoziation ändern	ein Aktivitätsdiagramm so ändern, dass zuvor sequentielle Aktivitäten anschließend parallel dargestellt werden
5.2	Transformationsaufgabe		–	Klassendiagramm der objektorientierten Analyse in ein Klassendiagramm des objektorientierten Entwurfs überführen
6	Konstruktionsaufgabe		–	zu gegebenem Text Klassendiagramm konstruieren

Tabelle 10: Beispiele für Aufgaben(-klassen) zur Verknüpfung von Gegenständen und Aufgabentypen

Die Übertragung der „Transformation in ein OOE-Modell“ vom Modell auf ein Modellelement führt zu bereits bestehenden Verknüpfungen. Alle Komponenten werden spezifiziert (Modellelement(e) spezifizieren) und ggfs. neue Modellelemente aufgrund von Entwurfserwägungen hinzugefügt (Modell(e) modifizieren). Ein einzelnes Modellelement wird nicht konstruiert, sondern einem, gegebenenfalls auch leeren, vorhandenen Teilmodell hinzugefügt (Modell(e) modifizieren). Prinzipiell denkbar wären Aufgaben zur Modifikation, Transformation und Konstruktion von OO-Konzepten. Das setzt allerdings extrem leistungsstarke Lerngruppen voraus, die auf der Basis vorhandener Konzepte und neuen Anforderungen selbstständig Konzepte konstruieren. Zu beachten ist allerdings, dass in diesem Fall der Konstruktionsbegriff eher lerntheoretischer Natur ist und nicht informatischer, wie bei den Modellen.

Klassifikationsstruktur

Durch die Klassifizierung von Fachkern, Gegenstand und Aufgabentyp lassen sich elementare Aufgabenklassen in eine dreidimensionale Struktur einordnen (vgl. Abbildung 7).

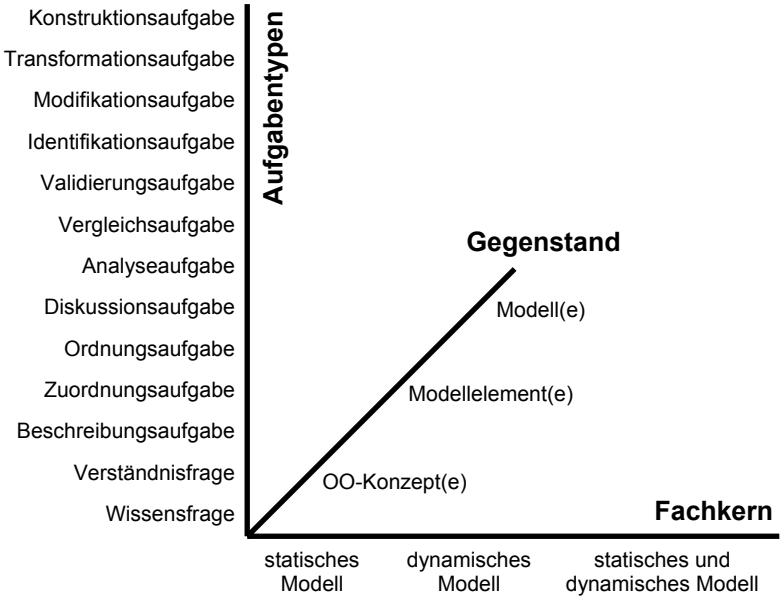


Abbildung 7: Klassifikationsstruktur für (elementare) Aufgabenklassen

Komplexe Aufgabenklassen verbinden einen Fachkern mit verschiedenen Gegenständen und bzw. oder verschiedenen Aufgabentypen. Sie korrespondieren daher zu mehreren Einträgen in die Struktur.

Ergebnisse

1. Statisches Modell – Erste Fassung einer strukturierten Aufgabenklassensammlung

Bei den Aufgabenklassen zum statischen Modell ist der Fachkern gegeben. Als Gegenstände wurden aufgrund der ausgewählten Aufgabenteilmenge zunächst nur Modellelemente[62] und Modelle und als Aufgabentypen Beschreibungs-, Zuordnungs-, Spezifikations-, Analyse-, Identifikations-, Modifikations- und Konstruktionsaufgaben berücksichtigt. Zur Darstellung der Zusammenhänge wurde eine Baumstruktur gewählt. Die Verknüpfung von Aufgabentypen und Gegenständen bildet in dieser Baumstruktur die Blätter. Der Wurzelknoten ist durch die Modellierung eines Informatiksystems gegeben. Die Konstruktion eines statischen Modells ist hierzu eine Teilaufgabe, die sich wiederum aus der Identifikation, Charakterisierung

[62] Bei der Entwicklung von Aufgabenklassen zum statischen Systemmodell wurde hierfür zunächst der Begriff „Merkmal“ verwendet, vgl. (Brinda 2000b) und (Brinda / Schubert 2001). Während der Überarbeitungsphase wurde er durch den Modellelementbegriff ersetzt, da er die Bedeutung besser wiedergibt.

und Strukturierung von Objekten, Klassen und Relationen zusammensetzt. Inneren Knoten und Blättern ist jeweils eine nichtnegative Anzahl von Aufgabenklassen zugeordnet. In dieser Struktur haben inneren Knoten entsprechende Aufgabenklassen Sicherungscharakter für in der Hierarchie tiefer befindliche Aufgabenklassen, da zu ihrer Bearbeitung Wissen bzgl. der Teile erforderlich ist (vgl. Abbildung 8).

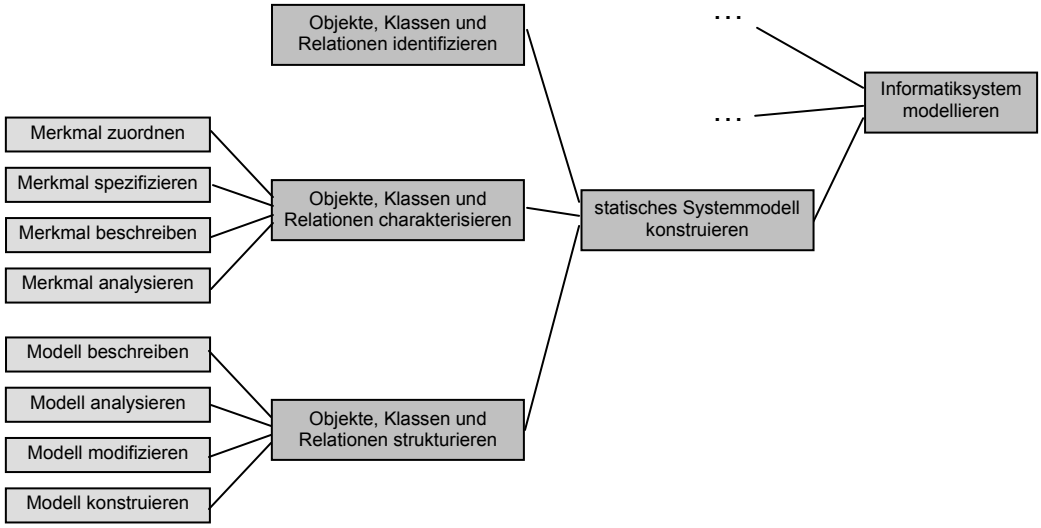


Abbildung 8: Baumstruktur von Aufgabenklassen zum statischen Systemmodell

In diese Struktur eingeordnete Aufgabenklassen bildeten die erste Fassung einer strukturierten Sammlung von Aufgabenklassen zum objektorientierten Modellieren (vgl. Anhang C.1). Diese Sammlung, publiziert mit ihrer Entwicklungsmethodik, Beispielaufgaben und Musterlösungen (Brinda 2000b), stellte die Grundlage für die im Rahmen dieser Arbeit durchgeführten empirischen Untersuchungen zu Aufgabenklassen (vgl. 4.5 und 4.5) dar.

2. Verfeinerte und überarbeitete Fassung einer Aufgabenklassensammlung

Für die Auswahl einer Aufgabenklasse für einen Lehr-Lern-Prozess hat der Fachkern die größte Bedeutung. Die Sammlung wird daher auf oberster Ebene nach dieser Dimension untergliedert. Eine Schwierigkeit stellt die Klassifizierung hinsichtlich Gegenstand und Aufgabentyp von komplexen Aufgabenklassen dar. Die beim statischen Modell verwendete Baumstruktur ist daher hier nicht praktikabel. Aus diesem Grund werden die Aufgabenklassen innerhalb einer Fachkern-Kategorie mit einer Kennung für Gegenstand und Aufgabentyp versehen und bezüglich dieser Kennung partiell geordnet. Die Kennung wird gebildet aus Komponenten von Tabelle 9, wie folgt:

$$(I|II|III).<Nr.>$$

Modellelemente vergleichen hat demnach die Kennung *II.4.3*[63]. Es ergeben sich damit vielfältige Strukturierungsmöglichkeiten der Aufgabenklassen für Lehr-Lern-Prozesse. Innerhalb einer Fachkern-Kategorie können die Aufgabenklassen partiell geordnet werden nach ihrer Zugehörigkeit zu einer Stufe der Bloom'schen Taxonomie (erster Bestandteil der „Nr." in Tabelle 9), nach dem Aufgabentyp (vollständige „Nr.") sowie nach dem Aufgabentyp bezogen auf den Gegenstand (vollständige Kennung). Durch die partielle Ordnung hinsichtlich der Kennung stehen innerhalb einer Fachkern-Kategorie Aufgabenklassen zu OO-Konzepten vor Aufgabenklassen zu Modellelementen und diese wiederum vor Aufgabenklassen zu Modellen. Innerhalb dieser Kategorien sind die Aufgabenklassen nach steigendem kognitivem Anforderungsniveau partiell geordnet. Beides ist allerdings lediglich eine Tendenz-Angabe, da

[63] Für die fachdidaktische Kommunikation und Anwendung wäre es wünschenswert, die hier gewählten durch „sprechendere" Bezeichner zu ersetzen.

sich komplexe Aufgabenklassen oft auf vielfältige Gegenstände in vielfältigen Aufgabentypen beziehen. Eine strikte Ordnung ließe sich auf der Basis elementarer Aufgabenklassen erreichen. Damit würden allerdings die Vorteile, die komplexe Aufgabenklassen in ihrer jeweiligen Struktur von Varianten und Ergänzungen bieten, verloren gehen. Die resultierende strukturierte Sammlung der Aufgabenklassen befindet sich im Anhang im Abschnitt C.2.

Repräsentation der Struktur von Aufgabenklassen

Zur besseren Veranschaulichung der Vernetzung der Aufgabenklassen sowie zur expliziten Darstellung von darin enthaltenen Strukturen von Varianten und Ergänzungen wurde deren grafische Darstellung untersucht. Verwendet werden darin drei Arten von Knoten. Die in Rechtecken dargestellten Texte repräsentieren Angaben bzw. Ergebnisse von Aufgabenklassen. Ergebnisse einer Aufgabenklasse können als Angabe für eine andere dienen. Texte in Rechtecken mit abgerundeten Ecken repräsentieren Arbeitsaufträge. Die Verknüpfung verschiedener Angaben für einen Arbeitsauftrag wird mittels eines Verknüpfungspunktes (Kreis mit „∧“-Zeichen) dargestellt. Ferner gibt es zwei Arten von gerichteten Kanten. Durchgezogene Linien repräsentieren Verbindungen zwischen Angaben und Aufträgen, die sich aus der Analyse der Aufgabenklassen ergaben. Gestrichelte Linien zeigen weitere Verknüpfungsmöglichkeiten ohne Analogon in den Aufgabenklassen. Abbildung 9 zeigt die Repräsentation der Vernetzung am Beispiel der Aufgabenklassen zu Objekten und Objektstrukturen (vgl. Anhang C.2.3.1).

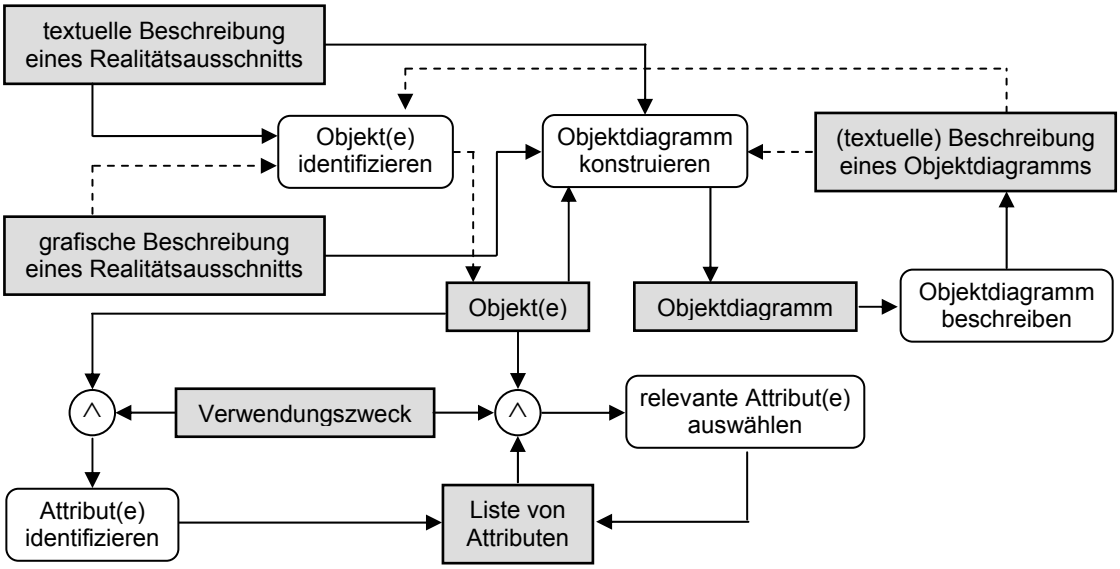


Abbildung 9: Beispiel für die Repräsentation der Vernetzung von Aufgabenklassen

Diese Repräsentationsform ist gut dazu geeignet, für einen abgeschlossenen Bereich die Struktur zu explizieren und Ideen für neue Verknüpfungsmöglichkeiten daran zu entwickeln. Für komplexere Teilbereiche (z.B. C.2.3.2) besteht aber die Gefahr der Unübersichtlichkeit, aufgrund von vielfältigen Komponenten und deren Vernetzung. Diese Strukturrepräsentation kann als Mittel für die Gestaltung von Übungsaufgaben angesehen werden (vgl. 4.4). Ausgangspunkt der Gestaltung kann eine beliebige Menge von Angaben sein (Rechtecke in Abbildung 9). In Richtung der davon ausgehenden Kanten zeigen sich mögliche Aufgabenklassen für die gewählten Angaben. Mehrere ausgehende Kanten repräsentieren verfügbare Varianten. Durch einen Pfad verbundene Aufträge zeigen potentielle Ergänzungen. Die Kommunikation, der Erfahrungsaustausch und die fachliche Diskussion zwischen Lehrenden und bzw. oder Fachdidaktikern in diesem Bereich werden hierdurch unterstützt und gefördert. Es zeigt sich die Bedeutung für das Berufsfeld.

4.4 Gestaltung von Aufgaben mittels Aufgabenklassen

4.4.1 Vorüberlegungen

Um Aufgaben zu einem bestimmten Unterrichtsinhalt zu konstruieren, sind zunächst geeignete Aufgabenklassen aus der Sammlung (vgl. Anhang C) auszuwählen. Diese Auswahl kann sich am Fachkern, am Gegenstand und bzw. oder am Aufgabentyp orientieren (vgl. 4.3.3). Verschiedene Aufgabenklassen sind hierbei kombinierbar, um umfangreichere Aufgaben zu entwerfen. Weitere Gestaltungsmöglichkeiten im Hinblick auf Schwierigkeitsgrad und Niveaustufungen ergeben sich durch geeignete Variation der mit einer Aufgabe bereitzustellenden Angaben und des Arbeitsauftrages (vgl. 4.4.2). Ferner ist für die Aufgabe ein geeigneter Kontext zu wählen (vgl. 4.4.3). Zu beachten sind dabei Ergebnisse zu typischen Missverständnissen im Bereich der Objektorientierung (vgl. Holland et al. 1997 bzw. Darstellungen dazu auf S. 34 im Abschnitt 2.3.2 der vorliegenden Arbeit).

4.4.2 Niveaustufungen

Durch die Variation des Aufgabentyps lassen sich Aufgaben für unterschiedliche kognitive Niveaustufungen gestalten (vgl. 4.3.3). Durch die Veränderung bestimmter Eigenschaften der mit einer Aufgabe bereit gestellten Angaben sind deren Schwierigkeitsgrad und Komplexität zusätzlich zu beeinflussen (vgl. 4.3.1). Einen Beleg für den Bedarf an solchen Empfehlungen zeigte die Erprobung des Aufgabenklassenkonzepts im Rahmen des Lehramtsstudiums (vgl. 4.6.1.1). Zur Bestimmung der zu variierenden Einflussgrößen wurden die Angaben der ausgewählten Aufgaben (vgl. Anhang B) einer Strukturanalyse unterzogen hinsichtlich folgender Kriterien:

- *Form von Angaben,*
- *Komplexität von Angaben,*
- *Verfügbarkeit von Angaben,*
- *Verwendungshäufigkeit von Angaben.*

Ergebnisse

Bei der *Form von Angaben* lassen sich unterscheiden:

- *Texte*
  Textuelle Angaben für den Arbeitsauftrag liegen entweder explizit vor (z.B. als Aufzählung) oder müssen selbstständig aus einem Fließtext erschlossen werden.
- *grafische Darstellungen*
  Grafische Darstellungen sind entweder frei gestaltete Abbildungen bzw. Schemata zu der Aufgabe oder ein oder mehrere Diagramme in einer Modellierungssprache, aus denen aufgabenrelevante Information vom Lernenden extrahiert werden muss.

Mischformen sind möglich. Die *Komplexität von Angaben* ergibt sich aus:

- *Anzahl der Komponenten,*
- *Komplexität der einzelnen Komponenten,*
- *Vernetzungsgrad der Komponenten,*
- *Auswahl der verwendeten Fachkonzepte.*

Dies gilt gleichermaßen für Texte und grafische Darstellungen. Die Auswahl der verwendeten Fachkonzepte beeinflusst die drei zuvor genannten Aspekte, ist somit orthogonal zu ihnen. Mehrere, komplexere und bzw. oder stärker vernetzte Komponenten, die mehrere Fachkon-

zepte berücksichtigen, führen zu einer Erhöhung des Schwierigkeitsgrades. Es lässt sich daraus allerdings nicht die Schlussfolgerung ziehen, dass ein längerer Text bzw. eine komplexere Grafik als Angabe zwangsläufig zu einer komplexeren Aufgabe führt. Am Beispiel einer textbasierten Modellierungsaufgabe sei dies illustriert. Die Aufgabe „Modellieren Sie eine Schulbibliothek." erfordert mehr Kreativität als die Aufgabe „Modellieren Sie eine Schulbibliothek, in der Lernende und Lehrende Bücher und CDs ausleihen können.". Noch einfacher wäre es, wenn eine genaue Beschreibung der Kunden- und Medienverwaltung sowie des Ausleihverfahrens und der über die einzelnen Objekte zu verwaltenden Attribute im Aufgabentext nachzulesen wären. In diesem Beispiel erhöht sich mit der Textlänge die *Verfügbarkeit von Angaben*, was zur Konsequenz hat, dass weniger Information selbstständig beigetragen werden muss. Folgende Varianten lassen sich hierbei unterscheiden:

- *aufgabenrelevante Angaben aus gegebener Aufzählung übernehmen,*
- *aufgabenrelevante Angaben in einem Fließtext oder einer Grafik identifizieren und auswählen,*
- *aufgabenrelevante Angaben in einem Fließtext oder einer Grafik identifizieren, auswählen und fehlende Angaben ergänzen.*

Je mehr Angaben Lernende selbstständig identifizieren bzw. selbst beitragen müssen, desto schwieriger wird die Aufgabe. Die Verfügbarkeit von Angaben korreliert hierbei mit dem Führungsgrad der Aufgabe. Je mehr Angaben Lernende selbstständig beitragen müssen, desto größer ist der Spielraum für eine Lösung. Mehr Variationsmöglichkeiten erfordern von den Lernenden mehr Bewertungskompetenz, ob es sich bei einer gestalteten Lösung um eine sinnvolle handelt. Es steigt aber auch die Gefahr der Fehlinterpretation einer Aufgabe.
Die *Verwendungshäufigkeit von Angaben* bezieht sich insbesondere auf Aufgabentypen, bei denen aufgabenrelevante Angaben als Aufzählung gegeben sind (s.o.). Mögliche Varianten sind hierbei:

- *jede Angabe genau einmal verwenden,*
- *jede Angabe mindestens einmal verwenden (1...*),*
- *jede Angabe beliebig oft verwenden (0...*).*

Der letzte Fall steht für für die Aufgabenlösung irrelevante Informationen im Text. Klar ist, dass eine größere Anzahl von Freiheitsgraden mehr Eigenarbeit der Lernenden und eine intensivere Analyse der Aufgabenstellung erfordert.
Durch die Variation dieser Ausprägungen von Aufgabenangaben lassen sich Schwierigkeitsgrad und Anforderungen an die Lernenden variieren. Zusammen mit der Variation des Aufgabentyps ist hierdurch das gesamte Spektrum von sehr einfachen Aufgaben für Anfängerinnen und Anfänger bis hin zu sehr schwierigen Aufgaben für fortgeschrittene Lernende der Sekundarstufe II abzudecken. Es ergeben sich dadurch auch Möglichkeiten der Binnendifferenzierung.

4.4.3 Auswahl des Kontextes

Auswahlkriterien

Für die Gestaltung von Aufgaben sind neben unterrichtsgeeigneten, abstrakten Aufgabenklassen auch konkrete und motivierende Kontexte erforderlich. Ein Kontext liefert den inhaltlichen Rahmen für die jeweilige Aufgabenstellung. Aufgrund der Analyse der Übungsaufgaben wurden folgende Kriterien für die Auswahl von Kontexten abgeleitet:

- *Eignung für OOM*: Nicht jeder Kontext legt ein objektorientiertes Vorgehen gleichermaßen nahe (vgl. z.B. Füller 1999). Keinesfalls sollte bei gegebenem Kontext ein objektorientiertes Vorgehen durch den Lehrenden erzwungen werden. Eine Ausnahme

hiervon kann lediglich ein Vergleich der Eignung verschiedener Programmierparadigmen zur Lösung derselben Problemstellung sein.

- *Leichte Änderbarkeit und Erweiterbarkeit:* Modellierungstechniken, wie Vererbungsstrukturen und abstrakte Klassen, zeigen ihre Qualität erst, wenn bestehende Strukturen verändert bzw. erweitert werden (Appelrath et al. 1998). Das setzt einen entsprechend offenen Kontext voraus, in dem Erweiterungen und Modifikationen der Struktur möglich und sinnvoll sind. In diesem Zusammenhang kann zwischen größeren Projekten und einer Verkettung kleinerer Beispiele unterschieden werden. Ein größeres Projekt kann von vornherein so gewählt werden, dass die Anwendung der o.g. Modellierungstechniken Vorteile bringt. Die Erstellung einer Gesamtlösung kann, je nach Projektgröße, sehr zeitintensiv sein und damit zum Motivationsverlust bei den Lernenden führen. Bei einer Folge aufeinander aufbauender kleinerer Beispiele kann das Ende flexibler gewählt werden. Beispielsweise können verschiedene Klassenstrukturen erstellt, modifiziert und die Qualität der Techniken in diesem Zusammenhang bewertet werden. Erweiterbare Kontexte, die auch Verknüpfungen unterschiedlicher Fachkonzepte ermöglichen, fördern eine angemessene Binnendifferenzierung.
- *Lebensweltbezug*: Der Kontext sollte aus der Lebenswelt der Lernenden stammen. Diese sollten die vielfältigen Zusammenhänge des Realitätsausschnitts aufgrund eigener Erfahrung kennen oder hinreichend viel Information darüber recherchieren können. Ein unbekannter Kontext erfordert zunächst eine zeitaufwendige Auseinandersetzung damit. Die Aufmerksamkeit kann dann nicht auf das Ziel, informatisches Modellieren zu erlernen, konzentriert werden.
- *Motivation*: Ein Kontext muss motivierend sein, um die Aufmerksamkeit der Lernenden zu binden und ihr Interesse für die weitere Beschäftigung mit den Inhalten zu wecken. Dies kann z.B. durch Kontexte geschehen, die den Lernenden aufgrund eines außerschulischen Freizeitbezugs Freude bereiten (z.B. Spiele) oder die ihnen einen persönlichen Nutzen versprechen. Von besonderer Bedeutung sind Kontexte zum Erlernen neuer Inhalte. In Verbindung mit den Aufgabenklassen sollte ein solcher Beispielkontext Spannung aufbauen und aufrechterhalten. Der Reiz des Neuen soll dazu genutzt werden, die Beschäftigung mit einer Aufgabenstellung zu motivieren. In der Praxis zeigt sich allerdings oft, dass das, was der Lehrende für die Lernenden als motivierend vermutete, diese nicht in der beabsichtigten Weise ansprach. Ferner ist das Abklingen positiver Neuigkeitseffekte über die Zeit empirisch belegt (z.B. Clark 1992). Die Motivierung der Lernenden innerhalb von Lehr-Lern-Prozessen stellt oft ein großes Problem dar.

Ein gegebener Fachkern lässt sich in verschiedenen Kontexten entwickeln. Ebenso eignet sich ein Kontext in der Regel dazu, verschiedene Fachkerne darin einzubetten.

Kontextklassen

Bei der Abstraktion der ausgewählten Übungsaufgaben (vgl. 4.3.2) wurden diese von ihren jeweiligen Kontexten getrennt. Die Kontexte wurden gesammelt und zu so genannten *Kontextklassen* abstrahiert. Tabelle 11 zeigt die identifizierten Kontextklassen[64], jeweils mit Beispielkontexten aus den ausgewählten Aufgaben (vgl. 4.3.1) und einer Bewertung hinsichtlich der Auswahlkriterien für Kontexte (s.o.). Diese Bewertung ist erforderlich, um zu beurteilen, inwieweit sich diese Kontextklassen auch für die Gestaltung von Übungsaufgaben für den Informatikunterricht eignen. Dunklere Einfärbung bzw. mehr Punkte in einer Tabellenzelle repräsentieren dabei einen stärkeren Erfüllungsgrad des jeweiligen Auswahlkriteriums im Hinblick auf die Kontextklasse. Der Tabellenzelleneintrag *ka.* steht für *kontextabhängig*. Je

[64] Damit soll nicht ausgeschlossen werden, dass weitere Kontextklassen auf der Basis zusätzlicher Quellen zu bilden sind.

nach gewähltem Kontext kann der Erfüllungsgrad hier variieren. Die angegebenen Beispielkontexte stammen aus den Literaturquellen. Eine Aussage über die Unterrichtseignung der konkreten Beispielkontexte (nicht der Kontextklassen) wird hier nicht getroffen.

Kontextklasse	Beispielkontexte aus Rumbaugh et al. (1993) und Balzert (1999)	Eignung für OOM	Leichte Änderbarkeit und Erweiterbarkeit	Lebensweltbezug	Motivation
Lebensweltbezogene Kontexte	Schule, Auto, Haushalt	●●	ka.	●●	ka.
Geräte und Steuerungen	Spielzeugeisenbahn, Bankautomat, Videorekorder	●●	●●	ka.	ka.
Spiele	Kartenspiel, Türme von Hanoi, Schach	●●	●●	ka.	ka.
Verwaltungssysteme	Bibliotheks-, Literaturstellen-, Sportartikelverwaltung	●●	●●	ka.	ka.
Metamodellierung	Metamodell objektorientierter Basiskonzepte / einer formalen Sprache	●●	●●	–	●
Mathematik	symbolische Manipulation von Polynomen, Berechnung von Körpern	●●	ka.	ka.	ka.
Simulationen	Flugsimulator, 4-Takt-Motor	●●	●●	ka.	ka.
Entwurfswerkzeuge	Zeitungs-Layoutsystem, CAD-System	●●	●●	ka.	ka.

Tabelle 11: Bewertung von Kontextklassen

Lebensweltbezogene Kontexte eignen sich besonders für einführende Beispiele. Geräte und Steuerungen bilden einen geeigneten Kontext für einführende Beispiele zur dynamischen Modellierung. In diesem Zusammenhang bilden insbesondere Automaten in vielfältigen Erscheinungsformen (Fahrscheinautomat, Getränkeautomat, Geldwechselautomat etc.) eine im schulischen Rahmen bereits etablierte Kontextklasse (vgl. Spolwig 1995, 2000; Hubwieser 2000a). Spiele lassen aufgrund des außerschulischen Freizeitbezugs eine hohe Motivation erwarten (abhängig vom konkreten Spiel). Bei schülergerechten Verwaltungssystemen (z.B. Abiturfeierorganisation) kann die Motivation aufgrund des erwarteten Nutzens erzielt werden. Mittels der Metamodellierung können die Lernenden unter Verwendung der erlernten Modellierungstechniken ihr diesbezügliches Wissen explizieren (vgl. auch Kapitel 6). Wenn sie dies als Strukturhilfe und Wissensnetz verstehen, ist eine solide Grundmotivation zu erwarten. Ein direkter Lebensweltbezug besteht hierbei aber nicht.
Bei den mathematischen Kontexten stellt insbesondere die Motivation der Lernenden eine potentielle Problemgröße dar. Simulationen und Entwurfswerkzeuge haben bei den genannten Beispielen wenig Lebensweltbezug für die Lernenden. Ferner werden dabei auch die zu gestaltenden Zusammenhänge sehr vielschichtig und komplex.
Von den identifizierten Kontextklassen können daher insbesondere die fünf zuerst genannten auch für den Informatikunterricht empfohlen werden. Eine fundierte Empfehlung ist allerdings erst auf der Basis eines konkreten Kontextes und einer konkreten Lerngruppe möglich.

4.4.4 Fachdidaktische Diskussion

Nachfolgend werden ausgewählte Aufgabenklassen[65] exemplarisch im Hinblick auf ihre unterrichtliche Einbettung in Form von Übungsaufgaben diskutiert. Für diese Aufgabenklassen befinden sich korrespondierende Übungsaufgaben im Anhang (vgl. D.5).

Aufgabenklasse 1 (Beispielaufgabe: D.5, Aufgabe 1)

!	Objektdiagramm
?	Klassendiagramm
?	Objektdiagramm beschreiben

Hiermit kann der Zusammenhang zwischen Objekten und Klassen geübt und gefestigt werden. Ein Objektdiagramm stellt eine Momentaufnahme eines Objektgeflechts dar und ist daher vom Abstraktionsniveau her sehr nah am modellierten Realitätsausschnitt, näher als ein Klassendiagramm, und daher leichter zu verstehen. Im Objektdiagramm können bereits Relationsarten verwendet werden, die auch im Klassendiagramm von Bedeutung sind (z.B. Assoziation, Aggregation). Damit wird die Konstruktion eines Klassendiagramms vorbereitet. Lernende abstrahieren dazu die Objekte zu Klassen. Relationen zwischen den Objekten im Objektdiagramm bestehen analog zwischen den zugehörigen Klassen im Klassendiagramm. Relationen zwischen einem Objekt und verschiedenen typgleichen anderen Objekten sind im Klassendiagramm durch Multiplizitäten zu spezifizieren, welche bei der Modellierung von Klassenstrukturen eine große Rolle spielen. Oft werden durch bestimmte Beziehungen innerhalb des modellierten Realitätsausschnittes implizit Multiplizitäten vorgegeben. Für die Gestaltung von Modellen sind daher häufig Kenntnisse über die Domäne erforderlich, die über die Problembeschreibung hinausgehen. Das erfordert eine intensive Beschäftigung mit der Problemstellung und den Zusammenhängen innerhalb der Domäne. Abhängig von der jeweils individuellen Sicht auf den Realitätsausschnitt sowie der Verwendung von Strukturierungstechniken, wie der Vererbungsbeziehung, sind unterschiedliche, aber ähnliche Klassendiagramme als Lösung zu erwarten.
Das Beschreiben eines Diagramms trägt neben der Reorganisation des Stoffs bei Lernenden auch mit dazu bei, insbesondere bei umfangreichen, selbst gestalteten Diagrammen die Kontrolle durch andere Lernende und den Lehrenden zu erleichtern.

Aufgabenklasse 2 (Beispielaufgabe: D.5, Aufgabe 2)

!	unvollständiges Klassendiagramm (Klassen ohne Attribute und Methoden verbunden durch nicht spezifizierte Assoziationen)
?	Assoziationen benennen
?	Assoziationen mit Multiplizitäten versehen
!	Liste mit Attributen
?	Attribute den Klassen zuordnen (Mehrfachverwendung möglich)
?	Klassendiagramm so modifizieren, dass es bestimmte Eigenschaft erhält

Ein unvollständiges Modell als Angabe gibt Lernenden für die Bearbeitung einer Aufgabe einen Orientierungsrahmen, vergleichbar zu einem Lückentext oder einem Quelltextrahmen für ein Programm, in dem fehlende Teile ergänzt werden müssen. Die kognitive Anforderung, ein vollständiges Modell selbstständig gestalten zu müssen, kann so vermieden werden, wenn die jeweilige Lerngruppe noch nicht sicher genug in der Modellierung ist. Dies gilt sowohl für Teilmodelle des statischen als auch des dynamischen Modells. Durch das Vergeben von As-

[65] zur Aufgabenklassen-Notation vgl. S. 69ff

soziationsnamen und Multiplizitäten wird die Analyse und Spezifikation von Klassenbeziehungen gefördert. Die zusätzliche Vergabe von Rollennamen (mögliche Ergänzung) kann zur genaueren Bestimmung der Beziehung zwischen zwei Klassen verwendet werden. Das trägt zusätzlich zur Verständlichkeit eines Modells bei. Durch die Zuordnung von vorgegebenen Attributen (und später Operationen) zu Klassen kann deren Mehrfachverwendung gegebenenfalls deutlicher erfasst werden, als wenn Attribute oder Operationen für jede Klasse neu identifiziert werden müssten (Problem der Synonyme). Auf diese Weise lassen sich Gemeinsamkeiten von Klassen leichter auffinden und damit Vererbungsbeziehungen motivieren. Zielgerichtete Modifikationen des Modells setzen ein solides Verständnis sowohl des Modells als auch der erforderlichen Modellierungstechniken voraus, um durch das selbstständige Verknüpfen die gewünschte Eigenschaft zu erzielen. Je nach Lerngruppe lassen sich hier einfache oder auch sehr komplexe Modifikationen vornehmen, und damit der Grad der freien Modellierung variieren.

Aufgabenklasse 3 (Beispielaufgabe: D.5, Aufgabe 3)

!	Liste von Klassen
?	Klassendiagramm mit vorgegebener Mindestanzahl von Relationen zwischen den Klassen erstellen, Hinzufügen weiterer Klassen möglich

Im Vergleich zu den beiden zuvor diskutierten Aufgabenklassen ist diese weniger geführt und eröffnet Lernenden damit deutlich mehr Gestaltungsfreiheit. Das erfordert ein intensiveres Auseinandersetzen mit der gewählten Problemdomäne. Relationen zwischen Klassen sind von den Lernenden selbstständig zu identifizieren und zu spezifizieren. Durch die selbstständige Identifikation von Relationen und weiteren Klassen ist hierbei eine Vielzahl von unterschiedlichen korrekten Lösungen zu erwarten. Die Entscheidung zwischen vielfältigen Modellierungsalternativen erfordert ein hohes Maß an Bewertungskompetenz der Lernenden, um zu beurteilen, ob eine potentielle Lösung auch sinnvoll ist.

Aufgabenklasse 4 (Beispielaufgabe: D.5, Aufgabe 4)

!	unvollständiges Klassendiagramm (Klassen mit einzelnen Attributen, ohne Methoden, verbunden durch nicht spezifizierte Assoziationen, einzelne Assoziationen benannt), textuelle Beschreibung einer Momentaufnahme eines Objektgeflechts
?	Assoziationen mit Multiplizitäten versehen
?	diskutieren, inwieweit Multiplizitätsentscheidungen von eigener Sicht auf den modellierten Realitätsausschnitt abhängen
?	Objektdiagramm

Hierbei sollen die Lernenden nach der Spezifikation der Relationen über ihr jeweiliges Modell reflektieren und feststellen, dass bestimmte Modellierungsentscheidungen mit ihrer individuellen Sicht auf die Welt zusammenhängen. Daraus können Anknüpfungspunkte für den Unterricht insofern entwickelt werden, als dass die zentrale Bedeutung von eindeutigen Modellen in einem gemeinschaftlichen Gestaltungs- und Kommunikationsprozess herausgestellt werden kann. Die Konstruktion eines Objektdiagramms aus der textuellen Beschreibung einer Momentaufnahme eines Objektgeflechts erfordert Kenntnisse über die Zusammenhänge zwischen Klassen- und Objektdiagrammen, die hier selbstständig angewendet werden müssen. So müssen textuelle Angaben konkreten Objekten zugeordnet werden. Diese sind allerdings unter Umständen unvollständig in Bezug auf das gegebene Klassendiagramm, so dass bestimmte Angaben selbstständig ergänzt werden müssen. Damit wird die Transformation von unpräzisen Formulierungen in ein formales Modell geübt.

4.4.5 Analogie zu anderen Unterrichtsfächern

Abschließend sei noch auf eine Analogie von Modellierungsaufgaben zu Aufgaben in anderen Unterrichtsfächern hingewiesen.

Sprachbezogene / geisteswissenschaftliche Fächer

Eine offene Modellierungsaufgabe in der Informatik ist bzgl. der Vielfalt korrekter Lösungen mit einer Textinterpretation in einem sprachbezogenen oder geisteswissenschaftlichen Unterrichtsfach zu vergleichen. Bei einer Textinterpretation werden in der Regel vielfältige Ergebnisse akzeptiert, sofern eine schlüssige Begründung der jeweiligen Sichtweise vorliegt. In deren Vorhandensein liegt ein wesentlicher Unterschied zum Ergebnis einer offenen Modellierungsaufgabe. Während bei einer Textinterpretation die verschiedenen Begründungschritte nachzulesen sind, ist bei der Modellierung das Modell selbst das Ergebnis und die Begründung steckt implizit im Aufbau. Gerade bei der Konstruktion von komplexeren Modellen ist es daher unbedingt sinnvoll, dass die Lernenden ihre Modelle begründen, damit deren logische Korrektheit von anderen Lernenden oder auch dem Lehrenden besser beurteilt werden kann. Modellierungswerkzeuge können lediglich bestimmte Fehlertypen identifizieren.

Mathematisch-naturwissenschaftliche Unterrichtsfächer

Die Vielfältigkeit korrekter Lösungen ändert sich mit dem Übergang zur Implementierung, sofern man sich auf ein einheitliches, zugrunde liegendes Modell einigt und präzise Anforderungen an die Lösung formuliert. Ein fertig implementierter Lösungsbaustein mit definiertem Verhalten hat somit eine starke Analogie zu Lösungen in den naturwissenschaftlichen Fächern. Allerdings ist in der Informatik das Vorbereiten einer Musterlösung oder die Bereitstellung von Teillösungen, sofern nicht schon vorhanden, sehr viel zeitintensiver als in anderen Fächern.

4.5 Empirische Fallstudien im Informatikunterricht der Jgst. 11 und 12

4.5.1 Konzeption der Fallstudien

Die hier beschriebenen empirischen Fallstudien entstanden im Rahmen einer Ersten Staatsarbeit für das Lehramt für die Sekundarstufe II, die vom Autor der vorliegenden Arbeit betreut wurde (vgl. Brinda / Ortmann 2002, Ortmann 2002). Ziel dieser empirischen Studien war es, die unterrichtliche Einbettung des didaktischen Systems mit besonderem Schwerpunkt auf der Komponente „Aufgabenklassen“ exemplarisch zu untersuchen und die Akzeptanz bei Lehrenden und Lernenden, die Motivation der Lernenden sowie deren typische Bearbeitungsstrategien und Fehler zu erkunden, um daraus erste Schlussfolgerungen für erforderliche Weiterentwicklungen dieser Komponente des didaktischen Systems im Sinne einer internen Qualitätssicherung ziehen zu können. Die Fallstudien erheben nicht den Anspruch, repräsentativ zu sein, da prinzipiell umstritten ist, inwieweit verallgemeinerbare Aussagen zur Wirksamkeit mit empirischen Studien in Lerngruppen gewonnen werden können. Die dazu erforderlichen Laborbedingungen sind in realem Unterricht faktisch kaum herzustellen.
Die Fallstudien wurden im ersten und zweiten Quartal 2002 in drei Informatikgrundkursen an zwei Gymnasien (hier mit A und B bezeichnet) im Raum Dortmund durchgeführt. Es wurden zwei Schulen ausgewählt, um eine Abhängigkeit von einer Lehrperson und deren Unterrichtsplanung zu verringern. Tabelle 12 zeigt die Geschlechterverteilung in den drei an der Studie beteiligten Lerngruppen.

Schule	Jgst.	#Jungen	#Mädchen	#gesamt
A	11	12	3	15
B	11	15	0	15
B	12	18	1	19

Tabelle 12: Geschlechterverteilung in den drei an der Studie beteiligten Lerngruppen

Zu den Beobachtungszeitpunkten (vgl. Tabelle 17 auf S. 91) waren nicht immer alle Lernenden anwesend. Alle drei Kurse wurden von (unterschiedlichen) männlichen Lehrpersonen unterrichtet.

Leistungsbild der Lerngruppen vor der Durchführung der Fallstudien

Zur Vorbereitung wurden die Kurse zweigestuft hospitiert: zunächst einige Wochen vor der Durchführung, um ein generelles Bild des Leistungsstandes zu gewinnen, und dann noch einmal unmittelbar vor der Durchführung der jeweils zwei Beobachtungsstunden, um mit der Einbettung an vorangegangenen Unterricht anzuknüpfen. Tabelle 13 zeigt das Leistungsbild der Lerngruppen vor der Durchführung der Fallstudien.

Schule	Jgst.	Beschreibung des Leistungsbildes
A	11	Die Lernenden im Kurs A-11 hatten geringe OOM-Vorkenntnisse (Klasse, Attribut, Methode; ist-, hat-, kennt-Relation) aus vorangegangenem Unterricht nach dem Konzept „Von Stiften und Mäusen (SUM)“ (LSW 1999). Klassendiagramme wurden bis zum Beobachtungszeitpunkt noch nicht erstellt. Klassenbeziehungen wurden durch entsprechende Benennung der daran beteiligten Attribute (z.B. kenntBildschirm(Assoziation)) ausgedrückt. Vorangegangene Unterrichtsbeispiele nach dem SUM-Konzept waren eine „Windmühle“ und ein „Ball“, der sich zwischen den Bildschirmrändern hin- und herbewegt. Diese Beispiele wurden jeweils im Klassenverband analysiert, erarbeitet und anschließend am Rechner implementiert. Eine grafische Darstellung (z.B. mittels UML) erfolgte nicht. Kurz vor den Beobachtungsstunden lernte der Kurs die Vererbung anhand von Strukturen geometrischer Objekte kennen. Während Assoziation und Komposition bereits an einigen Beispielen geübt wurden, galt dies noch nicht für die Vererbung.
B	11	Der Kurs B-11 hatte über die Vorkenntnisse des Kurses A-11 hinaus bereits selbstständig erste Klassendiagramme aus Aufgabenstellungen oder Quelltexten erstellt, war darin aber noch nicht sicher. Vererbungsstrukturen waren bereits bekannt. Auch hier folgte der Unterricht dem SUM-Konzept (LSW 1999). Zuletzt wurde eine Ereignisanwendung modelliert, in der ein durch einfache geometrische Figuren auf dem Bildschirm repräsentiertes „Auto“ per Maus gesteuert werden konnte. Diagramme der dynamischen Modellierung wurden noch nicht behandelt.
B	12	Die Lernenden des Kurses B-12 stammten von zwei verschiedenen Schulen, die ihre Informatikkurse zur Sicherung des Angebots nach der Jgst. 11 zusammenlegten. Bedingt durch diesen Zusammenschluss wurde die objektorientierte Modellierung mittels UML wiederholt bzw. neu eingeführt. Die Lernenden waren in der Lage, selbstständig Klassendiagramme nach vorangegangener Problemanalyse zu erstellen und diese anschließend, nach erfolgter Präsentation beim Lehrer, in Programme umzusetzen. Diese Lerngruppe hatte bereits Kollaborationsdiagramme zur Darstellung der Modelldynamik kennen gelernt und sich kurz vor dem Beobachtungszeitpunkt mit Client-Server-Strukturen befasst. Dabei wurden zuletzt Echoserver, Chatserver und -client modelliert und implementiert.

Tabelle 13: Leistungsbild der Lerngruppen vor der Durchführung der Fallstudien

Entwicklung von Arbeitsblättern mit Übungsaufgaben zum OOM

Anhand des Leistungsstandes der jeweiligen Lerngruppe wurden unter Verwendung der Aufgabenklassen (Basis für die empirischen Studien war die erste Fassung der Aufgabenklassen zum statischen Modell in Anhang C.1 und Brinda 2000b) und in Absprache mit dem jeweiligen Fachlehrer Unterrichtsentwürfe und Übungsaufgaben zum OOM entwickelt. Ferner wurden die beiden Quellen (Rumbaugh et al. 1993) und (Balzert 1999) für Beispielaufgaben herangezogen. Zur Anpassung an die jeweilige Lerngruppe wurden einige Aufgabenklassen angepasst, aufgeteilt und der Lösungsprozess somit stärker vorstrukturiert. Zuerst erfolgte die

Festlegung des Kontextes. Als Basis für die Gestaltung der Übungsaufgaben wurden innere Knoten des Baumes der Aufgabenklassen ausgewählt (vgl. Abbildung 8 auf S. 78). Konventionen des jeweiligen Kurses bezüglich Namensgebung von Klassen, Methoden etc. wurden berücksichtigt.

Zwischenresümee 4.1

Folgende Schlussfolgerungen für die Weiterentwicklung der Aufgabenklassen wurden aus dieser Vorgehensweise gezogen:

1. Um die Gestaltung von Aufgaben mittels Aufgabenklassen weiter zu vereinfachen, werden diese mit Beispielaufgaben verknüpft. Zunächst stammen diese Beispielaufgaben aus den analysierten Lehrbüchern. Wünschenswertes Ziel ist jedoch die Verknüpfung mit Beispielaufgaben für den Informatikunterricht. Dies ließe sich möglicherweise darüber realisieren, dass eine offene, webbasierte Aufgabenklassensammlung mit Diskussionsforum erstellt und in Fachpublikationen und Postverteilern für Informatiklehrende entsprechend beworben wird. Für die kostenfreie Nutzung könnten Lehrende dazu aufgefordert werden, mindestens eine schulerprobte Beispielaufgabe oder eine neue Aufgabenklasse (anonym oder mit Name des Autors) beizutragen.
2. Die Aufgabenklassen werden verfeinert und mit mehr Varianten verknüpft, um eine Anpassung an die jeweilige Lerngruppe z.B. durch stärkere Vorstrukturierung des Lernprozesses besser zu unterstützen. Ferner wurden in Folge dieser Erkenntnis die ausgewählten Aufgaben (vgl. 4.3.1) erneut analysiert und Merkmale für die Variation der Komplexität von Aufgaben daraus abgeleitet (vgl. 4.4.2).
3. Den Ausgangspunkt der Auswahl von Aufgabenklassen bildeten die inneren Knoten der Baumstruktur (vgl. Abbildung 8 auf S. 78). Dies unterstützt die Hypothese der Reihenfolge der Auswahl von Aufgabenklassen anhand von Eigenschaftsdimensionen, wie Fachkern, Gegenstand und Aufgabentyp (vgl. 4.3.3). Der Fachkern ist hier vorgegeben, so dass zunächst festzulegen ist, auf welchen Gegenstand sich die Aufgabe beziehen soll. Als Schlussfolgerung hieraus wurden die Aufgabenklassen nach der Ausweitung der zu analysierenden Aufgabenmenge verfeinert und auf weitere Gegenstände und Aufgabentypen hin untersucht (vgl. 4.3.3).

In den Kursen der Jgst. 11 wurden die Kontexte durch die Lehrpersonen vorgegeben. In den Kursen A-11 und B-11 wurden die Aufgaben als Arbeitsblätter verteilt (diese sind angefügt in den Anhängen D.1 und D.2). Im Kurs B-12 erfolgte die Aufgabenstellung mündlich durch den Lehrenden. Nachfolgend werden die jeweiligen Aufgabensammlungen kurz charakterisiert und mit den Aufgabenklassen aus Anhang C.1 verknüpft.

Kurs A-11 (Schule A, Jgst. 11)

Im Kurs A-11 ging es darum, einen Traktor mit Anhänger anhand bereits bekannter Klassen für geometrische Figuren zu modellieren und auf dem Bildschirm darzustellen. Zunächst sollten die Lernenden die Begriffe „Objekt“ und „Klasse“ voneinander abgrenzen sowie „Attribut“ und „Methode“ definieren. In einer Liste von Bezeichnern zum Traktorbeispiel (z.B. „XPosition“, „fahreVorwaerts“) waren dann Klassen-, Methoden- und Attributbezeichner zu identifizieren. Die gefundenen Methoden und Attribute sollten den gefundenen Klassen zugeordnet werden, wobei die Zuordnung zu mehreren Klassen möglich war. Anschließend sollte die Funktion der Methoden und Attribute innerhalb der Klasse kurz beschrieben werden. Weiterhin waren die Relationen zwischen Klassen („ist“, „hat“, „kennt“) zu identifizieren, zu benennen und schließlich ein Klassendiagramm zu erstellen. Hierzu wurde die benötigte Klassendiagrammnotation angegeben. Tabelle 14 zeigt die Verknüpfung von Aufgabentexten und -klassen für den Kurs A-11.

Nr.	Aufgabentext	Aufgabenklasse
1	Aus den letzten Stunden im Unterricht kennst Du die Begriffe Klasse und Objekt. Erkläre den Unterschied der beiden Begriffe mit eigenen Worten. Neben dem Klassenbegriff wird auch noch von Methoden und Attributen gesprochen. Erkläre den Begriff Attribut mit eigenen Worten. Erkläre den Begriff Methode mit eigenen Worten!	*nicht vorgesehen in Sammlung C.1*
2	Die folgenden Wörter bezeichnen Klassen-, Attribut- oder Methodennamen. Ordne die Wörter der richtigen Kategorie zu. <Liste>	*Sammlung 1: Objekte, Klassen und Relationen identifizieren* ! Liste von Klassen, Attributen und Methoden ? Begriffe den drei Kategorien zuordnen
3	Ordne den Klassen aus Aufgabe 2 (außer Bildschirm und BuntStift) die Methoden und Attribute zu, die sie benötigen. Beschreibe mit einem kurzen Satz deren Funktion innerhalb der Klasse. Hinweis: Einige Methoden und Attribute können in mehreren Klassen vorkommen.	*Sammlung 2a: Merkmal zuordnen* ! Liste von Klassen, Attributen und Methoden ? Attribute und Methoden den Klassen zuordnen
4	Bestimme und benenne die Beziehungen (Ist, Kennt, Hat) zwischen den einzelnen Klassen.	*Sammlung 2c: Merkmal analysieren* *Sammlung 2d: Merkmal spezifizieren* ! Liste von Klassen ? Relationen identifizieren ? Relationen spezifizieren
5	Erstelle ein Klassendiagramm.	*Sammlung 3d: Modell konstruieren* ! Liste von Klasse mit zugeordneten Attributen, Methoden und Relationen ? Klassendiagramm erstellen

Tabelle 14: Verknüpfung von Aufgabentexten und -klassen für den Kurs A-11

Kurs B-11 (Schule B, Jgst. 11)

Im Kurs B-11 sollte ein „Zug“, bestehend aus Lok und Waggons, anknüpfend an das zuvor behandelte Projekt „Auto“ modelliert und implementiert werden (vgl. Czischke 1998a). Ein aus einfachen geometrischen Objekten erstellter „Zug“ wurde auf einer Folie präsentiert und von den Lernenden beschrieben. In einem unvollständigen Klassendiagramm zum „Zug“ waren fehlende Klassen und Relationen zu ergänzen, die Relationen zu spezifizieren („ist“, „hat“, „kennt“) und anschließend in einer Liste gegebene Methodenbezeichner den Klassen zuzuordnen, wobei auch hier die Zuordnung zu mehreren Klassen möglich war. Tabelle 15 zeigt die Verknüpfung von Aufgabentexten und -klassen für den Kurs B-11.

Nr.	Aufgabentext	Aufgabenklasse
1	Zeichne in das folgende Klassendiagramm die Beziehungsarten (Ist, Hat, Kennt) ein. Welche Klassen oder Beziehungen fehlen?	*Sammlung 2d: Merkmal spezifizieren* ! Klassendiagramm ? Relationen spezifizieren *Sammlung 1: Objekte, Klassen und Relationen identifizieren* ! Klassendiagramm ? fehlende Klassen und Relationen identifizieren
2	Ordne den Klassen aus Aufgabe 1 die unten stehenden Methoden zu. <Liste> Beschreibe mit einem kurzen Satz deren Funktion innerhalb der Klasse. Hinweis: Einige Methoden können in mehreren Klassen vorkommen.	*Sammlung 2a: Merkmal zuordnen* ! Liste von Klassen ? den Klassen die Methoden zuordnen *Sammlung 2b: Merkmal beschreiben* ! Liste von Klassen mit zugeordneten Methoden ? Funktion der Methode in der Klasse beschreiben

Tabelle 15: Verknüpfung von Aufgabentexten und -klassen für den Kurs B-11

Abschließend wurde ein Schülerrollenspiel zum Nachrichtenaustausch zwischen den Objekten durchgeführt, um den Lernenden Erkenntnisse über die Modelldynamik zu ermöglichen (Ortmann 2002, 44ff).

Kurs B-12 (Schule B, Jgst. 12)

Das Spiel „Schnick-Schnack-Schnuck“ als Thema für den Kurs B-12 knüpfte an dessen Vorkenntnisse zu Client-Server-Strukturen an, da beide Spieler ihren Zug gleichzeitig ausführen. Eine Spielanleitung wurde präsentiert und im Schüler-Lehrer-Gespräch erarbeitet, wie das Spiel zwischen Spielern (Clients) und Spielleiter (Server) zu organisieren ist. In Partnerarbeit war dann schrittweise ein statisches Modell zu konstruieren: zunächst sollten die Klassen mit den zugehörigen Attributen und Methoden bestimmt und beschrieben werden. Anschließend waren die Relationen zwischen den Klassen zu identifizieren und schließlich ein Klassendiagramm zu zeichnen. Den Schluss bildete die Entwicklung eines Sequenzdiagramms für einen gegebenen Anwendungsfall. Tabelle 16 zeigt die Verknüpfung von Aufgabentexten und -klassen für den Kurs B-12.

Nr.	Aufgabentext	Aufgabenklasse
1	Bestimme Klassen mit zugehörigen Methoden und Attributen anhand der Spielbeschreibung und beschreibe diese.	*Sammlung 4: statisches Systemmodell konstruieren* *Sammlung 1: Objekte, Klassen und Relationen identifizieren* *Sammlung 2c: Merkmal analysieren* *Sammlung 2b: Merkmal beschreiben* ! textuelle Beschreibung einer Problemstellung ? Klassen, Methoden und Attribute identifizieren ? Klassen beschreiben
2	Finde die Beziehungen (Ist, Hat, Kennt) zwischen den Klassen.	*Sammlung 1: Objekte, Klassen und Relationen identifizieren* ! textuelle Beschreibung der Problemstellung, Liste von Klassen ? Relationen identifizieren
3	Zeichne ein Klassendiagramm für das Spiel Schnick-Schnack-Schnuck.	*Sammlung 3d: Modell konstruieren* ! textuelle Beschreibung der Problemstellung, Liste von Klassen mit Relationen zwischen diesen ? Klassendiagramm
4	Erstellt ein Sequenzdiagramm für die Aufnahme eines Spielers. Es soll der zweite Spieler sein, der aufgenommen wird. Beschreibt alle Nachrichten, die zwischen den beteiligten Objekten ausgetauscht werden.	*Sammlung 3d: Modell konstruieren* ! Klassendiagramm, Anwendungsfall ? Sequenzdiagramm

Tabelle 16: Verknüpfung von Aufgabentexten und -klassen für den Kurs B-12

Der Unterricht wurde vom Fachlehrer gehalten und vom Autor der Staatsarbeit gezielt beobachtet, um erste Erkenntnisse über typische Bearbeitungsstrategien und Fehler bezogen auf Aufgabenklassen zu gewinnen. Stundenschwerpunkt war jeweils die Bearbeitung der zuvor entwickelten Arbeitsblätter in Einzel- und Partnerarbeit. Diese wurden mit einem anonymisierten Identifikationsmerkmal versehen[66], eingesammelt und im Hinblick auf das oben genannte Ziel ausgewertet. Im Gegenzug wurde den Lernenden jeweils eine Musterlösung ausgehändigt.

Schriftliche Befragung der Lernenden

Im Anschluss an die Bearbeitung der Übungsaufgaben wurden die Lernenden jedes Kurses schriftlich befragt (Befragung mit Fragebogen, basierend auf einer Vorlage von Häußler et al.

[66] Die Lernenden sollten folgende Merkmale eintragen: Geburtsmonat, erster Buchstabe des Vornamens der Mutter, erster Buchstabe des Vornamens des Vaters.

(1998, 105). Die Fragebögen enthielten jeweils ca. 30 Aussagen aus den vier Bereichen

- *Lern- und Unterrichtsklima*,
- *Sozialformen des Unterrichts*,
- *Motivation in Bezug auf Aufgabenklassen* und
- *Einstellung zum Modellieren / Programmieren*,

zu denen die Lernenden durch Ankreuzen ihre Zustimmung oder Ablehnung (fünf vorgegebene Skalenwerte von „trifft völlig zu“ bis „trifft nicht zu“) äußern sollten. Wesentliches Ziel hierbei war generell die Erfassung des Unterrichtserfolges im nichtkognitiven Bereich und speziell die Erkundung der Motivation der Lernenden in Bezug auf Aufgabenklassen. In diesem Bereich war der Fragebogen an die Aufgaben der jeweiligen Lerngruppe angepasst, die übrigen Teile blieben bei den verschiedenen Gruppen unverändert. Gefragt wurde hier nach:

- *Bewertung des Schwierigkeitsgrades bezogen auf Aufgabenklassen (z.B. „Die Einordnung der Relationen fiel mir leicht.“),*
- *außerschulischer Beschäftigung mit dem Thema,*
- *themenspezifisches Interesse*,
- *Einschätzung des persönlichen Nutzens.*

Die übrigen Bereiche des Fragebogens dienten zur Erhebung ausgewählter Einflussgrößen bzw. Einstellungen der Lernenden, um deren Antworten leichter interpretieren zu können. Die ausgefüllten Fragebögen wurden von den Lernenden mittels des gleichen Verfahrens anonymisiert, wie die Arbeitsblätter, um eine Zuordnung zwischen Befragungsergebnissen und Schülerlösungen zu ermöglichen.

4.5.2 Auswertung der Beobachtungsstunden

Nachfolgend werden wesentliche Ergebnisse der Beobachtung sowie der Auswertung der Schülerlösungen insbesondere unter den Gesichtspunkten Bearbeitungsstrategien und typische Fehler in Bezug auf Aufgabenklassen präsentiert. Tabelle 17 gibt einen Überblick über die Beobachtungszeitpunkte in den verschiedenen Kursen.

Schule	Jgst.	Beobachtungsstunden
A	11	17.04.02, 08.05.02
B	11	24.04.02, 02.05.02
B	12	13.03.02, 19.03.02, 20.03.02

Tabelle 17: Überblick über die Beobachtungszeitpunkte

Zu beachten ist besonders der, schulorganisatorisch bedingte, große Zeitraum zwischen den Beobachtungsstunden im Kurs A-11. Für die Kurse A-11 und B-11 befinden sich Musterlösungen im Anhang in den Abschnitten D.1 und D.2, im Kurs B-12 wurde diese gemeinsam an der Tafel entwickelt.

Wissens- und Verständnisfragen

Bei den Wissensfragen (Kurs A-11) zu „Objekt“, „Klasse“, „Methode“ und „Attribut“ (vgl. Abschnitt D.1, Aufgabe 1) zeigte sich, dass die Lernenden mit den Begriffen „Attribut“ und „Methode“ kaum Schwierigkeiten hatten. 12 von 13 anwesenden Lernenden konnten „Methode“ definieren, der 13. Lernende listete lediglich einige Beispiele auf. Der überwiegende Anteil der Lernenden wusste, dass Attribute „Eigenschaften von Objekten“ sind und illustrierten dies mit Beispielen (Koordinaten, Länge, Größe). Bei der Abgrenzung von „Objekt“ und „Klasse“ waren sechs Lernende der Auffassung, dass eine Klasse ein Objekt sei (Klasse und

Objekt als Synonyme)[67].

> „Eine Klasse beinhaltet mehrere Objekte. Sie ist übergeordnet und kann nicht in einer anderen aufgerufen werden. Eine Klasse ist ein Objekt. Ein Objekt kann alles sein."
>
> „Eine Klasse ist ein Objekt, ein Objekt kann aber mehr sein, z.B. ein Stift."

Drei Lernende erklärten, dass eine Klasse ein „Oberbegriff für mehrere Objekte" sei.

> „Eine Klasse ist ein Oberbegriff für mehrere Objekte, das heißt die Vielecke sind eine Klasse und die Rechtecke gehören zur Klasse der Vielecke und sind Objekte."
>
> „Eine Klasse besteht aus mehreren Objekten. Ein Objekt ist zum Beispiel der Stift."

Lediglich zwei Lernende gaben die (aus dem Unterricht bekannten) Metaphern „Klasse ist Bauplan / Stempel für Objekte" an.

> „Eine Klasse ist eine Art Bauplan für Objekte. Eine Klasse ist immer ein Objekt."
>
> „Eine Klasse ist wie ein Stempel für Objekte"

Arbeitsblätter und Nachbesprechung der Aufgaben zeigten deutlich die Schwierigkeiten mit diesem Begriffspaar.

Identifikation von Klassen, Methoden und Attributen

Mit der Identifikation von Klassen, Methoden und Attributen hatten die Lernenden (A-11) keine Probleme (D.1, Aufgabe 2) und konnten die in einer Liste vorgegebenen Begriffe korrekt, und schneller als erwartet (10 statt 20 Minuten), den Kategorien zuordnen, obwohl die Begriffe von den im Unterricht bislang verwendeten Notationen (führendes „z" zur Kennzeichnung von Attributen, kleingeschriebene Methodennamen) abwichen. Bei der Bearbeitung dieser Aufgabe fielen bestimmte Vorgehensweisen besonders auf. Ein Lernender sah sich die Begriffsliste an, suchte zuerst die Klassennamen heraus und schrieb diese auf, im Anschluss daran die Methoden und zuletzt die Attribute (entsprechend der Reihenfolge in der auf den Arbeitsblättern vorbereiteten Tabelle). Nach erfolgter Zuordnung wurde der Begriff durchgestrichen. Ein anderer Lernender überlegte sich beim Durchlesen der Begriffsliste direkt die Zuordnung, kennzeichnete die Begriffe durch unterschiedliche Markierungen (unterstreichen, durchstreichen, Kreis) und übertrug sie anschließend in die Tabelle. Bei der Zuordnung wichen die meisten Lernenden nicht von der durch die Begriffsliste vorgegebenen Reihenfolge ab.

Zuordnung von Methoden und Attributen zu Klassen

Bei der Zuordnung von Methoden und Attributen zu Klassen (vgl. D.1, Aufgabe 3; D.2, Aufgabe 2) begannen die Lernenden im Kurs A-11 (Vererbung neu) mit den Zuordnungen, die ihnen aus vorherigen Unterrichtsstunden bekannt waren oder ähnlich erschienen. Es zeigten sich große Schwierigkeiten bei der Zuordnungsentscheidung von Methoden und Attributen zu Ober- oder Unterklasse. Vielfach wurden diese sowohl der Ober- als auch der Unterklasse zugeordnet (11 von 13 Lernenden ordneten „XPosition" und „YPosition" allen Klassen zu). Ferner hatten sie Probleme mit der Aufteilung von Funktionalität zwischen „Traktor" und „Anhänger", was sich z.B. darin zeigte, dass nur vier Lernende der Ansicht waren, dass der Anhänger die Methoden „fahreVorwaerts" und „fahreRueckwaerts" benötigt (11 Lernende bei „Traktor"). Nur drei Lernende begannen mit einer Beschreibung der Funktionalität der Methoden. Die Ursache hierfür könnte darin liegen, dass in der Tabelle kein eigenes Feld für die Beschriftung vorgesehen war, so dass viele Lernende dies einfach vergaßen.

[67] Im strengen objektorientierten Sinne ist die Aussage „eine Klasse ist ein Objekt" durchaus zutreffend. Dies ist hiermit aber nicht gemeint, da auf eine klare Unterscheidung der Begriffe geachtet wurde.

Im Kurs B-11 (Vererbung schon länger bekannt) konnten die Lernenden die Methoden sicherer zuordnen. Eine Ausnahme bildete die Methode „laenge“[68], bei der einige Lernende der Meinung waren, es handele sich um ein Attribut. Die Unterscheidung zwischen abstrakten und nicht abstrakten Methoden bereitete keine Schwierigkeiten, wohl aber die präzise Beschreibung der Funktionalität der Methoden. So schrieben einige Lernende bei der Methode „fahreVorwaerts“ der abstrakten Klasse „Waggon“: „Bewegt den Zug zum linken Bildschirmrand.“, obwohl diese Methode nur bewirkt, dass sich ein Wagen des Zuges bewegt und dem Nachfolger die Nachricht zur Bewegung weitergeleitet wird. Vielfach erfolgte die Beschreibung nur in Stichworten, nicht aber in zusammenhängenden Sätzen, wie im Klassenprotokoll gefordert.

(Identifikation und) Spezifikation der Relationen zwischen Klassen

Bei der Bestimmung und Benennung der Relationen zwischen den Klassen (Kurs A-11, vgl. D.1, Aufgabe 4) zeigten sich bei zwei Lernenden Schwierigkeiten mit dem Vererbungsprinzip darin, dass ein „BuntStift“ allen Unterklassen statt der Oberklasse zugeordnet wurde. Bei den Beziehungen der Klassen „Traktor“ und „Anhänger“ waren sechs Lernende der, zwar von der Musterlösung abweichenden, durchaus aber möglichen, Auffassung, dass dies „Figuren“ seien (Vererbung), zwölf Lernende sahen Kompositionsbeziehungen zwischen „Traktor“ und verschiedenen Figuren, acht Lernende stellten dies bei der Klasse „Anhänger“ fest. Bei den Beziehungen zwischen „Traktor“ und „Anhänger“ erkannten sechs von 15 anwesenden Lernenden keine Relation, die übrigen entschieden sich für eine der drei Varianten „Anhänger kennt Traktor“ (Assoziation), „Traktor kennt Anhänger“ (Assoziation) und „Traktor hat Anhänger“ (Komposition).
Bei der Spezifikation der Relationsarten (Kurs B-11, vgl. D.2, Aufgabe 1) im vorgegebenen, unvollständigen Klassendiagramm zum „Zug“ zeichneten zwei Schülerteams die fehlenden Symbole korrekt ein. Bei den anderen fünf Teams traten unterschiedliche Fehler auf: drei Teams hatten Probleme mit der Beziehung der Klassen „ZugAnwendung“ und „EreignisAnwendung“. Korrekt wäre eine „ist“-Beziehung (Vererbung), fehlerhafte Lösungen waren eine Komposition (zwei Teams) und eine falsch gerichtete Vererbung. Probleme mit der Semantik (Richtung) der Relationssymbole zeigten sich auch bei den anderen Teams. Bei der Beobachtung der Arbeitsweise der Lernenden fiel in den Kursen B-11 und A-11 (A-11 bei der Konstruktion des Klassendiagramms) auf, dass von ihnen zuerst die bekannten Beziehungen ausgewählt wurden, bevor sie neue Beziehungen einordneten. Probleme entstanden auch teilweise bei der Unterscheidung zwischen Assoziation und Komposition.

Identifikation fehlender Klassen und Relationen

Die Identifikation fehlender Klassen und Relationen im vorgegebenen, unvollständigen Klassendiagramm wurde nur von zwei Schülerteams (B-11) bearbeitet (vgl. D.2, Aufgabe 1). Das erste Team erkannte das Fehlen der Klasse „Stift“ oder „BuntStift“ zum Zeichnen der Wagen. Das zweite Team argumentierte, es fehle keine Klasse, da die „EreignisAnwendung“ einen „Stift“ besitze. Fehlende Relationen wurden von den Teams nicht spezifiziert.

Konstruktion eines Klassendiagramms

Die Konstruktion des Klassendiagramms war im Kurs A-11 durch die vorangegangenen Aufgaben gut vorbereitet und bereitete den Lernenden, obwohl der Umgang mit der UML-Notation noch nicht geübt wurde, keine Probleme (vgl. D.1, Aufgabe 5). Die Erkenntnisse der Nachbesprechung zur vorangegangenen Aufgabe wurden direkt in das Klassendiagramm übernommen. Da dort nicht auf die Beziehung der Klassen „Traktor“ und „Anhänger“ zu den Figurenklassen eingegangen wurde, zeichneten die Lernenden, die sich im Rahmen der voran-

[68] Diese Benennung wurde durch die Lehrperson vorgegeben. Attribute wurden in diesem Kurs durch ein vorangestelltes „z“ markiert, z.B. als „zLaenge“.

gegangenen Aufgabe für die Vererbungsbeziehung entschieden hatten, diese nun auch ein. Drei Lernende entschieden sich für eine ausschließliche Vererbungsbeziehung, drei weitere Lernende zeichneten neben der Vererbung auch die Komposition mit den einzelnen Figurklassen ein.

Konstruktion eines statischen Systemmodells

Der Kurs B-12 sollte selbstständig das Spiel „Schnick-Schnack-Schnuck" analysieren und ein statisches Systemmodell konstruieren. Einige Teilschritte waren vorgegeben, allerdings keine Teillösungen, wie bei den beiden anderen Kursen. Zur Analyse wurde das Spiel mit verteilten Rollen gespielt, um Klassen und deren Aufgaben zu identifizieren. Die meisten der sechs Schülerteams versuchten, jede identifizierte Klasse erst komplett zu beschreiben, bevor sie mit der nächsten Klasse fortsetzten. Nur wenige Teams bestimmten erst die benötigten Klassen, um diese im Anschluss genauer zu spezifizieren. Das Auffinden und Beschreiben der Klassen bereitete den Teams Probleme: zwei Teams fanden nur eine Klasse, beschrieben diese allerdings gut, ein Team ermittelte zwei Klassen und dokumentierte eine von beiden ausführlich, die anderen drei Teams beschrieben jeweils zwei von drei gefundenen Klassen ausführlicher. Deutliche Unterschiede gab es bei der Qualität der Dokumentation: einfache Klassen (z.B. Spielgegenstände „Stein", „Papier", „Schere") wurden ausführlich dokumentiert, komplexe Klassen (z.B. „Spielleiter") dagegen nur rudimentär. In einer Studie von Tholander (2001) trat dieses Phänomen auch auf. Bei der Beobachtung wurde deutlich, dass bei einigen Lernenden der Unterschied zwischen Analyse und Programmierung noch nicht ganz klar war. Mehrere Teams versuchten, eine pseudocode-ähnliche Beschreibung für die Auswertung einer Spielrunde zu finden, wo die Beschreibung gemäß Klassenprotokoll gefordert war. Da sie an dieser Stelle nicht umgehend eine befriedigende Lösung fanden, beschäftigten sie sich sehr lange mit dieser unnötigen Aufgabe und vergaßen dabei, die anderen Beschreibungen zu erstellen. Obwohl zunächst nur Klassen mit Methoden und Attributen bestimmt werden sollten, identifizierten die Lernenden auch Bezugsklassen. Allerdings hatte dies keine erkennbaren Auswirkungen hinsichtlich der Methodenzuordnung in Bezug auf einen späteren Nachrichtenaustausch. Nachdem die Klassen im Rahmen der Besprechung der Aufgabe festgelegt wurden, gelang es den Lernenden ohne große Schwierigkeiten, die Beziehungen zwischen den einzelnen Klassen zu bestimmen und das zugehörige Klassendiagramm zu erstellen (fünf von sieben Teams fehlerfrei). Die Identifikation und Spezifikation der Relationen lösten die Lernenden mit viel mehr Sicherheit als die Identifikation und Charakterisierung der Klassen. Dabei erkannten die Lernenden auch das Fehlen von Klassen in ihren individuellen Lösungen. Nachdem für alle Klassen die Bezugsklassen an der Tafel standen, gelang es fünf von sieben Teams, ein fehlerfreies Klassendiagramm zu erstellen. Probleme gab es mit der Angabe von Multiplizitäten, die entweder vergessen oder an der falschen Stelle notiert wurden. In einer Lösung wurde die Symbolik der Beziehungen Vererbung und Komposition verwechselt. Ansonsten war diese korrekt. Nach der gemeinsamen Erstellung eines Sequenzdiagramms (neu für die Gruppe) für die Aufnahme des ersten Spielers gelang es fast allen Teams, recht zügig und ohne Schwierigkeiten, die Aufnahme des zweiten Spielers entsprechend zu modellieren.

Zwischenresümee 4.2

Aus diesen Ergebnissen wurden folgende Schlussfolgerungen für eine Verfeinerung und Weiterentwicklung der Aufgabenklassen abgeleitet:

- *Begriffspaar Objekt / Klasse*
  Im Kurs A-11 zeigten sich Schwierigkeiten mit dem Begriffpaar Objekt-Klasse, die auch aus der Literatur bekannt sind (z.B. Holland et al. 1997, vgl. 2.3.2). Die Zuordnung gegebener Begriffe zu den Kategorien Klasse, Methode und Attribut bereitete keine Probleme, auch wenn im jeweiligen Kurs vereinbarte Notationskonventionen bzgl. Groß- und Kleinschreibung oder vorangestellten Buchstaben zur Auszeichnung (z.B.

vorangestelltes „z“ bei Zustandsvariablen / Attributen) bei den Begriffen nicht eingehalten wurden. Zuordnungsprobleme gab es mit einer Methode „laenge“ im Kurs B-11, deren Benennung auch eher auf eine Eigenschaft (Attribut) als auf eine Methode schließen lässt. Es sind Aufgabenklassen zu entwerfen, mit denen sich die Unterscheidung dieser Begriffe üben lässt, beispielsweise durch die Zuordnung gegebener Begriffe oder Aussagen in einem Text zu einer der beiden Kategorien Objekt bzw. Klasse. Ferner kann durch die Explikation von Objekt- und Klassenstrukturen sowie durch deren Verknüpfung die Abgrenzung der Begriffe besser gefestigt werden (vgl. 4.4.4, Aufgabenklassen 1 und 4). Als Konsequenz hieraus wurden bei der Überarbeitung der Aufgabenklassen eine Verfeinerung der inneren Struktur der Aufgabenklassen zum statischen Modell und eine explizite Zuordnung zu Objekten und Objektstrukturen, Klassen und Klassenstrukturen sowie zu deren Verknüpfung vorgenommen. Speziell die Verknüpfung eignet sich zum Üben der Unterschiede dieser beiden Kategorien.

- *Vererbung*
  Die partiellen Schwierigkeiten mit der korrekten Anwendung des Vererbungskonzeptes traten nur im Kurs A-11 auf, für den dieses Konzept relativ neu war. Einerseits hätte hier mit vorhandenen Aufgabenklassen (z.B. 4.4.4, Aufgabenklasse 2) die Anwendung des Vererbungskonzepts geübt werden können, andererseits besteht Entwicklungspotential für weitere Aufgabenklassen, beispielsweise Lösungsvarianten zu einem Modellierungsproblem diskutieren, bei denen ein Merkmal einmal der Oberklasse und einmal allen Unterklassen zugeordnet wurde.

- *Zuordnung von Methoden zu Klassen / Modelldynamik*
  Probleme zeigten sich bei allen Kursen bei der Zuordnung von Methoden zu Klassen im Hinblick auf die Modelldynamik und den späteren Nachrichtenaustausch zwischen Objekten. In besonderer Weise galt dies für die Verkettung von Klassen beim „Traktor“ und „Zug“. Auch Lernende mit wenig Erfahrung im Bereich der Modellierung von Objektkommunikation können eine solche Zuordnung zwar oft in großen Teilen korrekt vornehmen, Schwierigkeit bereiten aber Methoden, die ihrerseits Objektkommunikation auslösen. Für das Üben der Verantwortlichkeiten von Objekten wurden in der Literatur bereits „Objektspiele“ beschrieben (vgl. Bellin / Simone 1997, Andrianoff / Levine 2002; für den Informatikunterricht: Czischke 1998a, Schulte / Niere 2002). Aufgabenklassen zur dynamischen Modellierung (z.B. Szenarien aus textuell beschriebenen Ereignisketten aufstellen) werden benötigt und sollten an dieser Stelle stärker mit denen zur statischen Modellierung verknüpft werden, um für die Lernenden die nicht offensichtlich komplexe und nur scheinbar einfache Aufgabe der Methodenzuordnung in Teilaufgaben zu zerlegen. Als Konsequenz hieraus wurden auch Aufgabenklassen zum dynamischen Modell und zur Verknüpfung von statischem und dynamischem Modell entwickelt (vgl. Anhang C.2).

- *Identifikation fehlender Klassen / Relationen*
  Während das Identifizieren von Klassen aus einer Liste bzw. Relationen zwischen vorgegebenen Klassen mittels Tabelle als einfach empfunden wurde, so bereitete das selbstständige Identifizieren und Spezifizieren große Probleme, die darin bestanden, dass die Lernenden durch die Vielzahl der Lösungsmöglichkeiten „blockiert“ waren (B-12, Aufgabe 1). Im Zusammenhang mit diesen Aufgabenklassen sind systematische Vorgehensweisen zu erarbeiten, die durch geführte Aufgaben vorbereitet werden können.

- *Konstruktion eines statischen Systemmodells*
  Im Kurs B-12 zeigte sich eine teilweise Überforderung der Lernenden bei der selbstständigen Konstruktion eines statischen Systemmodells. Angebracht gewesen wäre hier

angesichts des Leistungsstands eine Aufteilung in kleinere, präzise formulierte und konkrete Teilaufgaben, da die Vielzahl der Lösungsmöglichkeiten die Lernenden „blockierte“. Die Aufgabenklassen sollten daher derart strukturiert werden, dass ersichtlich wird, welche Aufgabenklassen sich aus welchen Teilen (anderen Aufgabenklassen) zusammensetzen (hier bspw. „Identifikation von Klassen, Attributen, Methoden und Relationen“, deren Charakterisierung und Strukturierung als Teilaufgaben bei der Konstruktion eines statischen Systemmodells), damit die Lehrperson die Aufgabe im Bedarfsfall einfacher „zerkleinern“ kann. Solche Zerlegungen in Teilaufgaben sollten den Lernenden im Zusammenhang mit einer Aufgabenklasse bewusst gemacht werden, damit sie diese bei ähnlichen Problemstellungen selbstständig vornehmen können. Durch die Angabe von Strukturen von Varianten und Ergänzungen für Aufgabenklassen wurde dies bei der Überarbeitung und Verfeinerung der Aufgabenklassen berücksichtigt (vgl. 4.3.2).

Die Beschreibung von Klassen und Methoden bereitete Schwierigkeiten. Es zeigt sich darin der große Bedarf, dies zu üben und zu vertiefen. Probleme mit der Semantik der Relationssymbole (Richtung, Art) und anderer Darstellungsformen der UML sind in der Literatur bekannt (z.B. Raner 2000, vgl. 2.3.2). Vorgeschlagen wurden spezielle Ausbildungsdarstellungsformen („OVAL“, vgl. 2.3.2). Da diese allerdings das Problem der Darstellungssemantik nur für statische Modelle lösen, stellt sich die Frage, inwieweit ein Methodenmix (statisches Modell: OVAL, dynamisches Modell: UML) hier eine bessere Alternative darstellt. Ungewohnt ist für Lernende, dass das relationsrichtungsspezifizierende Darstellungselement (z.B. Raute bei Aggregation / Komposition) auf der Seite des Ausgangspunktes und nicht beim Ziel (wie z.B. bei einem Pfeil) zu finden ist. Auf diesen Unterschied sollte unbedingt im Unterricht hingewiesen werden.

4.5.3 Befragungsergebnisse

An der schriftlichen Befragung nahmen im Kurs A-11 alle 15 Lernenden teil, im Kurs B-11 12 von 15 und im Kurs B-12 16 von 19 Lernenden. Die vollständigen Befragungsergebnisse aufgeschlüsselt nach den drei Kursen befinden sich im Anhang im Abschnitt D.3. Zu beachten ist, dass alle Prozentangaben auf volle Prozente gerundet wurden. Es ist daher möglich, dass sich die Prozentangaben in einer Zeile nicht exakt auf 100% aufaddieren.

Ergebnisse zur Einschätzung des Schwierigkeitsgrades der Aufgabenklassen

Bei der Befragung der Lernenden in den Kursen A-11 und B-11 zu den Aufgaben stellte sich heraus, dass sie die Aufgaben vom Schwierigkeitsgrad her für angemessen hielten. Das Zuordnen von Attributen und Methoden bereitete im Kurs A-11 11 von 15 und im Kurs B-11 9 von 12 Lernenden keine Probleme. Das Auffinden und die Spezifikation der Relationsarten fiel im Kurs A-11 12 von 15 und im Kurs B-11 9 von 12 Lernenden leicht (vgl. Tabellen 18 und 19).

	trifft völlig zu		**trifft eher zu**		**trifft eher nicht zu**		**trifft nicht zu**		**kann ich nicht beurteilen**	
Das Zuordnen der Methoden und Attribute zu den einzelnen Klassen bereitete mir keine Probleme.	6	40%	5	33%	4	27%	0	0%	0	0%
Das Auffinden der Beziehungen fiel mir leicht.	3	20%	9	60%	2	13%	1	7%	0	0%
Die Erstellung des Klassendiagramms war einfach.	4	27%	9	60%	1	7%	1	7%	0	0%

Tabelle 18: Befragungsergebnisse zur Einschätzung des Schwierigkeitsgrades der Aufgaben im Kurs A-11

	trifft völlig zu		trifft eher zu		trifft eher nicht zu		trifft nicht zu		kann ich nicht beurteilen	
Die Einordnung der Beziehungen fiel mir leicht.	1	8%	8	67%	2	17%	1	8%	0	0%
Die Zuordnung der Methoden zu den einzelnen Klassen bereitete mir keine Probleme.	3	25%	6	50%	3	25%	0	0%	0	0%

Tabelle 19: Befragungsergebnisse zur Einschätzung des Schwierigkeitsgrades der Aufgaben im Kurs B-11

Die erstmalige Erstellung eines Klassendiagramms empfanden 13 von 15 Lernenden im Kurs A-11 als einfach. Als leicht empfundene Aufgaben (insb. Kurs A-11, Wissensfragen) führten allerdings nicht zwangsläufig zu korrekten Lösungen (vgl. 4.5.2).

Ergebnisse zu den Unterrichtsthemen

Befragt zu den Themen des Unterrichts waren alle drei Kurse der Auffassung, dass ihnen diese für ihr tägliches Leben eher nicht nützlich seien. In den Kursen A-11 und B-12 fanden mindestens die Hälfte der Lernenden die Themen dennoch eher interessant (vgl. 7. und 8. in Tabelle 44 auf S. 222).

> „Die Aussagen der Lernenden kommen meiner Meinung nach nicht unerwartet, da die Programme, die mit Hilfe der Klassenbibliothek des Konzepts ‚Von Stiften und Mäusen' [...] erstellt werden, sicher interessant und motivierend sind, aber das Bewegen von Zügen, Autos, Windmühlen und Bällen auf dem ‚Bildschirm' wenig persönlichen Nutzen bietet." (Ortmann 2002, 82)

Alle drei Kurse waren der Auffassung, dass sich der Unterricht nicht mit Aufgaben befasse, die ihnen im täglichen Leben begegnen (vgl. 2. in Tabelle 45 auf S. 223). Dennoch gaben zwischen 19% (B-12) und 53% (A-11) der Lernenden an, auch außerhalb des Unterrichts über die Aufgaben nachgedacht zu haben (Tabelle 45, 5.). Hier wird ein Zusammenhang mit dem themenspezifischen Interesse vermutet, da keine Hausaufgaben aufgegeben wurden, die dies hätten erklären können. Bei der Einschätzung des persönlichen Nutzens bestätigten die Kurse A-11 und B-12 mehrheitlich, etwas Neues dazu gelernt zu haben. Im Kurs B-11 war die einzige Neuerung das Rollenspiel zur Objektinteraktion (Tabelle 45, 3./7.). Deutlich weniger Lernende gaben an, dass sie das neue Wissen in Zukunft gebrauchen könnten (Tabelle 45, 8.). Dies entspricht den Antworten zur zukünftigen Verwendbarkeit des Unterrichtsstoffs aus dem Bereich der Modellierung (Tabelle 47, S. 224). Beim themenspezifischen Interesse fallen große Unterschiede auf (Tabelle 45, 4./9.). Im Kurs A-11 gaben ca. 70% der Lernenden an, dass sie die Themen interessierten und dass die Bearbeitung Spaß gemacht habe. Im Kurs B-12 äußerten sich noch 44% der Lernenden positiv, im Kurs B-11 hingegen noch 33%, obwohl 58% das Rollenspiel gut fanden. Da die Aufgaben nicht als zu schwierig empfunden wurden, könnte die Ursache darin bestehen, dass aus Sicht der Lernenden zu wenig Neues erlernt wurde.

Weitere Befragungsergebnisse

Es zeigte sich ein hohes Maß an Zufriedenheit mit dem Informatikunterricht (vgl. 1. und 2. in Tabelle 44 auf S. 222). Dass der Informatikunterricht ihnen Spaß mache, bestätigten zwischen 59% (B-11) und 86% (A-11) und zwischen 67% der Lernenden im Kurs B-11 und 76% der Lernenden im Kurs B-12 fühlten sich im Unterricht wohl. Die Selbsteinschätzung zum Verständnis des Unterrichtsstoffes war insbesondere in den Kursen A-11 und B-12 sehr hoch. Im Kurs A-11 gaben 93% der Lernenden an, den Stoff zu verstehen, im Kurs B-12 88%, im Kurs B-11 aber nur 42% (Tabelle 44, 3.). Die Ergebnisse zu den Wissensfragen im Kurs A-11 zeigten allerdings ein anderes Bild (vgl. 4.5.2). Alle drei Kurse waren der Meinung, dass es im Informatikunterricht eher um das Modellieren als das Programmieren gehe (Tabelle 44, 5./6.). Erstaunlich ist dies, weil die jeweiligen Lehrenden äußerten, „dass sie zu wenige Beispiele

hätten, um wirklich durchgängig Modellierungen durchzuführen“ (Ortmann 2002, 81). Vermutet wird ein Zusammenhang mit dem Befragungszeitpunkt, der nach einer mehrstündigen Modellierungsphase lag. Beim Vergleich des geschätzten individuellen Arbeitsanteils bei der Partnerarbeit beim Modellieren und Programmieren zeigten sich zwei besonders interessante Aspekte. So schätzten beim Programmieren maximal zwei Lernende ihren individuellen Arbeitsanteil in der geringsten Kategorie (0-25%) ein, beim Modellieren war dies maximal einer (vgl. Tabelle 46 auf S. 223). Ferner zeigte sich folgender Zusammenhang:

> „Bei der Betrachtung der Fragebögen zeigt sich, dass die Schüler, die ihren Arbeitsanteil im Rahmen der Modellierungsphasen als sehr gering einschätzen, ebenfalls bei der Programmierarbeit kaum mitarbeiten. Die Schüler, die nur bei der Programmierung diesen geringen Arbeitsteil angaben, schätzten ihren Arbeitsanteil an der Modellierung zwischen 25 und 50% ein.“ (Ortmann 2002, 84)

In allen Kursen äußerten sich die Lernenden positiver zum Programmieren als zum Modellieren (Tabelle 47 auf S. 224, 1./4). Während in den Kursen der 11 die Lernenden nur etwas lieber programmierten als zu modellieren, gab es im Kurs B-12 drei Programmierbegeisterte, die die Modellierung völlig ablehnten. Nur der Kurs A-11 gab mehrheitlich an, dass er eher gerne modelliert, trotz der geringsten Erfahrung. Der Kurs B-12 lehnte die Modellierung am deutlichsten ab. In allen Kursen halten die meisten Lernenden eine Problemanalyse für sinnvoll (Tabelle 47, 2.). Die Kurse A-11 und B-12 würden aber mehrheitlich bei einer neuen Aufgabe ohne Analyse direkt mit der Programmierung beginnen (6.). Die abweichenden Ergebnisse im Kurs B-11 legen die Vermutung nahe, dass hier die Vorteile der Analyse besser erkannt wurden. Ähnliche Ergebnisse wie bei der Problemanalyse zeigen sich bei der Einschätzung, ob die Fähigkeiten zu modellieren und zu programmieren einen individuellen Nutzen für die Zukunft hätten (3./5.):

> „Das Programmieren scheint eine zukunftsgewandtere Fähigkeit zu sein. Immerhin waren 75 bis 81 Prozent der Schüler in den beiden Kursen zum Thema Traktor und in der Jahrgangsstufe 12 der Meinung, dass sie diese Fähigkeit benötigen werden. In dem Kurs mit den Aufgaben zum Zug waren nur 41 Prozent davon überzeugt. Dieses ist das einzige Ergebnis, bei dem aus Sicht der Schüler das Modellieren wichtiger ist als das Programmieren.“ (Ortmann 2002, 89)

Zwischenresümee 4.3

Die Lehrenden gaben an, zu wenige Beispiele zu haben, um durchgängig Modellierungen durchführen zu können. Dies ist ein weiterer Beleg, dass es Bedarf für eine Sammlung von Aufgabenklassen zum OOM gibt. In Bezug auf diese zeigte sich bei den Lernenden, dass die erstellten Aufgaben nicht als zu schwierig empfunden wurden. Es zeigten sich aber kursabhängig unterschiedliche Ergebnisse hinsichtlich der Motivation und des themenspezifischen Interesses. Die geringsten Werte hierzu ergab die Befragung im Kurs B-11 („Zug“), in dem die Lernenden auch die geringsten Werte hinsichtlich des Wohlfühlens im Informatikunterricht und der Selbsteinschätzung ihres Stoffverständnisses angaben. Ferner zeigte sich, dass die Unterrichtsinhalte in den Kursen zwar weitestgehend als motivierend, nicht aber als nützlich für die Zukunft eingeschätzt wurden. Einerseits wird vermutet, dass von den Lernenden eher die jeweiligen Kontexte („Traktor“, „Zug“, „Schnick-Schnack-Schnuck“) bewertet wurden als die dahinter liegenden Aufgabenklassen. Dies zeigt erneut, dass die Suche nach geeigneten Kontexten ein schwieriges Problem darstellt, das durch das SUM-Konzept zumindest hier nicht befriedigend gelöst wurde. Andererseits spiegeln sich die Ergebnisse zum Zukunftsnutzen auch in denen zur Haltung gegenüber Modellierung und Programmierung. Dass die Lernenden sich jeweils positiver zur Programmierung als zur Modellierung äußerten, kann mit ihrer jeweiligen Lernbiografie und einer vermuteten stärkeren Programmierungsorientierung in ihrem früheren Informatikunterricht und der daraus resultierenden verstärkten Arbeit am Computer zusammenhängen. Um die Haltung der Lernenden gegenüber der Modellierung zu verbessern, müssten Notwendigkeit und Nutzen für sie greifbarer gestaltet werden, z.B. durch komplexere Beispiele mit höherem Praxisbezug oder bzw. und durch die Arbeit mit

Modellierungswerkzeugen mit integrierten Code-Generatoren. Im Kurs B-11 scheint dies, trotz der eher negativeren Befragungsergebnisse in anderen Bereichen, am besten gelungen zu sein. In der Arbeit mit Modellierungswerkzeugen steckt Chance und Risiko zugleich: einerseits besteht die Möglichkeit, auch während der Modellierungsphase am Computer zu arbeiten mit dem daraus resultierenden Motivationsgewinn handlungsorientierten Lernens. Andererseits kann durch die Wahl eines nicht unterrichtsgeeigneten Werkzeugs viel Unterrichtszeit durch das Erlernen der Handhabung gebunden werden. Diese Gefahr besteht besonders dann, wenn zu frühzeitig professionelle Werkzeuge im Unterricht zum Einsatz kommen. All die genannten Problembereiche liegen aber nicht im Gestaltungsbereich der Aufgabenklassen, sondern im Bereich des Unterrichts, in den sie eingebettet sind (vgl. 5.3.2).

4.6 Erprobung in der Informatiklehrerbildung

4.6.1 Lehramtsstudium

4.6.1.1 Anwendung des Aufgabenklassenkonzepts bei der Gestaltung von Übungsaufgaben

Konzeption

Diese Erprobung wurde durchgeführt im Rahmen der Hauptstudiumsvorlesung „Didaktik der Informatik II“ an der Universität Dortmund im Wintersemester 2001/2002. Die Vorlesung wurde von einer wöchentlichen Übung begleitet, in der die Studierenden 50% der erreichbaren Gesamtpunktezahl aller Übungsaufgaben (jeweils 10 Punkte / Blatt) für einen entsprechenden Nachweis (Übungsschein) erreichen mussten. Im Rahmen einer Lehreinheit wurde das Aufgabenklassen-Konzept auf der Basis von (Brinda 2000b) vorgestellt. An diese Vorlesung schloss sich eine Übung hierzu an. Es handelte sich um die dritte von 14 Übungen, mit der Konsequenz, dass bei den Studierenden noch eine hohe Motivation zur vollständigen Bearbeitung bestand. Das zugehörige Übungsblatt Nr. 3 ist beigefügt im Anhang D.4. Gefordert war darin die Auswahl eines Kontextes für die Gestaltung von Übungsaufgaben, dessen Bewertung mithilfe der Auswahlkriterien für Kontexte (vgl. 4.4.3), die Gestaltung von drei unterschiedlich schwierigen Aufgaben zu einer gegebenen Aufgabenklasse (zu einer textuellen Beschreibung eines Realitätsausschnittes ein Klassendiagramm entwerfen) zum selbstgewählten Kontext und schließlich der Transfer in den Unterricht durch Diskussion von Unterrichtsmethoden und Medieneinsatz. Vorlesung und Übung wurden von unterschiedlichen Personen gehalten (jeweils nicht vom Autor dieser Arbeit). Die Auswertung basiert auf den schriftlichen Lösungen von vier Lehramtsstudierenden (jeweils zwei männlich / weiblich), ohne unterrichtspraktische Erfahrungen im Bereich der Informatik. Die Studierenden werden im Folgenden mit $M_{2,1}$, $M_{2,2}$ (Studenten) und $W_{2,1}$, $W_{2,2}$ (Studentinnen) bezeichnet. Ihre Aufgabenlösungen sind anonymisiert über das Internet verfügbar (vgl. Tabelle 20).

Studierender	URL der Abgabe (alle geprüft am 25.11.03)
$M_{2,1}$	http://www.didaktik-der-informatik.de/DIE_BIB/lehre/ddi/ddi2/wise0102/ueb/3/3m21.pdf
$M_{2,2}$	http://www.didaktik-der-informatik.de/DIE_BIB/lehre/ddi/ddi2/wise0102/ueb/3/3m22.pdf
$W_{2,1}$	http://www.didaktik-der-informatik.de/DIE_BIB/lehre/ddi/ddi2/wise0102/ueb/3/3w21.pdf
$W_{2,2}$	http://www.didaktik-der-informatik.de/DIE_BIB/lehre/ddi/ddi2/wise0102/ueb/3/3w22.pdf

Tabelle 20: URLs der Studierendenlösungen zu Übung 3 (Didaktik der Informatik II)

Auswertung der Lösungen

Auswahl eines Kontextes

Im Rahmen der 1. Aufgabe sollten die Studierenden für in Folgeaufgaben zu entwickelnde Übungsaufgaben einen Kontext auswählen und mithilfe der Auswahlkriterien für Kontexte

(vgl. 4.4.3) überprüfen. Gewählt wurden folgende Kontexte:

- *Supermarkt* ($M_{2,1}$),
- *Auto* ($M_{2,2}$),
- *Schüler- und Lehrerbestenlisten im Abi-Jahrbuch* ($W_{2,1}$),
- *Einkaufsstraße* ($W_{2,2}$).

Auffällig ist, dass speziell $M_{2,2}$ und $W_{2,2}$ sehr geschlechtsbezogene Kontextwahlen trafen. $M_{2,1}$, $M_{2,2}$ und $W_{2,2}$ wählten Realitätsausschnitte als Kontexte ohne das Ziel einer späteren Implementierung als Programm. Dies war hier aber auch nicht gefordert. $W_{2,1}$ entwickelte einen Kontext, der sich auch für eine Umsetzung als Programm eignet.

Überprüfung der gewählten Kontexte

Die gewählten Kontexte sollten anhand der Auswahlkriterien für Kontexte (vgl. Abschnitt 4.4.3) überprüft werden. Dies bereitete den Studierenden keine Probleme. Als Begründung für die „Eignung für OOM“ wurde aufgeführt:

- *Es lassen sich Objekte identifizieren* ($M_{2,1}$, $M_{2,2}$),
- *Es lassen sich Klassen identifizieren* ($W_{2,2}$),
- *Es lassen sich Vererbungsstrukturen identifizieren* ($M_{2,1}$, $M_{2,2}$, $W_{2,2}$).

$W_{2,1}$ begründete ihren Kontext so:

> „Es geht hier kaum um komplizierte algorithmische Probleme. Die Herausforderung besteht tatsächlich darin, sich ein geeignetes Modell von dem Programm zu bilden, das man da konstruieren möchte.“

Keiner der Studierenden nahm explizit Bezug auf andere Relationen zwischen Klassen als die Vererbungsrelation. Deren Anwendbarkeit ist aber speziell für einführende Beispiele keine notwenige Voraussetzung.
$M_{2,1}$ und $M_{2,2}$ vermuteten einen starken Zusammenhang zwischen Lebensweltbezug und Motivation ihrer jeweiligen Kontexte:

> $M_{2,1}$: „Aus diesem Lebensweltbezug resultiert auch die Motivation, die ich in diesem Ansatz sehe. Ein Stück der Wirklichkeit, das jedem bekannt ist, wird mit Hilfe von informatischen Methoden auf einem Rechner abgebildet, und die meisten in der Realität vorkommenden Vorgänge werden sich in diesem Modell wieder finden können.“

> $M_{2,2}$: „Als Beispielkontext wähle ich das Beispiel ‚Verkehrsmittel’, insbesondere das Auto. Der Punkt des Lebensweltbezugs ist erfüllt, da die Lerngruppe ja hier die Sekundarstufe II ist und zu diesem Zeitpunkt sicher einige Schüler darüber nachdenken, den Führerschein zu machen. Aus gleichem Grund sollte auch eine gewisse Motivation vorhanden sein, sich mit dem Thema auch aus informatischer Sicht auseinanderzusetzen.“

Diese Schlussfolgerung ist allerdings sehr fraglich. Wird allerdings der Lebensweltbezug verknüpft mit einem starken Nutzen für die Lernenden, so kann dieser Bezug bestehen.

> $W_{2,1}$: „Die Schülerinnen und Schüler beabsichtigen wahrscheinlich, wie die Jahrgänge vor ihnen, solche ‚Charts’ zu erstellen. Dabei wäre das Programm, was sie modellieren, für sie nützlich, zumal es ihnen eine Menge Arbeit ersparen könnte. Die Chancen, dass die Lerngruppe, was den Kontext betrifft, motiviert ist, stehen also nicht schlecht.“

Gestaltung einer Übungsaufgabe in drei Komplexitätsstufen

Im Rahmen der 2. Aufgabe sollte unter Verwendung des Aufgabenklassenkatalogs eine Übungsaufgabe in drei Komplexitätsstufen entwickelt werden, bei der zu einer textuellen Beschreibung ein Klassendiagramm zu gestalten ist. Hierzu waren jeweils die kognitiven Anforderungen darzustellen. Allen Studierenden gelang es, Aufgaben zu der gegebenen Aufgabenklasse und des jeweils selbstgewählten Beispielkontextes zu entwerfen. Dabei lassen sich bereits deutliche Unterschiede bei den „einfachen“ Aufgaben feststellen. Bei $M_{2,1}$ sollen die Ler-

nenden Klassen im Problembereich identifizieren und mittels der Vererbungsbeziehung strukturieren (ohne Attribute / Methoden). Erstellt wird damit eine Vorstufe eines Klassendiagramms. Bei $M_{2,2}$ wird zunächst ein einfaches Klassendiagramm (ohne Attribute und Methoden) mit einer kleinen Anzahl von Klassen gestaltet. Alle Klassenrelationen sind erlaubt. Die Spezifikation der Relationen mittels Multiplizitäten wird gefordert und eine im Text explizit beschriebene Kompositionsbeziehung soll als solche erkannt werden. Bei $W_{2,1}$ ist ein Klassendiagramm zu einer textuellen Beschreibung zu erstellen. Gestaltungsvorgaben werden dabei keine gemacht. Bei $W_{2,2}$ ist die Aufgabenstellung strukturiert in drei Schritte:

> „a.) Identifizieren Sie geeignete Objekte und gruppieren Sie diese zu Klassen.
> b.) Identifizieren Sie Attribute und Methoden der Klassen.
> c.) Entwerfen Sie ein Klassendiagramm. Geben Sie darin die Multiplizitäten an!“

Ein differenziertes Bild zeigt sich bei der Ausgestaltung von jeweils zwei weiteren, jeweils eine Stufe komplexeren, Aufgaben zum Beispielkontext. Bei $M_{2,1}$ werden auf Basis der Vererbungsstruktur zunächst weitere Klassenrelationen und anschließend Attribute und Methoden ergänzt. $M_{2,1}$ entwickelte drei Aufgaben, die aufeinander aufbauen. Bei den übrigen Studierenden wurde jeweils die Basisaufgabe zur nächsten Komplexitätsstufe geringfügig modifiziert. Bei $M_{2,2}$ werden weitere Bestandteile eines Autos angegeben und eine Beziehung zum Fahrer hergestellt. In der dritten Ausbaustufe sind fünf weitere Klassen selbstständig zu ergänzen und das Klassendiagramm mittels abstrakten Klassen zu restrukturieren. $W_{2,1}$ ergänzte die Problembeschreibung in ihrer ersten Ausbaustufe geringfügig um eine Vererbungsbeziehung. In der zweiten Ausbaustufe sollen Attribute hinzugefügt und eine Listenklasse verwendet werden. Bei $W_{2,2}$ werden zuvor abstrakte Waren in der zweiten Stufe konkretisiert. Darüber hinaus ist keine Änderung erkennbar. In der dritten Stufe werden die Geschäfte und Waren ausgetauscht. Es soll ein Klassendiagramm erstellt, mittels Vererbung restrukturiert und schließlich die neuen Klassen mit Attributen versehen werden. Die Studierenden modellierten eine Zunahme der Komplexität also durch:

- *Erhöhung der Anzahl der verwendeten Fachkonzepte* ($M_{2,1}$, $M_{2,2}$, $W_{2,1}$, $W_{2,2}$),
- eine *stärkere Vernetzung der Klassen durch Relationen* ($M_{2,1}$),
- einen *komplexeren Realitätsausschnitt* bzw. daraus resultierend *mehr Komponenten* ($M_{2,2}$, $W_{2,1}$),
- die *Restrukturierung eines ersten Entwurfs mittels der Vererbungsrelation* ($M_{2,2}$, $W_{2,2}$).

Lediglich die Verwendung von mehr Fachkonzepten zur Erhöhung der Komplexität wurde von allen Studierenden in Betracht gezogen wurde.
Bei der Gestaltung der Aufgaben durch die Studierenden zeigten sich Schwierigkeiten mit der präzisen Formulierung in neutraler Sprache. In Bezug auf das Erstellen einer Klassenhierarchie gab $M_{2,1}$ an:

> „Achten Sie darauf, dass die Hierarchisierung nicht intuitiv geschieht, sondern auf den Problembereich zugeschnitten ist.“

Eine solche Formulierung wird mit hoher Wahrscheinlichkeit Nachfragen der Lernenden nach sich ziehen. Ferner gab $M_{2,1}$ folgenden Auftrag.

> „Fügen Sie in den [...] Klassen insgesamt mindestens 10 Methoden oder Attribute ein.“

Damit wären auch Lösungen mit 10 Methoden und 0 Attributen bzw. 0 Methoden und 10 Attributen zulässig. Zu vermuten ist, dass dies nicht so gemeint war. $M_{2,2}$ wählte für die Beschreibung einer Kompositionsbeziehung eine sehr martialische Formulierung:

> „Berücksichtigen Sie die Multiplizitäten sowie die Tatsache, dass der Kofferraum stets untrennbar mit dem Auto verbunden ist, die Zerstörung des einen also zur Vernichtung des anderen führt.“

Ergebnisse der Genderforschung zeigen, dass solche, offenbar typisch männlichen, Formulierungen insbesondere auf Frauen eher negativ wirken. Der Umgang mit quantitativen Angaben

in der Problembeschreibung ist in Teilen etwas lebensweltfern. $W_{2,2}$ gab an:

> „Jedes Geschäft beinhaltet Waren (100 verschiedene pro Geschäft) und kann diese an Kunden verkaufen."

Geschäfte mit exakt 100 verschiedenen Waren sind nicht sehr realitätsnah.
$M_{2,1}$, $W_{2,1}$ und $M_{2,2}$ beschrieben die kognitiven Anforderungen unter Angabe der erforderlichen OO-Konzepte. Die beste Lösung stammte von $M_{2,2}$, der seine Aufgaben unter Bezugnahme auf die Taxonomie kognitiver Lernziele nach Bloom (1976) diskutierte.

Unterrichtliche Einbettung

Im Rahmen der dritten Aufgabe sollte die Einbettung in Unterricht durch die Angabe von Unterrichtsmethoden und Medien geübt werden. Tabelle 21 und Tabelle 22 zeigen die Vorschläge der Studierenden für Unterrichtsmethoden und Medieneinsatz zu ihren Aufgabenentwürfen.

Student/in	leichte Aufgabe	mittelschwere Aufgabe	schwere Aufgabe
$M_{2,1}$	binnendifferenzierte Gruppenarbeit		Stillarbeit (Einzelarbeit)
$M_{2,2}$	Einzelarbeit	Hausaufgabe	Gruppenarbeit
$W_{2,1}$	kurze Frontalphase mit anschließender Gruppenarbeit		
$W_{2,2}$	Partnerarbeit oder fragend-entwickelnder Unterricht	Partnerarbeit oder fragend-entwickelnder Unterricht	Hausaufgabe

Tabelle 21: Vorschläge für Unterrichtsmethoden

Bei den Unterrichtsmethoden erfolgte eine Konzentration auf Sozialformen (hellgrau) und Handlungsmuster des Unterrichts (mittelgrau) in der Klassifikation von Meyer (1994).

Student/in	Medieneinsatz
$M_{2,1}$	OHP[69] für Präsentation, Computer / UML-Editor zum Erstellen der Diagramme
$M_{2,2}$	Tafel / Flipchart; Software-Werkzeuge, wenn Erfahrungen in der Lerngruppe hierzu vorliegen, sonst Papier und Bleistift
$W_{2,1}$	Papier und Bleistift für Zwischenergebnisse, Tapete / Tonpapier für Endergebnisse. „Vorteil: mehrere Lösungen vergleichen, Lösungen können hängen bleiben, Strukturen prägen sich durch ständige Präsenz besser ein."
$W_{2,2}$	OHP / Tafel; Präsentation der Ergebnisse anhand von Plakaten, Vorstellung / Diskussion am OHP

Tabelle 22: Vorschläge für Medien

Im Hinblick auf die Medienentscheidungen ist bemerkenswert, dass beide Studenten Software-Werkzeuge für die Erstellung von Modellen in Betracht zogen, und beide Studentinnen für die Ergebnispräsentation Tapete bzw. Plakate.

Zwischenresümee 4.4

Es zeigt sich hier die fehlende Erfahrung der Studierenden mit dem Entwurf von Informatikaufgaben zum OOM. Das Gestalten von Informatikübungsaufgaben ist selbst eine nichttriviale Aufgabe, insbesondere dann, wenn es darum geht, unterschiedliche Schwierigkeitsgrade und die unterrichtliche Einbettung zu berücksichtigen. Zur Komplexitätssteigerung verwendeten alle Studierenden eine Erhöhung der Anzahl der in der Aufgabe zu berücksichtigenden Fachkonzepte. Weitere Vorschläge, wie eine stärkere Vernetzung der Komponenten bzw. eine Erhöhung der Komplexität des Realitätsausschnittes wurden jeweils nur von einzelnen Studierenden unterbreitet. Dies trifft auch auf den Arbeitsauftrag zu, das jeweils zuvor erstellte Mo-

[69] Overhead-Projektor

dell zu restrukturieren. Es zeigt sich darin der Bedarf, Möglichkeiten der Komplexitätsbeeinflussung stärker zu explizieren. Diese Erkenntnis trug mit dazu bei, Empfehlungen für die Gestaltung von Niveaustufungen zu entwickeln (vgl. 4.4.2). In den entwickelten Empfehlungen korrespondieren die drei zuerst genannten Vorschläge der Studierenden mit einer Variation der Angaben zu einer Aufgabe, während es sich bei der Restrukturierung um eine Variation des Aufgabentyps handelt. Hier zeigt sich erneut eine Analogie zum Entwurfsmustergedanken (vgl. 4.2). Entwurfsmuster stellen Erfahrungswissen bereit. Um dies Wissen nutzen zu können, sind gewisse Grundkenntnisse als Voraussetzung erforderlich. Mit zunehmender Erfahrung können Entwurfsmuster dazu dienen, Anregungen für bewährte Varianten zu geben und Möglichkeiten der Qualitätsverbesserung aufzuzeigen. Die Verwendung von Aufgabenklassen erfordert eine gewisse Basiserfahrung in der Gestaltung von Informatikaufgaben. Dies hat Konsequenzen für die Lehrerbildung (vgl. 3.7). Der größte Gewinn wird für die Lehrenden vermutet, die über diese Basis verfügen, denen es aber noch an einem Überblick über den Variantenreichtum und die Möglichkeiten im Bereich des objektorientierten Modellierens mangelt.

4.6.1.2 Identifikation von Aufgabenklassen und Kontexten

Im Rahmen einer begleitenden Übung zur Spezialvorlesung „Vertiefung in der Didaktik der Informatik“ im Hauptstudium an der Universität Dortmund hatte ein Lehramtsstudierender im WS 2000/2001 die Aufgabe, Übungsaufgaben zu analysieren und Aufgaben- und Kontextklassen zu identifizieren. Dies gelang ohne weitere Schwierigkeiten.

4.6.2 Lehrerfortbildungen

Im Rahmen der dritten Ausbildungsphase wurden verschiedene Lehrerfortbildungen konzipiert, die neben einer Vorstellung des Konzepts des didaktischen Systems auch teilweise eine praktische Komponente zur Bearbeitung von Übungsaufgaben beinhalteten (vgl. Tabelle 23). Hellgrau unterlegt sind die Veranstaltungen, in denen die Teilnehmerinnen und Teilnehmer auch praktisch am PC arbeiteten.

Datum	Ort	Veranstaltungstitel	Veranstaltungsart	Teilnehmer/innen	Anzahl
07.12.00	Universität Dortmund	OOM ohne Programmiersprache	Workshop zur Lehrerfortbildung (Vortrag + Bearbeitung von Übungsaufgaben am PC)	Informatiklehrende aus dem Raum Köln, Informatiklehrende im Zweiten Bildungsweg aus NRW	7
06.03.01	Universität Dortmund	Unterrichtsbeispiele zum objektorientierten Modellieren mit Informatiksystemen praktisch umsetzen	Workshop zur Lehrerfortbildung (Bearbeitung von Übungsaufgaben am PC)	Informatiklehrende aus dem Raum Dortmund	6
21.05.01	Universität Aachen	Lehr-Lern-Konzept zum objektorientierten Modellieren im Informatikunterricht	Eingeladener Vortrag im Rahmen des fachdidaktischen Kolloquiums	Informatiklehrende, Informatiklehramtsstudierende, Universitätsmitarbeiter/innen	15

Datum	Ort	Veranstaltungstitel	Veranstaltungs-art	Teilnehmer/innen	Anzahl
08.11.01	Bjerringbro, Dänemark[70]	Didactic system for object oriented modeling	Workshop zur Lehrerfortbildung (Vortrag + Bearbeitung von Übungsaufgaben am PC)	Informatiklehrende und Lehramtsstudierende aus Dänemark	24
23.07.02	Universität Dortmund	Objektorientiertes Modellieren in der Sek. II	Vortrag im Rahmen des 1. Informatiktages NRW	Informatiklehrende und Lehramtsstudierende aus NRW	ca. 100
27.11.02	Rahel-Varnhagen Kolleg, Hagen	Objektorientiertes Modellieren mit UML im Informatikunterricht der Sek. II	Workshop zur Lehrerfortbildung (Vortrag + Bearbeitung von Übungsaufgaben am PC)	Informatiklehrende aus den Bereichen Kolleg / Abendgymnasium in NRW	17

Tabelle 23: Lehrerfortbildungen zum Aufgabenklassenkonzept

Wesentliches Ziel der Lehrerfortbildungen war der Praxistransfer der entwickelten Konzepte. Die Grundlage für Workshops mit praktischen Komponenten bildeten die Modellierungsaufgaben im Anhang D.5 (Aufgaben 1 bis 5: 07.12.00, 06.03.01 und 08.11.01[71]; Aufgaben 6-9: 27.11.02). Die jeweiligen Teilnehmerinnen und Teilnehmer bearbeiteten diese entweder in Partner- oder Einzelarbeit am PC unter Verwendung von verschiedenen Modellierungswerkzeugen. Am 07.12.00 und 06.03.01 wurden hierzu jeweils die Werkzeuge *TogetherJ 3* und *Rational Rose 4.0 Demo* eingesetzt, am 08.11.01 erfolgte die praktische Umsetzung mittels *Together 5.5* und am 27.11.02 mit *Poseidon Community Edition* (Weiterentwicklung von *ArgoUML*). Da in allen vier Workshops das Modellierungswerkzeug der überwiegenden Anzahl der Teilnehmerinnen und Teilnehmer nicht bekannt war, bildete ein experimentelles Umgehen damit den Auftakt. Für die praktische Arbeit standen jeweils zwischen 60 und 120 Minuten zur Verfügung. Das Ziel war es nicht, alle Aufgaben in dieser Zeit zu lösen, sondern neue Aufgabentypen mit der Zielgruppe zu erproben und damit Anregungen für den Unterricht zu geben. Es zeigte sich in allen vier Workshops, dass es sich jeweils um sehr vorwissensheterogene Gruppen handelte, in denen einige Teilnehmerinnen und Teilnehmer die Aufgaben zielstrebig und weitestgehend fehlerfrei bearbeiteten, während anderen dies nur mit mehr oder weniger großer Hilfestellung gelang. Schwierigkeiten bereiteten die Spezifikation von Relationen, insbesondere die Festlegung des Relationstyps, die aussagekräftige Benennung und die Bestimmung von Multiplizitäten. In diesem Zusammenhang gab es auch einzelne Handhabungsprobleme mit der Software Together, die darin bestanden, dass gezeichnete Relationen implizit die Relationsrichtung festlegten, und deren Richtungsumkehrung nur mittels Löschen und Neuzeichnen möglich war. Eine besondere Problematik ergab sich bei der Aufgabe zum Sportwettbewerb (Aufgabe 2 im Anhang D.5), bei der die Klasse „Start“ den Lehrenden Schwierigkeiten bereitete. Sie modelliert eine ternäre Assoziation zwischen Teilnehmer, Durchgang und Punktezahl, hat insofern kein direktes lebensweltliches Analogon. Am 27.11.02 wurde mit neuen Aufgaben experimentiert. Einzelne Gruppen zeigten gute Ergebnisse, bei anderen scheiterte es an relativ elementaren Fachproblemen. Zu großen Teilen lag dies daran, dass einige Lehrende in dieser Fortbildung nur über rudimentäre Vorkenntnisse zur objektorientierten Modellierung verfügten, die im Vormittagsteil der Fortbildung vom Autor der vorliegenden Arbeit durch einen Vortrag zu aktivieren versucht wurden. Besondere

[70] Der Autor dieser Arbeit trug seine Ergebnisse auf dem *World Congress for Computers in Education 2001* in Kopenhagen vor. In der Folge dieses Vortrags erging an den Autor die Einladung, im Rahmen einer vom dänischen Bildungsministerium finanzierten Informatiklehrerfortbildung sein Konzept vorzustellen.

[71] Am 08.11.01 wurde eine ins Englische übersetzte Fassung verwendet.

Schwierigkeiten zeigten sich bei einer Aufgabe, bei der Verwandtschaftsbeziehungen zwischen verschiedenen Personen als Objektdiagramm dargestellt waren und bei der gefordert war, hierzu ein Klassendiagramm zu erstellen (Aufgabe 6 im Anhang D.5). Nur wenige Teilnehmerinnen und Teilnehmer erkannten ohne massive Hilfestellung, dass es sich nur um eine einzige Klasse „Person“ mit verschiedenen Assoziationen mit dieser Klasse als Start- und Zielpunkt handelte, in denen die Verwandtschaftsbeziehungen durch Rollenangaben zu spezifizieren waren.
Diskutiert wurde teilweise im Workshopplenum und teilweise in Kleingruppen das unterrichtliche Potenzial dieser Aufgaben. Während einige Teilnehmerinnen und Teilnehmer starken Wert auf die spätere Implementierung eines Modells als Programm legten, konnten andere sich gut vorstellen, solche Modellierungsaufgaben auch ohne konkreten Informatiksystembezug zum Üben und Festigen bestimmter Fachkonzepte im Unterricht zu verwenden. Insbesondere wurden die Vorzüge solcher Modellierungsaufgaben im Vergleich zu Programmieraufgaben im Hinblick auf Klausuren herausgestellt. Eine Teilnehmerin, die an mehreren der o.g. Workshop-Termine teilnahm, berichtete dem Autor im Nachgang per E-Mail:

> „[...] dass [...] sich bei kleineren Übungsaufgaben und Klausuraufgaben folgende Aufgabentypen bewährt haben: Beschreibung der Anforderungen und Vorgabe der Klassen im Klassendiagramm. Attribute, Methoden und Assoziationen, Multiplizitäten, Rollen werden von den Studierenden ergänzt und begründet. Sehr hilfreich, das Verständnis zu testen, war die Erstellung von Objektdiagrammen. Die eigene Erstellung von Sequenzdiagrammen wurde von den Studierenden als schwierig empfunden und gelang nur den Besseren. Bei diesem Aufgabentyp konnte das Verständnis des Zusammenspiels der Objekte noch besser getestet werden.“

Diese Vorgehensweise wurde vom Autor in (Brinda 2000b) empfohlen. Aufgaben zur Erstellung von Sequenzdiagrammen waren darin noch nicht enthalten.

4.7 Zusammenfassung, Fazit und offene Fragen

4.7.1 Zusammenfassung

Bedarf

Ausgehend von der Identifizierung eines Bedarfs an vielfältigen Aufgaben zum objektorientierten Modellieren für den Informatikunterricht (vgl. 2.4.2, Punkt 2) wurde eine Vorgehensweise zur Entwicklung einer strukturierten Sammlung von Aufgabenklassen begründet und angewandt (vgl. 4.3). Das Verfahren wurde zweimal durchlaufen. Nach einem ersten Durchgang erfolgten Erprobungen in der informatischen Bildung, die zu einer Weiterentwicklung der Vorgehensweise führten.

Auswahl von Lehrbüchern

Den Ausgangspunkt bildete die Auswahl von fachwissenschaftlichen *Standardlehrbüchern* zum OOM mit der Voraussetzung darin enthaltener *Übungsaufgaben* (vgl. 4.1). Ausgewählt wurden die Lehrbücher von Rumbaugh et al. (1993) und Balzert (1999).

Auswahl von Aufgaben

Da die darin enthaltenen Aufgaben andere Zielgruppen adressierten, nämlich Informatikstudierende oder im Beruf stehende Informatikerinnen und Informatiker, wurden Kriterien entwickelt, mit denen unterrichtsgeeignete Aufgaben ausgewählt bzw. mit denen nicht unterrichtsgeeignete in geeignete Aufgaben transformiert werden konnten (vgl. 4.3.1). Beim ersten Durchgang wurden hierzu die Kriterien *Fachkonzepte des Informatikunterrichts*, *Betonung der Modellierung*, *Sprachenunabhängigkeit* und *Komplexität* begründet und angewandt. Es stellte sich heraus, dass die Komplexität von Aufgaben nicht schon deren Auswahl für die Sammlung beeinflussen sollte, da dies eine unnötige Einschränkung darstellt. Erst bei der

Gestaltung von Aufgaben für konkrete Lerngruppen können anhand deren Vorkenntnisstruktur Annahmen über geeignete Niveaustufungen getroffen werden. Deshalb wurde das Kriterium *Komplexität* beim zweiten Durchgang nicht weiter berücksichtigt. Ausgewählt wurden im ersten Durchgang schließlich 55 und im zweiten 142 von 256 Aufgaben aus (Rumbaugh et al. 1993) sowie 6 bzw. 46 von 70 Aufgaben aus (Balzert 1999), vgl. Anhang B.

Abstraktion von Aufgaben und Bildung von Aufgabenklassen

Nach der Auswahl wurden die Aufgaben zu so genannten Aufgabenklassen abstrahiert, indem sie von konkreten Erläuterungstexten, Kontexten und Bezeichnern getrennt wurden (vgl. 4.3.2). Jede Aufgabenklasse verknüpft eine Menge von Angaben (textuelle Information; Grafiken, z.B. Diagramme) mit einem Arbeitsauftrag (elementare Aufgabenklasse). Gibt es auf Arbeitsaufträge folgende Ergänzungsaufträge bzw. gibt es zu denselben Angaben alternative Arbeitsaufträge, so handelt es sich um eine komplexe Aufgabenklasse. Erforderlich war, unterschiedliche Begriffsverständnisse (z.B. Objektdiagramm) in den beiden Lehrbüchern zu identifizieren und bei der Bildung von Aufgabenklassen geeignet zu berücksichtigen. Die identifizierten Aufgabenklassen des ersten bzw. des zweiten Durchgangs sind dokumentiert in den Anhängen C.1 und C.2.

Strukturierung von Aufgabenklassen

Das nächste Ziel bestand darin, die Aufgabenklassen in einer Sammlung so zu strukturieren, dass der Zugriff für Lehrende und Lernende leicht möglich ist (vgl. 4.3.3). Dazu wurden die identifizierten Aufgabenklassen hinsichtlich der Eigenschaftsdimensionen *Fachkern*, *Gegenstand* und *Aufgabentyp* klassifiziert. Im Bereich des Fachkerns wurden Aufgaben zum *statischen Modell*, zum *dynamischen Modell* und zu deren *Verknüpfung* identifiziert. Bei den Gegenständen wurden Aufgaben zu *Fachkonzepten der Objektorientierung*, zu *Modellelementen* und zu *Modellen* unterschieden. Ferner wurden die Aufgabentypen *Wissens-* und *Verständnisfragen*, *Beschreibungs-*, *Zuordnungs-*, *Spezifikations-*, *Ordnungs-*, *Diskussions-*, *Analyse-*, *Vergleichs-*, *Validierungs-*, *Identifikations-*, *Modifikations-*, *Transformations-* und *Konstruktionsaufgaben* identifiziert. Diese Aufgabentypen wurden verknüpft mit der Bloom'schen Taxonomie für kognitive Lernziele und begründet, dass für alle kognitiven Anforderungsstufen Aufgaben gestaltet werden können. Dargelegt wurde ferner, welche Verknüpfungen von Gegenständen und Aufgabentypen sinnvoll erscheinen. Daraus ergab sich eine dreidimensionale Klassifikationsstruktur für elementare Aufgabenklassen (vgl. Abbildung 7 auf S. 77). Komplexe Aufgabenklassen korrespondieren zu mehreren Einträgen in diese Struktur. Es wurde eine Sammlung von Aufgabenklassen in zwei Fassungen erstellt. Zunächst erfolgte eine Vorstudie zur Überprüfung der Tragfähigkeit des Konzepts auf der Basis von Aufgaben nur zum statischen Modell (vgl. C.1). Diese Aufgabenklassen bildeten die Grundlage für Erprobungen im Informatikunterricht der Sek. II sowie in der Informatiklehrerbildung. Die Auswertung zeigte die Tragfähigkeit des Konzepts und den Bedarf für Aufgabenklassen auch zum dynamischen Modell sowie zur Verknüpfung von statischem und dynamischem Modell und initiierte damit einen zweiten Durchlauf (vgl. C.2).

Gestaltung von Aufgaben mittels Aufgabenklassen

Die Erkenntnisse aus der Erprobung in der Informatiklehrerausbildung führten zur Entwicklung eines Verfahrens zur Gestaltung von Aufgaben mittels Aufgabenklassen (vgl. 4.4). Dazu wurden die ausgewählten Aufgaben (vgl. 4.3.1, Anhang B) einer Strukturanalyse unterzogen hinsichtlich der Kriterien *Form*, *Komplexität*, *Verfügbarkeit* und *Verwendungshäufigkeit von Angaben* (vgl. 4.4.2) und begründet, wie sich durch deren Variation in Kombination mit der Wahl des Aufgabentyps (vgl. 4.3.3) Niveaustufungen gestalten lassen.

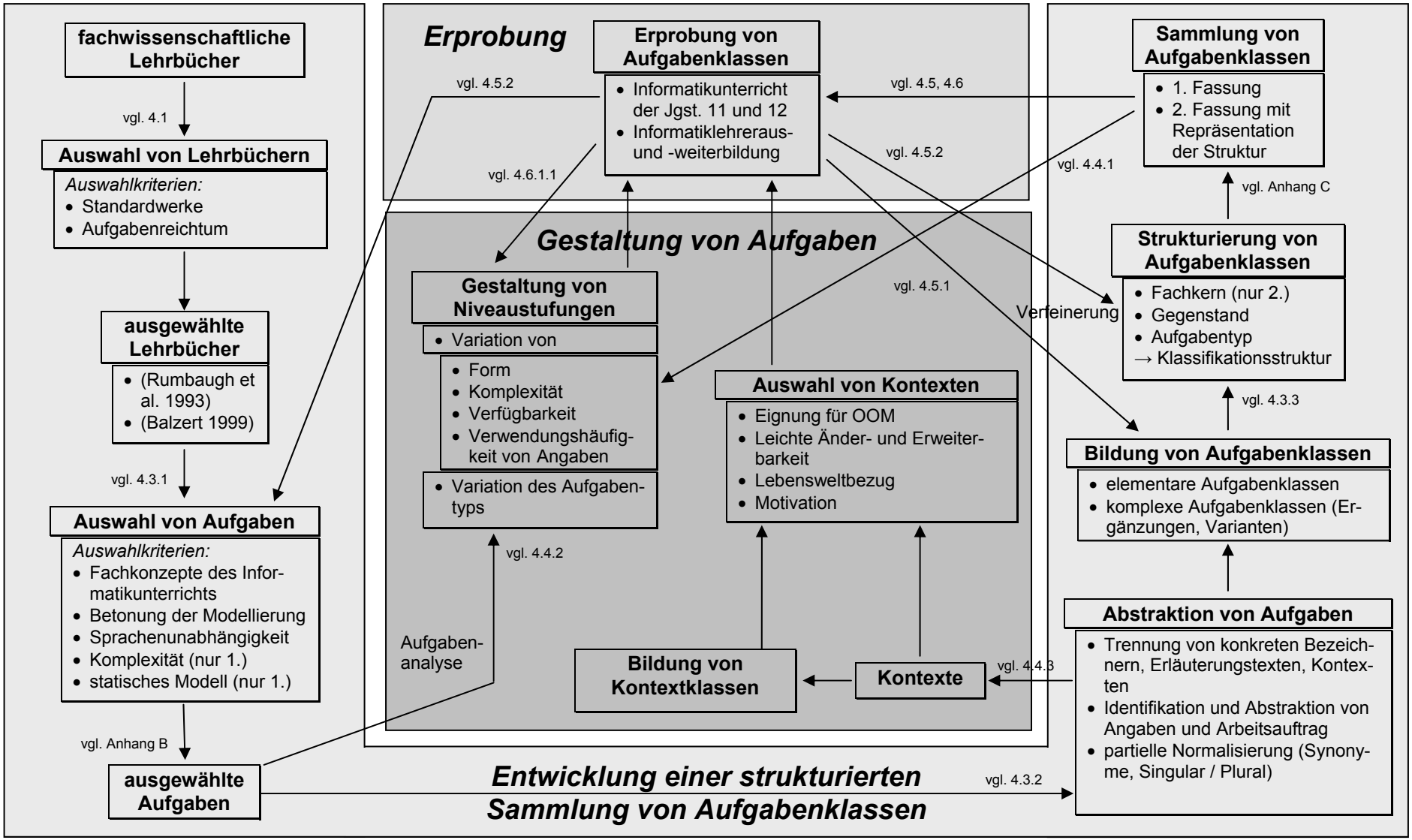


Abbildung 10: Vorgehensweise bei der Entwicklung und Erprobung von Aufgabenklassen und Aufgaben zum OOM

Da neben den Aufgabenklassen für die Gestaltung von Aufgaben auch motivierende Kontexte erforderlich sind, wurden die bei der Abstraktion von den Aufgaben getrennten Kontexte (vgl. 4.3.2) analysiert und die Kriterien *Eignung für OOM*, *leichte Änder- und Erweiterbarkeit*, *Lebensweltbezug*, und *Motivation* für deren Unterrichtseignung begründet (vgl. 4.4.3). Erläutert wurde exemplarisch anhand von ausgewählten Aufgabenklassen, welche Lernprozesse sich damit fördern lassen (vgl. 4.4.4).

Erprobung von Aufgabenklassen

Die erste Fassung der Aufgabenklassen stellte die Grundlage für drei empirische Fallstudien im Informatikunterricht der Jgst. 11 und 12 an zwei Gymnasien im Raum Dortmund dar. Auf der Basis dieser Aufgabenklassen wurden dazu Aufgabenblätter in Kooperation mit dem jeweiligen Fachlehrer entwickelt und eine schriftliche Befragung der Lernenden zur Erfassung des Unterrichtserfolgs im nichtkognitiven Bereich konzipiert (vgl. 4.5.1). Alle drei Kurse wurden im Frühjahr 2002 bei der Bearbeitung der Aufgaben an jeweils zwei bzw. drei Terminen hospitiert. Die anonymisierten Lösungen der Lernenden wurden eingesammelt und anschließend ausgewertet. Die Lernenden erhielten im Gegenzug jeweils eine vorbereitete Musterlösung. Den Abschluss der Fallstudien bildete die Bearbeitung des Fragenbogens durch die Lernenden. Insgesamt zeigte sich die Tragfähigkeit des Konzepts. Als wesentliche Erkenntnisse im Hinblick auf die Weiterentwicklung des Konzepts ergaben sich eine Verfeinerung der inneren Struktur der Aufgabenklassen zur Erleichterung der Auswahl durch Lehrende, eine Ausdehnung auf den Bereich des dynamischen Modells und dessen Verknüpfung mit dem statischen und eine Bereitstellung von Ergänzungen und Varianten, mit denen der Lösungsweg komplexer Aufgaben besser vorstrukturiert werden kann, wenn es das Leistungsbild einer Lerngruppe erfordert (vgl. 4.5.2).
Bei der Erprobung in der Informatiklehrerausbildung, in deren Kontext Lehramtsstudierende Aufgaben für unterschiedliche Niveaustufungen unter Verwendung des Aufgabenklassenkonzepts gestalten sollten, zeigte sich diesbezüglich die fehlende Erfahrung der Studierenden (vgl. 4.6.1.1). Deshalb wurde die Entwicklung einer Methodik zur Gestaltung von Aufgaben zu unterschiedlichen Niveaustufungen initiiert (s.o.). Im Rahmen von Lehrerfortbildungen (vgl. 4.6.2) zeigte sich eine breite Akzeptanz der Lehrenden, aber auch der Bedarf für konzeptorientierte Fortbildungen in diesem Bereich, da objektorientiertes Modellieren für viele Lehrende noch ein neues Thema ist.

4.7.2 Fazit

Zu den wissenschaftlichen Fragestellungen der Arbeit

Mit den Ausführungen zu Aufgabenklassen wurde eine wesentliche Komponente des didaktischen Systems ausführlich insbesondere unter den Aspekten Entwicklungsmethodik, deren Anwendung und Ergebnisse der exemplarischen Erprobung vorgestellt und diskutiert. Begründet wurde der Bedarf für Aufgabenklassen zum objektorientierten Modellieren. Dargelegt wurde ferner der generelle Gewinn durch ein Aufgabenklassenkonzept für Akteure in Lehr-Lern-Prozessen (vgl. 3.3.3, 4.2). Dies begründet, warum Aufgabenklassen eine wesentliche Komponente des didaktischen Systems darstellen (vgl. 2.4.3, Frage 1).
Mit der ausführlichen Beschreibung und Abstraktion der Vorgehensweise wurde dargelegt, wie man bei der Entwicklung von Aufgabenklassen vorgehen kann, um auch die Ausgestaltung weiterer didaktischer Systeme für andere Informatikteilgebiete bestmöglich zu unterstützen. Da Aufgabenklassen als wesentliche Komponente des didaktischen Systems begründet wurden, liefert diese Vorgehensweise damit auch einen Beitrag zur Entwicklung didaktischer Systeme (vgl. 2.4.3, Frage 2).

Mit der Einbeziehung der Aufgabenklassen in exemplarische Fallstudien im Informatikunterricht, mit der Durchführung von Erprobungen in der hochschulischen Informatiklehrerausbildung in Übungen zu Vorlesungen und in Projekt- und in Examensarbeiten konnte exemplarisch gezeigt werden, wie das Konzept der Aufgabenklassen in die informatische Bildung integriert werden kann. Hieraus konnten erste wertvolle Erkenntnisse für die Verfeinerung und Weiterentwicklung des Konzepts gewonnen werden. Mit gleicher Begründung, wie bei Frage 2, wurde damit ein wesentlicher Beitrag zur Beantwortung der dritten Fragestellung dieser Arbeit geleistet (vgl. 2.4.3, Frage 3).

Zur Vorbereitung von Bildungsstandards

Mit dem hier vorgestellten Aufgabenklassenkonzept wird ein Beitrag zur Vorbereitung von Bildungsstandards geleistet[72]. Im Gesamtkonzept der GI für die informatische Bildung an Schulen wurde informatisches Modellieren als eine von vier Leitlinien für die Gestaltung von Informatikunterricht begründet (vgl. 2.2.3). Diese Leitlinien wurden bereits bei der Konzeption einiger neuerer Informatikrichtlinien und -lehrpläne in unterschiedlichem Umfang berücksichtigt (z.B. TK 1999, SBW 2001, MBWK 2001, HK 2003, KSA 2003, HHBBS 2003). Als ein wesentlicher nächster Schritt ist ein Konsens darüber herzustellen, welche Kompetenzstufen Lernende bezüglich der Leitlinien erreichen sollen (vgl. Friedrich 2003) und Aufgaben welcher Aufgabenklassen sie dazu in der Lage sein müssen zu bewältigen. Puhlmann (2003, 148) schlägt im Rahmen dieser Diskussion die Kompetenzstufen *Anwendung*, *Gestaltung* und *Bewertung* vor. Im Hinblick auf das hier betrachtete Themenfeld objektorientiertes Modellieren wird die vorgeschlagene Klassifikationsstruktur für Aufgaben zum objektorientierten Modellieren (vgl. Abbildung 7 auf S. 77) als Grundlage für die Bildung von verfeinerten Kompetenzklassen empfohlen. Definierte Kompetenz- und Aufgabenklassen zu inhaltlichen Schwerpunkten des Informatikunterrichts fördern die Vergleichbarkeit von Bildungsergebnissen sowie die Berücksichtigung der informatischen Bildung bei zukünftigen, international vergleichenden Studien zu den Lernergebnissen von Schülerinnen und Schülern (z.B. PISA-Studie, vgl. OECD 2001), empirische Daten, welche die informatische Bildung in Deutschland für ihre Positionierung im internationalen Vergleich und ihre Weiterentwicklung dringend benötigt. Lehrende erhalten Orientierung hinsichtlich der Gestaltung von Lehr-Lern-Szenarien bei knapper, zur Verfügung stehender, Bildungszeit. Erwartet wird eine Verbesserung der Qualität informatischer Bildungsprozesse.
Auf der Basis von definierten Kompetenzklassen und Fehlerbildern von Lernenden im Hinblick auf Aufgabenklassen lassen sich ferner (softwarebasierte) Lernhilfen (z.B. Explorationsmodule, vgl. Kapitel 5) konzipieren.

4.7.3 Offene Fragen

Die offenen Fragen zu Aufgabenklassen lassen sich folgenden Bereichen zuordnen.

- Umfassende, systematische Evaluation mit Vergleichsgruppen im Hinblick auf Wirksamkeitsaspekte
  Eine umfassende, systematische Evaluation im Informatikunterricht sowie in der Informatiklehrerbildung mittels empirischer Begleitforschung unter Einbeziehung von Vergleichsgruppen über einen längeren Zeitraum im Hinblick auf Lernwirksamkeitsaspekte

[72] Die Gestaltung von Bildungsstandards ist als Konsequenz aus den Ergebnissen der PISA-Studie (OECD 2001) in den Fokus der zuständigen Politiker gelangt. Am 04.12.2003 wurden erste Bildungsstandards für die Fächer Deutsch, Mathematik und erste Fremdsprache (Englisch / Französisch) für den Mittleren Schulabschluss sowie eine Vereinbarung über Bildungsstandards für den Mittleren Schulabschluss (Jahrgangsstufe 10) von der Kultusministerkonferenz verabschiedet. Der Prozess der Standardentwicklung wird 2004 fortgesetzt, vgl. http://www.kmk.org/aktuell/pm031204b.htm (aufgerufen am 05.12.03).

ist noch offen. Zu überprüfen ist insbesondere die Beantwortung der Frage, ob die Aufgabenklassen für Lernende tatsächlich den vermuteten Gewinn im Hinblick auf die Unterstützung beim Lernen durch Analogieschluss bringen können (vgl. 3.3.3, 4.2).

- Weitere Verfeinerung / Erweiterung der Aufgabenklassen zum OOM
  Im Rahmen der vorliegenden Arbeit erfolgte die Entwicklung des Aufgabenklassenkonzepts auf der Basis von zwei Standardlehrbüchern zum objektorientierten Modellieren. Mit dem Ziel, das Aufgabenklassenkonzept weiter zu verfeinern und zu ergänzen, sollte weitere Literatur hinzugezogen und analysiert werden im Hinblick auf weitere Aufgaben- und Kontextklassen.
  Für das Lernen aus Fehlern sind die Lösungsbausteine (Musterlösungen) mit schülertypischen Fehlerbildern zu ergänzen. Diese können durch Befragung von Lehrpersonen erhoben werden.
- Übertragbarkeit des Konzepts auf andere Themengebiete der Informatik
  Zu überprüfen ist ferner exemplarisch, ob das hier vorgestellte Konzept auch für andere Teilgebiete der Informatik tragfähig ist, z.B. im Hinblick auf an OOM direkt angrenzende Gebiete, wie z.B. die objektorientierte Programmierung, oder im Hinblick auf völlig andere Teilgebiete der Informatik (z.B. Wirkprinzipien von Rechnernetzen oder Datenbanken). Zu beantworten sind in diesem Zusammenhang beispielsweise die Fragen, welche grundlegenden Aufgabentypen sich bei anderen Themenbereichen identifizieren lassen und ob es dabei Gemeinsamkeiten gibt bezüglich der Aufgabenklassen, die Rahmen der vorliegenden Arbeit identifiziert wurden.

5 Explorationsmodule

5.1 Überblick

In den Abschnitten 2.3.2 und 2.4.2 (Punkt 3) wurde auf der Basis von Literaturanalysen zum objektorientierten Modellieren im Informatikunterricht (vgl. 2.3) ein Mangel an unterrichtsgeeigneten Informatiksystemen zur Unterstützung von Lernenden beim Aneignen von Fachkonzepten des objektorientierten Modellierens festgestellt. Unterricht wird häufig projektorientiert organisiert. Zu schwierige Projektziele und Erfolgsdruck führen nicht selten zu starker Steuerung durch den Lehrenden. Arbeitsteilung zur Förderung der Teamfähigkeit ist zwar ein wertvolles Unterrichtsziel, beeinträchtigt aber die Möglichkeiten für individuelle Lernerfahrungen. Zeitmangel aufgrund von fachlichen Vermittlungszielen erschwert die Einbeziehung „spielerischer" bzw. „experimenteller" Herangehensweisen an Informatikkonzepte.
Keinesfalls soll damit diese bewährte Form der Unterrichtsgestaltung in Frage gestellt werden. Im Kapitel 5 werden Ergebnisse zur Entwicklung und zur Erprobung von neuen, handlungsorientierten Lernangeboten, so genannten *Explorationsmodulen (EM) (*synonym: *Explorationsbausteine)* zum objektorientierten Modellieren, vorgestellt und diskutiert. Vorrangiges Ziel von Entwicklung und Erprobung dieser EMs ist es, traditionelle Lehr-Lern-Szenarios des Informatikunterrichts durch die Einbeziehung von Elementen des explorativen Lernens zu bereichern und damit zu einer Verbesserung der Qualität der Lehr-Lern-Prozesse beizutragen. Den Ausgangspunkt bildet die Erörterung des Stellenwerts von Explorationsmodulen im didaktischen System (vgl. 5.2). Der Prozess der Entwicklung von EMs wird im Abschnitt 5.3 dargestellt. Hierbei werden Grundlagen zum explorativen Lernen (vgl. 5.3.1.1) und zum explorativen Lernen mit Informatiksystemen (vgl. 5.3.1.2) mit den Ergebnissen einer fachdidaktischen Analyse von Werkzeugen zum objektorientierten Modellieren (vgl. 5.3.2) zu einem Konzept für Explorationsmodule verknüpft (vgl. 5.3.3) und dieses dann bei der Entwicklung von Prototypen exemplarisch implementiert (vgl. 5.3.4). Ein Architekturkonzept zur Vereinfachung der Gestaltung von Software zur Exploration wird vorgestellt im Abschnitt 5.3.5. Das Lernen mit Explorationsmodulen wird im Abschnitt 5.4 eingebettet in ein Lernkonzept. Komponenten dieses Konzepts wurden in den Jahren 2001 bis 2003 erfolgreich in der informatischen Bildung erprobt (vgl. 5.5). Zusammenfassung mit grafischem Überblick über den Entwicklungs- und Erprobungsprozess (vgl. Abbildung 22 auf S. 169), Fazit und offene Fragen bilden den Abschluss im Abschnitt 5.6.

5.2 Stellenwert der Explorationsmodule im didaktischen System

Um den Stellenwert von Explorationsmodulen im didaktischen System zu beurteilen, werden Vor- und Nachteile für Akteure im Kontext des didaktischen Systems erörtert (vgl. 3.4.3). Wesentliche Akteure sind

- *Lehrende,*
- *Lehramtsstudierende*,
- *Lernende* und
- *Entwicklerinnen und Entwickler* des didaktischen Systems.

Exploratives Lernen gilt im Unterricht als besonders erstrebenswert. Da Explorationsprozesse stark lernergesteuert sind, ergibt sich daraus für *Lehrende* (und *Lehramtsstudierende*) die Anforderung einer besonders intensiven Unterrichtsplanung, wenn sie entsprechend flexibel und beweglich im Unterricht agieren möchten (vgl. 5.3.1.1). Durch die Heranführung von Lernenden an die Prozesse explorativen Lernens in softwarebasierten, geschlossenen Explorationsumgebungen (vgl. 5.4.2) können Explorationsprozesse durch Auswahl und Kombination von

Explorationsmodulen durch den Lehrenden beeinflusst und das Erkenntnisinteresse der Lernenden auf die für den jeweiligen Lernprozess relevanten Aspekte gelenkt werden. Hierin liegt Vor- und Nachteil zugleich, weil damit nur die Entdeckungen möglich sind, die vom Entwickler des jeweiligen Explorationsmoduls vorgesehen wurden. Wenn die Lernenden entsprechend lernkompetent sind, ihre individuellen Lernprozesse zielgerichtet auf unterrichtsrelevante Lernziele zu lenken, können geschlossene Explorationsumgebungen schrittweise geöffnet werden. Generell ist Unterricht, der auf explorativem Lernen basiert, deutlich weniger steuerbar als traditioneller Frontalunterricht. Für Lehrende ergibt sich die Aufgabe, Problemsituationen zu entwerfen, diese mit Medien für explorativen Unterricht zu verknüpfen und Lernenden Handlungsrahmen für ihre Explorationsprozesse zu vermitteln (vgl. Straka / Macke 1981). Durch Leitfragen können Lehrende Lernende dazu stimulieren, ihre individuellen Erkundungs- und Entdeckungsprozesse auf die relevanten Aspekte zu lenken (Neber 1988, vgl. 5.3.1.1). Ein potentielles Problem besteht darin, dass der Zeitaufwand im Unterricht deutlich höher ist als bei traditionellem Frontalunterricht (vgl. Steindorf 1995, 64).
Für *Lernende* liegt das Potenzial von Explorationsmodulen im Wesentlichen in den Bereichen

- *Endecken fachlicher Zusammenhänge,*
- *Entwicklung der individuellen Lernkompetenz.*

Das Entdecken fachlicher Zusammenhänge zum objektorientierten Modellieren fördert die Sach- und Methodenkompetenz der Lernenden. Durch Explorationsmodule und deren Einbettung in den Unterricht wird ein handlungsorientierter, lernerzentrierter Zugang zu theoretischen Informatikfachkonzepten ermöglicht, der sonst vielfach und oft ausschließlich auf dem Wege der Programmierung erfolgte. Verschiedene Lehrervorträge werden dadurch überflüssig. Informatikunterricht der Vergangenheit setzte auf die Entwicklung von kleinen Programmen, um daraus Erkenntnisse über Informatiksysteme in der Lebenswelt abzuleiten. Das funktionierte, solange beiden, den Unterrichtsbeispielen und den professionellen Problemlösungen, die gleichen Vorgehensmodelle, Strategien und Methoden zugrunde lagen. Lernende handelten dabei wie Expertinnen und Experten, im Vergleich zu diesen aber mit Gegenständen von stark reduzierter Komplexität. Inzwischen sind die Vorgehensmodelle, Methoden, Algorithmen und Datenstrukturen der professionellen Entwicklung großer Informatiksysteme prinzipiell andere, so dass der Transfer vom „Programmieren im Kleinen“ auf das „Programmieren im Großen“ nicht empfohlen wird (vgl. Appelrath et al. 1998). Im Vordergrund des Unterrichts steht häufig auf Anwenden von Standard-Software und Problemlösen in einer Programmiersprache. Durch die bloße Anwendung komplexer Informatiksysteme bleiben ihre innere Struktur und die dieser zugrunde liegenden Wirkprinzipien oft verborgen. Es lassen sich in diesem Kontext einzelne Probleme identifizieren, die für einen Zugang über die Programmierung im Unterricht zugänglich sind. Es besteht aber die große Gefahr, das Zusammenspiel verborgener Einzelkomponenten nicht angemessen zu berücksichtigen. Mit Explorationsmodulen kann diese Lücke geschlossen werden, da damit auch ein Zugang zu der inneren Gestaltung von komplexen Informatiksystemen möglich wird, der Teile ihrer Modellierung bei Verzicht auf Programmierung offen legt und eine Exploration der ihnen zugrunde Wirkprinzipien unterstützt. Die Fachkonzepte und deren innere Zusammenhänge können so von Lernenden exploriert werden, ohne dass durch langwierige, vorgelagerte Programmierphasen, in denen sehr viel sekundäres Faktenwissen erworben werden muss, zu viel Unterrichtszeit gebunden wird. Durch Explorationsmodule kann der informatische Inhalt in den Mittelpunkt des Unterrichts gerückt werden, und somit durch besondere, zielgruppenspezifische Formen der Veranschaulichung und der Interaktion eine Konzentration auf das Wesentliche erfolgen (vgl. Steinkamp 1999). Lernende erhalten die Möglichkeit der aktiven Auseinandersetzung mit dem Stoff. Mit auftretenden Problemen werden sie nicht allein gelassen. Diese werden zu einem wichtigen Element der Diskussion in der Lerngruppe (vgl. Kerres 2001, 278ff). In der Interaktion mit den Explorationsmodulen können sie logische Strukturen und Gedankenmo-

delle in Form von Hypothesen mit Experimenten auf Wirksamkeit prüfen. Dadurch wird ihre Hypothesenbildung gefördert. Durch die Kombination aus Veranschaulichungen und Rückmeldungen des Systems werden ferner das Lernen aus Beispielen und das Lernen aus Fehlern gefördert. Orientiert an Leitfragen explorieren die Lernenden fachliche Zusammenhänge. Damit werden auch ihre Fähig- und Fertigkeiten beim explorativen Lernen sowie bei der selbstbestimmten Organisation von Lernprozessen geschult (vgl. Kerres 2001, 217ff). Das steigert ihre Lernkompetenz und stellt damit eine gute Vorbereitung auf lebensbegleitende Lernprozesse dar. Durch das aktive, lernerzentrierte Lernen kann der Grad an intrinsischer Motivation und des Interesses am Unterricht erhöht werden (ebd., 139f).
Für *Entwicklerinnen und Entwickler* des didaktischen Systems stellen vorhandene Explorationsmodule Anregungen und Ausgangspunkte für weitere Entwicklungen dar. Insbesondere können über die Analyse von Gemeinsamkeiten Werkzeuge gestaltet werden, die den Entwicklungsprozess rationalisieren (vgl. Przygienda 2002, Hoffmann 2003; Abschnitt 5.3.5 in der vorliegenden Arbeit). Für den Inhaltsbereich von Explorationsmodulen stellt die Sammlung der Aufgabenklassen (vgl. Kapitel 4) einen reichen Vorrat an Ideen bereit. Vorhandene Explorationsmodule können die Entwicklung von Informatikunterricht insgesamt bereichern, weil dadurch die fachdidaktische Diskussion angestoßen und neue Impulse für zukünftige Gestaltungsdimensionen gegeben werden können.
Lernen mit Explorationsmodulen soll traditionellen Unterricht keinesfalls *ersetzen*. Es stellt eine interessante Zugangsalternative zu fachlichen Inhalten in geeigneten Phasen des Lehr-Lern-Prozesses dar.

5.3 Entwicklung von Explorationsmodulen

5.3.1 Grundlagen eines Konzeptes für Explorationsmodule

5.3.1.1 Exploratives / entdeckendes Lernen

Formen, Grundmuster und Charakterisierung
Da exploratives[73] bzw. entdeckendes Lernen (vgl. Bruner 1960, Neber 1981) oftmals mit großem Engagement und Begeisterung, selbstreguliert und scheinbar mühelos, erfolgt, wurde und wird auch in der informatischen Bildung sowohl in Schulen (z.B. Magenheim 2001) als auch in Hochschulen (z.B. Turner 1998, Diehl / Kerren 2001) angestrebt, entsprechende Lernprozesse zielorientiert anzuregen, um Lernende damit insbesondere auf lebensbegleitendes Lernen (life-long learning – LLL) vorzubereiten. Unter explorativem Lernen werden nach Neber (1981, 45ff) die Lernformen

- *induktives,*
- *forschendes,*
- *aktives,*
- *problemorientiertes* und
- *sokratisches*

Lernen zusammengefasst. Nach Paul (1994, 87ff) lassen sich diese Formen auf die drei Grundmuster

- *Lernen aus Beispielen*,

[73] „**Ex|plo|ra|tion** *[...zion*]: die; -, -en: Untersuchung u. Befragung; Nachforschung; das Explorieren.“ (Drosdowski et al. 1990, 239, Hervorhebungen im Original)
„**ex|plo|rie|ren** [*lat.*]: 1. erforschen, untersuchen, erkunden (z.B. Boden, Gelände). 2. [Personen]gruppen zu Untersuchungs-, Erkundungszwecken befragen, ausforschen; (Verhältnisse) durch Befragung u. Gespräche untersuchen, erkunden (Psychol., Med.)“ (Drosdowski et al. 1990, 239, Hervorhebungen im Original)

- *Lernen durch Experimentieren* und
- *Lernen durch Konfliktlösung*

zurückführen. Diese Grundmuster gehören zum Standardrepertoire der Unterrichtsgestaltung (Meyer 1994). Unterricht, der sich an diesen Grundmustern orientiert, ermöglicht exploratives Lernen, ist aber keineswegs hinreichend hierfür. Kerres (2001, 217) charakterisiert exploratives Lernen als *entdeckendes*, *forschendes* und *autonomes* Lernen, betont die hohe Selbstverbindlichkeit und nennt folgende Merkmale (ebd., 218):

- „Die Person steckt sich selbst ein Lernziel: Sie möchte etwas wissen oder können.
- Die Person initiiert verschiedene Handlungen, um dieses selbst gesteckte Ziel zu erreichen: Sie entscheidet, welche Lernaktivitäten in welcher Sequenz auszuführen sind.
- Das Lernen vollzieht sich dabei nicht als linearer Prozeß von einer Thematik bzw. Schwierigkeitsstufe zur nächsten, sondern eher spiralenförmig: Die Person tastet sich in verschiedene Richtungen weiter, sie kann sich in Sackgassen begeben, bevor sie an eine frühere Stelle zurückkehrt etc.
- Der Vollzug dieser Aktivitäten wird als befriedigend erlebt (und nicht [nur] das Handlungsergebnis)."

Unterscheiden lassen sich folgende Erscheinungsformen der Exploration (Paul 1994):

1. *Manipulative Exploration* hängt von der Tatsache ab, dass die explorierende Person etwas ändern kann und darauf ein sofortiges Feedback erhält.
2. Bei der *perzeptiven Exploration* müssen den explorierenden Sinnen Strukturen angeboten werden, die entdeckt werden können. Für die Entdeckung von Unbekanntem sollte es dennoch bekannte Bereiche geben, die als Grundlage für das neue Wissen dienen.

Explorationsstrategien

In Abgrenzung vom nicht angestrebten, unsystematischen Ausprobieren (Versuch-Irrtum-Strategie) wurden systematische Vorgehensweisen für die oben genannten Grundmuster beschrieben (Neber 1988). So sollen Lernende beim Lernen durch Experimentieren dazu angeregt werden, allgemeine Fragen in handlungsleitende bzw. hypothesentestende Fragen umzuformen, um so zu einer aktiven Auseinandersetzung mit dem kennen zu lernenden Gegenstand zu gelangen. Von Neber werden hierzu sechs Frageniveaus unterschieden, die vom Allgemeinen zum Speziellen führen:

- Was ist das?
- Wie funktioniert das?
- Was passiert, wenn ich dieses oder jenes tue?
- Wie kann ich ein bestimmtes Ziel erreichen?
- Hängt die Vorgehensweise zum Erreichen dieses Ziels von bestimmten Bedingungen ab?
- Gibt es Beziehungen zwischen den verschiedenen Bedingungen für das Erreichen des Ziels?

Beim Lernen durch Konfliktlösung wird der Problemlösevorgang durch ein fünfstufiges Schema erklärt:

1. kognitiver Konflikt, allgemeine Orientierung,
2. Identifikation des Problems, Trennung von Bekanntem und Unbekanntem,
3. vorläufiger Lösungsplan, Transfer von Bekanntem auf Unbekanntes; Umwandlung vager Ziele in genaue, testbare Alternativen,
4. Ausführung des vorläufigen Plans, Überprüfung der Hypothese, genaue Registrierung auftretender Konsequenzen,
5. Überprüfung des Ergebnisses der Planausführung, Vergleich der eingetretenen Konsequenzen mit Erwartungen, ggfs. Modifikation des Modells.

Durch die Thematisierung von Explorationsstrategien vor explorativ gestalteten Unterrichtsphasen sollen Lernende dazu angeregt werden, ihre Lernprozesse zielorientiert zu organisieren und somit effektiver zu gestalten (Kerres 2001, 222f).

Eigenschaften

Die Neugier- und Fragehaltung der Lernenden wird durch exploratives Lernen geweckt, ihre Hypothesenbildung gefördert und selbst organisiertes Lernen angeregt. Es zeigt sich allerdings ein scheinbarer Widerspruch, der zwischen mühelosen, nicht angeordneten, explorativen, durch Lernende kontrollierten und somit pädagogisch besonders wertvollen Lernaktivitäten einerseits und planvollen bzw. durch Lehrende organisierten Lehr-Lern-Prozessen andererseits besteht (ebd., 217ff). Dieser Widerspruch lässt sich jedoch auflösen, indem nicht die *Herstellung* explorativer Lernprozesse, sondern durch entsprechende Gestaltung des Unterrichtsprozesses und in ihm verwendeter Medien deren *Ermöglichung* bzw. *Anregung* im Mittelpunkt des „didaktischen Designs" steht (ebd., 220). Die Gestaltung explorativer Lernphasen im Unterricht erfordert eine besonders intensive Planung (ebd., 221), weil viele daraus resultierende, mögliche Verlaufsstrukturen berücksichtigt werden müssen.
Persönlichkeitsmerkmale, die bei den Lernenden durch exploratives Lernen einerseits ausgeprägt und gefördert werden (sollen), stellen andererseits in Teilen auch Voraussetzungen hierfür dar. So stellt die Ausprägung des Neugiermotivs bei Personen eine wesentliche Bedingung für exploratives Lernen dar (ebd., 221f). Ebenso müssen gewisse Fertigkeiten zur Selbstkontrolle des Lernprozesses sowie ein Mindestmaß an fachlichem Vorwissen bei den Lernenden vorliegen, damit exploratives Lernen zuverlässig zu Lernerfolgen führt (Steindorf 1995, 64). Ferner stellen subjektive Theorien über das Lernen sowohl bei Lehrenden als auch bei Lernenden ein Problem dar, da beide oft glauben, „dass Lernen ein Vorgang ist, der einer ‚Unterweisung' bedarf. Sie hängen dem […] Kopiermodell des Wissens an, was den Erfolg explorativen Lernens grundsätzlich in Frage stellt" (Kerres 2001, 222).
Bei Lernenden in der Anfangsphase, mit noch nicht genügend ausgeprägten Fertigkeiten zur Selbstkontrolle ihrer individuellen Lernprozesse, ist der Erfolg beim explorativen Lernen also wenig prognostizierbar. Durch die Verknüpfung von traditionellem Unterricht zum Aufbau eines fachlichen Fundaments mit überschaubaren, explorativen Lernphasen, können Lernende schrittweise zum Aufbau eigener, systematischer Explorationsstrategien und damit zu einer Verbesserung der Selbstorganisation ihrer individuellen Lernprozesse angeleitet werden. Die Zielorientierung kann in der Anfangsphase durch vom Lehrenden in den Unterricht einzubringende, oder von den Lernenden in der Gruppe vereinbarten, Leitfragen festgelegt werden. Später strukturieren die Lernenden ihre explorativen Lernprozesse zunehmend selbstständiger.
Lernhilfen sind erforderlich, die Schülerinnen und Schüler in ihren individuellen Lernprozessen unterstützen.

5.3.1.2 Exploratives Lernen mit Informatiksystemen

Informatik-Experimente im Schullabor

Eine erste eigene konzeptionelle und softwaretechnische Erkundung hierzu fand im Rahmen einer vom Autor dieser Arbeit in der Fachgruppe „Didaktik der Informatik" an der Universität Dortmund begleiteten Diplomarbeit statt, in der ein Konzept für „Informatik-Experimente im Schullabor" entwickelt wurde (vgl. Steinkamp 1999). Wie in Abschnitt 5.3.1.1 dargestellt, ermöglicht Lernen durch Experimentieren explorative Lernprozesse. Im Rahmen der Diplomarbeit wurde der Experimentbegriff des naturwissenschaftlichen und technischen Unterrichts in die Informatik übertragen. In seiner Studie zum Experimentbegriff kommt Steinkamp zu dem Ergebnis, dass

- *entdeckende* und *erkundende Experimente*
  (Experimente, die zu einer Hypothese über den Untersuchungsgegenstand führen),
- *prüfende Experimente*
  (Experimente, bei denen eine zu überprüfende Hypothese den Ausgangspunkt bildet),
- *strukturermittelnde Experimente*
  (Experimente, bei denen Erkenntnisse zum Zusammenspiel mehrerer Komponenten im Mittelpunkt des Interesses stehen)

im Informatikunterricht nicht hinlänglich vertreten sind (ebd., 38). Bei der Diskussion von Anwendungsbereichen bezieht er sich auf die Leitlinien des „Gesamtkonzepts der informatischen Bildung“ in der Arbeitsfassung von 1999 (GIFA 1999, vgl. auch S. 13 in der vorliegenden Arbeit) und begründet, dass sich „Informatik-Experimente [...] hauptsächlich zur Veranschaulichung der Bereiche Wirkprinzipien [von Informatiksystemen] und informatische Modelle“ eignen (Steinkamp 1999, 11f). Für das Experimentieren im Informatikunterricht schlägt er unter Bezug auf Schubert (1991) den Aktivitätszyklus vor, der in Abbildung 11 dargestellt ist.

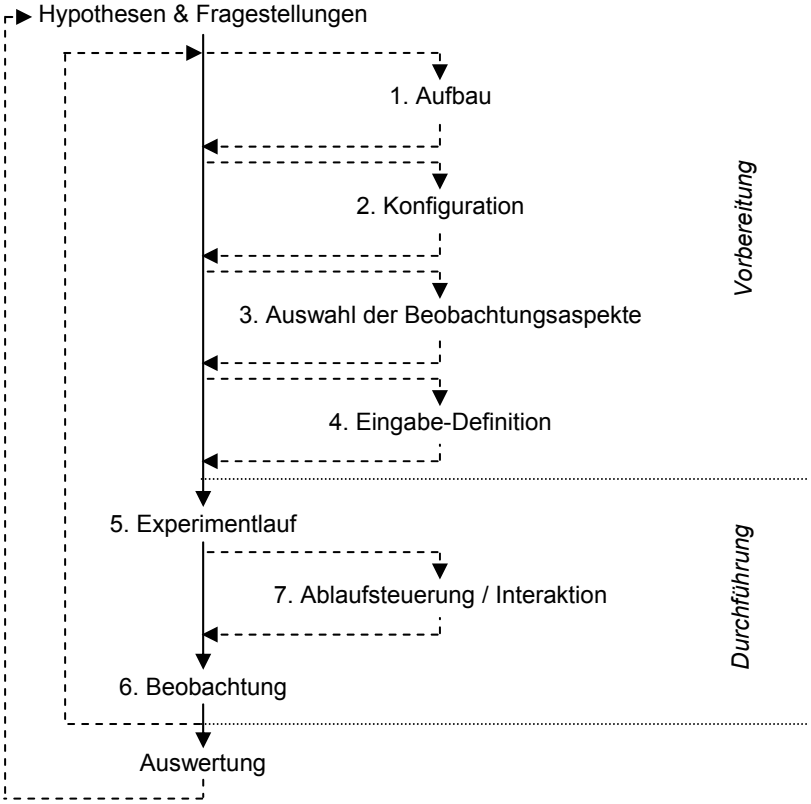


Abbildung 11: Aktivitätszyklus beim Experimentieren (vgl. Steinkamp 1999, 42)

Im Gegensatz zu anderen Fächern (z.B. Chemie, Physik) besitzt die Informatik aber noch keine Experimentkultur:

> „**Experiment**: Methode, um eine Vermutung, eine These oder eine Theorie zu überprüfen, um einen Effekt zu untersuchen oder um die Einsatzmöglichkeiten zu demonstrieren. Da Systeme der Informatik oft unüberschaubar sind, überprüft man gewisse Eigenschaften durch Experimente. Man spricht aber erst von Experimenten, wenn hierbei entweder umfangreiche Messreihen erforderlich sind [...] oder wenn der Einsatz von Methoden, Oberflächen, Schnittstellen usw. sich nur in der Praxis feststellen lässt [...] Typische Experimente in der Informatik sind: Tests (↑Testen), Messreihen, ↑Fallstudien, ↑Feldbeobachtungen und kontrollierte Experimente. Die Informatik besitzt noch keine ‚Experimentkultur', vielmehr enthalten Artikel über Experimente meist viele ungenaue Anhaltspunkte und die Resultate lassen sich dann nur angenähert oder auch gar nicht reproduzieren, vor allem, wenn Menschen handelnd mitwirken. Dennoch sind Ergebnisse von Expe-

rimenten mangels anderer Kriterien in der Praxis oft die Basis für Entwurfs- und Gestaltungsentscheidungen.“ (Claus / Schwill 2001, 233f, Hervorhebung im Original)

Anders als in den o.g. Fächern sind Experimente als Methode des Erkenntnisgewinns in informatischen Lehr-Lern-Prozessen noch nicht etabliert. Experimente in der Informatik in diesem Sinne erfordern spezielle Experimentierumgebungen als Lernhilfen, die als Informatiksysteme realisiert werden können. Durch entsprechende Gestaltung können damit ansonsten verborgene Wirkprinzipien von Informatiksystemen für Lehr-Lern-Prozesse beobachtbar und zugänglich gemacht werden. Die naturwissenschaftlichen Fächer verwenden hierfür Experimentieranlagen. Die Technikausbildung hat eine lange Tradition mit Baukästen. Auf der Basis der Schritte des Aktivitätszyklus beim Experimentieren (vgl. Abbildung 11) entwickelte Steinkamp eine Architektur für Umgebungen für Informatik-Experimente (vgl. Steinkamp 1999, 98 und Abbildung 12).

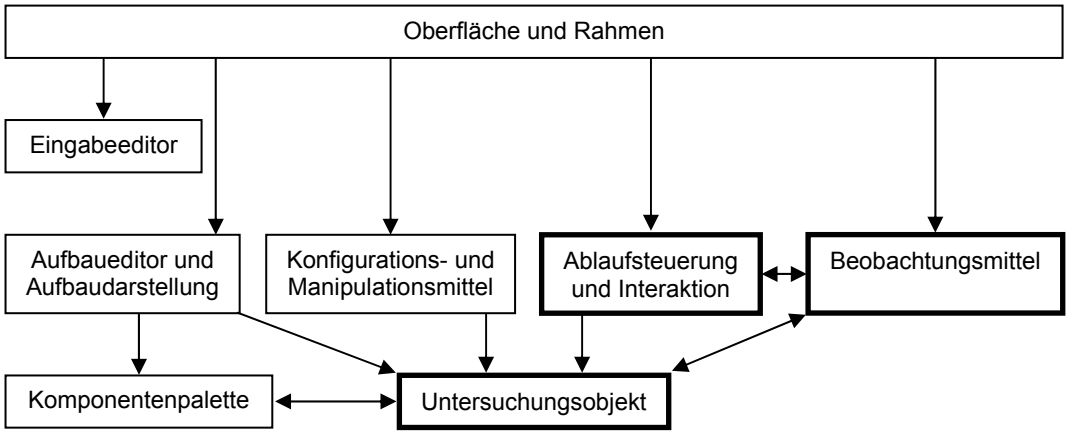


Abbildung 12: Architektur für Umgebungen für Informatik-Experimente (vgl. Steinkamp 1999, 47)

Den unverzichtbaren Kern bilden darin die besonders hervorgehobenen Subsysteme, die mit den Komponenten des Model-View-Controller-Paradigmas korrespondieren (Model – Untersuchungsobjekt, View – Beobachtungsmittel, Ablaufsteuerung und Interaktion – Controller), die übrigen Komponenten sind in Abhängigkeit vom konkretem Experiment fakultativ.
Auf dieser Basis lieferte Steinkamp Vorschläge für Informatik-Experimente. Dabei thematisierte er auch den Bereich der informatischen Modelle und wählte als Beispiele die „Turing-Maschine“ und die „(verallgemeinerte) Register Maschine“ (ebd., 59f). Objektorientiertes Modellieren wurde nicht berücksichtigt.

Exemplarische Implementierung des Konzepts

Das Konzept wurde exemplarisch implementiert. Entwickelt wurde eine Experimentierumgebung, in der Lernende (in einem Ausschnitt der Internet-Architektur) Netzwerkkomponenten, die an der Anforderung einer Webseite beteiligt sind, konstruieren, verknüpfen, konfigurieren und dann die Datenkommunikation zwischen diesen durch Protokollsimulation überprüfen können (ebd., 66ff). Bei der Anwendung eines Webbrowsers bei der Informationsrecherche im Internet bleiben den Lernenden die zugrunde liegenden Wirkprinzipien verborgen. Abbildung 13 zeigt den Arbeitsbereich, in dem Lernende die für die Anforderung einer Webseite erforderlichen Strukturen von Netzwerkkomponenten konstruieren, verknüpfen und konfigurieren können. Im „Kommunikationsmonitor” (vgl. Abbildung 14) werden Kommunikationsprozesse zwischen den Komponenten in einer an das Hypertext-Transfer-Protokoll (http) angelehnten Darstellungsweise visualisiert.

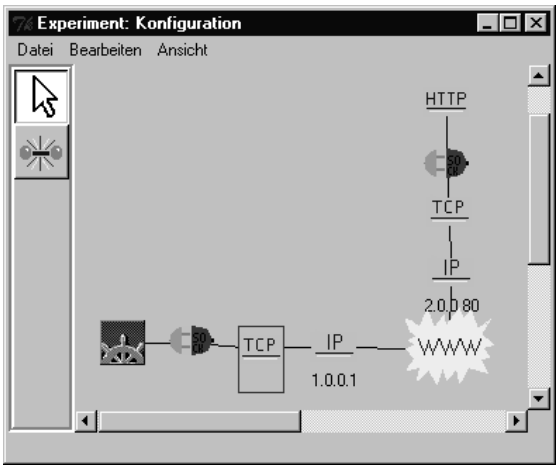


Abbildung 13: Arbeitsbereich für die Exploration der Kommunikation zwischen Netzwerkkomponenten, die an der Anforderung einer Webseite beteiligt sind

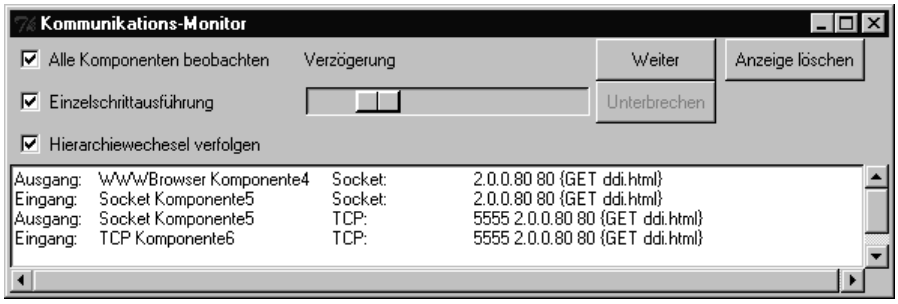


Abbildung 14: Kommunikationsmonitor

Damit Lernende feststellen können, ob der gewählte Aufbau und die Konfiguration der Komponenten die gewünschte Datenkommunikation ermöglichten, wird ein simuliertes Browserfenster für die Ergebnispräsentation angeboten. Im positiven Fall erscheint darin, wie bei einem realen Webbrowser, entweder das angeforderte Dokument oder die Nachricht, dass dieses nicht verfügbar ist. Die Software wurde im Rahmen von Studienwerbeveranstaltungen an der Universität Dortmund (Tag der offenen Tür, Schnupperuni[74] zwischen 1999 und 2002) sowie in begleitenden Übungen zur Grundvorlesung für Informatik-Lehramtsstudierende „Didaktik der Informatik I“ mehrfach erprobt und sowohl von Schülerinnen und Schülern als auch von Lehramtsstudierenden und Informatik-Lehrenden mit großem Interesse aufgenommen. Steinkamp begründete folgende Vorteile des Experimentkonzeptes für den Informatikunterricht (ebd., 4ff):

- Erweiterung des handlungsorientiert zugänglichen Spektrums,
- Unterstützung bei der Erarbeitung von Themen der theoretischen Informatik,
- Konzentration auf das Wesentliche,
- Wirksamkeitsprüfung von logischen Strukturen,
- geschützte Umgebung,
- die Experimentierumgebung als didaktisch motiviertes Informatiksystem.

Zwischenresümee 5.1

Durch die Arbeiten von Steinkamp und durch die erfolgreiche Erprobung ist ein exemplarischer Nachweis dafür gegeben, dass ein handlungsorientierter Zugang zu Wirkprinzipien von Informatiksystemen über Informatik-Experimente möglich ist. Steinkamp konzentrierte sich

[74] URL: http://www.schnupperuni.de/ (aufgerufen am 12.11.03)

bei seinen Arbeiten auf einen Zugang über Experimente. Durch den Zusammenhang von Experiment und Exploration (vgl. 5.3.1.1 sowie Paul 1994 im folgenden Abschnitt) lieferte er damit auch einen Beitrag zur Gestaltung des explorativen Lernens von Fachkonzepten der Informatik unter Verwendung von Informatiksystemen (vgl. Brinda / Schubert 2003, 111f). Durch eine Verknüpfung des Aktivitätszyklus für den Experimentierprozess (vgl. Abbildung 11 auf S. 116) mit den Frageniveaus bei der Exploration nach Neber (1988, vgl. 5.3.1.1, S. 114) kann dem Explorationsprozess von Lernenden in der Anfangsphase die erforderliche Orientierung gegeben werden, wobei noch genauer zu beschreiben ist, wie dieser Prozess in einer softwarebasierten Explorationsumgebung zu gestalten ist. Die Fragen nach Neber konkretisieren im Aktivitätszyklus den Punkt 3 „Auswahl der Beobachtungsaspekte“. Durch die Thematisierung solcher Strategien kann die grundlegende Fähigkeit von Schülerinnen und Schülern, mit softwarebasierten Lernangeboten zu lernen, geschult werden. Dazu ist es allerdings wichtig, dass die Lernenden ihre individuellen Lernprozesse durch Interaktionen mit dem Lernangebot selbst kontrollieren (vgl. Kerres 2001, 217ff).

Gestaltung von explorationsfreundlicher Anwendungssoftware

Um Software zur Unterstützung explorativen Lernens zu gestalten, sollte diese zunächst *Gestaltungsprinzipien explorationsfreundlicher Anwendungssoftware* berücksichtigen. Nach Herrmann (1986) beginnt die Exploration bei der Benutzungsoberfläche, wendet sich dann der Systemfunktionalität zu, um schließlich die Bearbeitung der Arbeitsaufgabe mit dem System zu erproben bzw. zusätzliche Nutzungsmöglichkeiten zu erkunden. Bei Software zur Unterstützung explorativen Lernens von Fachkonzepten der Informatik besteht die „Arbeitsaufgabe“ in der Exploration von fachlichen Zusammenhängen (vgl. Abbildung 15).

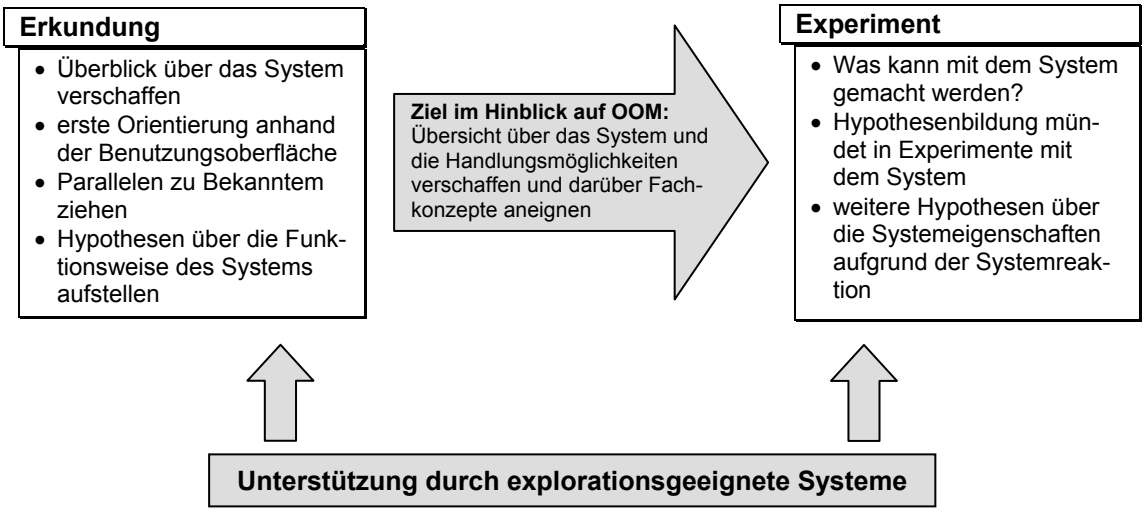


Abbildung 15: Exploration: Erkundung und Experiment

Paul (1994) befasste sich mit der Entwicklung von Unterstützungsformen für explorationsfreundliche Anwendungssoftware. Er versteht darunter Software, die sich durch ihren Gebrauch erschließt (ebd., 55). Den Ausgangspunkt seiner Untersuchung bildete die Frage, wie Software zu gestalten ist, damit neue Anwenderinnen und Anwender deren Funktionsweise besonders leicht erlernen können. Bei der Entwicklung von Gestaltungselementen für explorationsfreundliche Software nimmt er Bezug auf Carroll (1982, 54ff), die folgende Eigenschaften einer Explorationsumgebung beschrieb (vgl. Paul 1994, 149):

- *leichtes Erzeugen von ausführlichen Systemantworten,*
- *Angebot von Orientierungspunkten,*
- *Akzeptieren von Unsicherheit als Handlungsbasis,*
- *Vorhandensein von Sicherungen,*

- *learning-by-doing,*
- *Übertragbarkeit von Erlerntem,*
- *Initiative des Systems bei stagnierender Interaktion,*
- *Eindruck beim Benutzer, den Interaktionsvorgang jederzeit zu kontrollieren.*

Auf der Basis von Erkenntnissen aus der Informatik, insbesondere der Software-Ergonomie, und der Psychologie begründete Paul eine Reihe von Software-Funktionen, die neuen Anwenderinnen und Anwendern die Aneignung der Funktionsweise einer Software erleichtern sollen. Er begründete, dass ein System mit Explorationsmöglichkeiten einem System ohne Explorationsmöglichkeiten aus ergonomischer Sicht überlegen ist. Paul unterscheidet bei der Exploration zwischen den Interaktionsformen Erkundung und Experiment und empfiehlt für deren Unterstützung folgende Systemfunktionalität:

- Interaktionsform *Erkundung* (Paul 1994, 154ff)
  Im Mittelpunkt hierbei stehen das Traversieren, Durchlaufen, Betrachten und Analysieren der Software.
  - *Übersichten* (Liste der gesamten Systemfunktionalität),
  - *Interaktionshistorien und -graphen* (Interaktionsprotokoll),
  - *Filter* (Ausblenden bestimmter Systemfunktionen),
  - *Neutralmodus* (Systemfunktionen durch Beschreibungen ersetzen),
  - *Szenario-Maschine* (vollständige Lösungswege für bestimmte Arbeitsaufgaben zum Nachschlagen).
- Interaktionsform *Experiment* (vgl. Paul 1994, 173ff)
  Im Mittelpunkt stehen das aktive Überprüfen von Hypothesen über das Systemverhalten und die Wirkung einzelner Systemfunktionen.
  - *Stornierung* („Undo“),
  - *Wiederaufsetzpunkte* (Speicherung von Zwischenzuständen),
  - *Spieldaten* (Beispieldateien),
  - *Explorationskarten* (Karten mit folgenden Angaben: Arbeitsaufgabe, erforderliche Arbeitsschritte, (Miss-)Erfolgskontrolle, Abhilfe bei Problemen),
  - *anpassbare Systeme* (Zugang zur Systemfunktionalität, Verfügbarkeit von Funktionen).

Zwischenresümee 5.2

Die von Paul beschriebenen Funktionen richteten sich an Anwenderinnen und Anwender, die sich *selbstständig* in eine für sie neue Anwendungssoftware einarbeiten sollen oder möchten. Dieses Szenario weist Gemeinsamkeiten mit dem Anliegen auf, Lernende dazu zu stimulieren, selbstständig fachliche Zusammenhänge zu explorieren. Ein wesentlicher Unterschied besteht aber in der Verfügbarkeit des Lehrenden sowie der Mitschülerinnen und Mitschüler im Unterricht. Bestimmte der o.g. Unterstützungsformen und Systemeigenschaften, wie z.B. „Übersichten“, „Explorationskarten“ (Paul 1994) sowie „Initiative des Systems bei stagnierender Interaktion“ (Carroll 1982), sind deshalb entbehrlich, weil hierfür die Lerngruppe zur Verfügung steht. Keinesfalls ist es das Ziel, traditionellen Unterricht zu ersetzen oder Lernende mit computerbasierten Lernangeboten „allein“ zu lassen. Gestaltet werden soll hier keine Selbstlern-Software, sondern Lernangebote zur Förderung des explorativen Lernens im Sinne des „Blended Learning“, die durch ihren didaktischen Mehrwert zu einer Bereicherung traditionellen Unterrichts führen (Kerres 2001, 278ff). Andere der o.g. Funktionen sind aufgrund der verschiedenen Zielsetzungen von Lernangeboten und Anwendungssoftware entbehrlich. Der „Neutralmodus“ kann in explorationsfreundlicher Anwendungssoftware aktiviert werden, damit durch das Agieren noch keine zustandsändernden Aktionen in Bezug auf den Arbeits-

gegenstand ausgelöst werden. Durch das Interagieren mit einem Lernangebot zur Förderung des explorativen Lernens sollen aber gerade Aktionen ausgelöst werden, damit sich der Systemzustand ändert und Lernende so Aktionen und Zustandsänderungen verknüpfen, diese in der Lerngruppe mit der Theorie verbinden, strukturieren und so schließlich Wissen aufbauen.
Die übrigen, von Paul (1994) benannten, Funktionen sowie ausgewählte, der von Carroll (1982) beschriebenen, Systemeigenschaften können auch für Gestaltung von Software-Angeboten zur Förderung des explorativen Erlernens fachlicher Zusammenhänge herangezogen werden. Das „leichte[...] Erzeugen von ausführlichen Systemantworten" (Carroll 1982) kann durch die Bereitstellung von vielfältigen Sichten auf den Lerngegenstand ermöglicht werden. Das „Vorhandensein von Sicherungen" (Carroll 1982), „Stornierung" und „Wiederaufsetzpunkte" (Paul 1994) kann durch die Möglichkeit der Speicherung bzw. des Ladens von Systemzuständen sowie durch eine mehrstufige Undo-Funktion implementiert werden. Diese Funktionen sind für die Exploration zentral, um Auswirkungen von Aktionen mittels Experimenten systematisch untersuchen und ungewünschte Ergebnisse zurücknehmen zu können. Gespeicherte Daten zum Interaktionsprozess („Interaktionshistorien und -graphen", Paul 1994) eignen sich prinzipiell zur Realisierung von Protokollfunktionalität, die vom Lehrenden in Verknüpfung mit anderen Daten dazu genutzt werden kann, aus den Interaktionsmustern der Lernenden Rückschlüsse über deren aktuelle Lernprozesse zu gewinnen (vgl. Freudenreich / Schulte 2002)[75]. Für Lernende kann es beispielsweise von Interesse sein, die Interaktionen mit dem System zwischen einem definierten Start- und Zielpunkt aufzunehmen und hinterher, unter Umständen auch mehrfach, die Konsequenzen ihres Handelns in Bezug auf den Lerngegenstand in verschiedenen Sichten schrittweise und zwischen den Sichten synchronisiert zu analysieren. „Filter" und „anpassbare Systeme" (Paul 1994) können z.B. über Konfigurationsdateien oder entsprechende Dialoge realisiert werden, die entweder sowohl den Lernenden als auch dem Lehrenden oder aber nur dem Lehrenden zugänglich sind, um das Lernangebot zu individualisieren. Eine „Szenario-Maschine" ist aufgrund der Verfügbarkeit des Lehrenden und der Lerngruppe nicht erforderlich. Eine prinzipielle Einführung in die Lernmöglichkeiten des Angebots im Sinne einer „geleiteten Tour" kann aber für die erste Anwendung hilfreich sein. „Spieldaten" können auf verschiedene Weisen realisiert werden: entweder durch eine Folge „kleiner" Lernangebote für jeweils eine fest damit verbundene Menge an zu explorierenden Fachkonzepten (vgl. 5.3.4.2 und 5.3.4.3) oder durch Basisangebote, in denen Lerngegenstände über eine definierte Schnittstelle ausgetauscht werden können (vgl. 5.3.4.4).

Explorationsfreundliche Gestaltung von interaktiven Lernangeboten

Kerres wandte Kriterien des Erziehungswissenschaftlers Peter Petersen zur Gestaltung von Arbeitsmitteln, die ein selbstständiges Erarbeiten von Lerninhalten ermöglichen, auf interaktive Medien an und gelangte zu folgenden Anforderungen (Kerres 2001, 224, Hervorhebungen im Original):

> „Das Multimedium sollte nicht nur äußerlich attraktiv gestaltet sein, wesentlich ist vielmehr, dass es zu bestimmten Lern-*Aktivitäten* motiviert. Es muss darüber hinaus sofort ersichtlich sein, was das Thema ist, aber auch wie das System zu nutzen ist. Das Multimedium sollte Wiederholungen anbieten, d.h. es muss ein Pool von verschiedenen Darstellungen, Aufgaben etc. vorliegen […] Interessant ist auch die Forderung nach der *Weiterführung*: Das Medium sollte Verweise zu anderen Lernangeboten beinhalten und so zu Vertiefungen etc. motivieren. Die *wertvolle Arbeitshaltung*, heute etwa als *metakognitive Fertigkeit* diskutiert, meint insbesondere, wie das Lernen mit dem Medium über die Lerninhalte hinaus die grundlegende Fähigkeit zu lernen schult."

In Teilen zeigt sich hier eine Überschneidung mit den Gestaltungsprinzipien explorationsfreundlicher Anwendungssoftware. Der Gestaltung der Benutzungsschnittstelle von explorati-

[75] Die Autoren verknüpften Video- und Audio-Aufzeichnungen der Lernenden mit elektronischen Protokollen ihrer Interaktionen mit der verwendeten Software.

ven Lernangeboten ist nicht nur unter funktionalen, sondern auch unter ästhetischen Gesichtspunkten besondere Bedeutung beizumessen (äußerliche Attraktivität), um Lernende zu Lernaktivitäten zu motivieren. Dies kann einerseits durch die Wahl der Gestaltungswerkzeuge (z.B. Macromedia Flash, Java), andererseits durch die Wahl ansprechender Beispiele in diesen Angeboten beeinflusst werden. Die Forderung nach Verweisen zu anderen Lernangeboten wird im Rahmen dieser Arbeit nicht *innerhalb* einzelner Lernangebote erfüllt, sondern durch die Einbettung von Aufgabenklassen (vgl. Kapitel 4) und explorativen Lernangeboten in Wissensstrukturen (vgl. Kapitel 6). Diese Einbettung gibt Lernenden und Lehrenden Orientierung in Bezug auf den Lehr-Lern-Prozess.
Als Gestaltungselemente explorativer Lernangebote kommen Hypertexte, Animationen, Videos, Simulationen, Mikrowelten, Edutainment etc. in „Reinform" oder als Verknüpfung zum Einsatz (Kerres 2001, Magenheim 2001). Wie im Abschnitt 5.3.1.1 dargestellt, ist für die Anregung explorativen Lernens ein besonders günstiges Verhältnis zwischen Freiraum und Führung erforderlich. Lineare Medien, wie computergenerierte Filmsequenzen (Animationen) oder Videos sind daher allein nicht sonderlich explorationsfreundlich (vgl. Kerres 2001, 228f, 230). Erweiterte Eingriffsmöglichkeiten ergeben sich bei Simulationen, bei denen Lernende die Möglichkeit haben, das jeweils zugrunde liegende, formale Simulationsmodell zu parametrisieren und die Auswirkungen dieser Aktionen zu verfolgen.
Keil-Slawik (2001) hält in diesem Zusammenhang die Verknüpfung von Konstruktion und Simulation für besonders geeignet:

> „Explorationen sind unabhängig von verschiedenen didaktischen Konzeptionen und Herangehensweisen einsetzbar und ermöglichen den Studierenden, interaktive Experimente durchzuführen, die sonst nur in gut ausgestatteten Labors oder mit einem hohen manuellen Rechenaufwand möglich sind. Neuartig am Konzept der Explorationen ist die Möglichkeit (im Gegensatz zur Simulation), vorgegebene Experimente oder technische Konstruktionsaufgaben nicht nur mit unterschiedlichen Parametereinstellungen zu versehen, sondern Aufbau und Struktur zu modifizieren oder ganz selbsttätig durchzuführen. Entscheidend ist dabei die Kopplung von Konstruktion, Simulation und mathematischer Beschreibung, wobei zwischen diesen drei Ebenen flexibel hin und her gewechselt werden kann."

Die von ihm geleitete Fachgruppe für „Informatik und Gesellschaft" entwickelte im Rahmen eines vom Universitätsverbund Multimedia NRW[76] geförderten Verbundprojektes der Universitäten in Paderborn und Siegen (Laufzeit: 01.04.1999 – 31.03.2001) in Kooperation mit Fachgruppen aus der Maschinentechnik Software zur Exploration in der Mechanikausbildung (als „Explorationen" bezeichnet). In der Mechanikausbildung gelingt die Verknüpfung von Konstruktion und Simulation deshalb so einfach, weil den Ausbildungselementen jeweils wohldefinierte, numerisch berechenbare, mathematische Modelle zugrunde liegen. Konstruktionsprozesse sind auch beim objektorientierten Modellieren zentral, da es sich dabei um eine Methode zur Konstruktion von Software handelt. Ein numerisch berechenbares Modell liegt der Objektorientierung aber nicht zugrunde. Liegen zu einem OOM aber entsprechende Quelltexte als Implementierung vor, können mittels geeigneter Werkzeuge Laufzeitobjekte instanziiert und Objektkommunikation simuliert werden.

5.3.2 Fachdidaktische Analyse von Werkzeugen zum objektorientierten Modellieren

Konzeption

Mangels geeigneter Lernhilfen kommen im Informatikunterricht zum objektorientierten Modellieren bereits frühzeitig professionelle, integrierte Entwicklungsumgebungen zum Modellieren und bzw. oder zum Codieren zum Einsatz (vgl. 2.3.1.2). Produktentwicklung und die

[76] http://www.uvm.nrw.de/ (aufgerufen am 12.11.03)

Ausbildung von Software-Entwicklerinnen und -entwicklern sind keine Ziele informatischer Allgemeinbildung. Im Falle einer Berufsbildung müsste diese Aussage relativiert werden. Um Handlungsorientierung zu ermöglichen, passt sich der Informatikunterricht oft an diese Werkzeuge an (vgl. 2.3.1.2). Im Hinblick auf einen Unterrichtseinsatz ist es ein zentrales Problem dieser Werkzeuge, dass sie mit dem Ziel konzipiert wurden, die Entwicklung von möglichst fehlerfreien Software-Produkten in der Software-Industrie zu rationalisieren und deshalb mehr bzw. andere Funktionen bereit stellen müssen als sie für Lernende zur Förderung ihres informatischen Konzepterwerbs erforderlich wären (vgl. Abbildungen 16 und 17).

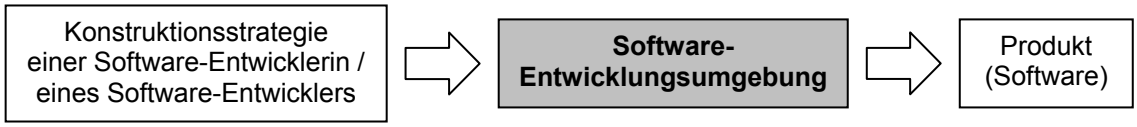


Abbildung 16: Professionelle Entwicklung von Software mit Werkzeugen

Abbildung 17: Erlernen von Fachkonzepten mit Explorationsmodulen

Damit sind diese Werkzeuge allerdings für den Einsatz im Informatikunterricht überladen und der unterrichtliche Weg bis zu ihrer Gewinn bringenden Anwendung erfordert eine intensive Einarbeitung und Orientierung von Lehrenden und Lernenden, Zeit, die für wichtige Unterrichtsinhalte verloren geht. Engagierten Informatiklehrenden gelingt es, ausgewählte Funktionen solcher Werkzeuge zur Bereicherung der Lehr-Lern-Prozesse einzusetzen, ohne die Lernenden mit der parallelen Aneignung der Handhabung und der Aneignung von Fachkonzepten zu überfordern (vgl. 2.3.1.2). Zur Identifikation solcher Funktionen und Eigenschaften wurden ausgewählte Modellierungswerkzeuge genauer untersucht (vgl. Tabelle 24).

Werkzeug	Version	Größe	Art der Lizenz
ArgoUML	0.7	1.2 MB	GNU-Lizenz
Together	3	27.5 MB	Schullizenz
Rational Rose	4.0.3	10 MB	Student-/Demo-Version
Objekttechnologie Werkbank (OTW)	2.4	62 MB	Private Edition

Tabelle 24: Informatikdidaktisch analysierte Werkzeuge zum OOM

Auswahl der Werkzeuge

Grundlage für die Auswahl war eine Übersicht über Werkzeuge zum objektorientierten Modellieren auf der Internet-Seite http://www.jeckle.de/umltools.htm in der Fassung vom 17.02.2000. Diese Übersichtsseite wird seitdem weitergepflegt und ermöglicht den Zugang zu aktuellen Versionen der o.g. Produkte sowie zu zahlreichen Neuentwicklungen (aufgerufen an 23.10.2003)[77]. Die Auswahl der Werkzeuge für die Analyse erfolgte aufgrund von zwei Kriterien:

1. *Kostenfreiheit*: Im Hinblick auf die finanziellen Möglichkeiten von Schulen wurden nur Werkzeuge ausgewählt, die entweder als Public-Domain-Software verteilt oder für die kostenfreie Lizenzen für Bildungsinstitutionen angeboten werden.
2. *Unterstützung der Quellcodegenerierung (in Java):* Um einen möglichst hohen unterrichtlichen Nutzen zu garantieren, wurden Werkzeuge mit integriertem Quellcodegene-

[77] Eine weitere Übersicht dieser Art befindet sich auf der Internet-Seite: http://www.cetus-links.org/oo_ooa_ood_tools.html (aufgerufen am 23.10.03)

rator und Unterstützung der Programmiersprache Java ausgewählt. Die Festlegung der Sprache erfolgte zur Gewährleistung der Vergleichbarkeit generierten Codes. Java wurde aufgrund der großen Verbreitung im schulischen Umfeld gewählt.

Schwerpunkte der Analyse

Die Analyse konzentrierte sich auf folgende Schwerpunkte:

1. *Unterrichtseignung*: Im Rahmen einer vom Autor dieser Arbeit an der Universität Dortmund im SoSe 2000 betreuten Projektarbeit[78] von zwei Informatik-Lehramtsstudierenden im Hauptstudium (Koch / Ortmann 2001) wurden ausgewählte Modellierungswerkzeuge im Hinblick auf ihre Unterrichtseignung untersucht, um damit den Einsatz im Hinblick auf die Zielgruppe besser zu verstehen sowie Gestaltungsanregungen für Software zur Exploration objektorientierter Fachkonzepte abzuleiten.
2. *Erkenntnisfördernde / erkenntnishemmende Funktionen:* Dieselben Werkzeuge wurden im Hinblick auf erkenntnisfördernde und erkenntnishemmende Funktionen analysiert mit dem Ziel, auch daraus Gestaltungshinweise abzuleiten.

Nachfolgend werden zunächst relevante Ergebnisse dieser Analysen dargestellt. Die gewonnenen Erkenntnisse werden anschließend mit den Ergebnissen zu Lern-Software zum OOM (vgl. 2.3.2) und zum explorativen Lernen mit Informatiksystemen (vgl. 5.3.1.2) aus den Literaturanalysen verknüpft. Daraus wird dann ein Konzept für die Gestaltung von explorativen Lernangeboten für den Bereich des objektorientierten Modellierens abgeleitet und diskutiert, inwieweit dieses Konzept auch für andere Bereiche der Informatik tragfähig ist (vgl. 5.3.3).

Ergebnisse

zu 1.) Unterrichtseignung

Die Analyse der Unterrichtseignung erfolgte insbesondere unter folgenden Gesichtspunkten:

- *Erstellung eines Modells,*
- *Benutzungsfreundlichkeit,*
- *Qualität der automatisch erzeugten Projektdokumentation,*
- *Qualität des generierten Codes.*

Für die Durchführung wurde ein einheitliches Referenzmodell bestehend aus Klassen- und Sequenzdiagramm ausgewählt (vgl. Anhang E.1). Dieses zweiteilige Modell wurde (soweit möglich) mit den Werkzeugen erstellt und der Konstruktionsprozess im Hinblick auf die o.g. Kriterien ausgewertet und verglichen. Klassen- und Sequenzdiagramm wurden als Darstellungsformen ausgewählt aufgrund ihres im Vergleich zu anderen Darstellungsformen (z.B. Anwendungsfalldiagramm, Aktivitätsdiagramm) deutlich höheren Stellenwerts im Informatikunterricht.
Im Weiteren erfolgt eine zusammenfassende Darstellung ausgewählter Aspekte dieser Analyse, soweit sie für das hier verfolgte Ziel der Konzeption von Software zur Exploration objektorientierter Fachkonzepte relevant sind. Für die vollständigen Ergebnisse und insbesondere die eingehende Diskussion der einzelnen Produkte sei an dieser Stelle auf die Projektarbeit verwiesen (vgl. Koch / Ortmann 2001).
Insbesondere folgende Eigenschaften von Software-Entwicklungswerkzeugen trugen zu einer positiven Bewertung der Unterrichtseignung bei und führten zu Gestaltungshinweisen für Software zur Förderung des explorativen Erlernens objektorientierter Fachkonzepte:

[78] An der Universität Dortmund ist die Projektarbeit eine Wahlpflichtkomponente im Rahmen des Informatik-Lehramtsstudiums. Studierende im Hauptstudium wählen aus einem definierten Katalog von Informatikteilgebieten eines als Vertiefungsgebiet aus (hier: „Didaktik der Informatik“), in dem dann neben anderen Studienleistungen auch die Projektarbeit verfasst werden muss. Der Umfang beträgt 6 SWS für ein Semester. Sowohl Einzel- als auch Gruppenprojektarbeiten sind möglich.

- *Bereitstellung vielfältiger Sichten auf ein Modell*
  Modellierungswerkzeuge stellen vielfältige Sichten auf ein Modell zur Verfügung, um den Modellierungsprozess zu rationalisieren. Verwendete Sichten sind:

  - *Modellübersicht*,
  - *Arbeitsfläche / Editor* (mit single / multiple document interface[79]),
  - *Eigenschaften / Inspektor*,
  - *Quelltext*.

  Zur grafischen Darstellung des Modells wird in der Regel zumindest eine Auswahl der UML-Diagrammtypen unterstützt. Mittels der Sichten kann ein Modell unter verschiedenen Aspekten betrachtet werden, z.B.

  - *statisches Modell – dynamisches Modell,*
  - *grafische Modelldarstellung – textuelle Modelldarstellung,*

  etc. Der Anwenderin bzw. dem Anwender werden so verschiedene Zugangsmöglichkeiten zur Modellierung eröffnet.
  Die Anwendung dieses Prinzips stellt auch eine Anforderung an Software zur Förderung des explorativen Lernens von Fachkonzepten dar (vgl. Zwischenresümee 5.2 auf S. 120). Lernende können einen Fachgegenstand so aus verschiedenen Sichten betrachten und dabei unterschiedliche Perspektiven einnehmen. Sie lernen exemplarisch verschiedene Sichten auf einen Fachgegenstand kennen und können leicht und individuell zwischen diesen Sichten hin- und herwechseln. Bei Lernangeboten zum objektorientierten Modellieren für den Hochschulbereich erfolgte eine Konzentration auf das statische Modell (vgl. 2.3.2).

- *Verknüpfung / Synchronisation von Modellsichten*
  Modellierungswerkzeuge unterscheiden sich dadurch, wie konsequent verschiedene Modellsichten miteinander verknüpft bzw. wie gut sie synchronisiert sind. Durch Umsetzung des Model-View-Controller-Paradigmas werden Änderungen in der einen Sicht automatisch oder auf Anforderung in alle anderen Sichten übertragen. So kann beispielsweise bei den Werkzeugen „Together“ und „Rational Rose“ ein Sequenzdiagramm automatisch in ein Kollaborationsdiagramm überführt werden. Ferner werden bei diesen beiden Werkzeugen auch Klassen- und Interaktionsdiagramme synchronisiert, indem im Interaktionsdiagramm automatisch die Methoden angeboten werden, die das Klassendiagramm bereitstellt. Im Interaktionsdiagramm hinzugefügte Methoden werden in das Klassendiagramm übernommen.
  In der Synchronisation von Sichten liegt im Hinblick auf die Gestaltung von Software für exploratives Lernen großes fachdidaktisches Potenzial. Lernenden kann es so besser gelingen, aus vielen Einzelsichten auf einen Fachgegenstand (z.B. ein objektorientiertes Modell) ein Gesamtbild zu gewinnen und eine bekannte Lernbarriere damit zu überwinden. Sie können somit leicht erkennen, welche Konsequenzen ihr Handeln in Bezug auf die statische bzw. die dynamische Seite des Lerngegenstands hat, indem sie zwischen verschiedenen Sichten auf diesen wechseln und die jeweiligen Änderungen analysieren. Dieses Prinzip wird im Folgenden als *Sichtenwechsel* bezeichnet (vgl. 5.3.3, 5.4).

- *Trennung zwischen Modell und Implementierung*
  Ein weiteres Gütekriterium im Hinblick auf die Unterrichtseignung von Modellierungswerkzeugen ist die konsequente Trennung zwischen Modell und Implementierung. Bei einzelnen Werkzeugen war die Spezifikation von Attributen oder Methoden nur bei Kenntnis der Syntax der jeweils vom Werkzeug unterstützten Programmiersprache

[79] vgl. Balzert (1999, 196)

(hier: Java) möglich. Im Hinblick auf die strikte Trennung zwischen Modellierung und Implementierung ist dies fachdidaktisch bedenklich.
Für die Konzeption von Software zur Exploration von Fachkonzepten bedeutet dies, dass in grafischen Modellsichten auf die Verwendung von Programmiersprachensyntax zu verzichten ist. Möglich und wünschenswert ist hingegen eine eigene Programmsicht, in der Änderungen am statischen oder dynamischen Modell direkt auf Quelltextebene nachvollzogen werden können. Der Quelltext sollte dabei den üblichen Anforderungen an Lesbarkeit und Übersichtlichkeit genügen. Durch die direkte Verknüpfung von Modell und Programm (Quelltext) wird Lernenden so ein Zugang zu Konzepten und Notationen von Programmiersprachen ermöglicht.

zu 2.) Erkenntnisfördernde / erkenntnishemmende Eigenschaften und Funktionen

Bei der Analyse der Modellierungswerkzeuge wurden folgende *erkenntnisfördernden Eigenschaften und Funktionen* identifiziert (vgl. Brinda / Schubert 2003):

- *Förderung der syntaktischen Korrektheit eines Modells*
  Modellierungswerkzeuge stellen syntaktisch korrekte Modellelemente als vordefinierte Gestaltungselemente bereit. Eine Anwenderin bzw. ein Anwender wählt aus diesen korrekten Alternativen aus. Die syntaktische Korrektheit des Modells bleibt solange erhalten, wie gültige Verknüpfungen gewählt werden. An viele Modellierungswerkzeuge sind Quelltextgeneratoren mit den zugehörigen Compilern angeschlossen. Eine erfolgreiche Übersetzung erfordert ein syntaktisch korrektes Modell.
  Rückmeldungen des Systems zu syntaktischen Fehlern, sofern hinreichend aussagekräftig, helfen den Lernenden, diesbezügliche Schwächen ihrer Modelle zu identifizieren und zu beheben.

- *Förderung der logischen Korrektheit eines Modells*
  Die Verletzung syntaktischer Regeln ist vergleichsweise einfach mit Werkzeugen festzustellen. Deutlich schwieriger gestaltet sich die Identifikation logischer Fehler. Bestimmte Teilmodelle (z.B. Klassendiagramme) ermöglichen zunächst ein sehr informales Modellieren. Das ist in frühen Phasen des Modellierungsprozesses durchaus erwünscht. Beim Übergang zu formaleren Darstellungen sind abstrakte Konzepte unverzichtbar, wie z.B. Konstruktor, formale Spezifikation vorher informal definierter Attribute und Methoden, Zugriffsoperationen für die Attribute. In vielen Fällen kann ein Modellierungswerkzeug das Fehlen solcher Angaben erkennen und die Anwenderin bzw. den Anwender zur Vervollständigung des Modells auffordern. Zusätzlich können Modellierungswerkzeuge die Konsistenz von Teilmodellen (z.B. Klassen- und Sequenzdiagramm) zueinander sichern.
  Lernende verstehen so, welche Komponenten zu einem vollständigen Modell gehören und wie die Konsistenz verschiedener Teilmodelle gestaltet wird. Tiefer liegende Probleme, die eine weiter gehende Kenntnis der Semantik des modellierten Realitätsausschnittes erfordern, müssen in Diskussionsprozessen innerhalb der Lerngruppe erkannt und gelöst werden. Es wäre unabgebracht, für die Fülle der potentiellen semantischen (logischen) Probleme mit Systemunterstützung zu rechnen.

  Die genannten Mechanismen zur Überprüfung helfen, Probleme im bearbeiteten Modell zu identifizieren. Eine automatische Korrektur ist in der Regel nicht möglich. Lernende werden dadurch auf ihre eigenen Fehler aufmerksam. Es bleibt ein komplizierter Lernprozess zu verstehen, warum es zu einem Fehler gekommen ist. Noch schwieriger ist es, aus dem Fehlerverständnis bessere Strategien für zukünftige Modellierungsaufgaben abzuleiten. Der traditionelle Unterrichtsprozess ermöglicht es, in der Lerngruppe erarbeitete Korrekturempfehlungen umzusetzen.

- *Visualisierung abstrakter Konzepte mit Wechsel des Beobachtungsschwerpunktes*
  Modellierungswerkzeuge bieten die Möglichkeit, die Sicht auf einen Modellausschnitt zu verändern, z.B. durch Veränderung des darzustellenden Detaillierungsgrades bzw. durch Bereitstellung von „Zoom"- und „Unzoom"-Funktion, mit denen zwischen Modellüberblick und der Betrachtung ausgewählter Details gewechselt werden kann. Lernende können die dargestellten Zusammenhänge so auf verschiedenen Abstraktionsniveaus betrachten und sich auf die für eine konkrete Problemlösungsphase wesentlichen Elemente beschränken. Um z.B. die Struktur einer Klassenhierarchie zu analysieren, ist die Darstellung von Detailinformationen einzelner Klassen (Attribute und Operationen) nicht erforderlich, sofern es nicht um die Diskussion von Vererbungen geht.
  Dies ergänzt die Forderung nach Filtern und Anpassungsmöglichkeiten für explorationsfreundliche Anwendungssoftware (vgl. 5.3.1.2).

Ferner wurden folgende *erkenntnishemmende Eigenschaften und Funktionen* identifiziert (Brinda / Schubert 2003):

- *Voraussetzung der fundamentalen Ideen / Zielsetzung von Modellierungswerkzeugen*
  Im Hinblick auf den Lehr-Lern-Prozess besteht ein Hauptproblem darin, dass die fundamentalen Ideen des Arbeitsgegenstandes, hier also des Erstellens objektorientierter Modelle, vorausgesetzt werden. Professionelle Modellierungswerkzeuge richten sich an Anwenderinnen und Anwender mit Vorwissen zur Gestaltung objektorientierter Problemlösungen und unterstützen diese bei der effizienten und möglichst fehlerfreien Erstellung von Software-Produkten, indem sie dazu beitragen, alles Wesentliche bei der Modellierung zu berücksichtigen, bestimmte Routineaufgaben zu automatisieren und die Qualität der Lösung, wo möglich, zu erhöhen.
  Für die Aneignung von Modellierungskonzepten sind diese Systeme nicht konzipiert und nur sehr bedingt geeignet (vgl. 2.3.2: Arbeiten von Kölling / Rosenberg). Software zur Exploration hingegen soll Fachkonzepte nicht voraussetzen, sondern Lernenden Erkenntnisgewinn zu diesen Konzepten durch die Interaktion mit ihr ermöglichen (vgl. Abbildungen 16 und 17 auf S. 123).

- *Funktionsumfang / Komplexität der Benutzungsschnittstelle*
  Aus der im vorherigen Punkt dargestellten Zielsetzung von professionellen Modellierungswerkzeugen resultiert die Notwendigkeit für einen sehr großen Funktionsumfang. In diesem und in der daraus folgenden Komplexität ihrer Benutzungsschnittstellen liegt ein gravierender Nachteil, weil sich Anfängerinnen und Anfänger in der Vielzahl der Einstellungsmöglichkeiten leicht verlieren können. Die Identifikation einer unterrichtsrelevanten Funktionsteilmenge und die Aneignung der Handhabung binden viel Unterrichtszeit und viel Vorbereitungszeit für den Lehrenden.
  Dies korrespondiert mit der Forderung, dass Software zur Förderung explorativen Lernens sich durch ihren Gebrauch erschließen soll (vgl. 5.3.1.2). Der Funktionsumfang und die Interaktionsstrategie müssen dem Vorwissen der Lernenden angemessen sein.

- *Fehlerbegrenzungsstrategien*
  Ein wesentliches Ziel von Software-Entwicklungsumgebungen ist es, möglichst viele Fehler durch entsprechende Gestaltung der Benutzungsschnittstelle direkt zu vermeiden. Das korrespondiert mit der ergonomischen Anforderung der Fehlerrobustheit an Software (vgl. ISO 9241). Die für die Vermeidung bestimmter Fehlerklassen sehr sinnvolle Interaktionsstrategie der „Fehlerbegrenzung durch Handlungsverbot" (z.B. Auswahl des Rückgabetyps einer Methode aus einer Liste von Datentypen) ist als ständig präsente Funktionalität aus fachdidaktischer Sicht kritisch zu bewerten, da Lernende auf diese Weise vom System vor Fehlern bestimmter Klassen bewahrt werden und ein Lernen aus diesen somit verhindert wird. Beim Arbeiten ohne ein Modellierungswerkzeug oder mit

einem, das diese Funktionalität nicht anbietet, werden die Lernenden dann mit solchen Fehlern konfroniert, ohne entsprechende Problemlösungsstrategien erlernt zu haben. Aus Gründen der Gestaltungsflexibilität bieten viele Systeme die Möglichkeit, vorbereitete Auswahlfelder um eigene Freitexteingaben zu ergänzen. Damit wird dieser Punkt zum Teil relativiert.
Im Hinblick auf Software zur Förderung des explorativen Lernens sollten Fehler nicht grundsätzlich verhindert werden, da sie wesentlicher Teil des menschlichen Handelns sind und eine Voraussetzung zum Kompetenzerwerb darstellen (vgl. Kerres 2001). Besser ist es, den Prozess der Fehlerbewältigung angemessen zu unterstützen, z.B. durch die geeignete Darstellung von Problemsituationen in verschiedenen Sichten. Für die Software-Entwicklung bedeutet dies aber eine Erhöhung der Komplexität, wenn die Konsistenz von Modelldarstellungen nicht erzwungen wird (vgl. Przygienda 2002, 67ff).

Zwischenresümee 5.3
Software zur Exploration von Fachkonzepten sollte vielfältige, synchronisierte Sichten auf diesen Gegenstand bereit stellen. Der Inhalt dieser Sichten konstituiert sich durch den jeweiligen Fachgegenstand. Enthält dieser Fachgegenstand statische und dynamische Elemente, so sollten beide in entsprechenden Sichten dargestellt werden. Da einem objektorientierten Modell entweder ein konkreter oder abstrakter Realitätsausschnitt zugrunde liegt, der dann zunächst grafisch modelliert und später in einem Programm implementiert wird, sollten diese drei Betrachtungsebenen durch Sichten unterstützt werden. Ein Lernender kann so durch den Wechsel zwischen diesen Sichten erkennen, welche Gegebenheiten und Aktionen im Realitätsausschnitt welche Repräsentation und Konsequenzen im Modell haben und zu welchen Anweisungen auf Quelltextebene dies schließlich führt.
Funktionen zur Modellüberprüfung sind, wie oben dargestellt, erkenntnisförderlich. Da Software zur Exploration, die eine Darstellung eines Realitätsausschnittes bereitstellt, Wissen über über die Semantik des zugrundeliegenden Modells haben muss, können zusätzliche Rückmeldungen bei semantischen Problemen realisiert werden. Vielfältige Modellsichten stellen andererseits in sich eine indirekte Form der Rückmeldung und Modellüberprüfung dar, da Lernende zu jedem Zeitpunkt die Konsequenzen ihres Handelns auf unterschiedlichen Ebenen verfolgen können. Sie können damit selbst zu der Erkenntnis gelangen, ob oder ob nicht die erzielten Ergebnisse die gewünschten waren. Um unterschiedliche Betrachtungsebenen zu fördern, sollten auch Funktionen zum Vergrößern bzw. Verkleinern betrachteter Sichtenausschnitte (Zoom, Unzoom) angeboten werden. Fehlerbegrenzungsstrategien sind, soweit wie möglich, zu vermeiden. Der Funktionsumfang muss dem Vorwissen der Lerngruppe angemessen sein. Dies erfordert entweder die Möglichkeit, die Software an den Kenntnisstand der Lernenden anzupassen oder die Implementierung separater Angebote für unterschiedliches Vorwissen.

5.3.3 Konzept für Explorationsmodule zum OOM

Auf der Basis der Erkenntnisse zum explorativen Lernen mit Informatiksystemen (vgl. 5.3.1.2), den Ergebnissen der fachdidaktischen Analyse von Modellierungswerkzeugen (vgl. 5.3.2) und den Erkenntnissen zu Lernumgebungen zur Objektorientierung im Hochschulbereich (vgl. 2.3.2) wird nachfolgend ein Konzept für Software zur Exploration von Fachkonzepten aus dem Bereich der Objektorientierung, im Rahmen dieser Arbeit als *Explorationsmodule* oder synonym als *Explorationsbausteine* bezeichnet, präzisiert (vgl. Brinda / Schubert

2001, 2002b, 2003; Brinda 2003, Alex et al. 2002, Przygienda 2002, Hoffmann 2003[80]). Lernende sollen mittels Explorationsmodulen zum OOM sowohl zur Exploration von

- *Fachkonzepten aus dem Bereich der Objektorientierung einzeln und in ihrer Verknüpfung mit anderen Fachkonzepten*

als auch zur Exploration von

- *Elementen des objektorientierten Gestaltungsprozesses bis hin zu Software-Produkten*

angeregt werden. Dabei stehen Interaktionen zwischen den Lernenden und den Explorationsmodulen im Vordergrund. An Lerntexte ist nicht gedacht. Gute Lehrbücher zum OOM stehen im Unterricht zur Verfügung. Mit der Verwendung des Explorations*modul*- bzw. Explorations*baustein*-Begriffs wird im Rahmen dieser Arbeit eine implizite Größenbeschränkung zum Ausdruck gebracht. Angestrebt werden kleine, überschaubare Software-Bausteine zur Anregung des explorativen Lernens für jeweils eine begrenzte Menge von Fachkonzepten (vgl. Zwischenresümees 5.2 auf S. 120 und 5.3 auf S. 128). Da diese Bausteine zur Bereicherung und zur Unterstützung des traditionellen Lehr-Lern-Prozesses im Sinne des „Blended Learning“ zum Einsatz kommen (vgl. Kerres 2001, 278ff), nicht aber als reine Selbstlernmaterialien, ist ein zu großer Funktionsumfang problematisch, da dies längere Einarbeitungszeiten zur Folge hat. Vom Funktionsumfang her übersichtliche Lernangebote hingegen können vom Lehrenden oder den Lernenden flexibel für den Lehr-Lern-Prozess ausgewählt und kombiniert werden.

Technische Rahmenbedingungen

Die Forderung nach einer Größenbeschränkung wird zusätzlich unterstützt durch die aus Kostengründen in der Regel eher unterdurchschnittliche, technische Ausstattung von Schulen. Ferner sind in Schulen alle verbreiteten Betriebssystemwelten (Linux, MacOS, Windows) in verschiedenen Versionen vorzufinden. Explorationsmodule für den Einsatz in Schulen bzw. auf den Heimarbeitsplätzen von Lehrenden und Lernenden sollten plattformunabhängig sein, d.h. entweder in einem Browser ausgeführt werden können (Animationen bei installiertem, kostenfreien Browser-Plugin; Applets) oder in einer betriebssystemabhängigen Laufzeitumgebung (z.B. Java-Anwendung). Da sich außerdem in den internationalen Publikationen bis 2003 ein Bedarf für Software zur Unterstützung des explorativen Lernens im Bereich des objektorientierten Modellierens weltweit zeigt, sollte diese leicht an den jeweiligen Sprachraum angepasst werden können[81].

Sichtenkonzept

In den Abschnitten 5.3.1 und 5.3.2 wurden Argumente für die Berücksichtigung vielfältiger interaktiver Sichten auf den zu explorierenden Fachgegenstand geliefert. Sichten präsentieren den Lernenden die Konsequenzen ihres Handelns im Hinblick auf den Explorationsgegenstand. Im Abschnitt 2.3.2 wurde festgestellt, dass bestehende Lern-Software zur Objektorientierung sich auf Teilaspekte, insbesondere auf das statische Modell konzentriert, Lernenden so aber kein Gesamtbild vermitteln kann. Die Frage ist also, *welche* (und wie viele) Sichten auf den zu explorierenden Fachgegenstand von der Software angeboten werden sollten, wie diese auszugestalten und zu verknüpfen sind. Ferner stellt sich die Frage, ob es hierbei „allgemeine“ Sichten gibt, also solche, die vom zu explorierenden Fachgegenstand unabhängig sind,

[80] Bei den nicht eigenen Referenzen handelt es sich um vom Autor dieser Arbeit betreute, studentische Arbeiten: Alex et al. 2002 (studentische Projektgruppe), Przygienda 2002, Hoffmann 2003 (Diplomarbeiten).

[81] Java bietet hierzu alle erforderlichen Werkzeuge: Sprachcodierung / Ländercodierung (Klasse java.util.Locale), ResourceBundle zur Verwaltung von Dialogtexten in Dateien, Werkzeug Jilkit (Java Internationization and Localization Toolkit, http://java.sun.com/products/jilkit (aufgerufen am 12.11.03)) zur automatischen Extraktion von Dialogtexten aus Java-Quelltexten.

d.h. die auch bei Gestaltung von Explorationssoftware für andere Themenbereiche der Informatik und darüber hinaus Anwendung finden könnten.

Auswahl von Sichten I

Die Aneignung von fachlichen Gegenständen und Zusammenhängen ist eng verbunden mit der Aneignung von Elementen des Gestaltungsprozesses (vgl. Zwischenresümee 5.3 auf S. 128). Elemente des Gestaltungsprozesses werden durch den Fachgegenstand konkretisiert. Aus diesem Grunde sollten sich wesentliche *Etappen des Gestaltungsprozesses* auch in Form von interaktiven Sichten wieder finden. Als wesentliche Etappen werden hier vorgeschlagen:

- *Realitätsausschnitt,*
- *Modell,*
- *Produkt.*

Im Bedarfsfall kann dieser Dreischritt verfeinert werden. Aus den Etappen lassen sich direkt drei korrespondierende Sichten ableiten:

- *Realsicht,*
- *Modellsicht,*
- *Produktsicht.*

Diese Etappen des Gestaltungsprozesses finden sich nicht nur beim objektorientierten Modellieren wieder, sondern in vielen Gestaltungsbereichen der Informatik, so z.B. beim Entwurf von Datenbanken oder Rechnernetzen. Die Produktsicht kann z.B. durch eine Sicht auf ein Programm (Quellcode), auf eine Benutzungsschnittstelle, o.ä. ausgestaltet werden.

Einschub: Anschaulichkeit vs. Gestaltungsfreiheit

Die Einbeziehung einer Realsicht in das Sichtenkonzept hat Konsequenzen für die Flexibilität von Modell- und Produktsicht. Freies Gestalten ist hier nur insoweit möglich, wie es dafür definierte Zustände in der Realsicht gibt. Der damit verbundenen Einschränkung an Gestaltungsflexibilität steht ein deutlicher Gewinn an Anschaulichkeit gegenüber, die hier als vorrangiges Ziel verfolgt wird. Für freies Gestalten existieren Modellierungswerkzeuge, die diesen Prozess unterstützen, vgl. z.B. 5.3.2. Die Berücksichtigung sowohl von Anschaulichkeit als auch Gestaltungsfreiheit hat daher gestalterische Konsequenzen. Anstelle eines großen Explorationssystems wird hier ein Verbund vielfältiger Explorationsmodule propagiert, die dann leicht bzgl. Anschaulichkeit und Gestaltungsfreiheit ausdifferenziert werden können.
Unterstützt wird dies zusätzlich durch die Forderung nach „Spieldaten“, die entweder durch eine Folge „kleiner“ Lernangebote für jeweils eine fest damit verbundene Menge an zu explorierenden Fachkonzepten oder durch Basisangebote mit über eine definierte Schnittstelle austauschbaren Lerngegenständen realisiert werden können (vgl. Zwischenresümee 5.2 auf S. 120).

Auswahl von Sichten II

Zentral im Gestaltungsprozess von Informatiksystemen ist weiterhin die Berücksichtigung von *Statik und Dynamik* (z.B. Balzert 1999). Statik und Dynamik lassen sich im Hinblick auf alle o.g. Etappen des Gestaltungsprozesses identifizieren und diskutieren. In Bezug auf die Sichten bedeutet dies, dass entweder

- *jeweils (mindestens) eine Statik- und Dynamiksicht auf Realitätsausschnitt, Modell bzw. Produkt* bereitzustellen sind oder dass
- *statische und dynamische Aspekte jeweils in (mindestens) einer hybriden Realsicht, Modellsicht bzw. Produktsicht*

darzustellen sind (vgl. Abbildung 18).

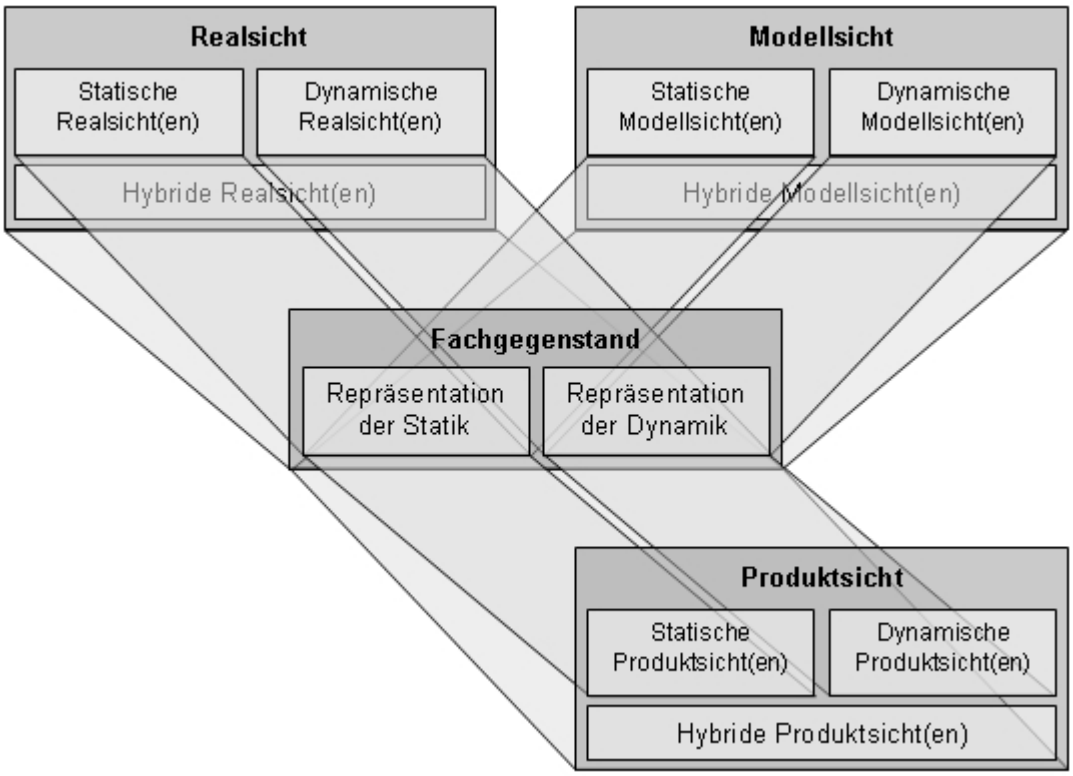


Abbildung 18: Allgemeines Sichtenkonzept für Software zur Exploration von informatischen Fachgegenständen

Eine Mischung der Varianten ist möglich. Inhaltlich werden die einzelnen Sichten durch den jeweiligen *Fachgegenstand* (hier: objektorientiertes Modellieren) konkretisiert.
Lernende können durch *Wechsel zwischen diesen Sichten* („Sichtenwechsel“) erkennen, welche Gegebenheiten und Aktionen im Realitätsausschnitt welche Repräsentation und Konsequenzen im Modell bzw. im Produkt haben. Durch die Analyse des Sachzusammenhangs in anderen Sichten, mit anderen Repräsentationen, kann sich beim Lösen von Problemen, die z.B. mit dem Erreichen eines Lernziels zusammenhängen, eine günstigere Ausgangssituation für deren Lösung ergeben („funktionale Fixierung“, vgl. Anderson 2001, 267). Es werden *vielfältige Zugangsmöglichkeiten* für Lernende unterschiedlichen Vorwissens möglich, da die fachlichen Zusammenhänge auf verschiedenen Abstraktionsniveaus exploriert werden können. Während Anfängerinnen und Anfänger beispielsweise fachliche Gegenstände und Zusammenhänge zunächst in der Realsicht erkunden, in der diese an anschaulichen Beispielen aus der Lebenswelt veranschaulicht werden, können fortgeschrittene Lernende die Beziehungen zwischen Realsicht, Modellsicht und Produktsicht untersuchen.
Um die Exploration des Fachgegenstandes auf verschiedenen Abstraktionsstufen zu unterstützen und um damit eine geeignete Binnendifferenzierung zu ermöglichen, sollte jede Sicht im Bedarfsfall einzeln ein- und auszublenden sowie in Vollbilddarstellung zu betrachten sein. Die Komplexität der Sichten sollte anpassbar sein, indem bestimmte Details aufgedeckt und verborgen werden können („anpassbare Systeme“, „Filter“, vgl. Zwischenresümee 5.2 auf S. 120; „(Un-)Zoom“-Funktion, vgl. Zwischenresümee 5.3 auf S. 128). Ferner ist es nicht erforderlich, dass ein Explorationsmodul alle der o.g. Sichten implementiert, wenn das Ziel die Fokussierung bestimmter Teilaspekte ist, z.B. die Gestaltung eines vereinfachten Klassendiagramms auf der Basis vordefinierter Gestaltungselemente zu einer statischen Realsicht.
Es wird ein handlungsorientierter Zugang zu Fachkonzepten geschaffen, die im Informatikunterricht sonst oft nur auf dem Weg der Programmierung erarbeitet wurden (vgl. 2.3.1.2). Hierzu werden alle Vorgänge von den Lernenden selbst ausgelöst, beobachtet, protokolliert und diskutiert. Durch Auswertung der Beobachtungsergebnisse entscheiden die Lernenden

jeweils selbst, ob ein gewünschtes Lernziel erreicht wurde. Unerwünschte Ergebnisse können mittels „Stornierung“, „Sicherungen“ und „Wiederaufsetzpunkten“ (vgl. Zwischenresümee 5.2) rückgängig gemacht werden. Explorationsmodule greifen nicht im Sinne „intelligenter tutorieller Systeme“ in Lehr-Lern-Prozesse ein. Alle Rückmeldungen finden sich in den vielfältigen Visualisierungen der Handlungskonsequenzen der Lernenden in den Sichten sowie in einigen wenigen, textuellen Meldungen. Lernende entscheiden zu jedem Zeitpunkt selbst, mit welchen Aktivitäten sie fortfahren. Auch nicht in den Sichten repräsentierte fachliche Zusammenhänge können durch geeignete Einbettung der Software in den Unterrichtsprozess über entdeckende Lernaktivitäten angeeignet werden, indem Lernende dazu stimuliert werden, im unterrichtlichen Lehr-Lern-Prozess oder in ihrer Freizeit weitergehende Zusammenhänge zu erkunden und zu recherchieren. Die Anregung Lernender hierzu ist aber neben der Gestaltung der Software von vielen Faktoren abhängig, wie z.B. motivierende Unterrichtsgestaltung durch den Lehrenden, Lerninteresse, Lernkompetenz und Neugiermotiv des jeweiligen Lernenden etc. (vgl. 5.3.1.1).
Im Hinblick auf eine Realisierung ist zu beachten, dass aufgrund von technischen Rahmenbedingungen (Bildschirmgröße und dessen maximale Auflösung) nicht unbegrenzt viele Sichten parallel angezeigt werden können. In einer schulischen Umgebung verschärfen sich diese Rahmenbedingungen weiter, da dort aus Kostengründen eher unterdurchschnittlich leistungsstarke Geräte vorzufinden sind. Das bedeutet, dass hybriden Modellsichten, wo ohne Informationsverlust möglich, der Vorzug zu geben ist, da es die Sichtenzahl reduziert und es Lernenden so ermöglicht wird, mehr Sichten gleichzeitig zu betrachten.

Synchronisation, Transfer und Evaluation von Sichten

Es stellt eine große kognitive Anforderung an Lernende dar, verschiedene, aus dem Gestaltungsprozess, aus Statik und Dynamik und deren Konkretion in einem zu explorierenden Fachgegenstand resultierende, Sichten (z.B. verschiedene UML-Diagramme[82]) auf diesen ohne Systemunterstützung zu einem Gesamtbild zu verknüpfen (vgl. Brinda / Schubert 2003, 115). Systemunterstützte Synchronisation kann hierbei als „didaktische Brücke“ zwischen verschiedenen Sichten dienen und Lernende dabei unterstützen, verschiedene Betrachtungsebenen miteinander zu verknüpfen. Das Zusammenwirken verschiedener Elemente des Explorationsgegenstandes wird auf verschiedenen Betrachtungsebenen veranschaulicht. Aktionen des Lernenden in der einen Sicht werden an alle von ihr abhängigen Sichten propagiert und führen dort zu einer Zustandsänderung (Implementierung möglich mittels des Beobachter-Musters, vgl. Gamma et al. 1996, 257ff). Durch direkte Beobachtbarkeit der Konsequenzen des Handels auf verschiedene Betrachtungsebenen des Explorationsgegenstands gelingt es, die einzelnen Sichten zu einem Gesamtbild zusammenzuführen und damit eine kognitive Barriere zu überwinden.
Die Synchronisation zwischen Sichten kann entweder

- *vollautomatisch* oder
- *halbautomatisch*

erfolgen. Bei der *vollautomatischen Synchronisation* werden Änderungen in einer Sicht automatisch an alle von ihr abhängigen Sichten propagiert (s.o.). Ein Beispiel aus dem Bereich der Objektorientierung ist die Darstellung eines einfachen Realitätsausschnittes, z.B. Verschiebe-Operation auf einem grafischen Objekt (Realsicht), deren Repräsentation in einen animierten Kollaborationsdiagramm (Modellsicht) und schließlich die synchronisierte Darstellung des zugehörigen, dynamisch erzeugten Quelltextes (Produktsicht), vgl. Anhang F.3.1.

[82] Im Rahmen der vorliegenden Arbeit erhält die UML damit eine Multifunktion: im Rahmen der Konzeption von Software zur Exploration dient sie als Entwurfshilfsmittel, in den Modellsichten wird sie zur Visualisierung des zu explorierenden, objektorientierten Modells verwendet. Da es sich um eine Standardnotation zur Objektorientierung handelt erfüllt sie als Lerngegenstand zusätzlich auch eine wissenschaftspropädeutische Funktion.

Nicht immer ist es möglich oder erwünscht, dass das System automatisch abhängige Sichten synchronisiert. Um den Lernenden zu befähigen, selbstständig Darstellungen ineinander zu überführen, müssen Funktionen bereitgestellt werden, mit denen die Konsistenz der Sichten überprüft werden kann (vgl. Zwischenresümee 5.3, S. 128). Am Beispiel der Erzeugung von Objektdiagrammen zu einem gegebenen Klassendiagramm und umgekehrt sei dies illustriert. Von bestimmten Problembereichen[83] abgesehen ist es möglich, zu einem Objektdiagramm ein eindeutiges Klassendiagramm (auch automatisch) zu ermitteln. Umkehrt ist das nicht möglich, da zu jedem Klassendiagramm beliebig viele Objektdiagramme existieren, sofern die Anzahl der möglichen Objekte und deren Attributwerte nicht beschränkt sind. Es lässt sich aber überprüfen, ob ein Objektdiagramm eine gültige Instanz eines Klassendiagramms darstellt oder ob, und wenn ja, wo, die Struktur verletzt wird. Die Konsistenzüberprüfungen erfolgen entweder in der Lerngruppe oder unterstützt durch die Software. Dies wird dann hier als *halbautomatische Synchronisation* bezeichnet.
Um insbesondere dynamische Vorgänge in verschiedenen Sichten schrittweise nachvollziehen zu können, kann ein „Aufnahmewerkzeug“ hilfreich sein, mit dem eine Sequenz von Vorgängen zunächst aufgezeichnet und dann abgespielt werden kann (vgl. Zwischenresümee 5.2, S. 120).

Konkretisierung für den Bereich der objektorientierten Modellierung

Einem objektorientierten Modell liegt entweder ein konkreter oder abstrakter Realitätsausschnitt zugrunde, der zunächst grafisch modelliert und anschließend (in der Regel) in einem Programm implementiert wird. Das im vergangenen Abschnitt vorgestellte Sichtenkonzept ist somit anwendbar. Nachfolgend wird ein Vorschlag für die Konkretisierung der einzelnen Sichten für den Bereich der objektorientierten Modellierung präsentiert.
In der ***Realsicht*** kann für einen vorgegebenen Realitätsausschnitt eine Nachbildung bzw. Veranschaulichung durch Simulation der darin enthaltenen Objekte, ihrer Eigenschaften und der mit ihnen verbundenen Aktivitäten erfolgen. Die Realsicht auf einen Realitätsausschnitt zeigt die Objekte, deren Eigenschaften und Verhalten auf realitätsnah veranschaulichte Weise, also einfache geometrische Objekte, Mobiltelefone, Simkarten und Provider, eine Kamera und ihr Zubehör o.ä. Die zentrale Idee ist, dass der Umgang mit und die Manipulation von Objekten leichter zu verstehen sind als deren Klassifizierung und der Aufbau von Klassenbeziehungen. Lernende sollen zuerst in Kontakt mit Objekten kommen, zunächst ohne etwas über Klassen und deren Beziehungen wissen zu müssen. Das zugrunde liegende Modell ist abgeschlossen und nur insoweit erweiterungsfähig, wie es vom Ersteller der Realsicht vorgesehen wurde. Die Wahl einer geeigneten Modellwelt ist aus didaktischer Sicht eine zentrale Aufgabe, da motivierende Wirkung, Ausdrucksstärke und Handhabbarkeit in Einklang gebracht werden müssen.
Die ***Modellsicht*** besteht beim objektorientierten Modell in der Regel aus mehreren Einzelsichten, die durch statische und animierte (dynamische) UML-Diagramme (Booch et al. 1998) dargestellt werden können. Nicht alle verfügbaren UML-Diagrammtypen sind für ein System zur Exploration wesentlicher Fachkonzepte des objektorientierten Modellierens gleichermaßen geeignet (s. Tabelle 25 und Przygienda 2002, 40ff).
Besonders geeignet für Lehr-Lern-Prozesse in der Anfangsphase sind Objekt- und Klassen-, bzw. Kollaborations- und Sequenzdiagramme. Diese erfüllen die Forderung nach der Berücksichtigung von Statik (Objekt- und Klassendiagramm) und Dynamik (Kollaborations- und Sequenzdiagramm).

---

[83] Schwierigkeiten bereiten z.B. Multiplizitäten. Deren untere und obere Grenze kann automatisch nur aus den Gegebenheiten im Objektdiagramm abgeleitet werden. Vom Modellierer beabsichtigte, offene Intervalle (z.B. 1..*) lassen sich nicht automatisch ermitteln, sondern müssen manuell ergänzt werden. Bei der Abstraktion typgleicher Objekte zu einer Klasse kann anschließend noch in den Klassen nach potentiellen Vererbungsstrukturen gesucht werden. Auch hierbei geht die Eindeutigkeit verloren.

Diagrammtyp	auswahlrelevante Eigenschaften	besonders geeignet
Anwendungsfalldiagramm	• als Diagramm ohne textuelle Spezifikation von Anwendungsfällen wenig aussagekräftig • verwendbar als Ergänzung oder in spezialisierter Software	-
Klassendiagramm	• zentrale Darstellungsform für Übersicht über Systemaufbau	x
Objektdiagramm	• Momentaufnahme eines Objektgeflechts • Nähe zum Realitätsausschnitt	x
Verhaltensdiagramme[84]		
Aktivitätsdiagramm	• besonders geeignet für Visualisierung von nebenläufigen Prozessen bzw. funktionalen Abhängigkeiten in komplexen Systemen • verwendbar als Ergänzung oder in spezialisierter Software	-
Kollaborationsdiagramm	• Visualisierung des Nachrichtenaustausches • strukturelle Ähnlichkeit zum Objektdiagramm	x
Sequenzdiagramm	• Visualisierung des Nachrichtenaustausches in zeitlicher Ordnung	x
Zustandsdiagramm	• Visualisierung des Lebenszyklus von Objekten • viele Objekte haben trivialen Lebenszyklus	-
Implementierungsdiagramme		
Komponentendiagramm	• besonders geeignet für komplexe Systeme	-
Verteilungsdiagramm	• besonders geeignet für komplexe Systeme	-

Tabelle 25: Auswahl von UML-Diagrammtypen für Modellsichten

Das Objektdiagramm erleichtert Lernenden den kognitiven Schritt zwischen einem Realiätsausschnitt und einem Klassendiagramm (vgl. Brinda 2001 und 6.2.3). Das Objektdiagramm kann seinerseits später zu einem Kollaborationsdiagramm erweitert werden und so in sich statische und dynamische Elemente vereinen. Für eine stärkere Trennung der Aspekte spricht die Repräsentation der Dynamik im Sequenzdiagramm. Ein Verzicht auf das Klassendiagramm ist z.B. dann möglich, wenn Objekte durch Interaktion in einer Realsicht erzeugt bzw. manipuliert werden können (s.o. und vgl. Tabelle 26).

Diagrammtyp	Modellsicht	V_1	V_2	V_3	V_4	V_5
Klassendiagramm	Klassensicht	x	x	x		
Objekt-/Kollaborationsdiagramm	Objektsicht	x	x		x	x
Sequenzdiagramm	Sequenzsicht	x		x	x	

Tabelle 26: Kombinationsmöglichkeiten (Varianten V_i) von Klassen-, Objekt-, Kollaborations- und Sequenzdiagramm

Die *Objektsicht* steht in direkter Verbindung mit der Realsicht und stellt eine formale Darstellung der Zusammenhänge in Form von Objektdiagrammen (Objekte, Assoziationen) dar. Die Sicht kann unabhängig von der Realsicht verwendet werden. Lernende können hier Objekte erzeugen, deren Attributen Werte zuweisen, Beziehungen zu anderen Objekten herstellen. Diese Sicht ist zentral, da sie bzgl. der Abstraktion einen Zwischenschritt bei der Konstruktion von Klassendiagrammen darstellt. Zur Visualisierung von Nachrichtenflüssen wird die Objektsicht zur Kollaborationssicht erweitert. Botschaften zwischen Objekten werden darin animiert dargestellt. Dazu kann eine Sequenz von Benutzerinteraktionen mit Objekten mittels eines Aufnahmewerkzeugs aufgezeichnet werden. Anschließend werden der zeitliche Ablauf der Kommunikation zwischen den Objekten simuliert und Veränderungen der Objektzustände visualisiert.
In der *Klassensicht* wird die Klassenstruktur des Modells dargestellt. Da ein besonderer Schwerpunkt auf der anschaulichen Visualisierung in der Realsicht liegt, kann die Klassen-

[84] Diese Unterteilung in Verhaltens- und Implementierungsdiagramme findet sich bei Oestereich (1998, 204).

sicht nicht verändert werden, da es in der Realsicht ansonsten zu nicht definierten Zuständen kommen könnte. Möglich ist aber ein Austausch gegen andere Klassenmodelle (vgl. 5.3.4.4). Werkzeuge, die hier ein freies Modellieren unterstützen, ermöglichen dies zulasten realitätsnaher Veranschaulichungen (z.B. BlueJ, vgl. Kölling / Rosenberg 1996, 2000, 2001). Lernende können in der Klassensicht Klassenmethoden, insbesondere Konstruktoren, aufrufen. Stellt die Realsicht ihrerseits Möglichkeiten zur Erzeugung und Manipulation von Objekten bereit, so kann die Klassensicht ausgeblendet werden (vgl. 5.3.4.3). Dadurch wird ein echter „objects first"-Zugang unterstützt. Durch Auswahl einer Klasse wird in der statischen Produktsicht ihre Implementierung (Quelltext) angezeigt.
In der *Sequenzsicht* werden Objekterzeugungen und Objektkommunikation, die durch Benutzerinteraktionen ausgelöst wurden, direkt als UML-Sequenzdiagramm angezeigt. Durch Auswahl und Aktivierung eines Objektes in der Sequenzsicht werden den Lernenden die verfügbaren Methoden angezeigt, die dann auch von dort aus aufgerufen werden können.
Zu beachten und zu betonen im Lehr-Lern-Prozess ist, dass sich sowohl mit Sequenz- als auch mit Kollaborationsdiagrammen das Verhalten einer Methode nur unvollständig spezifizieren lässt, da z.B. Kontrollstrukturen nur rudimentär unterstützt werden. Beim imperativen Problemlösen wurden hierfür Struktogramme verwendet. Ergänzungen der UML zu einer vollständigen, visuellen Programmiersprache durch so genannte „Story Boards" (vgl. Zündorf 2001) erfordern ihrerseits umfangreiche Kenntnis der nichttrivialen Gestaltungsmittel. Da das Ziel aber das Erlernen objektorientierter Basiskonzepte ist, ist diese Lösung hier ungeeignet. In fortgeschrittenen Lerngruppen lässt sich mit „Story Boards" anspruchsvoller und attraktiver Unterricht gestalten (vgl. 3.8 und Diethelm et al. 2002, 2003).
Die ***Produktsicht*** wird durch zwei Quelltextsichten implementiert. In der statischen Produktsicht wird bei Wahl einer Klasse in der Klassensicht der zugehörige Quelltext angezeigt. In der dynamischen Produktsicht werden bei zustandsändernden Benutzerinteraktionen (z.B. interaktives Erzeugen von Objekten, Aufrufen von Methoden) diese Aktionen durch dynamisch erzeugten Quellcode, der zu diesen Aktionen korrespondiert, nachvollzogen.

5.3.4 Explorationsmodule zum OOM

5.3.4.1 Zur Entwicklungsreihenfolge

Nachfolgend wird die im Rahmen des hier dargestellten Forschungsprojektes entwickelte Software vorgestellt, klassifiziert und jeweils diskutiert, inwieweit die Gestaltungserkenntnisse verallgemeinerbar auch für andere Themenbereiche der Informatik sind. Zu beachten ist, dass die Reihenfolge der Darstellung mit zunehmenden Interaktionsmöglichkeiten für die Lernenden korrespondiert. Diese stimmt allerdings nicht mit der Entwicklungsreihenfolge überein. Bei der Entwicklung bildete den Ausgangspunkt der „Explorer für geometrische Objekte - EGO". Auf der Basis dieser ersten Designstudie konnte das Konzept für die Gestaltung von Explorationsmodulen präzisiert werden (vgl. 5.3.3). Die resultierenden Erkenntnisse stellten die Basis für die Beantragung einer studentischen Projektgruppe an der Universität Dortmund dar, in der schließlich elf Informatikstudierende im Hauptstudium eine „Lernumgebung für objektorientiertes Modellieren im Informatikunterricht – LEO" entwickelten. Parallel zum Start der Projektgruppe entstanden, motiviert durch den „Explorer für geometrische Objekte", die theoretischen Erkenntnisse und die Arbeit am Projektantrag, kleine Software-Bausteine (Animationen mit unterschiedlich stark ausgeprägten Interaktionsmöglichkeiten, realisiert mittels Macromedia Flash 5.0) zum Explorieren ausgewählter Konzepte des OOM, die das hier vorgestellte Konzept ausgezeichnet ergänzen.

5.3.4.2 Animationen mit Interaktionsmöglichkeiten: Puzzles und Experimentierumgebungen

Im Frühjahr 2001 wurde unter Begleitung des Autors in der Fachgruppe „Didaktik der Informatik“ an der Universität Dortmund exemplarisch untersucht (Weigend 2001a, 2001b, Brinda 2002), inwieweit sich objektorientiertes Modellieren im schulischen Kontext vermitteln lässt

- mittels *webbasierten, interaktiven Animationen* als Lernhilfe für die anschauliche Einführung objektorientierter Basiskonzepte und
- mittels *Werkzeugen zur Gestaltung von webbasierten Animationen*, z.B. als zweites Beispiel neben einer „vollwertigen“ Programmiersprache mit der Möglichkeit des Transfers von Fachkonzepten des objektorientierten Modellierens, beispielsweise in den Bereichen „Gestaltung von Benutzungsoberflächen“, „Client-Server-Programmierung“. Dieser Aspekt steht im Rahmen der vorliegenden Arbeit nicht im Vordergrund und wird deshalb hier nicht weiter betrachtet. Weiterführende Informationen hierzu liefert Weigend (2003).

Zur Realisierung wurden auf der Basis von Vorarbeiten des Autors („Explorer für geometrische Objekte“, Brinda / Schubert 2001, IML 2001, Entwicklungsfassungen von Brinda / Schubert 2002b, 2003) webbasierte, interaktive Animationen zu ausgewählten Konzepten des objektorientierten Modellierens als Prototypen mittels der Software Macromedia Flash 5.0 entwickelt und im Unterricht der Sekundarstufe II erprobt (vgl. 5.5.2). Die Erstellung der Animationen diente auch dazu, Möglichkeiten und Grenzen der Gestaltungssoftware als Entwicklungsumgebung zur Erstellung multimedialer Lernangebote insbesondere zur Vermittlung von Fachkonzepten des objektorientierten Modellierens zu erkunden.

Animationstypen

Folgende Typen von Animationen konnten im Rahmen der Erkundung identifiziert und anhand ihrer Interaktionsmöglichkeiten für Lernende (von oben nach unten zunehmend) unterschieden werden:

- *Filme mit Stopppunkten,*
- *Puzzles,*
- *Experimentierumgebungen,*
- *(Grafische) Editoren.*

Nachfolgend werden diese Animationstypen näher charakterisiert, die realisierten Beispiele kurz vorgestellt und das Explorationspotential analysiert.

Filme mit Stopppunkten

Typische Merkmale sind (vgl. Weigend 2001a):

- lineare Folge oder Netzwerk von computergenerierten, multimedialen Filmsequenzen (u.U. verknüpft mit anderen Medienelementen, z.B. Klänge) mit einer definierten Startsequenz,
- Anhalten des Films am Ende jeder einzelnen Filmsequenz (Stopppunkt),
- Navigationsentscheidung des Lernenden an einem Stopppunkt: mittels Navigationsschaltflächen weiteren Ablauf steuern (letzte Filmsequenz wiederholen, nächste Filmsequenz auswählen, etc.),
- in der Regel keine weiteren Interaktionsmöglichkeiten.

Filme mit Stopppunkten entsprechen der Filmmetapher für Animationen und bieten als Interaktionsmöglichkeiten für Lernende lediglich Eingriffsmöglichkeiten in die Ablaufsteuerung (Navigation) an. Zur Visualisierung dynamischer Prozesse gilt diese Form der Animation als

didaktisches Hilfsmittel (Claus / Schwill 2001, 49). Über das mehrfache Anschauen und Beobachten der dargestellten Zusammenhänge hinaus bieten sie Lernenden keine Möglichkeiten zur Manipulation und zur Erkundung daraus resultierender Konsequenzen (Kerres 2001, 230).

Puzzles

Typische Merkmale sind (vgl. Weigend 2001a):

- vorgegebene, endliche Menge an grafischen Gestaltungselementen („Puzzleteile“),
- Gestaltungselemente können mittels „Drag and Drop“ verschoben und zu einem Ganzen zusammengefügt werden,
- genau eine, korrekte Lösung existiert,
- Lösungsversuch kann ausgewertet werden, der Lernende erhält eine positive oder negative Rückmeldung,
- positive Rückmeldung erfolgt, wenn das Puzzle korrekt im Sinne der im System codierten Musterlösung zusammengesetzt ist.

Puzzles ermöglichen Lernenden eine individuelle Rückmeldung zu einer Gestaltungsaufgabe. Da im Puzzle die Gestaltungsaufgabe fest codiert ist, ist eine negative Rückmeldung auch bei semantischen Problemen und nicht nur bei syntaktischen oder ausgewählten logischen Problemen möglich, wie z.B. bei Modellierungswerkzeugen (vgl. 5.3.2). Tabelle 27 gibt eine Charakterisierung der realisierten Beispiele.

Nr.	Name	Beschreibung
1.	***Rechtecke***	***Klassendiagramm mit Aggregationen*** *Abbildung 40 auf S. 239, Abbildung 41 auf S. 240*
2.	***Ventilator***	***Klassendiagramm mit Aggregationen*** *Abbildung 42 auf S. 241, Abbildung 43 auf S. 241*
	zu 1. und 2.	Zu einer aus mehreren Gestaltungselementen zusammengesetzten Animation (Aggregat aus mehreren Teilen) können Lernende interaktiv per „Drag and Drop“ ein vereinfachtes Klassendiagramm erstellen und testen. Klassen werden darin durch Rechtecke mit darin enthaltenem Klassennamen dargestellt, auf Attribute und Methoden wird verzichtet.
3.	***Raumschiff***	***komplexeres Klassendiagramm mit Aggregationen*** *Abbildung 44 auf S. 242, Abbildung 45 auf S. 242*
	zu 3.	Die Grundstrategie entspricht 1. und 2. Als zusätzliche Schwierigkeit kommt hierbei hinzu, dass eine Klasse („Ankopplungsstück“) Teil mehrerer Klassen sein kann.
4.	***Ventilator 2***	***Klassendiagramm mit Assoziationen, Aggregationen, Attributen und Methoden*** *Abbildung 46 auf S. 243, Abbildung 47 auf S. 243*
	zu 4.	Zu einer aus mehreren Gestaltungselementen zusammengesetzten Animation (Aggregat aus mehreren Teilen) können Lernende interaktiv per „Drag and Drop“ ein vereinfachtes Klassendiagramm erstellen und testen. Klassen werden darin durch Rechtecke dargestellt, die oben den Klassennamen tragen, und unten einen freien Bereich besitzen, in den Lernende per „Drag and Drop“ Attribute und Methoden „hinein schieben“ können. Es sollen nur die für die Funktionsweise des Systems unerlässlichen, nach außen hin sichtbaren Attribute, Methoden und Beziehungen dargestellt werden. Anstelle der Testmöglichkeit des Lösungsversuches, startet zu Beginn eine Uhr, die anhält, sobald das korrekte Ergebnis erreicht wurde.

Tabelle 27: Animationstyp „Puzzle“ – Realisierte Beispiele

Das Prinzip ist auf beliebige andere, grafische Modellierungsaufgaben übertragbar. In der vorliegenden Form werden keine weiteren Sichten auf den Gegenstand angeboten. Puzzles eignen sich daher eher zum Experimentieren als zum Explorieren. Das Experimentieren wurde andererseits als Teil des Explorierens beschrieben (vgl. 5.3.1.2). Erweiterungspotential gibt es in folgenden Bereichen:

- Bei genau einer korrekten Lösung könnte die binäre Korrektheitsentscheidung durch eine *mehrstufige Hilfekomponente* ersetzt werden, z.B.:
  - *Anzahl der falsch angeordneten Gestaltungselemente anzeigen*: Da es innerhalb eines Lösungsversuches unter Umständen mehrere korrekte, aber falsch mit einander verbundene Teillösungen gibt, kann die Fehleranzahl z.B. definiert werden als die Anzahl aller Gestaltungselemente minus der Anzahl der in der größten korrekten Teillösung innerhalb des Lösungsversuchs enthaltenen Gestaltungselemente. Ein Lernender erhält somit bei einem fehlerhaften Lösungsversuch eine genauere Angabe über dessen Qualität.
  - *Markierung der falsch angeordneten Gestaltungselemente*: Entweder die größte korrekte Teillösung oder die falsch daran angeschlossenen Gestaltungselemente könnten markiert werden. Als Vorstufe hierzu könnte auch nur ein einzelner Fehler markiert werden. Ein Lernender erhält zusätzlich zur vorherigen Stufe eine Angabe über den Ort eines bzw. der Fehler und kann entsprechend eine Modifizierung vornehmen.
  - *Tipp-Funktion*: Ein noch nicht platziertes Gestaltungselement wird automatisch an der richtigen Stelle in die Struktur eingefügt.
- Insbesondere Modellierungsaufgaben zeichnen sich dadurch aus, dass es nicht nur genau eine korrekte Lösung gibt. Es sollte eine *endliche Anzahl von korrekten Lösungen* unterstützt werden. Auch hier ist ein mehrstufiges Feedback möglich. Die größte korrekte Teillösung innerhalb des Lösungsversuchs muss bzgl. aller korrekten Lösungen bestimmt werden.

Die Umsetzung der hier skizzierten Erweiterungsvorschläge mit dem verwendeten Gestaltungswerkzeug Macromedia Flash ist technisch prinzipiell möglich. Es stellt sich aber die Frage, inwieweit dieses Gestaltungswerkzeug zu diesem Zweck verwendet werden sollte oder ob dafür nicht eine vollwertige Programmiersprache, wie z.B. Java, sinnvoller einzusetzen ist. Bei Implementierung mittels einer Programmiersprache wäre es leicht möglich, die konkrete Gestaltungsaufgabe austauschbar zu gestalten, was insbesondere für die Kosten-Nutzen-Relation bei der Realisierung einer mehrstufigen Hilfekomponente sehr sinnvoll wäre. Es ist hier keinesfalls das Ziel, dass Lernende sich über einen längeren Zeitraum ausschließlich mit E-Learning-Elementen des hier vorgestellten Typs befassen. Auch im traditionellen Lehr-Lern-Prozess gibt es die Notwendigkeit, dass Lernende zuvor angeeignete fachliche Zusammenhänge üben. Für Übungen in diesem Sinne konnten sich die Puzzles bewähren (vgl. 5.5.2).

Experimentierumgebungen

Typische Merkmale sind (vgl. Weigend 2001a):

- relativ eng umgrenzte Palette von Handlungsmöglichkeiten,
- jede Benutzeraktion führt zu einer wahrnehmbaren Reaktion des Systems,
- keine vorgegebene, richtige oder falsche Lösung; Lernende entscheiden selbst, ob und wann sie mit einem Ergebnis zufrieden sind.

Dieser von Weigend (2001a) als „Experimentierumgebung“ bezeichnete Typus kommt dem Gedanken des „Explorationsmoduls“ deutlich näher als die „Puzzles“. Unterscheidungskriterien sind die deutlich geringere Anzahl und die weniger strikte Trennung von Sichten sowie eingeschränktere Interaktionsmöglichkeiten. In Bezug auf ihre Charakteristika stellen sie aber eine Vorstufe zu Explorationsmodulen dar und können daher in einer sehr frühen Unterrichtsphase eingesetzt werden. Tabelle 28 gibt einen Überblick über die realisierten Beispiele.

Nr.	Name	Beschreibung
1	***Kreise***	***Steuerung von Objekten*** *Abbildung 48 auf S. 244, Abbildung 49 auf S. 244*
		Mittels über die Tastatur einzugebender Botschaften (Java-Methodenaufrufe) können Kreisflächen in ihrer Größe, Farbe und Position verändert werden. Lernende interagieren über Schaltflächen und ein Texteingabefenster mit dem System.
2	***Springender Ball***	***Modell und Beobachter*** *Abbildung 50 auf S. 245*
		Veranschaulicht wird das Beobachterprinzip. An ein Objekt (Modell eines springenden Balls), können verschiedene Beobachterobjekte angeschlossen werden. Dargestellt wird der aktuelle Zustand des Modell-Objektes auf verschiedene Weisen. Die angeschlossenen Beobachter werden über jede Veränderung des Modellzustandes über eine entsprechende Botschaft informiert und aktualisieren sich entsprechend.
3	***Kursverwaltung***	***Klassendiagramm mit Feedback*** *Abbildung 51 auf S. 246, Abbildung 52 auf S. 246*
		Ziel ist die Modellierung eines stark vereinfachten Systems zur Verwaltung von Oberstufenkursen. Angeboten werden ein Konstruktions- und ein Prüfmodus, zwischen denen per Knopfdruck gewechselt werden kann. Im Konstruktionsmodus kann (vergleichbar zu den Puzzles, vgl. Tabelle 27 auf S. 137) durch direkte Manipulation von vorbereiteten Gestaltungselementen ein vereinfachtes Klassendiagramm mit Klassen, Assoziationen und Multiplizitäten erstellt werden. Bei Wechsel in den Prüfmodus erscheint nach Auswahl eines Klassensymbols eine verbale Interpretation der Beziehungen der ausgewählten Klasse.

Tabelle 28: Animationstyp „Experimentierumgebung" – Realisierte Beispiele

(Grafische) Editoren

Wesentliche Merkmale sind (vgl. Weigend 2001a):

- durch direkte Manipulation von Objekten per „Drag and Drop" bzw. Betätigen von Schaltflächen sowie textuelle Eingaben über die Tastatur kann ein geistiges Produkt (z.B. Klassendiagramm) konstruiert werden,
- Ziel hierbei ist nicht primär Erkenntnisgewinn, sondern die Herstellung eines Produktes,
- Gestaltungselemente sind nicht, wie bei den Puzzles, mengenbegrenzt, sondern können dynamisch erzeugt bzw. gelöscht werden,
- große Palette an Gestaltungsmöglichkeiten,
- Editionsprodukt kann als Datei gespeichert, später wieder geladen und weiterentwickelt oder mit anderen Werkzeugen weiterverarbeitet werden.

Realisiert wurde ein „Spielzeug-Editor" für Klassendiagramme. Eine in Flash realisierte Fassung eines Modellierungswerkzeugs ist nicht mehr oder weniger explorationsfreundlich als eine entsprechende Anwendung. Zu beachten ist, dass aufgrund technischer Randbedingungen viele Funktionen von Modellierungswerkzeugen mit diesem Gestaltungswerkzeug, wenn überhaupt, nur auf Umwegen realisiert werden können[85]. Leicht möglich ist hingegen die Gestaltung didaktisch reduzierter Werkzeuge, mit denen Lernende an das Gestaltungsprinzip herangeführt werden können. Für realistische Gestaltungsaufgaben sollten aber vom Funktionsumfang her für die Lernenden geeignete Modellierungswerkzeuge verwendet werden.

[85] „Technischer Hinweis: Der letzte Punkt ist mit Flash nur schwer zu realisieren. Flash ist für das Erstellen von Web-Applikationen konzipiert. Das heißt: eine Flash-Animation läuft in der Regel auf einem Server, der über das Internet erreichbar ist. Ein Abspeichern ist folgerichtig (aus Sicherheitsgründen) nicht vorgesehen, da es direkten Schreibzugriff auf den entfernten Server bedeuten würde. Mit Flash kann man ein Editionsprodukt (technisch: den Zustand des Desktops zu einem bestimmten Zeitpunkt) grundsätzlich nur durch kontrollierte Kommunikation mit einem Server persistent machen. Der Server (z.B. ein Python-Script) erledigt das Abspeichern in einer Datenbank. Dabei werden im Prinzip XML-Objekte oder Variablen über die cgi-Post-Methode verschickt." (Weigend 2001a)

Als ein wesentlicher Vorteil in Bezug auf alle hier dargestellten Materialien konnte festgestellt werden:

> „Unsere Erfahrungen im Zusammenhang mit der Entwicklung von Online-Studienmaterialien an der Universität Dortmund ergaben eine Aufwandersparnis von mindestens 90% gegenüber dem Einsatz von Java-Entwicklungsumgebungen." (Weigend 2001b, 57)

Weigend (2003) unterscheidet ferner „Modellierung als Experiment" mittels geeigneter Materialien und „Modellierung als Entwicklung einer Klassenstruktur". Aufgrund fehlender Erfahrung benötigen Anfängerinnen und Anfänger eine sofortige Rückmeldung bezüglich ihrer Modellierungsversuche. Dadurch werden vielfältige Entwicklungszyklen möglich.

5.3.4.3 Explorer für geometrische Objekte – EGO

Der „Explorer für geometrische Objekte" („ObjectExplorer", EGO) entstand in den Jahren 2001 und 2002 in der Fachgruppe „Didaktik der Informatik" an der Universität Dortmund unter Leitung des Autors dieser Arbeit. Ziel war die Realisierung eines Explorationsmoduls zum objektorientierten Modellieren, in dem Lernende einen Fachgegenstand in Real-, Modell- und Produktsicht explorieren können. Da die Synchronisierung der Sichten von vornherein berücksichtigt wurde, diente dies auch als Designstudie für den Sichtenwechsel (vgl. 5.3.3). Eine Serie von Bildschirmfotos findet sich im Anhang F.3.1 ab S. 247. Als Modellwelt wurden einfache geometrische Objekte (Polygone) gewählt aufgrund ihrer einfachen Darstellbarkeit und um Interaktivität in der Realsicht zu ermöglichen. Die Modellwelt ist aufgrund ihrer Nähe zur Mathematik informatikdidaktisch nicht unumstritten, wurde aber aufgrund ihrer Einfachheit und Überschaubarkeit bereits in mehreren Konzeptionen aufgegriffen (z.B. „Von Stiften und Mäusen", vgl. 2.3.1.2, S. 29). Um diesbezüglicher Kritik zu begegnen, wurde bei der Gestaltung von LEO (vgl. 5.3.4.4) auf die Berücksichtigung alternativer Modellwelten großer Wert gelegt.

Kurzbeschreibung des Explorers für geometrische Objekte, der Beobachtungs- und Interaktionsmöglichkeiten

Das EGO-System stellt Lernenden den Explorationsgegenstand in drei verschiedenen Sichten dar.

- Realsicht,
- Objektsicht,
- Quelltextsicht.

Momentaufnahme: Realitätsausschnitt und zugehöriges Objektdiagramm

In der Realsicht erzeugen, löschen oder manipulieren (verschieben, skalieren, drehen, färben) Lernende beliebige Polygone. Sie beobachten die synchronen Veränderungen in der Objektsicht. Erzeugt wird dort automatisch eine Objektdiagrammdarstellung des Zustandes der Realsicht (vgl. Abbildung 53 auf S. 247). Die Erstellung eines weiteren bzw. die Veränderung eines vorhandenen Polygons in der Realsicht bewirkt die Erzeugung eines neuen Polygonobjektes und neuer Punktobjekte bzw. die Modifikation von Attributwerten in der Objektsicht (Abbildung 54 auf S. 247). Umgekehrt können die Lernenden in der Objektsicht Objekte löschen und manipulieren und die synchronen Veränderungen in der Realsicht verfolgen. Dazu werden Methoden der Objekte (aktivierbar über Kontextmenü des Objektes) aufgerufen und erforderliche Parameterwerte übergeben (Abbildung 55 auf S. 248). Durch eine zustandsverändernde Interaktion in der einen Sicht können jeweils synchron die Zustandsveränderungen in der anderen Sicht beobachtet werden. Durch diesen Sichtenwechsel kann ein tieferes Verständnis für den Zusammenhang zwischen Realitätsausschnitt und dessen Objektrepräsenta-

tion gefördert werden. Rückschlüsse von den Objektdiagrammen auf ein zugrunde liegendes Klassendiagramm sind leicht möglich.

Objektkommunikation festhalten und analysieren

Das EGO-System verfügt über eine Aufnahmefunktion für Aktionen in der Realsicht und die daraus resultierende Objektkommunikation. Von einem definierten Startzeitpunkt an (Aktivierung der Aufnahme im Buttonpanel) werden alle Erzeugungs-, Manipulations- und Löschoperationen in der Realsicht aufgezeichnet. Nach Ende der Aufnahme kann in einen Wiedergabemodus gewechselt und die aufgezeichneten Aktionen können schrittweise in ihren Auswirkungen auf alle Sichten nachvollzogen werden. Dazu werden die durch die Aktionen in der Realsicht ausgelösten Nachrichten zwischen Objekten animiert in der Objektsicht dargestellt, die damit Elemente einer Objekt- und Kollaborationssicht vereint. Jeweils synchron werden in der Realsicht die Zustandsveränderungen nachvollzogen. In der Quelltextsicht werden die Erzeugungs- und Manipulationsoperationen als Pseudocode dargestellt (vgl. Abbildung 56 auf S. 249). Damit werden die Schritte der Interaktion auf drei verschiedene Weisen dargestellt und bieten somit einen guten Zugang für den Sichtenwechsel. Lernende verknüpfen so Aktionen, daraus resultierende Objektkommunikation und deren Repräsentation in Pseudocode.

Zu beachten ist, dass es bei der Arbeit mit dem EGO-System kein korrektes oder falsches Ergebnis gibt. Es ist kein definierter Zielzustand fest vorgegeben. Dies korrespondiert mit den Experimentierumgebungen im Abschnitt 5.3.4.2. Für einen gemeinsamen Lehr-Lern-Prozess könnten die Lernenden dennoch einen gewünschten Zielzustand der Realsicht beschließen. Sie stellen diesen durch Interaktion in der Real- bzw. Objektsicht her, zeichnen die Aktionen auf und analysieren hinterher ihre Lösungswege und die Interobjektkommunikation.

Erweiterungs- und Verallgemeinerungspotenzial: Explorer für elektronische Dokumente

Als nachteilig am EGO-System kann der *begrenzte Funktionsumfang in der Realsicht* (Polygone erzeugen, löschen, manipulieren) aufgefasst werden. Selbstverständlich wären hier Erweiterungen möglich im Hinblick auf typische Eigenschaften und Funktionen von Werkzeugen zur Vektorgrafikbearbeitung, z.B. mehr und andere Objekttypen (Kreise, Ellipsen etc.) und bzw. oder weitere Funktionen (kopieren, einfügen, spiegeln, etc.). Die Frage ist allerdings, ob die Erweiterung des Funktionsumfangs zu einer prinzipiellen Erweiterung der Erkenntnismöglichkeiten führen würde. Im Sinne der Erweiterung der Einsatzmöglichkeiten ist eine andere, prinzipielle Überlegung anzustellen, nämlich inwieweit die mit dem EGO-System untrennbar verknüpfte *Modellwelt* (einfache Geometrie der Ebene) im System *austauschbar* gestaltet werden kann. Eine Möglichkeit besteht darin, die hier gewählte Realsicht als stark vereinfachte Fassung eines Werkzeugs zur Bearbeitung von Vektorgrafiken aufzufassen, in dem hier das zugrunde liegende Dokument nicht austauschbar ist. Wenn das elektronische Dokument austauschbar wäre und ebenso das „Werkzeug“, könnte ein Explorer für elektronische Dokumente entstehen, also z.B. für elektronische Textdokumente, Tabellen, Pixelgrafiken, Audiodokumente etc. Damit könnte der didaktische Zugang zur Objektorientierung über elektronische Dokumente (vgl. 2.3.1; Vertiefung zur Textverarbeitung: Voß 2003) und darin zu identifizierender Objekte noch besser unterstützt werden. Für eine Realisierung sind zwei Varianten denkbar:

- *Realisierung einer Schnittstelle zwischen einer realen Standardanwendung und einem ObjectExplorer-Basissystem*
  Als Grundvoraussetzung muss eine Programmierschnittstelle zur Standardanwendung verfügbar sein. Problematisch ist, dass reale Dokumente in der Regel eine kaum zu handhabende Anzahl von Objekten enthalten (z.B. einzelne Zeichen in mehrseitigem

Textdokument). Hier sind geeignete Konventionen und Filter erforderlich, um zu einer überschaubaren Anzahl von Objekten zu gelangen. Vorteil wäre, dass reale Dokumente auf ihre Struktur analysiert werden könnten.

- *Realisierung eigener Mini-Editoren für verschiedene Dokumenttypen mit Schnittstelle zum ObjectExplorer-Basissystem*
  Wesentlicher Vorteil ist, dass die Funktionalität des Editors genau so umfangreich gestaltet werden kann, wie der Explorer Objektstrukturen handhaben kann.

Abschließend werden ausgewählte Architekturüberlegungen zu einem solchen System unter Verwendung von Entwurfsmustern (vgl. Gamma et al. 1996) dargestellt. Die Basis bildet eine interne Repräsentation eines elektronischen Dokumentes mit einem darauf operierenden Editor als *Beobachter* (ebd., 257). Durch einen *Adapter* (ebd., 151) kann einheitlich auf die Objektstruktur des Dokuments zugegriffen werden. Zur Objektrepräsentation existiert eine Objektsicht als *Beobachter*. Erzeugungs-, Lösch- und Manipulationsoperationen in der Objektsicht bzw. im Editor (Realsicht) werden in der jeweils anderen Sicht synchron aktualisiert. Die einzelnen Operationen werden mittels des *Befehl*smusters (ebd., 245) realisiert. Ausgeführte Operationen werden in einer Befehlsliste gespeichert. Mittels der Aufnahmefunktion kann eine Teilmenge der Befehlsliste definiert werden. Zu dieser Teilmenge existieren eine Kollaborations- und Quelltextsicht als *Beobachter.*

5.3.4.4 Lernumgebung für objektorientiertes Modellieren im Informatikunterricht – LEO

Zur Schwierigkeit, eine studentische Projektgruppe zu beginnen

Die Vorarbeiten für die Entwicklung einer Lernumgebung für objektorientiertes Modellieren im Informatikunterricht begannen in der Fachgruppe „Didaktik der Informatik“ an der Universität Dortmund im WiSe 2000/2001. Die Entwicklung sollte erfolgen im Rahmen einer studentischen Projektgruppe mit Beginn des Sommersemesters 2001. Die Grobspezifikation des Projektziels ist dokumentiert (Brinda / Schubert 2001, 32ff). An der Universität Dortmund ist die Projektgruppe eine Wahlpflichtveranstaltung im Hauptstudium, in der jeweils acht bis zwölf Informatikstudierende gemeinsam (mit in der Regel mindestens zwei Universitätsbetreuerinnen bzw. -betreuern) für ein Jahr eine große Programmieraufgabe selbstständig bearbeiten[86]. In der Regel gibt es ein Überangebot an Projektgruppen durch die verschiedenen Lehrstühle des Fachbereiches Informatik, so dass pro Semester nicht immer alle Angebote zustande kommen können. Dieses Schicksal ereilte auch das Projektgruppenangebot des Autors. Die Ursache hierfür lag vermutlich im zu wenig eingängigen Projektgruppentitel „Entwicklung einer Umgebung zum Erlernen von und zum Experimentieren mit Techniken objektorientierter Modellierung im Informatikunterricht“. Deshalb wurde im zweiten Anlauf im WiSe 2001/2002 die Projektgruppe umbenannt in „Lernumgebung für objektorientiertes Modellieren im Informatikunterricht – LEO“. Die Projektgruppenpräsentation vor den Studierenden erfolgte anstatt mit Folien mit Präsentationstechnik. Eine Projektgruppe mit elf Informatikstudierenden kam zustande[87].

Treffen mit Informatiklehrenden

Den Ausgangspunkt bildete ein offenes Treffen mit Informatiklehrenden an der Universität Dortmund am 23.10.2001, um bei den Projektgruppenmitgliedern (Informatik-Studierende,

[86] Projektgruppenseiten des Fachbereiches Informatik der Universität Dortmund, vgl. http://ls4-www.cs.uni-dortmund.de/PGB/info.html (aufgerufen am 12.11.03)

[87] Projektgruppe Nr. 403, vgl. http://ls4-www.cs.uni-dortmund.de/PGB/alles/node13.html (aufgerufen am 12.11.03)

nicht Informatik-Lehramtsstudierende) ein Problembewusstsein für die Vermittlung objektorientierter Fachkonzepte zu schaffen und um andererseits im Gespräch gemeinsam erste Gestaltungshinweise zu erarbeiten. Diskutiert wurden insbesondere die konkurrierenden Entwicklungsziele Anschaulichkeit und Gestaltungsfreiheit (vgl. hierzu Alex et al. 2002, 21ff und 5.3.3 in der vorliegenden Arbeit). Im Ergebnis ergab sich schließlich der Wunsch nach einer Umgebung, in der Lernende in einer abgeschlossenen Modellwelt (als „Sandkasten"-Welt bezeichnet) möglichst viele Gestaltungsmöglichkeiten bei maximaler Anschaulichkeit haben (vgl. 5.3.3). Damit sollte der diesbezüglichen Beschränkung des EGO-Systems (vgl. 5.3.4.3) begegnet werden. Unterstützt werden sollte dies durch eine „didaktische Landkarte", in der Lernenden Vorschläge für Lernwege durch die Menge der zu explorierenden Fachkonzepte angeboten werden (vgl. Alex et al. 2002, 90ff. und 6.5 in der vorliegenden Arbeit). Das LEO-System wurde konzipiert und implementiert im WiSe 2001/2002 und im SoSe 2002. Der Entwicklungsprozess ist dokumentiert im Projektendbericht (Alex et al. 2002). Der entstandene Prototyp wird seit dem Frühjahr 2003 unter Leitung des Autors dieser Arbeit weiterentwickelt.

Kurzbeschreibung des LEO-Systems, der Beobachtungs- und Interaktionsmöglichkeiten

LEO besteht aus einem Basissystem und austauschbaren Szenarios.

Basissystem

Im Basissystem (realisiert in Java) wird die szenariounabhängige Funktionalität bereitgestellt. Die Hauptaufgabe bildet die Bereitstellung und die Organisation der Synchronisation der *Modellsichten*:

- *Klassensicht,*
- *Objektsicht,*
- *Sequenzsicht,*

und der *Produktsichten*:

- *statische Programmsicht,*
- *dynamische Programmsicht.*

Durch Verarbeitung eines geladenen Szenarios werden diese Sichten mit Inhalt gefüllt. Eine *Realsicht* wird ebenfalls bereitgestellt. Aus der Perspektive des Basissystems besteht diese lediglich aus einer Zeichenfläche, auf der das geladene Szenario eine Darstellung erzeugen kann.

Szenarios

Jedes Szenario besteht aus einer Menge von speziell aufbereiteten Klassen (hier: realisiert in Java), einer dazugehörigen Implementierung der Darstellung von Objekten dieser Klassen in der Realsicht und einer XML-Datei, in der Darstellungsoptionen gespeichert sind. Darstellungsrelevante Aktionen in den Szenario-Klassen müssen sich durch Aufruf von bestimmten Methoden des Basissystems an diesem registrieren und wieder abmelden.
Durch ein Szenario wird die in LEO zu explorierende Modellwelt festgelegt. Realisiert wurden im Rahmen der Projektgruppe folgende Szenarien:

- *Kreis,*
- *Bibliothek,*
- *Kamera,*
- *Mobilfunk.*

Das Kreis-Szenario diente als „Proof of Concept", um anhand eines möglichst kleinen Szena-

rios das Zusammenspiel mit dem Basissystem zu überprüfen. Bei den Szenarios lassen sich zwei Typen unterscheiden:

- *Aufgabenbezogene Szenarios,*
- *Szenarios für freies Explorieren.*

Aufgabenbezogene Szenarios sind hier die Szenarios „Kamera“ und „Mobilfunk“. Beim Kameraszenario ist es das Ziel, Kamera- und Zubehörobjekte so zusammenzusetzen, dass es schließlich gelingt, ein richtig belichtetes, nicht verwackeltes Foto zu schießen. Beim Mobilfunkszenario müssen Provider-, Mobiltelefon- und Simkartenobjekte erzeugt und so richtig verbunden werden, dass der Versand und das Empfangen von Kurznachrichten (SMS) möglich sind. Das Erreichen dieses Ziels dient der Bestätigung, Objekte korrekt mit Attributwerten belegt und verknüpft zu haben zur Erhöhung der Motivation (Kerres 2001, 202). Kreis und Bibliothek ermöglichen freies Explorieren. Beim Kreisszenario können Kreise erzeugt, gelöscht, positioniert, skaliert und mit Farbobjekten für Linien bzw. Füllfarbe verknüpft werden. In der Bibliothek können mit Leser- und Buchobjekten Ausleihvorgänge simuliert werden. Bei Überschreiten von Leihfristen werden Mahnungsobjekte erzeugt und dem jeweiligen Leserobjekt zugestellt. Bei diesen beiden Szenarios gibt es keine richtigen oder falschen Zustände. Der Lernende entscheidet selbst, wann ein gewünschtes Ziel erreicht wurde.
Die Schnittstelle für die Entwicklung eigener Szenarien ist dokumentiert (Alex et al. 2002, 78ff). Bereitgestellt werden Quelltextrahmen zur Unterstützung dieses Prozesses.

Zusammenwirken von Basissystem und geladenem Szenario

Jeweils ein Szenario kann im Basissystem geladen und ausgeführt werden. Dazu werden die Java-Quellen des Szenarios vom Basissystem mittels der Java-Reflection-Klassen analysiert und die Klassensicht (UML-nahes Klassendiagramm) erzeugt. In dieser Klassensicht sind alle Klassenmethoden, insbesondere die Konstruktoren der Klassen, aktiv und können von den Lernenden durch Aktivieren mit der Maus aufgerufen werden. Wird ein Konstruktor aufgerufen, so wird in der Objektsicht ein Objekt der zum Konstruktor gehörenden Klasse erzeugt. Handelt es sich um ein sichtbares Objekt, so wird es in der Realsicht entsprechend dargestellt. Durch Auswahl einer Klasse in der Klassensicht wird zudem der ihr zugrunde liegende Quelltext in der statischen Programmsicht dargestellt. Die Objektsicht-Darstellung ist an Objektdiagramme angelehnt, entspricht ihnen aber nicht völlig. Wesentlicher Unterschied ist, dass bei den Objekten, zusätzlich zu Objektnamen und -typ und den Attributwerten, die für dieses Objekt verfügbaren Methoden angezeigt werden. Lernende können so auf einen Blick erkennen, über welche Aktionsmöglichkeiten ein Objekt verfügt. Der Prozess der Objekterzeugung wird in der Sequenzsicht in einer an UML-Sequenzdiagrammen angelehnten Notation und gleichzeitig in einer dynamischen Programmsicht der zugehörige Quellcode dargestellt, z.B. „`kreis1 = new Kreis(100);`“. In der Objektsicht sind die Methoden eines Objektes aktiv, d.h. durch Aktivierung können sie aufgerufen, gegebenenfalls erforderliche Parameter übergeben und dadurch z.B. Attributwerte verändert, Relationen zu anderen Objekten aufgebaut oder gelöscht werden etc. Methodenaufrufe werden in der Real-, Objekt-, Sequenz- und dynamischen Programmsicht analog zum Konstruktoraufruf nachvollzogen. In der dynamischen Programmsicht können über ein Texteingabefester Konstruktoren von Klassen und Methoden von Objekten aufgerufen werden. In der Sequenzsicht können Objekte ausgewählt und aktiviert werden. Objekt-Kontextmenüs ermöglichen dort das Aufrufen der verfügbaren Methoden.
Das dynamische Erzeugen, Löschen und Manipulieren von Objekten erfolgt unter Verwendung der Java-Reflection-Klassen. Wie schon beim EGO-System (vgl. 5.3.4.3) liegt der Schwerpunkt hier bei der Interaktion der Lernenden mit dem System in jeweils einer Sicht und der Beobachtung der Auswirkung ihrer Aktionen auf von dieser Sicht abhängige Sichten.

Lernende betrachten ihre Aktionen daher immer unter statischen und dynamischen Gesichtspunkten und verknüpfen Realitätsausschnitt, Modell und Produkt.

LEO wurde erfolgreich im Informatik-Unterricht und in Lehrerfortbildungen erprobt (vgl. 5.5.2) und wird in einem Lehrbuch zur „Programmentwicklung mit UML“ empfohlen (vgl. Mielke / Sagorny 2003, 132ff).

Erweiterungspotential

Ein Schwachpunkt von LEO in der vorliegenden Version ist, dass die Realsicht noch keine Interaktivität ermöglicht. Dies ließe sich ändern, wenn, vergleichbar zu den Puzzles (vgl. 5.3.4.2), beliebige Realsicht-Objekte[88] per „Drag and Drop“ bewegt und angeordnet werden könnten. Eine bestimmte Anordnung von Realsicht-Objekten müsste mit einem bestimmten Zustand der Objektsicht verknüpft werden. Im Kreisszenario könnte z.B. in der Realsicht ein durch einen Farbeimer visualisiertes Farbobjekt mit der Maus in das Innere eines Kreisobjektes gezogen und damit dort dessen Füllfarbe festgelegt werden. In der Objektsicht würde dadurch eine Beziehung zwischen den beiden Objekten aufgebaut. Mit dieser Erweiterung ließen sich bereits bestimmte Aktionen realisieren, z.B.

- Konstruieren eines Ganzen aus Teilen (Kameraszenario, Bibliotheksszenario),
- Änderung von Zuordnungen (z.B. Verschieben eines Buchobjekts vom Bibliotheksobjekt zum Leserobjekt im Bibliotheksszenario), etc.

5.3.4.5 Vergleich der Entwicklungsergebnisse

In den vergangenen Abschnitten wurden Ergebnisse zur Entwicklung von Software zur Exploration von objektorientierten Fachkonzepten präsentiert. Wie bereits dargestellt, erfolgte die Entwicklung von Konzept (vgl. 5.3.3) und Umsetzung nicht strikt sequentiell, sondern beeinflussten einander wechselseitig. Abschließend soll betrachtet werden, inwieweit die Entwicklungsergebnisse die im Konzept beschriebenen Anforderungen erfüllen (vgl. Tabelle 29).

	Animationen							**Anwendungen**	
	Puzzles				**Experimentierumgebungen**				
	Rechtecke	Ventilator	Raumschiff	Ventilator 2	Kreise	Springender Ball	Kursverwaltung	EGO	LEO
Realsicht	I	I	I	I	P	P	-	I	P/I
Objektsicht	-	-	-	-	P	I	-	I	I
Klassensicht	I	I	I	I	-	-	I	-	I
Kollaborations-/Sequenzsicht	-	-	-	-	-	-	-	P	I
Programmsicht	-	-	-	-	I	-	-	P/I	I
Sichten ausblenden	-	-	-	-	-	-	-	X	X
Detaillierungsgrad von Sichten ändern	-	-	-	-	-	-	-	-	X
Synchronisation von Sichten	-	-	-	-	X	-	-	X	X
Automatische Sichtenerzeugung	-	-	-	-	X	-	-	X	X

[88] Gemeint sind kleine, lebensweltnahe Grafiken (z.B. Mobiltelefon, Kamerastativ) zur Visualisierung. Damit soll eine Verwechselung mit der Benennung von Objekten in der Objektsicht vermieden werden.

	Animationen							Anwendungen	
	Puzzles				Experimentierumgebungen				
	Rechtecke	Ventilator	Raumschiff	Ventilator 2	Kreise	Springender Ball	Kursverwaltung	EGO	LEO
Aufnehmen und Abspielen	-	-	-	-	-	-	-	X	-
Modellüberprüfung	X	X	X	X	-	-	X	-	X
Plattformunabhängigkeit	X	X	X	X	X	X	X	X	X
Internationalisierung	-	-	-	-	-	-	-	X	X
Stornierung / Undo – Redo	-	-	-	-	-	-	-	-	X
Speicherung	-	-	-	-	-	-	-	-	X
Geleitete Tour	-	-	-	-	-	-	-	-	X
Beispielwelt austauschbar	-	-	-	-	-	-	-	-	X

Tabelle 29: Software zur Exploration und die Erfüllung der Anforderungen des Konzepts

Bei den Sichten bedeutet der Eintrag „P“, dass es sich um eine reine Präsentationssicht handelt. Sichten mit Interaktionsmöglichkeit sind mit einem „I“ markiert. Der Eintrag „P/I“ bedeutet, dass eine gegenwärtig reine Präsentationssicht mit Interaktionsmöglichkeiten angereichert werden kann. Ein „X“ zeigt an, dass die Anforderung bei der jeweiligen Software erfüllt ist.

Die „Anwendungen“ in Tabelle 29 sind prototypische Implementierungen des Explorationsmodulkonzepts. Ersichtlich wird, dass das LEO-System die Anforderungen am besten erfüllt. Lediglich das Aufnehmen von Aktionen und das Abspielen der Konsequenzen in verschiedenen Modellsichten sind gegenwärtig noch nicht realisiert. Durch die Verfügbarkeit einer mehrstufigen Undo- bzw. Redo-Funktion wird diese Funktion in LEO dennoch indirekt bereitgestellt. Bei den Animationen zeigt sich, dass die Anforderungen nur teilweise erfüllt sind. Damit verbunden sind beschränkte Aktionsmöglichkeiten. Darin liegen allerdings auch Vorteile, weil diese Systeme so sehr gezielt für Teillernaufgaben hinzugezogen werden und insbesondere die Anfangsphase gut begleiten können. Zur Unterstützung von Prozessen des selbstgesteuerten Lernens und zur Verbesserung einer geeigneten Binnendifferenzierung sind derartige Materialien ausgezeichnet geeignet. Zum Schuljahresbeginn 2003/2004 startete in Nordrhein-Westfalen die Initiative „Abitur Online NRW“ an ca. 170 Schulen der Sekundarstufe II, die ähnliche Ziele in verschiedenen Schulfächern verfolgt[89]. Die „Kreise“ (vgl. Tabelle 29) kommen von ihrer Interaktionsstrategie her den Anwendungen sehr nahe. Sie stellen damit eine ausgezeichnete Vorbereitung hierfür dar. Beim „springenden Ball“ wird ein anderes Konzept visualisiert, nämlich die Verknüpfung eines Modells mit verschiedenen Sichten (Beobachterkonzept). Im Hinblick auf eine spätere Analyse der Realisierung der Anwendungen (vgl. 5.4.1), kann dies einen Ausgangspunkt für die Erarbeitung des Beobachterkonzepts darstellen, das auch in den Anwendungen seine Realisierung findet. Bei der „Kursverwaltung“ wird ein Klassendiagramm mit Multiplizitäten in eine textuelle Beschreibung übersetzt, eine textuelle Interpretation der Semantik. Wie sich in Erprobungsstudien im Informatikunterricht der Sek. II (vgl. 4.5) und in Lehrerfortbildungsworkshops (vgl. 4.6.2) zum Aufgabenklassenkonzept zeigte, stellt die korrekte Spezifikation von Relationen eine Lernschwierigkeit dar. Lernende können hier verschiedene Spezifikationen erproben und eine Interpretation nachlesen. Hierin liegt Potential für eine weitere Sicht im Sichtenkonzept, nämlich eine tex-

[89] URL: http://www.abitur-online.nrw.de/ (aufgerufen am 12.11.03)

tuelle Interpretation eines Modells. Ein Ausschnitt aus einem Objekt- oder Klassendiagramm könnte beispielsweise automatisch beschrieben werden unter Verwendung der Bezeichnungen „ist“ (Vererbung), „hat“ (Aggregation) und „kennt“ (Assoziation), z.B. „Ein Objekt vom Typ X kennt drei Objekte vom Typ Y“. Wie schon das allgemeine Sichtenkonzept wäre auch eine solche Sicht übertragbar auf andere Bereiche, z.B. Gestaltung mit „endlichen Automaten“ oder „Entity-Relationship-Diagramme“.

5.3.5 Architekturkonzept für Software zur Exploration im Bildungskontext

Auf der Basis des Konzeptes „Informatik-Experimente im Schullabor“ (Steinkamp 1999), Software zur Exploration in der Mechanik-Ausbildung (Keil-Slawik 2001)[90], den hier vorgestellten Explorationsmodulen zum OOM (vgl. 5.3.4) sowie weiteren Analysen zu entsprechender Software aus der Bio-Chemie entwickelte Hoffmann (2003) im Rahmen seiner, vom Autor der vorliegenden Arbeit betreuten, Diplomarbeit eine „Theoretisch begründete Vorgehensweise bei der Entwicklung von Software zur Exploration im Bildungskontext“. Den Kern der Arbeit bildete die Entwicklung und Begründung eines Architekturkonzepts. Bewusst erfolgte bei diesem Architekturkonzept keine Einschränkung auf das Gebiet OOM. Ziel war es, das Verallgemeinerungspotential des Ansatzes zu überprüfen.
Hierzu wurden zunächst durch Analyse der oben genannten Bildungssoftware mit Explorationsanspruch funktionale, technische und softwareergonomische Anforderungen an Explorationssoftware herausgearbeitet, die in ein objektorientiertes Analysemodell mündeten. Folgende, funktionale Anforderungen an Explorationssoftware wurden begründet:

- Eigenes und freies Konstruieren,
- Darstellung von verschiedenen Sichtweisen,
- Zulassen von Fehlern und Möglichkeiten zur Aufhebung,
- Simulations- und / oder Experimentiermöglichkeiten,
- Möglichkeit, geleistete Aktionen nachzuvollziehen,
- Hilfestellungen.

Die technischen Anforderungen zielten auf einen möglichst hohen Wiederverwendungsgrad und auf Plattformunabhängigkeit hin. Auf dieser Basis wurde ein objektorientiertes Analysemodell entwickelt (vgl. Abbildung 19).

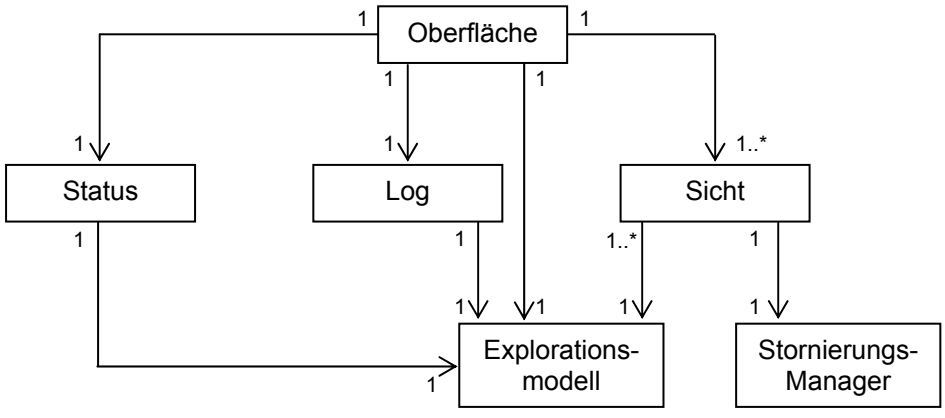


Abbildung 19: Architekturkonzept – OOA-Modell (nach Hoffmann 2003, 53)

Dieses Analysemodell wurde zu einem objektorientierten Entwurfsmodell verfeinert. Bereitgestellt wird darin eine Klassenstruktur, von der konkrete Explorationsanwendungen abgeleitet werden können.

[90] vgl. 5.3.1.2

Die Verfeinerung des Modells erfolgte verschränkt mit der Entwicklung von zwei Fallstudien aus der Digitaltechnik und dem sprachwissenschaftlichen Bereich (vgl. Abbildung 20).

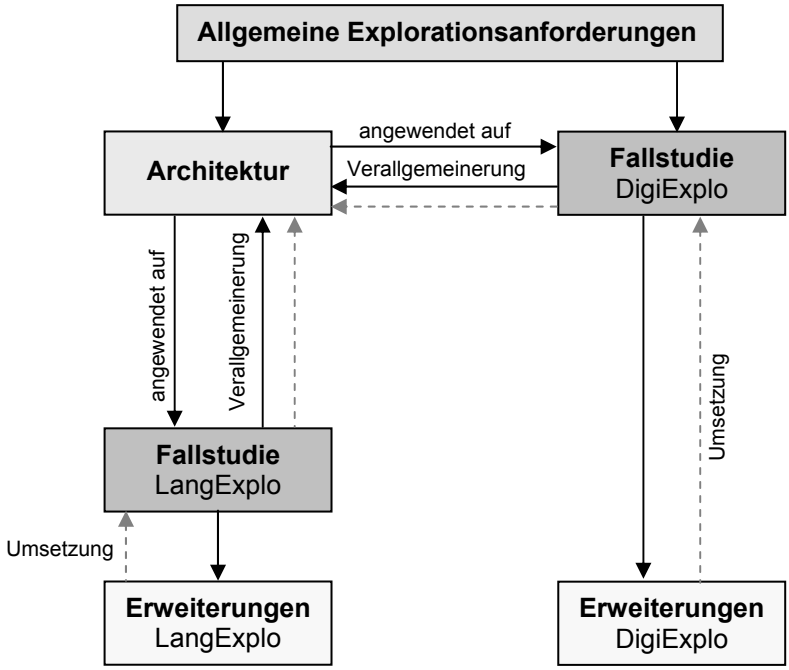


Abbildung 20: Vorgehensweise bei der Entwicklung von Architektur und Fallstudien (vgl. Hoffmann 2003, 89)

Abbildung 21 zeigt die beim objektorientierten Entwurf verfeinerte Fassung des Modells. Dieser Entwurf orientiert sich an der Model-View-Architektur (vgl. Balzert 1999, 373ff), die ihren Ursprung in der Model-View-Controller-Architektur hatte, die erstmalig für Smalltalk-80 verwendet wurde. View und Controller werden hierbei aus Gründen der Vereinfachung kombiniert. Der Explorationsgegenstand wird im *ExploModell* als Modell zur Verfügung gestellt, verschiedene Sichten auf diesen Gegenstand werden als Views modelliert. Des Weiteren wird das Beobachter-Muster (Gamma et al. 1996, 257ff) für die Synchronisation des Explorationsmodells und der vielfältigen Sichten auf dieses verwendet. Jede Komponente, die über eine Zustandsänderung des Modells benachrichtigt werden möchte, muss sich als Beobachter registrieren. Tritt eine Zustandsänderung ein, so werden alle registrierten Beobachter darüber informiert und können ihren Zustand, also z.B. ihre Darstellung, dementsprechend aktualisieren.
Durch die zwei realisierten Fallstudien zeigte sich die Tragfähigkeit des Architekturkonzepts. Die Gestaltung von Software zur Exploration im Bildungskontext wird damit systematisiert und vereinfacht. Wenn die Gestaltung solcher Software entsprechend einfach möglich ist, fällt es leichter, interessierte Personen dazu zu motivieren, entsprechende Mittel für Lehr-Lern-Prozesse zu gestalten. Eine entsprechend breite Verfügbarkeit solcher Software stellt eine wesentliche Voraussetzung für die Verbreitung des hier vorgestellten Ansatzes auch in andere Themenbereiche der Informatik und darüber hinaus dar.

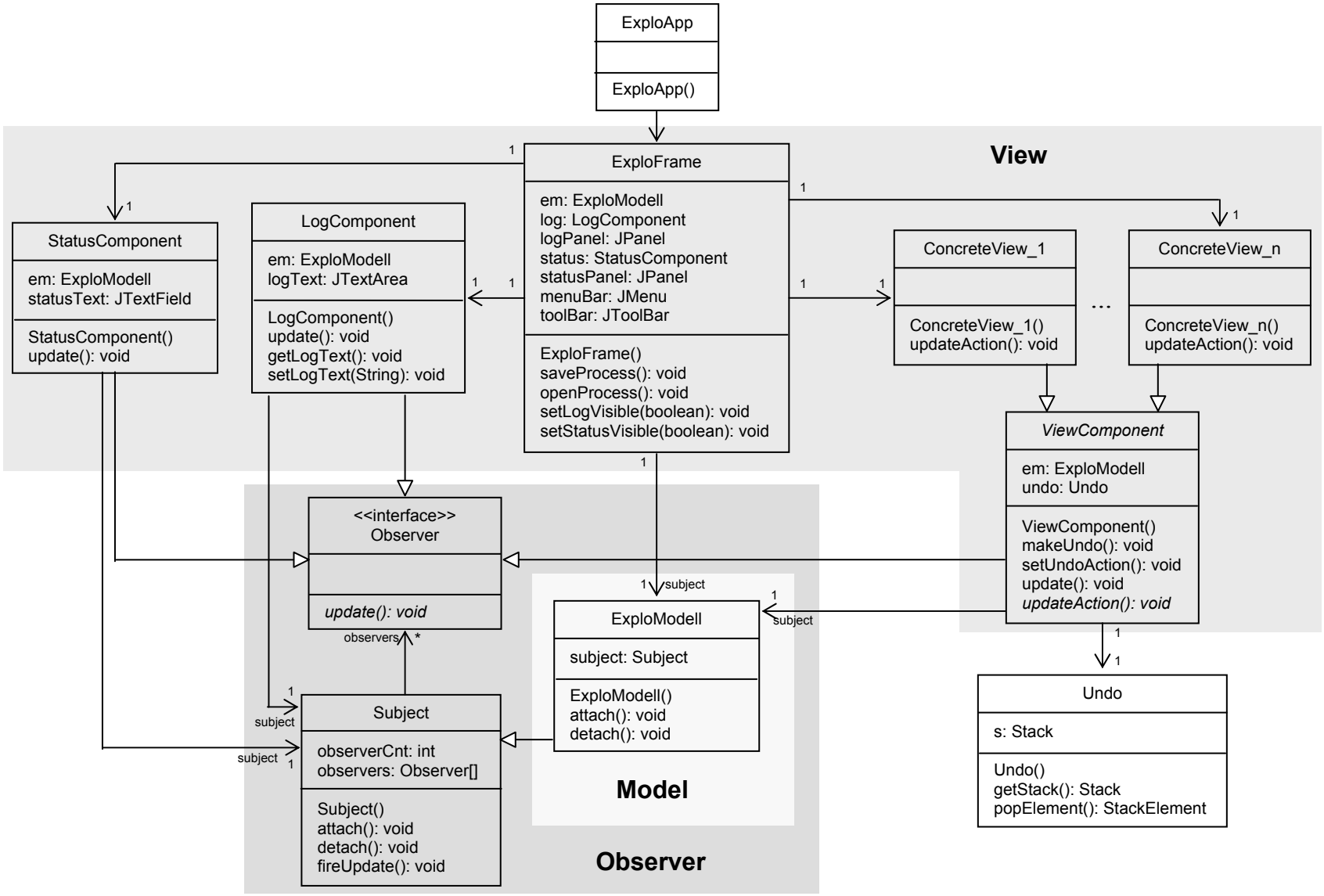


Abbildung 21: Architekturkonzept – OOE-Modell (nach Hoffmann 2003, 64)

5.4 Konzept für das Lernen mit Explorationsmodulen

5.4.1 Pädagogische Doppelfunktion von Explorationsmodulen

Explorationsmodule haben eine pädagogische Doppelfunktion. Je nach Lerngruppe können sie entweder als Unterrichtsmedien verwendet oder als Beispiele für komplexe Software-Systeme analysiert und diskutiert werden. Im zweiten Fall werden sie damit zum Unterrichtsinhalt.

> „In der Pädagogik wird als Medium ein *Mittel* oder ein *Mittler* bezeichnet, mit dessen Inhalt der Unterrichtsinhalt an die Schüler vermittelt werden kann." (Meyer 1994, 148, Hervorhebungen im Original)

> „Der Unterrichtsinhalt ist die Vergegenständlichung der vom Lehrer und den Schülern geleisteten zielgerichteten Arbeit, sozialen Interaktion und sprachlichen Verständigung." (ebd., 80)

Meyer weist auch darauf hin, dass Medien nicht „an sich" existieren. Nur im Zusammenhang mit einem konkreten Lehr-Lern-Prozess kann ein bestimmter Aspekt des Unterrichts als Medium bezeichnet werden.

> „Dies liegt daran, daß die Entscheidung darüber, ob ein bestimmter Aspekt des Unterrichts bzw. eine bestimmte Handlung oder Kompetenz des Lehrers oder der Schüler als ‚Medium' bezeichnet werden kann, nicht objektiv vorgegeben ist, sondern von der hermeneutisch vermittelten Interpretation der Zielsetzung der Stunde bzw. der ganzen Unterrichtseinheit abhängt. [...] Es gibt keine Medien ‚an sich', deren theoretischer Status unabhängig von realen Unterrichtsprozessen bestimmt werden könnte [...]." (ebd., 150)

Kerres (2001, 23) präzisiert zur didaktischen Qualität von Medien:

> „Die didaktische Qualität oder Wertigkeit eines Mediums lässt sich nicht an Merkmalen des Mediums selbst (seien sie inhaltlicher, konzeptueller oder gestalterischer Art etc.) feststellen, sondern nur in dem kommunikativen Zusammenhang, in dem das Medium Verwendung findet."

Aus diesen Gründen wird hier, auf der Basis der Vorarbeiten der vergangenen Abschnitte, zusammengefasst und präzisiert, wie Explorationsmodule in Lehr-Lern-Prozesse integriert werden können. Der Grundgedanke dabei ist, dass Lernende in der Anfangsphase mit Explorationsmodulen als Unterrichtsmedien an OOM-Fachkonzepte herangeführt werden. Fortgeschrittene Lernende (z.B. im Leistungskurs Informatik) können im Rahmen von Unterrichtsprojekten in Teamarbeit in begrenztem Umfang selbst Explorationsmodule für nachfolgende Schülergenerationen erstellen.

> „Der Einsatz von Medien kann die Selbständigkeit der Schüler fördern, weil sie eigenständig an und mit den Medien arbeiten und weil die Aneignung des Unterrichtsinhalts in größerer Distanz zur Sichtweise des Lehrers stattfinden kann. Dies gilt erst recht, wenn sich die Schüler ihre Medien selbst herstellen können." (Meyer 1994, 151)

Dass Lernende selbst Explorationsmodule herstellen, mit denen sie sich explorativ Fachkonzepte des objektorientierten Modellierens aneignen, die sie aber bereits verstanden haben müssen, um diese Software zu analysieren, zu entwerfen und zu implementieren, stellt einen nicht aufzulösenden Zyklus dar. Möglich ist aber, dass fortgeschrittene Lernende kleine Explorationsmodule für nachfolgende Schülergenerationen entwickeln. In der Fachgruppe „Didaktik der Informatik und E-Learning" an der Universität Siegen wurde im November 2003 eine vom Autor dieser Arbeit betreute Diplomarbeit fertig gestellt, in der ein Architekturkonzept für Software zur Exploration im Bildungskontext gestaltet wurde (vgl. 5.3.5, Hoffmann 2003). Erwartet wird hierdurch eine Vereinfachung der Entwicklung von Software zur Förderung explorativer Lernprozesse zu Fachkonzepten. Explorationsmodule ergänzen den Einsatz von Software-Entwicklungswerkzeugen als Unterrichtsmittel. Die starke Belastung der Lernenden durch Syntax und Semantik einer Programmiersprache im Anfangsunterricht (Schubert 2001c) wird durch den Einsatz von Explorationsmodulen vermieden.

5.4.2 Explorationsstrategien für objektorientiertes Modellieren

Das hier dargestellte Explorationskonzept wird zur Bereicherung des Methodenrepertoires traditionellen Informatikunterrichts vorgeschlagen. Deshalb sind sowohl lerntheoretische als auch informatikdidaktische Prinzipien und Methoden für die Gestaltung entscheidend. Als Lerntheorie wird der Konstruktivismus (vgl. Vygotsky 1978) angewendet, der sehr geeignet ist für die vielfältigen konstruktiven Anforderungen des Informatikunterrichts. Zu den übergeordneten Zielen von Unterricht in der Sekundarstufe II gehört die Ausbildung von Sach-, Methoden-, Sozial- und Selbstkompetenz der Lernenden (vgl. z.B. MSWF 1999, MBWK 2001, MKJS 2001, HK 2003). Im Kontext der objektorientierten Modellierung sollen Lernende:

- *Strukturen kennen lernen, analysieren und entwerfen*,
- *Prozesse verstehen und steuern.*

Damit dies gelingt, müssen sie Basiskonzepte und Modellelemente kennen lernen (vgl. Tabelle 31 auf S. 152, Spalte A), diese in Modellsichten verknüpfen (Spalte B) und Modellsichten durch Sichtenwechsel zu einem mentalen Modell eines Systems verknüpfen (Spalte C). Fortgeschrittene Lernende können diese Ziele anhand der Entwicklungsdokumente und Quelltexte fertiger, didaktisch besonders geeigneter Informatiksysteme vertiefen (Spalte D).

Selbstständigkeitsstufen der Exploration

In Anlehnung an Selbstständigkeitsstufen beim Experimentieren (vgl. Pochanke 1980) können auch bei explorativen Herangehensweisen an fachliche Zusammenhänge verschiedene Selbstständigkeitsstufen der Lernenden unterschieden werden (vgl. Tabelle 30).

Exploration	in geschlossenen Umgebungen	in offenen Umgebungen
nach gegebenen Anleitungen	*Exploration mit Explorations-modulen*	
als selbstständiger Weg zur Lösung von Problemen		

Tabelle 30: Selbstständigkeitsstufen der Exploration

Steinkamp (1998, 6) begründete Vorteile einer „geschützten Umgebung“ für Informatik-Experimente in speziellen Experimentierumgebungen im Wesentlichen über die Vermeidung von Gefahren für reale Informatiksysteme (z.B. Rechnernetze), die durch Experimente mit ihnen beschädigt werden könnten (vgl. 5.3.1.2). Übertragen auf Explorationsumgebungen ist es ein wesentlicher Vorteil einer geschützten bzw. abgeschlossenen Umgebung, dass Lernende zwar einerseits explorieren können, andererseits der Lehrende durch die Auswahl von Explorationsmodulen den Explorationsprozess auf die für den Lernprozess wesentlichen Erkenntnisziele richten kann. Wenn Lernende entsprechend lernkompetent sind, können abgeschlossene Umgebungen schrittweise geöffnet werden, z.B. indem Lernende im Internet selbstständig bestimmte fachliche Zusammenhänge recherchieren und im Unterricht präsentieren. Das Explorieren nach Anleitungen stellt einen scheinbaren Widerspruch dar, da Exploration selbstgesteuert erfolgt (vgl. 5.3.1.1). Um unsystematisches Ausprobieren (Versuch-Irrtum-Strategie) zu vermeiden, können Lernende durch explorationsanregende Leitfragen und Hinweise Orientierung für ihre Lernprozesse erhalten und sich die Methodik explorativen Lernens aneignen. Sie lernen damit nicht nur etwas über den fachlichen Zusammenhang, sondern entwickeln auch ihre Lernkompetenz, indem sie exploratives Lernen als systematische Lernstrategie verinnerlichen. Fortgeschrittene Lernende wählen Explorationsmodule selbstständig aus, um gegebene fachliche Probleme zu bewältigen. Das Ziel ist es, dass Lernende

selbstständig bei Auftreten kognitiver Probleme in ihnen zur Verfügung stehenden Wissensbasen nach Strategien zu deren Lösung suchen.

Exploration anhand von Leitfragen in geschlossenen Umgebungen

In der *Anfangsphase* kann der Explorationsprozess der Lernenden z.B. durch Leitfragen, die sich an den Frageniveaus bei der Exploration von Neber (vgl. 5.3.1.1) orientieren, angeregt werden. Die Lernenden benutzen für eine gegebene Lernaufgabe ein vom Lehrenden ausgewähltes Explorationsmodul, in dem sie mit einer Repräsentation von abstrakten Fachkonzepten interagieren und Reaktionen des Systems beobachten können. Schrittweise explorieren sie die Antworten auf die Leitfragen. Während für die Beantwortung der ersten Frageniveaus (Fragen nach Gegebenheiten) Modellelemente und einzelne Modellsichten im Vordergrund stehen, ist dies bei den nachfolgenden (Fragen nach Zusammenhängen) zunehmend der Sichtenwechsel (vgl. Markierungen in Tabelle 31). Bei Fragestufe 1 (Tabelle 31, Zeile 1) analysieren die Lernenden einzelne Modellelemente und Sichten. Sie wechseln zwischen den Sichten, um unterschiedliche Repräsentationen zu erkunden. Die Lernenden beobachten, sie stellen erste Hypothesen über Zusammenhänge auf.

			Explorationsgegenstand			
			A	**B**	**C**	**D**
			Modellelemente	**Modellsicht**	**Sichtenwechsel**	**Informatiksystem**
Fragestufen bei der Exploration	**1.**	**Was ist das?**	●●●	●●●	●●	●
	2.	**Wie funktioniert das?**	●●	●●●	●●●	●
	3.	**Was passiert, wenn ich dieses oder jenes tue?**	●●	●●	●●●	●
	4.	**Wie kann ich ein bestimmtes Ziel erreichen?**	●	●●	●●●	●
	5.	**Hängt die Vorgehensweise zum Erreichen des Ziels von bestimmten Bedingungen ab?**	●	●●	●●●	●
	6.	**Gibt es Beziehungen zwischen den verschiedenen Bedingungen für das Erreichen des Ziels?**	●	●●	●●●	●

Tabelle 31: Explorationsstrategie zum OOM

Diese münden in der Stufe 2 (Zeile 2) in erste Experimente mit dem System, die in Stufe 3 intensiviert werden. Zunehmend erhält der Sichtenwechsel eine zentrale Rolle. Lernende führen in einer Sicht Aktionen aus und beobachten die Reaktionen des Explorationsmoduls in den anderen Sichten. Durch die Exploration der Zusammenhänge zwischen den verschiedenen Sichten gelingt es den Lernenden, Hypothesen über fachliche Zusammenhänge aufzustellen und diese experimentell zu überprüfen. Damit sind sie in der Lage, einen von ihnen gewünschten Zustand des Explorationsmoduls zielgerichtet herzustellen (Stufe 4). Gelingt dies nicht direkt, so können sie durch weitere, zielgerichtete Analyse des Bedingungsgefüges ihre Vorgehensweise zur Erreichung eines Ziels präzisieren (Stufen 5 und 6).

Aktionsmöglichkeiten der Lernenden

Bezogen auf die verschiedenen Explorationsgegenstände ergeben sich damit u.a. folgende Aktionsmöglichkeiten der Lernenden:

- *Basiskonzepte / Modellelemente*
  - Basiskonzepte und Modellelemente einander zuordnen,
  - Analyse eines Basiskonzepts / Modellelements,
  - Basiskonzepte / Modellelemente identifizieren.
- *Modellsicht*
  - Modellsicht analysieren,
  - Modellsicht überprüfen,
  - Modellsicht modifizieren,
  - Zuweisen und Verknüpfen von Modellelementen / Konstruktion einer Modellsicht.
- *Sichtenwechsel*
  - Modellsichten identifizieren,
  - Modellsichten vergleichen,
  - Konsistenz von Modellsichten überprüfen,
  - Modellsichten transformieren.

Zu beachten ist hierbei der Zusammenhang mit Aufgabentypen beim objektorientierten Modellieren (vgl. 4.3.3).

Fortgeschrittene Lernende (z.B. Jgst. 13 im Informatikleistungskurs) können den Explorationsprozess nachvollziehen, indem sie die Umsetzung verschiedener informatischer Fachkonzepte (z.B. Entwurfsmuster, vgl. Gamma et al. 1996) in einem Explorationsmodul analysieren und selbst daraus resultierende, kleine Erweiterungen des Systems vornehmen. Die Software wird damit zum Unterrichtsinhalt (vgl. Tabelle 31, Spalte D).

Experimente beim Explorieren

Die im Rahmen des Explorationsprozesses von den Lernenden durchgeführten Experimente sind in der Regel qualitativ.

> „Most exploratory experiments concerned with discovering new facts or with giving new theories a preliminary test, are qualitative or semi quantitative: only if they are either definitely favorable or altogether inconclusive it is worth while to attempt the attainment of quantitative accuracy. [...] The design of quantitative experiments is clearly more complex than that of qualitative experiments but not necessarily more ingenious: the use of measurement instruments presupposes that the variables concerned have been objectified and that measurement techniques have been evolved, whereas a qualitative experiment may ‚demonstrate' (i.e. objectify or render manifest) certain variables and relations for the first time. Here again originality may be more valuable than accuracy." (Bunge 1967, 252)

Bei diesen Experimenten geht es um das Entdecken, Erkunden, Prüfen von Hypothesen und die Ermittlung von Strukturen (vgl. 5.3.1.2) und nicht um das Messen bestimmter Variablen. Typische Phasen des Experimentierprozesses (Vorbereitung, Durchführung, Auswertung, vgl. Sachs 1997, 250) verlaufen bei der Exploration wiederholt in relativ kurzer Zeit ab. Durch das Explizieren von Explorationsstrategien im Unterricht kann hierbei eine Abgrenzung vom unsystematischen Ausprobieren erfolgen. Zentral ist, dass die Erkenntnisse aus einem Experiment handlungsleitend für weitere Experimente sind (formative Tätigkeit beim entdeckenden Lernen, vgl. Hameyer 2002). Betrachtet man die Phasen des Experimentierprozesses genauer, so stehen bei der Vorbereitung zu beantwortende Fragestellungen und daraus resultierende Konsequenzen für den Versuchsaufbau im Vordergrund. Als Leitfragen eignen sich hier die Fragestufen bei der Exploration nach Neber (s.o.). Durch vielfältige Interaktions- und Beobachtungsmöglichkeiten können die Lernenden die Fragen im Rahmen der Durchführung von

Experimenten mit dem Explorationsmodul schrittweise beantworten. Der jeweilige Explorationsgegenstand wird so nach einem begründeten Plan beobachtet und verändert. Die Konsequenzen des Handelns werden analysiert und bewertet. Hameyer (2002) unterscheidet in diesem Zusammenhang explorative, reflexive, konstruktive und formative Tätigkeiten beim entdeckenden Lernen. Interaktionen der Lernenden mit dem System und die Notwendigkeit, Entscheidungen zu treffen, sind möglich bei allen Parametrisierungs- und Konstruktionsprozessen objektorientierter Modelle, so z.B. beim halbautomatischen Sichtentransfer (z.B. vom Klassendiagramm zum Objektdiagramm, vgl. 5.3.3). Die Experimentauswertung besteht jeweils darin, dass die Lernenden ihr erwartetes Systemverhalten mit dem tatsächlichen abgleichen, Handlungen bei nicht erwartetem Verhalten stornieren und dann zielgerichtet die Systemreaktionen auf sich daraus ergebende alternative Eingaben explorieren. Nach dem so modifizierten Plan werden Beobachtung und Manipulation fortgesetzt. Dieser iterative Prozess wird anschließend in der Lerngruppe analysiert, diskutiert und bewertet.

Kompetenzentwicklung

Durch die Exploration fachlicher Zusammenhänge schärfen die Lernenden ihre Sach- und Methodenkompetenz. Da sie ihre Strategien in kleinen Teams erarbeiten und anschließend in der Lerngruppe präsentieren und diskutieren, wird auch ihre Sozialkompetenz gefördert. Durch die Einbettung im Informatikunterricht in die Theorie des objektorientierten Modellierens werden die Beobachtungen und Analyseergebnisse der Lernenden strukturiert und zu Wissen. Explorationsmodule ermöglichen es, theoretische Erkenntnisse deduktiv zu überprüfen und explorativ gefundene Erkenntnisse induktiv als Anstoß für theoretische Folgerungen zu verwenden. Der systematische, iterative Prozess aus Beobachten, Manipulieren und Vergleich der Systemreaktion mit der jeweiligen Erwartung schärft die Hypothesenbildung der Lernenden. Durch die mentale Verknüpfung verschiedener Sichten beim Sichtenwechsel zu einem Gesamtbild eines Systems wird das vernetzte Denken der Lernenden gefördert. Rückmeldungen der Explorationsmodule fördern ihr Fehlerbewusstsein. In traditionellem Unterricht ist es kaum möglich, die Lösungen aller Lernenden zu einem gegebenen Problem zu diskutieren. Explorationsmodule können dazu beitragen, dass mehr Lernende individuelle Rückmeldungen erhalten. All diese Aspekte steigern die Selbstkompetenz der Lernenden. Damit liefert das in traditionellen Unterricht eingebettete Lernen mit Explorationsmodulen einen Beitrag zur Qualität der informatischen Bildung in Schulen.
Eine erfolgreiche Umsetzung ist an zahlreiche Bedingungen gebunden, z.B. im Hinblick auf die Qualität der Unterrichtsmedien (vgl. 5.3.3) und die Bildung der Informatiklehrenden im Hinblick auf Exploration. Lehramtsstudierende der Informatik müssen einerseits auf Exploration als neue Lehrmethode vorbereitet werden und andererseits Explorationsmodule sowohl als Lernhilfen als auch als Unterrichtsinhalt einbeziehen können (vgl. 3.7). Hierzu fanden Erprobungen im Rahmen des Informatiklehramtsstudiums an der Universität Dortmund statt (vgl. 5.5.3.1).

5.4.3 Verknüpfung mit Handlungsphasen problemorientierten Unterrichts

Ein Zugang über Exploration ist weder für alle Bereiche des objektorientierten Modellierens noch für alle unterrichtlichen Handlungsphasen gleichermaßen geeignet. Deshalb sind realistische Erwartungen angebracht und überzogene zu vermeiden. Nachfolgend wird dargestellt, für welche Phasen des Unterrichts ein Zugang über Explorationsmodule besonders geeignet erscheint (vgl. Brinda / Schubert 2003, 117f). Zugrunde gelegt wird hierbei ein problemorientiertes Stufenkonzept (nach Heinrich Roth, vgl. Meyer 1994, 183ff).
Explorationsmodule eignen sich sehr gut zur Motivation einzelner Modellelemente und Modellsichten, insbesondere im Bereich des Sichtenwechsels (vgl. Tabelle 32, Zeile 1). Die be-

sondere Eignung eines explorativen Zugangs für die frühen Phasen des Lehr-Lern-Prozesses (Stufen 1 bis 3) ergibt sich aufgrund der spezifischen Visualisierung und der Interaktionsmöglichkeiten. In besonderer Weise gilt dies für den Sichtenwechsel, bei dem die Lernenden durch die computerunterstützte Synchronisation von Modellsichten und deren Visualisierung den speziellen didaktischen Mehrwert der Explorationsmodule für ihre individuellen Lehr-Lern-Prozesse feststellen können. Wenn Lernende die Bedeutung eines Sichtenwechsels in einer Beispielwelt entdecken können, werden sie dazu motiviert, auch zukünftig mit dieser Methode zu arbeiten, auch wenn keine diesbezügliche Systemunterstützung zur Verfügung steht. Weniger geeignet sind Explorationsmodule allerdings für das Erproben von eigenen Lösungswegen. Explorierbar sind nur die Lösungsmöglichkeiten, die der Explorationsmodul bereitstellt, z.B. durch Laden anderer Beispiele (Szenarien in LEO, vgl. 5.3.4.4). Durch die Menge der zur Verfügung stehenden, implementierten Beispiele ergibt sich hier eine Beschränkung (Tabelle 32, Zeile 4). Das gilt analog auch für das Behalten und Üben (Zeile 5). Andere Lernformen, z.B. das Bearbeiten von Übungsaufgaben (vgl. Kapitel 4), können den Lehr-Lern-Prozess wirkungsvoll ergänzen. Die Verknüpfung verschiedener Sichten beim Sichtenwechsel (Zeile 6, Spalte C) birgt in sich großes Potential für einen spezifischen Transfer in andere Bereiche der Informatik, in denen auch verschiedene Repräsentationen fachlicher Zusammenhänge zu einem mentalen Bild des Gesamtsystems verknüpft werden müssen. Insgesamt wird hierdurch das vernetzte Denken der Lernenden gefördert.

			Explorationsgegenstand			
			A	**B**	**C**	**D**
			Modellelemente	**Modell sicht**	**Sichtenwechsel**	**Informatiksystem**
Problemorientiertes Stufenkonzept	**1.**	**Motivation / Lernwunsch**	●●	●●	●●●	●
	2.	**Identifikation von Schwierigkeiten im Lerngegenstand**	●●	●●	●●●	●
	3.	**Entdecken eines neuen Lösungsweges**	●●●	●●●	●●●	●
	4.	**Erproben des Lösungsweges**	●	●	●	●
	5.	**Behalten und Einüben**	●	●	●	●
	6.	**Bereitstellen, Übertragen und Integration des Gelernten**	●	●	●●	●

Tabelle 32: Anwendung der Exploration in der informatischen Bildung

Die Einbeziehung eines Explorationsmoduls als zu analysierendes und zu modifizierendes Informatiksystem in den Lehr-Lern-Prozess ist prinzipiell eine geeignete Vorgehensweise auch für problemorientierten Unterricht, vorausgesetzt Vorwissen der Lerngruppe, Komplexität der Software und Qualität der Entwicklungsdokumente sowie der Implementierung stehen in einem ausgewogenen Verhältnis zueinander. Zur Lösung gegebener Problemstellungen analysieren die Lernenden Explorationsmodule und deren Entwicklungsdokumentation. Neben der Software gehört hierzu die Problembeschreibung, Analyse- und Entwurfsdokumente und der vollständige Quelltext. Das Arbeiten mit diesen Dokumenten erfordert fundiertes Wissen über objektorientierte Modellierungs- und Programmiertechniken und kann deshalb

erst in einem weit fortgeschrittenen Stadium der Informatikausbildung stattfinden. Erfolgreiche Strukturierungsideen werden identifiziert und in der Lerngruppe verallgemeinert. Die Lernenden reflektieren die Erkenntnisse und beziehen sie in die Lösung gegebener Problemstellungen mit ein. Sie verwenden die Dokumentation auch, um herauszufinden, wie verschiedene Modellelemente in der verwendeten Programmiersprache implementiert werden. Dies ist jedoch nur möglich, wenn die Explorationsmodule in derselben Programmiersprache implementiert wurden, in der die Lernenden ihre eigenen Problemlösungen erstellen.

5.4.4 Exploration in Hochschulen

Für traditionelle Vorlesungen ist das Lernen mit Explorationsmodulen wenig geeignet. Im Rahmen von „Notebook University"-Konzeptionen werden innovative Lernszenarien entwickelt, um von dozentenzentrierten Lehrveranstaltungen zu mehr Lernerzentrierung zu gelangen. Beispiele hierfür sind „interaktive Vorlesungen und Seminare", in denen interaktive Vorlesungs- und Übungsmaterialien von den Lernenden direkt in der Vorlesung am eigenen Notebook annotiert und bearbeitet werden können (z.B. Brehm et al. 2003). Hier wären auf die jeweilige Lerngruppe abgestimmte Explorationsmodule geeignete Lernmedien. Bereits ohne den technischen Aufwand von „Notebook University"-Konzeptionen können in traditionellen Übungen, Seminaren und Praktika mit Explorationsmodulen unterstützte Lernszenarien gestaltet werden.

Beispielszenario „Bibliothek"

Nachfolgend wird ein Beispielszenario zum Lernen mit Explorationsmodulen vorgestellt, von dem Elemente im Rahmen der Übung zur zweistündigen Vorlesung „Informatisches Modellieren" des Zertifikatsstudiengangs „Medien und Informationstechnologie in Erziehung, Bildung und Unterricht" an der Universität Dortmund mit Lehramtsstudierenden erfolgreich erprobt wurden (vgl. Brinda / Schubert 2002b). Diese Studierenden hatten zuvor im Rahmen einer weiteren zweistündigen Vorlesung einen Überblick über und Einblicke in verschiedene Teilgebiete der Informatik erhalten. Da es sich um Lehramtsstudierende anderer Fächer (also nicht der Informatik) ohne weitere spezifische Informatikbildung handelte, ist die Ausgangslage mit dem Lernen in der Sekundarstufe II in etwa vergleichbar.
Ziel der Lernenden war es hier, die Arbeitsprozesse in einer Bibliothek mit einem Informatiksystem zu vereinfachen. Vorwissen bildeten statische und dynamische Basiskonzepte des objektorientierten Modellierens (vgl. Fowler / Scott 1997) und die Skriptsprache Python als Programmiersprache. Eine geeignete Systemunterstützung hierfür bildet LEO mit dem Bibliotheksszenario (vgl. 5.3.4.4). Die Lernenden beginnen damit, das Problem in Teilprobleme zu zerlegen. Sie diskutieren und identifizieren Bibliothek, Leser und Bücher als wesentliche Objekte. In der Realsicht des LEO-Systems können diese exploriert werden. Um zur Objektsicht zu gelangen, ist ein Abstraktionsschritt erforderlich. Ein zum Anwendungsszenario passendes Objektdiagramm kann in der Objektsicht bereitgestellt und inspiziert werden (vgl. Abbildung 58 auf S. 251). Um das höhere Abstraktionsniveau „Klassendiagramm" zu verstehen, können Objekte mit gemeinsamen Attributen abstrahiert und zu Klassen kombiniert werden (vgl. Abbildung 59 auf S. 252). Begleitend zu den explorativen Phasen konstruiert die Gruppe Lösungsteile des Bibliotheksproblems. Vermutungen der Lernenden in Bezug auf problemspezifische Objekte, Attribute und Methoden werden im Team gesammelt und die Erkenntnisse gemeinsam systematisiert. Hier kann der Prozess durch einen weiteren Explorationsmodul begleitet werden, der eine Verknüpfung von Exploration und Konstruktion ermöglicht. Dieser stellt den Lernenden eine präzise, textuelle Beschreibung des Bibliotheksproblems zur Verfügung. Darin haben sie die Möglichkeit, in einem ersten Schritt die Klassenkandidaten auszuwählen. Nach dieser Auswahl können sie mit Hilfe von vom System bereitgestellten Check-

listen entscheiden, ob ein Klassenkandidat eine Klasse ist oder nicht. Weiterhin stellt das System die Möglichkeit bereit, Klassenkandidaten gemäß bestimmter Heuristiken aufzulisten. Zu jedem Zeitpunkt ist es möglich, die getroffenen Entscheidungen dadurch zu visualisieren, dass die identifizierten Klassen als Symbole in der Klassensicht dargestellt werden. Attribute-, Methoden- und Relationskandidaten, die im Text identifiziert wurden, werden als Produktionselemente für das Klassendiagramm angeboten. Sie können dort verknüpft werden und unterstützen deshalb den Lernprozess des Strukturierens. Die Lernenden konstruieren individuelle Klassendiagramme und diskutieren Vor- und Nachteile von Varianten in der Gruppe. Für die weitere Gestaltung stimmt die Gruppe ein gemeinsames Diagramm ab. Um Verständnis dafür zu entwickeln, wie im fertigen Bibliothekssystem die Objekte kooperieren, um das Systemziel zu realisieren, ist es erforderlich, den normalerweise versteckten Nachrichtenaustausch durch kooperierende Objekte zu visualisieren. Im LEO-System können Lernende, ausgehend von einem fertig gestellten Klassendiagramm der Bibliothek, Objekte instanziieren und ein Objektdiagramm konstruieren. Durch Multiplizitäten gegebene Objektrelationen können automatisch vom System erzeugt werden. Ein Objektdiagramm, das von Lernenden konstruiert wurde, kann auf seine Konsistenz mit einem gegebenen Klassendiagramm überprüft werden. Im fertig gestellten Objektdiagramm können die Lernenden Methoden der Objekte aufrufen, den Nachrichtenaustausch mit anderen Objekten beobachten und deren Zustandswechsel verfolgen.

5.5 Erprobungen in der informatischen Bildung

5.5.1 Vorbemerkung

Die im Folgenden dargestellten Ergebnisse entstanden im Rahmen von verschiedenen exemplarischen Erprobungen zwischen 2001 und 2003. Erkundet wurden verschiedene Einsatzszenarios in der Sekundarstufe (vgl. 5.5.2), in der Ausbildung von Informatiklehramtsstudierenden (vgl. 5.5.3.1) und der Fortbildung von Informatiklehrpersonen (vgl. 5.5.3.2) auf die Gestaltung von Unterricht unter Verwendung von Explorationsmodulen sowie die Entwicklung von Software zur Exploration mit Informatikstudierenden im Hauptstudium (vgl. 5.5.4). Die hier dargestellten Ergebnisse erheben nicht den Anspruch, repräsentativ zu sein, sondern dienten im Rahmen einer internen Qualitätssicherung dazu, die Akzeptanz bei Lehrenden und Lehrenden zu erkunden und um erste Erkenntnisse für eine Weiterentwicklung dieser Komponente des didaktischen Systems ableiten zu können.

5.5.2 Sekundarstufe

Im Bereich der Sekundarstufe wurden ausgewählte Animationen (die „Puzzles“, vgl. 5.3.4.2) mit Lernenden eines Grundkurses Informatik der Jahrgangsstufe 12 sowie mit Schülerinnen der 10. Jahrgangsstufe im Rahmen der „Schupperuni für Schülerinnen“ des Fachbereiches Informatik der Universität Dortmund exemplarisch erprobt. Ferner wurde LEO im Informatikunterricht eines Berufskollegs zur Einführung in das Modellieren mit UML eingesetzt.

Animationen: Puzzles

Informatikgrundkurs einer Jahrgangsstufe 12 (Gesamtschule)

Im Juni 2001 wurden die Puzzles von Lernenden eines Informatik-Grundkurses der Jahrgangsstufe 12 einer Gesamtschule im Raum Dortmund erprobt (vgl. Weigend 2001a). Es zeigte sich die Praxistauglichkeit. Im Einzelnen führte die Erprobung zu folgenden Erkenntnissen bzgl. typischer Fehler bzw. Schwächen in den Lösungen der Lernenden:

- *Fehler:*
  - Die Richtung der Aggregationssymbole wurde von einigen Lernenden systematisch vertauscht.
  - Von einigen Lernenden wurde keine mehrstufige Aggregationshierarchie aufgebaut, wie es aber zur Lösung erforderlich gewesen wäre.
- *Schwächen:*
  - Einige Lernende wählten eine ungewöhnliche Anordnung mit der Wurzel der Aggregationshierarchie unten.
  - Bei weiteren Lernenden waren die Klassen und Aggregationssymbole ungeordnet platziert.
  - Teilweise waren die Rauten der Aggregationssymbole durch Klassensymbole verdeckt.

Damit Lernende die Richtung von Relationssymbolen korrekt wählen, wäre bei den Puzzles die Verknüpfung mit einer Realsicht sinnvoll, in der eine Visualisierung zum gegenwärtigen Status eines Klassendiagramms angezeigt werden kann. Lernende können so erkennen, ob ihr Modell die von ihnen beabsichtigte Semantik trägt oder nicht. Ferner können die in Abschnitt 5.3.4.2 dargestellten Erweiterungsmöglichkeiten dazu beitragen, diese Fehler zu vermeiden bzw. schrittweise selbstständig zu beheben. Die im Bereich der „Schwächen“ genannten Anordnungsprobleme sollten mit den Lernenden in der Gruppe erörtert werden. Insbesondere die systematische Anordnung von Komponenten erleichtert das Lesen von Diagrammen durch

andere erheblich. Lernenden mit Programmiervorerfahrung kann dies über den Vergleich mit der übersichtlichen Gestaltung von Quelltexten (Einrückungen etc.) zur Erhöhung der Lesbarkeit veranschaulicht werden

Schnupperuni für Schülerinnen am Fachbereich Informatik der Universität Dortmund

Seit 1998 findet an der Universität Dortmund am Fachbereich Informatik zu Beginn des Wintersemesters jährlich eine „Schnupperuni" für Schülerinnen statt mit dem Ziel, mehr Studentinnen für ein Informatikstudium zu gewinnen[91]. In der Schnupperuni 2001 (18.-20.10.01) gestaltete die Fachgruppe „Didaktik der Informatik" am 19.10.2001 eine Veranstaltung zum Thema „E-Learning" bestehend aus Vorlesung und Übung[92]. In der Übung, an der ca. 50 Schülerinnen der Klasse 10 mit unterschiedlichen Informatikvorkenntnissen teilnahmen, wurden u.a. die Puzzles eingesetzt (Aufgabentext vgl. E.2). Die den Puzzles zugrunde liegenden objektorientierten Prinzipien und die Semantik der verwendeten Symbole wurden kurz erläutert.
Einigen Schülerinnen mit Vorkenntnissen zur Objektorientierung gelang die Bearbeitung der Puzzles ohne Schwierigkeiten. Insbesondere bei den Schülerinnen ohne diesbezügliche Vorkenntnisse zeigten sich neben den Problemen, die sich auch beim Einsatz im Informatikgrundkurs der Jgst. 12 ergaben (s.o.), prinzipielle Schwierigkeiten mit der Semantik aufgrund fehlender Vorkenntnisse. Es bestätigt sich hierbei, dass für effektives, exploratives Lernen gewisse fachliche Vorkenntnisse vorliegen müssen. Die Schülerinnen zeigten sich dennoch sehr interessiert, fragten die begleitenden Tutoren um Hilfe und halfen einander.

LEO

Seit 2001 besteht eine Kooperation der Fachgruppe „Didaktik der Informatik" bzw. „Didaktik der Informatik und E-Learning" mit einem Informatiklehrer und langjährigen Fachmoderator im Unterrichtsfach Informatik für IT-Berufe für die Bezirksregierung Arnsberg, der kürzlich auch ein Lehrbuch zur „Programmentwicklung mit UML" veröffentlichte (vgl. Mielke / Sagorny 2003). Im Rahmen seiner Moderatoren- und Lehrertätigkeit setzte er LEO in Lehrerfortbildungen und im Informatikunterricht der Höheren Berufsfachschule im Bildungsgang „Informationstechnischer Assistent" ein. Sagorny berichtet von folgenden Erfahrungen mit LEO (vgl. E.3):

> „Die Vermittlung von objektorientierten Konzepten bereitet dem Lernenden, insbesondere in der Anfangsphase, große Probleme bei der Vorstellung der Arbeitsweise objektorientierter Programmsysteme. Dies liegt vermutlich daran, dass der Lernende zum einem vor dem Problem der Identifikation von Klassen steht und zu anderem eine Vorstellung entwickeln muss, wie das Verhalten der Anwendung aus Objekten daraus resultiert. Eine weitere pädagogische Schwierigkeit liegt in der Auswahl geeigneter Beispiele und deren anschauliche Modellierung. Hier helfen professionelle Tools wie z.B. Together Control Center zwar bei der Darstellung der Analyse und der Generierung von Klassenrümpfen. Dem Lernenden hilft dies aus meiner Erfahrung jedoch wenig beim Verständnis des Modellierungsvorgangs.
> An dieser Stelle lässt sich die Lernumgebung LEO phantastisch einsetzen. Die gewählte Darstellungsform, mit der Sicht aus fünf verschiedenen Perspektiven, ermöglicht dem Lernenden zu jedem Zeitpunkt eine anschauliche Sichtweise auf die Problemstellung. Ein weiterer Pluspunkt ist die Wahl der [...] gewählten Beispiele. Diese lagen bei allen Lerngruppen, in denen ich das Tool bisher eingesetzt habe, im täglichen Erfahrungsbereich. Besonders gut aufgenommen wurde hier das Beispiel Mobiltelefon mit dem Ziel Versand einer SMS.
> Der Einsatz des Tools erfolgte in meinem Unterricht ohne Einweisung auf das Tool. Es wurde lediglich der Import der Szenarios erläutert, nicht jedoch der Umgang mit dem Tool selbst. Die Vorkenntnisse der Lerngruppen beschränkten sich zum Zeitpunkt des Einsatzes auf die objektorientierten Grundbegriffe, so wie die Grundzüge der Darstellung des Klassendiagramms mit Hilfe der

[91] Schnupperuni-Homepage: http://www.schnupperuni.de/ (aufgerufen am 12.11.03)
[92] Programm der Schnupperuni 2001: http://ls10-www.cs.uni-dortmund.de/~schnupperuni/uni2001/schnuppern.html (aufgerufen am 12.11.03)

> UML. Die Lernenden fanden durchweg zügig die Möglichkeit der Erzeugung von Objekten durch Anklicken eines Konstruktors im Klassendiagramm. Die dann im Objektdiagramm sichtbar werdenden Operationen bereiteten den Lernenden keine Probleme mehr. Das Tool visualisiert in dieser Phase sehr anschaulich den Schablonencharakter einer Klasse, da außer der Objektsicht das entsprechende Objekt auch in der Realsicht sichtbar wird. Der Aufruf von Operationen zeigt außerdem den Versand von Botschaften zwischen den Objekten. Diese Art der Darstellung ist mir aus keinem anderen bisher bekannten Tool bekannt. Sie hilft jedoch dem Lernenden enorm beim Verständnis der objektorientierten Konzepte."

Es zeigt sich hier, dass LEO insbesondere in der Anfangsphase gut geeignet ist, Lernenden einen explorativen Einblick in die fachlichen Zusammenhänge objektorientierten Modellierens zu ermöglichen. Sagorny lieferte eine Gestaltungsanregung für die Verbesserung der Sequenzsicht:

> „Sehr anschaulich ist auch die Darstellung der Operationen im Sequenzdiagramm und in der Programmsicht. Schön wäre hier noch, wenn das Sequenzdiagramm automatisch scrollen würde, um immer die aktuell ausgewählten Operationen sehen zu können."

Diese Erweiterung wurde im Rahmen der Pflege und Weiterentwicklung des LEO-Systems aufgegriffen und verwirklicht. Mielke und Sagorny empfehlen LEO im Rahmen ihres Lehrbuchs zur Programmentwicklung mit UML (vgl. Mielke / Sagorny, 132ff).

5.5.3 Informatiklehrerbildung

5.5.3.1 Lehramtsstudium Informatik

Erörterung unterrichtlicher Einsatzmöglichkeiten und der Implementierung eines Explorationsmoduls

Konzeption

Diese Erprobung wurde durchgeführt im Rahmen der Grundstudiumsvorlesung „Didaktik der Informatik I" an der Universität Dortmund im Sommersemester 2001. Laut Studienordnung wird die Vorlesung für das 4. Fachsemester empfohlen. Die Vorlesung wurde von einer wöchentlichen Übung begleitet, in der die Studierenden 70% der möglichen Gesamtpunktezahl aller Übungsaufgaben (jeweils 10 Punkte / Blatt) für einen Nachweis (Übungsschein) erreichen mussten. Die hier beschriebenen Ergebnisse beziehen sich auf die achte und neunte von insgesamt zwölf Übungen. Die Aufgabenblätter für die Übungen 8 und 9 sind angefügt im Anhang E.4. In einer vorangegangenen Übung beschäftigten sich die Studierenden mit dem Konzept für Informatik-Experimente von Steinkamp (1999), vgl. 5.3.1.2.

Grundlage für die Blätter 8 und 9 war eine von einem Student erstellte Arbeitsfassung eines Explorationsmoduls[93]. Im Anhang E.4 befinden sich zwei Bildschirmfotos dieser Software (vgl. Abbildung 38 und Abbildung 39). Inhaltlich stellt sie eine frühe Fassung des „Explorers für geometrische Objekte - EGO" dar (vgl. 5.3.4.3 und F.3.1). Nach dem Programmstart erscheint ein zweigeteiltes Fenster. Im oberen Bereich können einfache geometrische Objekte erstellt werden (Rechteck, Linie). Im unteren Bereich wird das jeweils ausgewählte geometrische Objekt als Objektdiagramm dargestellt. Änderungen am geometrischen Objekt (Verschiebung, Skalierung) werden in der Objektdiagrammrepräsentation nachvollzogen (Abbildung 38). Ist die Option „Nachrichtenaustausch verfolgen" aktiviert, so werden die im oberen Teil durchgeführten Aktionen im unteren Teil mittels Nachrichten zwischen den beteiligten Objekten dargestellt (Abbildung 39). Wie sich zeigte, hatte der realisierende Student die Idee zwar weitestgehend korrekt implementiert, die Quelltexte zeigten aber, dass er das Wesen objektorientierter Programmierung selbst nicht wirklich verstanden hatte. Diese Software wurde

[93] Die Software inklusive Quellen steht im Internet bereit zum Download: URL: http://www.didaktik-der-informatik.de/DIE_BIB/lehre/ddi/ddi1/sose01/ueb/8/modul.zip (aufgerufen am 12.11.03)

bewusst für den Übungsbetrieb ausgewählt, da daran einerseits (ohne Kenntnis der Implementierung) unterrichtliche Einsatzmöglichkeiten erörtert werden konnten (Blatt 8, vgl. E.4) und andererseits die Implementierung reichlich Diskussionspotential in sich barg (Blatt 9, vgl. E.4). Für Informatiklehrende stellt es eine Basisqualifikation dar, einerseits von Lernenden erstellte Software-Bausteine auf ihre Korrektheit und ihre Angemessenheit hin zu analysieren und andererseits von anderen Informatiklehrenden erstellte Lern-Software zu bewerten im Hinblick auf ihre Unterrichtseignung bzw. auf die Anpassbarkeit an den eigenen Unterricht.
Die Übungen wurden vom Autor dieser Arbeit gehalten. Die Auswertung basiert auf den schriftlichen Lösungen von vier Lehramtsstudenten (alle männlich), ohne unterrichtspraktische Erfahrungen im Bereich der Informatik. An der 8. Übung nahmen die vier Studenten $M_{1,1}$ bis $M_{1,4}$ teil, an der 9. Übung die drei Studenten $M_{1,2}$ bis $M_{1,4}$. Die vollständigen Lösungen der Studenten befinden sich anonymisiert im Internet (vgl. Tabelle 33).

Student	Blatt	URL der Abgabe (alle geprüft am 12.11.03)
$M_{1,1}$	8	http://www.didaktik-der-informatik.de/DIE_BIB/lehre/ddi/ddi1/sose01/ueb/8/8m11.pdf
$M_{1,2}$	8	http://www.didaktik-der-informatik.de/DIE_BIB/lehre/ddi/ddi1/sose01/ueb/8/8m12.pdf
$M_{1,3}$	8	http://www.didaktik-der-informatik.de/DIE_BIB/lehre/ddi/ddi1/sose01/ueb/8/8m13.pdf
$M_{1,4}$	8	http://www.didaktik-der-informatik.de/DIE_BIB/lehre/ddi/ddi1/sose01/ueb/8/8m14.pdf
$M_{1,1}$	9	(keine Abgabe)
$M_{1,2}$	9	http://www.didaktik-der-informatik.de/DIE_BIB/lehre/ddi/ddi1/sose01/ueb/9/9m12.pdf
$M_{1,3}$	9	http://www.didaktik-der-informatik.de/DIE_BIB/lehre/ddi/ddi1/sose01/ueb/9/9m13.pdf
$M_{1,4}$	9	http://www.didaktik-der-informatik.de/DIE_BIB/lehre/ddi/ddi1/sose01/ueb/9/9m14.pdf

Tabelle 33: URLs der Studierendenlösungen zu Übung 8 und 9 (Didaktik der Informatik I)

Auswertung der Lösungen

Erörterung unterrichtlicher Einsatzmöglichkeiten (Blatt 8)

Zunächst (*Aufgabe 2*) sollten die Studenten analysieren, die Einführung welcher Basiskonzepte mittels der Software unterstützt werden kann. Tabelle 34 zeigt die Angaben der Studenten. Auffällig ist, dass nur die Konzepte „Objekt“ und „Botschaft“ bzw. „Nachrichtenaustausch“ (ergänzen sich in Tabelle 34) von allen Studenten erkannt wurden.

Basiskonzept	$M_{1,1}$	$M_{1,2}$	$M_{1,3}$	$M_{1,4}$
Aggregation	X	X	-	X
Assoziation	-	X	X	X
Attribut	X	X	-	X
Botschaft	-	X	-	-
Methode	X	X	-	X
Nachrichtenaustausch	X	-	X	X
Objekt	X	X	X	X
Objektorientierung	X	-	-	-
Parameter	-	-	-	X

Tabelle 34: Lösungen der Studierenden zu Blatt 8, Aufgabe 2, 1.Teil

Im zweiten Teil der Aufgabe sollte dargestellt werden, welche Erkenntnisprozesse mittels der Software unterstützt werden können. Die Lösungen hierzu waren vielfältig:

- Objektorientierte Repräsentation eines Realitätsausschnittes ($M_{1,1}$, $M_{1,2}$, $M_{1,3}$),
- Verknüpfung von Änderungen im Realitätsausschnitt und Nachrichtenaustausch zwischen Objekten ($M_{1,1}$, $M_{1,4}$),
- Aufteilung der Datenspeicherung zwischen Objekten ($M_{1,1}$),
- teilweise Förderung des Verständnisses für UML ($M_{1,1}$),
- Darstellung eines Objektdiagramms ($M_{1,3}$),
- Visualisierung von Objekt, Assoziation und Nachrichtenaustausch ($M_{1,3}$),
- technische Umsetzung der Manipulation von grafischen Objekten ($M_{1,3}$),

- Realisierung bestimmter Programmierdetails durch Analyse des Quellcodes ($M_{1,3}$),
- Was sind Attribute? ($M_{1,4}$),
- Zusammenhang zwischen Methoden und Attributwerten ($M_{1,4}$).

Dies sind alles korrekte Lösungen, wobei zu beachten ist, dass nur die ersten beiden Punkte von mehr als einem Student genannt wurden. $M_{1,4}$ gelangte ferner zu folgender Auffassung:

> „Das Lernmodul fördert das Gesamtverständnis der OOM. Da jeder Vorgang im Realitätsausschnitt (z.B. [...] verschieben) im Objektdiagramm nachzuvollziehen ist."

Da es sich nur um einen sehr kleinen Ausschnitt aus einem eingeschränkten Beispiel handelt, ist diese Aussage allerdings so nicht haltbar. Bemerkenswert ist ferner, dass die Objektdiagrammdarstellung nicht allen Studenten bekannt war:

> „Erkennbar sind all diese Prinzipien deutlich in dem [...] Diagramm, welches zwar einem Klassendiagramm enorm ähnelt, jedoch statt aus Klassen eindeutig aus Objekten besteht." ($M_{1,2}$)

Im Rahmen der *3. Aufgabe* ging es um den Entwurf von jeweils drei unterrichtlichen Einsatzszenarios für die Software. Tabelle 35 gibt einen Überblick über die Ergebnisse. Der Eintrag „k.A." in einer Tabellenzelle steht für „keine Angabe".

Student	Jgst.	GK/LK	Unterrichtsreihe	Thema der Einzelstunde
$M_{1,1}$	11	k.A.	Einführung in das OOM	Klasse und Objekt
	k.A.	k.A.	k.A.	Kommunikation von Klassen untereinander
	k.A.	k.A.	k.A.	Sequenzdiagramme
$M_{1,2}$	11	GK	Einführung in die objektorientierte Modellierung	Eine erste Betrachtung von Methodenaufruf und Zusammenspiel verschiedener Objekte innerhalb eines Programms – also innerhalb von Software
	12	LK	Einführung in die objektorientierte Modellierung	Heranführung an den Objektbegriff und einige von dessen grundlegenden Eigenschaften
	13	LK	Erstellung eines Grafikprogramms in der objektorientierten Programmiersprache Java	Einführung in die „Darstellung" von Grafikobjekten mittels Objektorientierung
$M_{1,3}$	11	GK	Einführung in die objektorientierte Modellierung	Nachrichtenaustausch in einem objektorientierten Modell
	13	LK	Objektorientierte Programmierung mit Java	Behandlung von Ausnahmen in Java
	13	LK	Objektorientierte Programmierung mit Java	Sicherung des Umgangs mit Dateien und dem Lernmodul „Vom Realitätsausschnitt zum Objektdiagramm"
$M_{1,4}$	11	k.A.	Grundlagen der OOM	Objekte
	11	k.A.	Grundlagen der OOM	Beziehungen
	11	k.A.	Grundlagen der OOM	Nachrichtenaustausch

Tabelle 35: Lösungen der Studierenden zu Blatt 8, Aufgabe 3

Die Mehrzahl der entworfenen Szenarien bezieht sich auf eine Einführungssequenz in die objektorientierte Modellierung, aber auch fortgeschrittene Möglichkeiten, bei denen die Software als zu analysierende bzw. zu erweiternde Grundlage für weitere Arbeitsaufträge genutzt wird, wurden berücksichtigt ($M_{1,3}$). Die Studenten boten also von sich aus Einsatzszenarien mit Explorationsmodulen als Medien bzw. als Unterrichtsinhalt an (vgl. 5.4.1). Die Beschreibungsqualität der Szenarien war sehr unterschiedlich und variierte zwischen ein paar Sätzen pro Szenario ($M_{1,1}$) bis hin zu einer sehr ausführlichen Darstellung aller geforderten Angaben ($M_{1,2}$). Bei $M_{1,1}$ zeigten sich Schwierigkeiten bei der Definition von Lernzielen:

> „Das Lernziel könnte in diesem Fall darin bestehen, dass herausgestellt werden soll, dass z.B. Rechtecke unabhängig von ihrer Form und Größe von ihrer Konzeption her immer gleich sind und sich diese Unterschiede allein durch Attribute definieren lassen."

Das eigentliche Lernziel ist implizit in dieser Aussage enthalten. Ferner zeigten sich bei ihm Schwierigkeiten mit der Unterscheidung zwischen Objekt- und Klassendiagramm und der Zuordnung der Kommunikation zu Objekten bzw. Klassen (vgl. auch Tabelle 35, Zeile 3):

> „Eine weitere Einsatzmöglichkeit bestünde darin, das Verständnis der Kommunikation der Klassen untereinander zu untermauern. In dieser Software wird sie in ein Klassendiagramm eingebaut und auch nur in Einzelschritten angezeigt, was der Übersichtlichkeit und dem Verständnis sehr zu gute kommt. Besonders, dass keine neue Darstellungsform, sondern das normale Klassendiagramm verwendet wird, macht die Zusammenhänge in diesem Fall deutlicher."

Bei *Aufgabe 4* waren alle vier Studenten der Auffassung, dass die Software sowohl Stärken (+) als auch Schwächen (–) im Hinblick auf einen Unterrichtseinsatz besitzt (vgl. Tabelle 36).

	Eigenschaft der Software im Hinblick auf Unterrichtseinsatz	**$M_{1,1}$**	**$M_{1,2}$**	**$M_{1,3}$**	**$M_{1,4}$**
I.	Attraktivität durch Interaktivität	+			
	Schrittweises Durchlaufen des Nachrichtenaustausches	+			
	OOM- / UML-Vorkenntnisse erforderlich	–			
	eingeschränkter Funktionsumfang	–	–		–
	Intuitivität durch Interaktivität		+		
	Übertragbarkeit der Erkenntnisse durch Exemplarität und Einfachheit des Moduls		+		
	Arbeit mit dem Modul animiert zum eigenen Programmieren			+	
	Erklärungsbedürftigkeit			–	
II.	Quellcode kann als Vorlage für eigene Programme dienen			+	
	„schlechter" Quellcode als Vorlage für eigene Programme			–	
III.	Versuch-Irrtum-Strategien statt zielgerichtetes Explorieren	–		–	
	Förderung selbsttätigen Erkundens / Experimentierens, bessere Behaltensleistung		+	+	+
	Motivationssteigerung durch „persönliche Bindung" zu „selbst Erstelltem"		+		
	Interesse der Lernenden steigt durch Praxisnähe beim Umgang mit dem Modul			+	
	Abwechslung zum Alltagsunterricht			+	
	Eignung für Einsatz in Gruppenprojekten			+	
	weniger Kommunikation zwischen Lehrendem und Lernenden				–/+
	Förderung der Binnendifferenzierung				+

Tabelle 36: Lösungen der Studierenden zu Blatt 8, Aufgabe 4

Die diskutierten Eigenschaften beziehen sich auf eine Anwendungssicht auf die Software (I.), auf deren Implementierung (II.) und auf Eigenschaften bzw. Gestaltungselemente des Lehr-Lern-Prozesses (III.).
Es zeigt sich hier, dass diese Studierenden prinzipiell dazu in der Lage sind, Unterrichtsszenarios und unterrichtliche Einsatzmöglichkeiten für eine gegebene Software zur Exploration zu entwerfen. Schwierigkeiten resultierten vermutlich aus fachlichen Problemen zur objektorientierten Modellierung und fehlender, eigener Unterrichtserfahrung.

Erörterung der Implementierung (Blatt 9)

Gegenstand der *2. Aufgabe* war die Analyse des Quelltextes unter softwaretechnischen Gesichtspunkten. Als wesentliche Schwächen (–) wurden die mangelnde Objektorientierung, die Verstöße gegen das Prinzip der Kapselung durch direkten Zugriff auf Attribute fremder Klassen sowie der Widerspruch zwischen der Darstellung an der Benutzungsoberfläche und der internen Realisierung insbesondere vor dem intendierten Zweck als Lernmodul identifiziert (vgl. Tabelle 37). Zur mangelnden Objektorientierung bemerkte $M_{1,2}$:

> „In diesem Zusammenhang sollte vor allem erwähnt werden, dass die Software im Allgemeinen auf mich eher einen ‚imperativen' als einen objektorientierten Eindruck machte."

Als Stärke (+) wurde von allen Studenten die gute Lesbarkeit des Quelltextes herausgestellt.

Eigenschaften der Quelltexte	$M_{1,2}$	$M_{1,3}$	$M_{1,4}$
mangelnde Objektorientierung	–		–
Verstoß gegen das Prinzip der Kapselung (direkter Zugriff auf Attribute von „außen“)	–	–	
an der Benutzungsoberfläche vorhandene Klassen (Rechteck, Punkt, ...) existieren in der Implementierung nicht: mangelnde Klassenbildung	–	–	–
Kommentierung fehlt oder zu knapp	–		
ausreichende Kommentierung		+	+
Grundstruktur des Aufbaus der GUI: Mainframe mit zwei Panels	+		
Startklasse zum Zweck der Instanziierung von Mainframe	+		
gute Lesbarkeit der Quelltexte: Berücksichtigung von Einrückungen, Notationskonventionen	+	+	+
häufige Verwendung von Java-Standardfunktionen			–/+

Tabelle 37: Lösungen der Studierenden zu Blatt 9, Aufgabe 2

Die Übersichtlichkeit, Wiederverwendbarkeit und Erweiterbarkeit der Quelltexte wurde von den Studenten (insb. $M_{1,2}$ und $M_{1,3}$) vor dem Hintergrund der identifizierten Schwächen und Stärken diskutiert und bewertet. $M_{1,2}$ gelangte zu einer insgesamt leicht negativen, aber sehr differenziert dargelegten, Haltung bezüglich der genannten Kriterien. $M_{1,3}$ begründete seine negative Haltung anhand der von ihm identifizierten Schwächen. $M_{1,4}$ nahm keine direkte Bewertung im Hinblick auf die Kriterien vor.
Im Rahmen von *Aufgabe 3* sollten die Studenten begründen, ob sich die Quelltexte dazu eignen, um mit Lernenden daran Prinzipien und Methoden des objektorientierten Programmierens zu erarbeiten. Ferner sollte eine Position zur Analyse der Realisierung fertiger Systeme im Informatikunterricht bezogen werden. Alle drei Studenten hielten die Quelltexte für nicht oder nur bedingt geeignet, um Prinzipien und Methoden des OOP daran zu erarbeiten. Begründet wurde dies zu großen Teilen mit den in Aufgabe 2 (s.o) beschriebenen, negativen Eigenschaften der Quellen. Insbesondere die mangelnde Objektorientierung und die Verletzung des Prinzips der Kapselung wurden hierzu herausgestellt (s.a. Tabelle 38, I.). Zu einem späteren Zeitpunkt, d.h. deutlich nach der Erarbeitung, sei es möglich, Fehler bzw. Defizite exemplarisch anhand der Quellen zu diskutieren. Weitere Argumente der Studenten für (+) bzw. gegen (–) eine Eignung der Quellen zeigt Tabelle 38.

Eigenschaft		$M_{1,2}$	$M_{1,3}$	$M_{1,4}$
I.	durchdachte Klassenstruktur fehlt		–	
	„schlechter“ Programmierstil		–	
II.	Komplexität der Software	–	–	
III.	Verwendung vieler Standard-Grafikroutinen von Java	–		
	GUI-zentrierte Programme eignen sich für die Erarbeitung von OO-Prinzipien und -Methoden prinzipiell schlecht			–
IV.	prinzipielle Realisierung einer solchen Software erkunden		+	
	Nachrichtenaustausch und Parameterübergabe gut erkennbar			+

Tabelle 38: Teillösungen der Studierenden zu Blatt 9, Aufgabe 3

Hervorhebenswert sind zwei Aussagen von $M_{1,3}$:

> „Da Java kaum eine andere Programmierform bietet, als die objektorientierte, kann jedes darin verfasste Programm den Schülern einen Einblick in die Welt der Objektorientierung gewähren.“

> „Außerdem denke ich, dass ein Informatiklehrer seine Schüler lieber eigene Programme schreiben lässt, als sie einen fremden Quelltext analysieren zu lassen, was eine ganze Reihe von Stunden kosten und unter Umständen wenig Früchte tragen kann.“

Selbstverständlich ist Java eine OOP-Sprache. Die korrekte Verwendung relevanter Prinzipien der Objektorientierung (Klassenbildung etc.) wird ermöglicht, aber nicht oder nur zu gewissen Teilen erzwungen. Dieses Missverständnis war speziell in der Anfangsphase des Paradigmenwechsels unter Informatiklehrenden nicht nur auf Schulebene weit verbreitet (vgl.

2.3.1.2 und Mitchell 2000). Die Aussage bzgl. der Informatiklehrenden ist ebenfalls gewagt. Sowohl eigene Gestaltung als auch Analysephasen haben im Unterricht ihren festen Platz. Wie ertragreich sowohl Gestaltung als auch Analyse sind, hängt nicht von der Entscheidung für eine dieser beiden Schwerpunktsetzungen ab, sondern von anderen Parametern, wie Vorwissen der Lernenden, Geschick des Lehrenden, etc.
Zur „Analyse der Realisierung fertiger Systeme" äußerten sich nur $M_{1,2}$ und $M_{1,4}$. $M_{1,2}$ bewertete es als „sehr gute Ergänzung zum aktiven Programmieren, sofern der verwendete Quellcode entsprechende Qualitäten besitzt". Vorteile sah er für Lernende mit Schwierigkeiten bei der eigenständigen Programmierung. Ferner werde durch fertige Quellen der Realitätsbezug der theoretischen Konzepte hergestellt. Zu beachten sei aber, die Quelltextportionen am Anfang nicht zu groß zu wählen. $M_{1,4}$ sah die wesentlichen Vorteile in der Förderung abstrakten Denkens sowie in der Veranschaulichung von wichtigen Programmieraspekten. Zu analysierende Quellen sollten in der Anfangsphase aber eher dem Fachkonzept als der Benutzungsschnittstelle entstammen. Ferner sei eine Analyse gut arbeitsteilig in Kleingruppen zu erledigen, was Gruppenarbeitsprozesse fördere.
Es zeigt sich auch hier, dass die Studierenden dazu in der Lage waren, die Realisierung der Software zu analysieren, wesentliche softwaretechnische Stärken und Schwächen zu identifizieren und daraus Schlussfolgerungen für eine unterrichtliche Einsetzbarkeit zu ziehen.

Entwurf von Unterrichtsbeispielen unter Verwendung von Explorationsmodulen, Spezifikation von Anforderungen an Explorationsmodule

Konzeption

Diese Erprobung wurde durchgeführt im Rahmen der begleitenden Übung zur Hauptstudiumsvorlesung „Didaktik der Informatik II" an der Universität Dortmund im Wintersemester 2001/2002, wie schon die Erprobung zu „Aufgabenklassen" (vgl. 4.6.1.1). Die Übung wurde durchgeführt unter denselben Rahmenbedingungen und mit denselben Studierenden $M_{2,1}$, $M_{2,2}$, $W_{2,1}$ und $W_{2,2}$, wie im Abschnitt 4.6.1.1 beschrieben. Die hier beschriebenen Ergebnisse beziehen sich auf das vierte Übungsblatt zur oben genannten Lehrveranstaltung (vgl. E.4). Erprobt wurde, inwieweit Informatiklehramtstudierende einerseits dazu in der Lage sind, Informatikunterrichtsbeispiele unter Verwendung von Explorationsmodulen zu entwerfen und andererseits Anforderungen an Explorationsmodule vor dem Hintergrund eines Unterrichtseinsatzes zu spezifizieren. Die vollständigen Lösungen der Studierenden befinden sich anonymisiert im Internet (vgl. Tabelle 39).

Studierender	URL der Abgabe (alle geprüft am 25.11.03)
$M_{2,1}$	http://www.didaktik-der-informatik.de/DIE_BIB/lehre/ddi/ddi2/wise0102/ueb/4/4m21.pdf
$M_{2,2}$	http://www.didaktik-der-informatik.de/DIE_BIB/lehre/ddi/ddi2/wise0102/ueb/4/4m22.pdf
$W_{2,1}$	http://www.didaktik-der-informatik.de/DIE_BIB/lehre/ddi/ddi2/wise0102/ueb/4/4w21.pdf
$W_{2,2}$	http://www.didaktik-der-informatik.de/DIE_BIB/lehre/ddi/ddi2/wise0102/ueb/4/4w22.pdf

Tabelle 39: URLs der Studierendenlösungen zu Übung 4 (Didaktik der Informatik II)

Auswertung der Lösungen

Ziel von *Aufgabe 2* war es, jeweils drei Unterrichtsbeispiele zu entwerfen, in denen Explorationsmodule zur Anwendung kommen. Die Lösungen der Studierenden gingen leicht an dieser Aufgabenstellung vorbei, indem sie im Wesentlichen jeweils drei Explorationsmodule beschrieben, deren unterrichtliche Einbettung aber nur andeuteten. Tabelle 40 gibt einen Überblick darüber, welche Explorationsmodule die Studierenden vorschlugen. Sofern in den jeweiligen Lösungen angegeben (sonst „k.A. – keine Angabe) wird mit aufgeführt, welche Eingaben Lernende machen können und welche Ausgaben vom System zu erwarten sind.
Die Lösungsvorschläge sind nach ihren Interaktionsmöglichkeiten in Gruppen eingeteilt (Spalte „Gr."). Die Vorschläge der Gruppe 1 bieten als einzige Interaktionsmöglichkeit die Übergabe bzw. Veränderung von Parametern an.

Gr.	Stud.	Explorationsmodul	Eingaben des Lernenden	Ausgaben des EM
0	$W_{2,1}$	CSCL[94]-System	k.A.	k.A.
	$W_{2,1}$	Sicherheit in Rechnernetzen (Firewall, FTP-Dienst, Hackerangriffe)	k.A.	k.A.
1	$M_{2,2}$, $W_{2,1}$	Visualisierung von Algorithmen	• Geschwindigkeit der Ausführung • Ausführungsschritt	• Visualisierung der Ausführung des Algorithmus
	$W_{2,2}$	Rekursion	• Modifikation von Parametern	• rekursives Zeichnen von Figuren • Markierung des Schritts im Programm
	$M_{2,1}$	Funktionsweise einer Internet-Suchmaschine	• Suchanfrage	• Verbesserungsvorschläge für die Anfrage • Visualisierung der Datenwege
	$W_{2,2}$	Verschlüsselungsverfahren	• Text • Schlüssel	• Code
	$M_{2,1}$, $M_{2,2}$	von-Neumann-Rechnerkonzept	• (kleines) Assembler Programm	• Betrachtung der Verarbeitung auf verschiedenen Detailebenen • Datenaustausch zwischen Komponenten
2	$M_{2,1}$	Rechner-Kommunikation	• Manipulation des Netzwerks (Leitungen kappen) • Paket verzögern • Veränderungen des Protokolls	• Beobachtung der Paketvermittlung
	$M_{2,2}$	Mealy-/Moore-Automat	• Automat aufbauen und manipulieren	• Darstellung des Automaten • Zustandsübergänge beobachten

Tabelle 40: Lösungen der Studierenden zu Blatt 4, Aufgabe 2

Bei den Vorschlägen der Gruppe 2 ist zudem eine Strukturänderung möglich, was mit erhöhten Explorationsmöglichkeiten korrespondiert (vgl. 5.3.1.2). Bei den zwei Vorschlägen der Gruppe 0 fehlten Angaben bzgl. Ein- bzw. Ausgaben des Systems. Je nach Realisierung ist der zweite Vorschlag der Gruppe 1 oder 2 zuzuordnen. Ein CSCL-System an sich ist eine Anwendungssoftware und kein Explorationsmodul. Ein Explorationsmodul zu CSCL-Systemen wäre aber denkbar.

In *Aufgabe 3* waren alle Studierenden der Auffassung, dass sie an EMs die gleichen softwareergonomischen Anforderungen stellen, wie an andere benutzungsfreundliche Software. Betont wurde die Berücksichtigung von Standards zur GUI-Gestaltung auch für diese Art von Software. Die Förderung von Kooperation und Kommunikation ist nach Auffassung von $M_{2,1}$ durch die Realisierung von EMs als verteiltes System möglich. $M_{2,2}$ fordert die Unterstützung von Persistenz, um über Dateiaustausch Kooperation und Kommunikation zu fördern. $W_{2,1}$ stellte heraus, dass kooperations- bzw. kommunikationsförderliche Software notwendig aber nicht hinreichend für die tatsächliche Erreichung dieser Ziele im Lehr-Lern-Prozess sei:

> „Durch die Art und Weise des Einsatzes kann sie jedoch kommunikationsförderlich wirken. Ich stelle hier also keine Anforderungen an Softwareentwickler, sondern an Informatiklehrende und solche, die es werden wollen, also z.B. an mich." ($W_{2,1}$)

Das korrespondiert mit dem Sachverhalt, dass die Qualität eines Lernmediums erst im Rahmen des kommunikativen Zusammenhangs, in dem es eingesetzt wird, bewertet werden kann (vgl. 5.4.1). Bei der Diskussion der Einsatzbreite schlugen $M_{2,1}$ die Berücksichtigung mehrerer Komplexitätsstufen und $M_{2,2}$ eine plattformunabhängige Implementierung vor, um die jeweiligen Möglichkeiten zu maximieren. $W_{2,1}$ hingegen sah in der geringen Einsatzbreite kein Problem, forderte aber eine Serie von Modulen mit ähnlicher Benutzungsstrategie zu

[94] CSCL – computer supported co-operative learning

verwenden, um sich nicht immer wieder in konzeptionell andere Benutzungsschnittstellen einarbeiten zu müssen.
Die genannten Anforderungen werden im Konzept für Explorationsmodule berücksichtigt (vgl. 5.3.3). Bei $W_{2,1}$ zeigten sich Missverständnisse bzgl. des Wesens von Explorationsmodulen. Es zeigte sich, dass diese Studierenden überwiegend dazu in der Lage waren, eigene Software zur Exploration zu entwerfen, dass sie Interaktionsmöglichkeiten hierbei aber nicht immer in den Vordergrund stellten. Hieraus ergeben sich Anforderungen an die Informatiklehrerbildung, in denen dies entsprechend zu betonen ist (vgl. 3.7).

5.5.3.2 Lehrerfortbildungen

Im Rahmen der im Kontext dieses Forschungsprojekts durchgeführten Lehrerfortbildungen (vgl. Tabelle 23 auf S. 104) wurde die entwickelte Software zur Exploration von OOM vorgestellt. Nachfragen der Lehrenden während und nach den Veranstaltungen zeigten das große Interesse. Zusätzlich zu den genannten Veranstaltungen fand am 23.07.2003 an der Universität Dortmund im Rahmen des 1. Informatiktages NRW ein Workshop zum Thema „LEO - eine Lernumgebung für objektorientierte Basiskonzepte“ statt[95], an dem ca. 30 Informatiklehrende aus NRW teilnahmen. Die Veranstaltung wurde im Wesentlichen gestaltet von Mitgliedern der studentischen Projektgruppe, die LEO entwickelte (Alex et al. 2002). Gegeben wurde ein kurzer Entwicklungsbericht, an den sich eine ausführliche Präsentation anschloss. Die Nachfragen der Lehrenden konzentrierten sich auf die Entwicklung und Integration eigener Szenarien. Im Projektbericht (Alex et al. 2002) bzw. im Handbuch ist ausführlich dokumentiert, wie dies möglich ist.

5.5.4 Informatikstudium

Entwicklung von Software zur Exploration

Im Rahmen der Projektgruppe „LEO“ entwickelten elf Informatikstudierende an der Universität Dortmund eine Software zur Exploration von objektorientierten Fachkonzepten (vgl. 5.3.4.4, Alex et al. 2002). Einen Ausgangspunkt bildete die Analyse bestehender Software zur Modellierung und zum Erlernen von OOM. Dies korrespondiert mit der Komponente „Informatiksystem“ des Lernkonzepts (vgl. 5.4), bei der bestehende Software als Ausgangspunkt für Modifikationen bzw. eigenen Entwicklungen verwendet wird. Die individuellen Fazits der Teilnehmerinnen und Teilnehmer zum Projekt (ebd., 105ff) zeigen, dass die Projektarbeit generell positiv bewertet wurde. Offen ist eine diesbezügliche Erprobung mit Informatik*lehramts*studierenden. Sowohl in Dortmund als auch in Siegen ergab sich hierzu bis 2003 keine Gelegenheit. An der Universität Dortmund können Lehramtsstudierende im Hauptstudium ein Vertiefungsgebiet wählen, in dem sie dann eine Projektarbeit erstellen. „Didaktik der Informatik“ konkurriert hierbei mit anderen möglichen Themen. Bedingt dadurch fanden bis 2002 Projektarbeiten maximal als Partnerarbeiten statt (z.B. Koch / Ortmann 2002). An der Universität Siegen erreichen die ersten Informatiklehramtsstudierenden ihr Hauptstudium. Offen ist weiterhin die Erprobung der Entwicklung von Software zur Exploration mit Schülerinnen und Schülern. In Relation zu LEO wäre hierbei aber nur ein *deutlich* kleineres Entwicklungsergebnis möglich.

Analyse und Weiterentwicklung von Software zur Exploration

Im Rahmen der Vorlesung „E-Learning“ an der Universität Siegen hatten 12 Studierende der Angewandten Informatik mit Anwendungsfach „Medienwissenschaften“ (Medieninformatik)

[95] URL: http://www.die.informatik.uni-siegen.de/dortmund2002/nrw/ (aufgerufen am 12.11.03)

im WiSe 2002/2003 den Auftrag, im Team, im Rahmen der begleitenden Übung, jeweils einen von vier vorgegebenen Analyseaufträgen zum LEO-System zu bearbeiten. Voran gingen der Übung drei Vorlesungseinheiten zur Konzeption und zur Realisierung von LEO. Die Aufträge waren:

- Welche Konzepte lassen sich mit LEO explorieren? Wie kann LEO erweitert / modifiziert werden, so dass weitere Konzepte explorierbar werden oder die Exploration / Visualisierung bereits realisierter Konzepte verbessert wird? (Team 1)
- LEO stellt den Nachrichtenaustausch zwischen Objekten in einem generierten Sequenzdiagramm dar. Eine andere, weit verbreitete Darstellungsform ist die des Kollaborationsdiagramms. Wie kann eine solche Sicht (evtl. unter Verwendung der Objektsicht) in LEO integriert werden? (Team 2)
- Wie kann LEO erweitert werden, so dass die Quelltexte von Klassen / Methoden / Konstruktoren an (allen) geeigneten Stellen angezeigt werden können? Lassen sich dabei auch verschiedene Programmiersprachen unterstützen? (Team 3)
- Wie könnte durch LEO / durch ein Begleitwerkzeug die Entwicklung eigener Szenarien vereinfacht / teilautomatisiert werden? (Team 4)

Auch diese Erprobung korrespondiert mit der Komponente „Informatiksystem“ des Lernkonzepts (vgl. 5.4). Es zeigte sich, dass die Studierenden auf der Basis der vorliegenden Dokumente (LEO-System, Analyse- und Entwurfsdokumente, Quelltexte) dazu in der Lage waren, eine Strategie für die Bearbeitung der Aufgaben zu entwickeln. Die vollständigen Lösungen der Studierenden befinden sich anonymisiert im Internet (vgl. Tabelle 41).

Team	URL der Abgabe (alle geprüft am 12.11.03)
1	http://www.didaktik-der-informatik.de/DIE_BIB/lehre/ele/ele/wise0203/ueb/leo/team1.pdf
2	http://www.didaktik-der-informatik.de/DIE_BIB/lehre/ele/ele/wise0203/ueb/leo/team2.pdf
3	http://www.didaktik-der-informatik.de/DIE_BIB/lehre/ele/ele/wise0203/ueb/leo/team3.pdf
4	http://www.didaktik-der-informatik.de/DIE_BIB/lehre/ele/ele/wise0203/ueb/leo/team4.pdf

Tabelle 41: Lösungen der Studierenden zu den LEO-Analyseaufträgen

Auch hierzu ist die Erprobung mit Lehramtsstudierenden und Lernenden in der Sekundarstufe II noch offen (s.o).

5.6 Zusammenfassung, Fazit und offene Fragen

5.6.1 Zusammenfassung

Bedarf

Den Ausgangspunkt für die Entwicklung und Erprobung von Explorationsmodulen bildete die Begründung des Bedarfs für unterrichtsgeeignete Informatiksysteme, mit denen sich Handlungsorientierung und explorative Lernstrategien beim Aneignen von Basiskonzepten des objektorientierten Modellierens besser verwirklichen lassen als dies mit zu diesem Zweck oftmals eingesetzten, professionellen Software-Entwicklungsumgebungen gelingt (vgl. 2.4.2, Punkt 3). Dargelegt wurde, wie Explorationsmodule für Lehrende und Lernende zu einer Bereicherung traditionellen Unterrichts beitragen können (vgl. 3.4.3, 5.2).

Exploratives Lernen (mit Informatiksystemen)

Bei der Entwicklung eines Konzeptes für Explorationsmodule (vgl. 5.3) wurde begonnen mit der Analyse von Grundlagen aus dem Bereich des explorativen Lernens (vgl. 5.3.1.1). Untersucht wurden Formen und Grundmuster sowie Anhaltspunkte zu Explorationsstrategien.

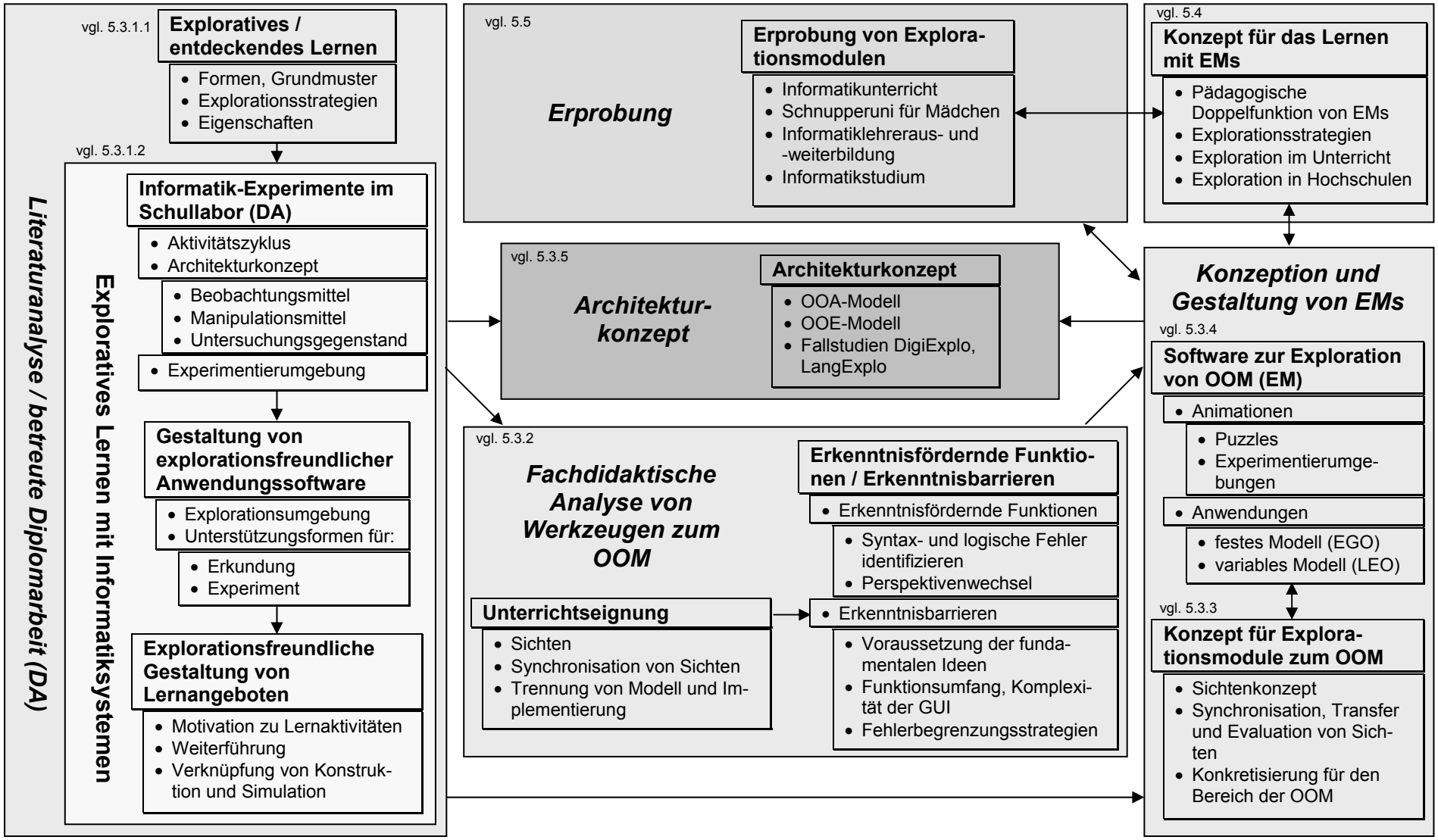


Abbildung 22: Vorgehensweise bei der Entwicklung und Erprobung von Explorationsmodulen zum OOM

Zeitlich parallel hierzu wurden Erkenntnisse zum explorativen Lernen mit Informatiksystemen eingehend analysiert (vgl. 5.3.1.2). Den Auftakt hierzu bildete eine, vom Autor der vorliegenden Arbeit betreute, Studie zu Informatik-Experimenten, als deren Ergebnis exemplarisch nachgewiesen werden konnte, dass ein handlungsorientierter Zugang zu Wirkprinzipien von Informatiksystemen möglich ist, der auf Programmierung verzichtet. Des Weiteren wurden Erkenntnisse zur explorationsfreundlichen Gestaltung von Anwendungssoftware und Lernangeboten im Hinblick auf die Gestaltung von Software zur Exploration objektorientierter Fachkonzepte zur Einbettung in traditionelle Lehr-Lern-Prozesse im Sinne von „Blended Learning"-Szenarios untersucht.

Fachdidaktische Analyse von Werkzeugen zum objektorientierten Modellieren

Mit dem Ziel der Entwicklung eines Gestaltungskonzeptes für explorative Lernangebote für das Themenfeld objektorientiertes Modellieren wurden professionelle OOM-Werkzeuge analysiert im Hinblick auf ihre Unterrichtseignung und damit verknüpft auf erkenntnisfördernde und erkenntnishemmende Eigenschaften (vgl. 5.3.2).
Als besonders förderlich im Hinblick auf die Unterrichtseignung erwiesen sich die *Bereitstellung vielfältiger Sichten auf ein Modell*, die Verknüpfung bzw. *Synchronisation von Modellsichten* und die *Trennung zwischen Modell und Implementierung*.
Bei den erkenntnisfördernden Funktionen wurden die Möglichkeiten zur *Identifikation von syntaktischen und logischen Fehlern* sowie die *Visualisierung abstrakter Konzepte mit Wechsel des Beobachtungsschwerpunktes* herausgestellt. Als potentiell problematisch für Lehr-Lern-Prozesse erwiesen sich die *Voraussetzung der fundamentalen Ideen* des Fachgegenstandes aufgrund der vom Einsatz in schulischen Lehr-Lern-Prozessen signifikant abweichenden *Zielsetzung von Modellierungswerkzeugen*, der daraus resultierende *Funktionsumfang* und die *Komplexität der Benutzungsschnittstelle* sowie die aus Rationalisierungserwägungen in diesen Werkzeugen implementierten *Fehlerbegrenzungsstrategien*.

Konzeption und Implementierung von Explorationsmodulen

Auf der Basis der genannten Vorarbeiten wurde ein Konzept für Explorationsmodule zum OOM entwickelt (vgl. 5.3.3). Damit Lernende einerseits zu aktivem Lernen, andererseits zur individuellen Einnahme vielfältiger Perspektiven auf den Lerngegenstand angeregt werden, bildete den Schwerpunkt die Entwicklung eines Sichtenkonzepts, das Lernenden die Interaktion mit einem Lerngegenstand unter vielfältigen Blickwinkeln ermöglicht. Da diese Sichten synchronisiert sind, können die Lernenden die Auswirkungen ihres Handels gleichzeitig aus verschiedenen Perspektiven verfolgen und damit zu weiteren Handlungen auf dem Weg zu einem individuellen Lernziel stimuliert werden. Die Synchronisation erleichtert es den Lernenden, verschiedene Einzelsichten auf den Lerngegenstand zu einem mentalen Gesamtbild zu verknüpfen, eine erhebliche kognitive Anforderung bei der Arbeit mit statischen Diagrammen (z.B. an der Tafel, auf Papier oder in einem Modellierungswerkzeug). Im Mittelpunkt hierbei stand die Frage, welche Sichten auf den Fachgegenstand angeboten und wie diese miteinander verknüpft werden sollten. Begründet wurde bei der Sichtengestaltung eine Verknüpfung von Etappen des Gestaltungsprozesses (*Realitätsausschnitt – Modell – Produkt*) mit der Unterscheidung von *Statik* und *Dynamik*. Das allgemeine Sichtenkonzept wurde für den Bereich des objektorientierten Modellierens konkretisiert und dargelegt, dass sich mit der Sichtenauswahl *Real-*, *Objekt-*, *Klassen-*, *Sequenz-* und *Programmsicht* bereits wesentliche Bildungsziele des Informatikunterrichts wirkungsvoll unterstützen lassen.
Verschränkt mit der Entwicklung der Konzeption entstanden verschiedene prototypische Implementierungen (vgl. 5.3.4). Entwickelt wurden Animationen mit unterschiedlich ausgeprägten Interaktionsmöglichkeiten (Filme mit Stopppunkten, Puzzles, Experimentierumgebungen und grafische Editoren, 5.3.4.2). Dargelegt wurde, dass insbesondere die Experimentierumgebungen dem hier vorgestellten Explorationsmodulansatz nahe kommen. Des Weiteren ent-

standen zwei prototypische Explorationsmodule (Explorer für geometrische Objekte – EGO, Lernumgebung für objektorientiertes Modellieren im Informatikunterricht – LEO, vgl. 5.3.4.3, 5.3.4.4). Analysiert wurde jeweils das Erweiterungs- und Verallgemeinerungspotential der Lösungen. Zur Rationalisierung der Entwicklung von Explorationsmodulen entstand im Rahmen einer, vom Autor der vorliegenden Arbeit betreuten, Diplomarbeit ein Architekturkonzept für Software zur Exploration im Bildungskontext, welches nicht auf den Bereich des objektorientierten Modellierens beschränkt ist (vgl. 5.3.5).

Konzept für das Lernen mit Explorationsmodulen

Da die Beurteilung der didaktischen Qualität von Lernmedien generell nur im Zusammenhang mit Lehr-Lern-Prozessen möglich ist, wurde ein Konzept für das Lernen mit Explorationsmodulen entwickelt (vgl. 5.4). Herausgestellt wurde die pädagogische Doppelfunktion von Explorationsmodulen als Unterrichtsmedien bzw. als Unterrichtsinhalt, je nach Gestaltung der Lehr-Lern-Prozesse (vgl. 5.4.1). Explorationsstrategien für objektorientiertes Modellieren wurden präzisiert (vgl. 5.4.2). Gezeigt wurde, welche Selbstständigkeitsstufen Lernende beim „Erlernen des explorativen Lernens“ durchlaufen können. Es wurde dargelegt, wie für mit systematischen Explorationsstrategien unerfahrene Lernende Explorationsprozesse anhand von Leitfragen in geschlossenen Explorationsumgebungen vorstrukturiert werden können. Exemplarisch erfolgte eine Verknüpfung mit den Handlungsphasen problemorientierten Unterrichts (vgl. 5.4.3).

Erprobung von Explorationsmodulen in der informatischen Bildung

Verschiedene Erprobungen von Konzeptausschnitten wurden im Abschnitt 5.5 vorgestellt und ausgewertet. In der Sekundarstufe wurden die Puzzles erprobt (vgl. 5.5.2). Es zeigte sich, dass die Einbindung in den Unterricht bei entsprechender Vorkenntnis der Lerngruppe leicht möglich ist. Identifizierte Schwächen bei den Lernenden führten zur Begründung weiterer Darstellungsmöglichkeiten. Im Hinblick auf LEO zeigte sich der besondere Gewinn in der Anfängerausbildung. Im Bereich der Informatiklehrerausbildung (vgl. 5.5.3.1) konnten unterrichtliche Einsatzmöglichkeiten und Implementierungen von Explorationsmodulen erfolgreich mit Lehramtsstudierenden analysiert werden. In Lehrerfortbildungen (vgl. 5.5.3.2) und auf Fachtagungen wurde die gesamte, im Rahmen dieser Arbeit entwickelte, Software mehrfach präsentiert. Es zeigte sich eine breite Akzeptanz und ein hohes Interesse.

5.6.2 Fazit

Zu den wissenschaftlichen Fragestellungen

Bis 2003 finden sich kaum Publikationen zur Förderung des explorativen Lernens mittels geeigneter Informatiksystemunterstützung im Bereich des objektorientierten Modellierens im Informatikunterricht. Vorhandene Angebote im Bereich der Hochschulbildung konzentrieren sich vorrangig auf statische Modellaspekte (vgl. 2.3.2). In den Ausführungen zu Explorationsmodulen wurde ausgehend von der Begründung des Bedarfs (vgl. 2.4.2, Punkt 3) ein Konzept für die Gestaltung entsprechender Software und damit verbundener Lehr-Lern-Prozesse vorgestellt. Die vom Bedarf ausgehende Darlegung des Gewinns für verschiedene, am Lehr-Lern-Prozess beteiligte, Akteure (vgl. 3.4.3, 5.2) lieferte eine Begründung für die Einbeziehung von Explorationsmodulen als Komponente des didaktischen Systems (vgl. 2.4.3, Frage 1).

Die Vorgehensweise bei der Entwicklung von Gestaltungs- und Lernkonzept wurde ausführlich dargelegt. Begründet wurde, wie aus der Verknüpfung von Erkenntnissen zum explorativen Lernen, mit und ohne Informatiksysteme, und der fachdidaktischen Analyse von professionellen Werkzeugen Gestaltungskriterien für geeignete Lernhilfen abgeleitet werden kön-

nen. Erfolgreiche, prototypische Implementierungen von Explorationsmodulen belegten die Tragfähigkeit des Gestaltungskonzepts. Durch die Diskussion der Verallgemeinerbarkeit der realisierten Lösungen sowie durch die Bereitstellung und Erprobung eines Architekturkonzeptes wurden Belege für die Übertragbarkeit des Ansatzes auch in andere Themengebiete geliefert. Erfolgreiche Umsetzungen von Ausschnitten des Lernkonzeptes in der informatischen Bildung führten zu Belegen für die Tragfähigkeit und Akzeptanz auch dieses Konzeptes. Durch die Begründung von Explorationsmodulen als Komponente des didaktischen Systems, lieferte die Vorgehensweise zur Entwicklung und Erprobung dieser Komponente einen Beitrag zur Entwicklung und Erprobung didaktischer Systeme (vgl. 2.4.3, Frage 2 und Frage 3).

Zur Vorbereitung von Bildungsstandards

Informatische Bildung ohne die Anwendung von Informatiksystemen ist nicht möglich. Eingesetzt werden vielfach und frühzeitig professionelle Software-Entwicklungsumgebungen, die weder für die Zielgruppe Informatiklernender in der Sekundarstufe noch für das Anwendungsziel der Aneignung informatischer Fachkonzepte entwickelt wurden (vgl. 5.3.2). Mit dieser Art von Medieneinsatz ist die gewünschte Bildungsqualität entweder nur auf Umwegen, was angesichts knapp bemessener Bildungszeit ein ernstes Problem darstellt oder überhaupt nicht zu erreichen. Explorationsmodule als eine andere Art von Software sind erforderlich, um Lernende beim Aneignen objektorientierter Fachkonzepte zu unterstützen und ihren Aufbau kognitiver Modelle zum Fachgegenstand zu fördern. Dazu werden realitätsnahe Darstellungen mit objektorientierten Modellen verknüpft, den Lernenden die Möglichkeit gegeben, in vielfältigen Sichten mit den jeweiligen Darstellungen zu interagieren und die Veränderungen in den jeweils anderen Sichten zu beobachten. Ferner besteht bei Explorationssoftware die Möglichkeit, bestimmte Modellsichten einzubeziehen, die zwar fachdidaktisch sinnvoll für die professionelle Softwareentwicklung aber weniger interessant sind (z.B. Objektdiagramme). Bestimmte Interaktionen mit einem objektorientierten Modell können in dynamisch erzeugten Sichten dargestellt werden, die in dieser Form bei Softwareentwicklungswerkzeugen nicht zu finden sind. Beispiele hierfür sind dynamisch aus Benutzeraktionen erzeugte Sequenzdiagramme oder dynamisch daraus erzeugter Programmcode (vgl. 5.3.4.4). Wenn Lernende die fachlichen Grundlagen durch Exploration erworben und bzw. oder gefestigt haben, können unterrichtsgeeignete Softwareentwicklungswerkzeuge bei der Bearbeitung von Gestaltungsaufgaben eingesetzt werden.
Explorationsmodule eigenen sich auch ausgezeichnet dafür, Lernende in der Sek. I für die Wahl des Faches Informatik in der Sek. II zu gewinnen. Diese Lernenden erhalten damit einen einerseits motivierenden andererseits aber auch realistischen Einblick in Inhalte des Informatikunterrichtes in der Sek. II. Diese Vorabinformation ist eine wichtige Aufgabe, da Lernende oftmals das Fach Informatik mit völlig falschen Erwartungen und Vorstellungen in Bezug auf dessen Inhalte wählen und es auch dementsprechend schnell wieder verlassen. Besser informierte Lernende wählen das Fach bewusster und auch dementsprechend weniger schnell wieder ab. Damit kann ein Beitrag zur Stabilisierung der Schulinformatik geleistet werden. Die Umsetzung von Bildungsstandards erfordert ein stabiles Unterrichtsfach.

5.6.3 Offene Fragen

Die offenen Fragen zu Explorationsmodulen lassen sich folgenden Bereichen zuordnen.

- Umfassende, systematische Evaluation mit Vergleichsgruppen im Hinblick auf Wirksamkeitsaspekte
  Eine systematische Evaluation des Ansatzes in der informatischen Bildung, mittels Instrumenten empirischer Sozialforschung unter Berücksichtigung von Vergleichsgruppen, ist noch offen. Zu untersuchen wäre beispielsweise die Frage, ob sich mit der Ver-

wendung von Explorationsmodulen für Informatiklernende in der Anfangsphase der Aufbau mentaler Modelle zur objektorientierten Modellierung besser fördern lässt als mit anderen Ansätzen.

- Systematisierung der Verknüpfung von Aufgabenklassen, Explorationsmodulen und Wissensstrukturen
  Explorationsmodule stehen in enger Beziehung zu Aufgabenklassen, da in der Software mögliche Aktionen der Lernenden zu Aufgabenklassen korrespondieren. Zu untersuchen ist auf der Basis von Fehlerbildern zu Aufgabenklassen, welche weiteren Explorationsmodultypen benötigt werden, um diesen Schwierigkeiten zu begegnen. Ebenso ist der Prozess der Ableitung von Explorationsmodulen zu gegebenen Wissensstrukturen, Aufgabenklassen und dazu beschriebenen Fehlerbildern zu untersuchen und zu systematisieren.
- Überprüfung der Übertragbarkeit des Gestaltungskonzepts auf andere Themengebiete (nicht nur) der Informatik
  Für das im Rahmen der vorliegenden Arbeit entwickelte Gestaltungskonzept ist dessen Anwendbarkeit in anderen Themengebieten der Informatik und anderen Fachdisziplinen zu untersuchen. Erste Untersuchungen hierzu erfolgten bereits mit der Entwicklung eines verallgemeinerten Architekturkonzepts für Software zur Exploration im Bildungskontext und dessen exemplarischer Anwendung auf ein Lernproblem im sprachwissenschaftlichen Bereich (vgl. 5.3.5).

6 Wissensstrukturen

6.1 Überblick

Im vorliegenden Kapitel wird, ausgehend von einer Vertiefung des in Abschnitt 2.4.2 betonten Bedarfs an expliziten Strukturierungsempfehlungen für Lehr-Lern-Prozesse im Bereich des objektorientierten Modellierens (vgl. 6.2), im Abschnitt 6.3 der theoretische Hintergrund zu Wissensrepräsentationsformen, insbesondere Mapping-Techniken, kurz dargestellt. Auf dieser Basis werden verschiedene didaktische Funktionen einer grafischen Repräsentation von Erarbeitungs- und Vorkenntnisstrukturen von Fachkonzepten (im Folgenden als Wissensstrukturen bezeichnet) begründet (vgl. 6.4.1), aus diesen Anforderungen an eine geeignete Darstellungsform abgeleitet (vgl. 6.4.2) und schließlich verschiedene Darstellungsformen aus Informatik, Psychologie und Pädagogik (Begriffsnetze, Klassen- und Objektdiagramme, Semantische Netze, Und-Oder-Graphen) im Hinblick auf die Erfüllung der Anforderungen untersucht (vgl. 6.4.3). Zusammen mit einer studentischen Projektgruppe wurde die Repräsentation von Wissensstrukturen unter Verwendung der Landkartenmetapher erprobt. Das dabei entstandene Vorgehensmodell wird im Abschnitt 6.5 dargestellt. Fazit und offene Fragen bilden den Abschluss des Kapitels im Abschnitt 6.6.

6.2 Fachwissenschaftliche und fachdidaktische Erarbeitungsstrukturen

6.2.1 Vorüberlegungen

Da im Unterricht die Entwicklung von Kompetenzen im Vordergrund steht, ist die sachlogische Struktur der Fachwissenschaft allein ungeeignet zur Strukturierung des Unterrichts. Aufgrund der Struktur der Fachkonzepte wird aber deutlich, welche Elemente als Vorkenntnisse für andere Elemente erforderlich oder hilfreich sind. Wie im Abschnitt 2.4.2 dargestellt wurde, fehlt es an *expliziten* Empfehlungen und Begründungen für die Strukturierung von Lehr-Lern-Prozessen im Bereich der objektorientierten Modellierung. Fachdidaktische Dokumente (z.B. Forschungsarbeiten, Lehrbücher, Handreichungen, Bildungsempfehlungen, Unterrichtsentwürfe) enthalten oft implizites (teilweise auch explizites) Strukturierungswissen, das entweder aus konkreten Einzellehrerfahrungen der jeweiligen Autoren resultierte (Crutzen / Hein 1995) oder das sich aus gesammelten Lehrerfahrungen und bzw. oder theoretischen Analysen der jeweiligen Autoren in Lehrheuristiken manifestierte. Das fachdidaktische Strukturierungswissen bezieht sich auf mehr oder weniger Erfolg versprechende Erarbeitungsreihenfolgen von Fachkonzepten, Unterrichtsmethoden, Lernschwierigkeiten etc. und stellt damit Wissen über das Wissen, so genanntes Metawissen dar. Im „pedagogical patterns project“ (Manns et al. 2000) wurden wiederkehrende Lehr-Lern-Situationen aus dem Bereich der Objektorientierung mit erprobten Lehr-Lern-Strategien verknüpft. Erfolg versprechende Erarbeitungsreihenfolgen wurden teilweise angedeutet, aber nicht weit genug ausgearbeitet.

6.2.2 Fachwissenschaftliche Lehrbücher

Obwohl fachwissenschaftliche Lehrbücher andere Zielgruppen adressieren als Lernende in der Sekundarstufe II, so sind die in ihnen dargestellten Heuristiken zur Begriffsbildung und zu Erarbeitungsreihenfolgen nicht ohne Erprobung für den Informatikunterricht abzulehnen, sofern es sich um Fachkonzepte handelt, die fachdidaktisch fundierten Selektionskriterien für den Informatikunterricht genügen (z.B. Schwill 1993a).

Begriffsbildung zu objektorientierten Basiskonzepten über Metaphern

Objektorientierte Basiskonzepte, wie Objekt, Klasse, Assoziation oder Vererbung, sind erforderlich, um objektorientierte Modellierungstechniken und die Lösung von Problemen durch ein Geflecht kooperierender Objekte verstehen zu können. Deshalb stehen sie zumeist am Anfang entsprechender Ausbildungsabschnitte. In Lehrbüchern zu OOM, analysiert wurden hier Publikationen (Standardwerke) von Wirfs-Brock et al. (1990), Jacobson et al. (1992), Rumbaugh et al. (1993), Booch (1994) und Meyer (1997), werden diese Basiskonzepte oft über Metaphern eingeführt, indem ein Bezug zu Begriffen der Lebenswelt hergestellt wird. Wirfs-Brock et al. veranschaulichen den Klassenbegriff als Schablone für eine spezielle Art von Objekten und als Fabrik, die Objekte produziert (1990, 22). Ein Objekt sehen sie als Black Box mit öffentlicher Schnittstelle und geheimem Inhalt (ebd., 18). Jacobson et al. führen die Klasse als Bauplan für den internen Aufbau strukturgleicher Objekte ein (1992). Booch beschreibt eine Beziehung zwischen zwei Objekten als Pfad, über den Nachrichten versendet werden können (Booch 1994, 129) und Aggregatobjekte als Container (ebd., 166).

Varianten von Erarbeitungsreihenfolgen / Bezüge zum imperativen und funktionalen Ansatz

Bei der Reihenfolge der Erarbeitung der Grundbegriffe lassen sich in der Fachliteratur z.B. folgende Varianten erkennen:

- *Objekt, Nachrichtenaustausch und Beziehungen zwischen Objekten, Klasse, Beziehungen zwischen Klassen* (Wirfs-Brock et al. 1990, Jacobson et al. 1992, Booch 1994),
- *Objekt, Klasse, Beziehungen zwischen Objekten und Klassen* (Rumbaugh et al. 1993),
- *Klasse, Objekt, Beziehungen zwischen Objekten und Klassen* (Meyer 1997).

Während die ersten beiden Varianten keine Vorkenntnisse aus dem imperativen Paradigma erfordern und mit einem lebensweltlichen Objektbegriff beginnen, setzt die dritte Variante auf dem Begriff des abstrakten Datentyps (ADT) auf und entwickelt diesen zum Klassenbegriff weiter. Die Grundbegriffe werden in der Regel schrittweise und systematisch entwickelt. Zuerst eingeführte Begriffe werden verwendet, um nachfolgende daraus abzuleiten bzw. darauf aufzubauen. Teilweise werden Begriffe bereits verwendet, bevor sie formal eingeführt worden sind. Vorwissen aus anderen Paradigmen wird dazu verwendet, um Lerninhalte der Objektorientierung daran anzuknüpfen. Die Erarbeitung des Klassenbegriffs aus dem ADT ist ein Beispiel hierfür. Bei Meyer (1997, 215) finden sich auch Ansatzpunkte für einen Bezug zum funktionalen Ansatz. Beim funktionalen Ansatz kann z.B. eine von zwei Variablen abhängige Funktion $f(x,y)$ überführt werden in eine Funktion höherer Ordnung $(g(x))(y)=f(x,y)$. Dies wird als „currying“ bezeichnet. Beim objektorientierten Ansatz gibt es einen analogen Zusammenhang. So kann die Funktion f hier transformiert werden in $x.g(y)$ bzw. $y.g'(x)$.
Objektorientierte Vorgehensmodelle geben Hinweise zur systematischen Erstellung von statischem und dynamischem Modell (z.B. Balzert 1999, 119ff) und betonen, dass beide aufgrund der Abhängigkeit parallel entwickelt werden sollten. Die Einführung in die verschiedenen Techniken erfolgt in der Literatur, ähnlich wie bei den Basiskonzepten, streng systematisch. Praktische Übungsbeispiele zum OOM (z.B. Rumbaugh et al. 1993 und Kapitel 4 der vorliegenden Arbeit) zeigen allerdings, wie objektorientierte Basiskonzepte, Modellierungstechniken und Vorgehensweisen erst einzeln, dann kombiniert eingeübt werden können.

Fachkonzepte werden schrittweise und systematisch entwickelt. Dabei werden Wissen über die Lebenswelt und Vorkenntnisse zu anderen Problemlösungsmethoden benutzt, um die neuen Konzepte zu veranschaulichen.

6.2.3 Fachdidaktische Problembereiche

Fehlererkennung durch grafische Modellierungstechniken

Aus Sicht der Fachwissenschaft Informatik ist es erforderlich, dass Anfängerinnen und Anfänger nicht nur die prinzipielle, sondern aus ökonomischen Gründen die Konstruktion guter, objektorientierter Modelle erlernen im Sinne von Kriterien, wie z.B. Wiederverwendbarkeit und Wartbarkeit. Fowler und Scott weisen darauf hin, dass bestimmte Modellierungstechniken diesbezügliche Mängel schnell ersichtlich machen. So kann z.B. in Interaktionsdiagrammen anhand des Nachrichtenaustausches zwischen Objekten schnell erkannt werden, ob Aufgaben gleichmäßig zwischen Objekten verteilt sind oder ob der Entwurf zu stark zentralisiert ist (Fowler / Scott 1997, 8). Die Verwendung grafischer Modellierungstechniken kann also dazu beitragen, dass bestimmte Fehlerklassen und strukturelle Mängel schneller entdeckt werden können als dies z.B. auf Quelltextebene möglich ist.

Vermeidung von Sprüngen im Abstraktionsniveau

Probleme im Lehr-Lern-Prozess können auftreten, wenn es bei der Erarbeitung zu Sprüngen im Abstraktionsniveau kommt, wie z.B. bei der Konstruktion von Klassenhierarchien zu in Aufgabenstellungen beschriebenen Realitätsausschnitten zu beobachten ist. Ein fachdidaktisches Problem vieler hierzu publizierter Methoden und damit eine potentielle Fehlerquelle stellt der Übergang von der objektorientierten Sicht auf einen Realitätsausschnitt hin zur klassenorientierten Sicht des Klassendiagramms dar. In einem Realitätsausschnitt werden zumeist sofort potentielle Klassenkandidaten gesucht, anstatt zunächst problemrelevante Objekte zu identifizieren. Für Fortgeschrittene ist das eine leichte Aufgabe. Für Anfängerinnen und Anfänger wird an dieser Stelle die Formalisierung eines Realitätsausschnitts durch Objekte und damit ein Abstraktionsschritt übersprungen. Um nicht zu frühzeitig zu formalisieren, werden im Informatikunterricht teilweise CRC-Karten[96] (Beck / Cunningham 1989) als Vorstufe zu Klassendiagrammen eingesetzt. Da eine einzelne Karte eine informell beschriebene Klasse repräsentiert, wird dadurch das oben genannte Problem nicht gelöst. Der Abstraktionsschritt von der Beschreibung eines Realitätsausschnitts hin zur Strukturierung von Klassen in einem Klassendiagramm kann vereinfacht werden, wenn Objektdiagramme verwendet werden (vgl. Abbildung 23).

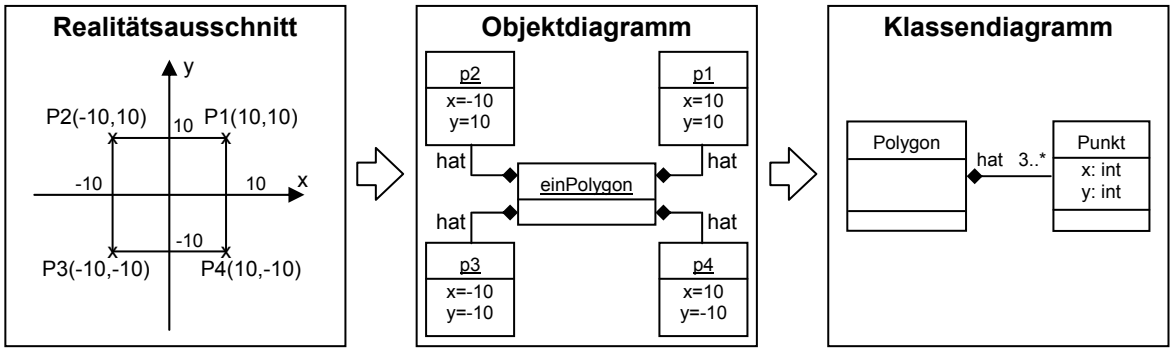


Abbildung 23: Realitätsausschnitt – Objektdiagramm – Klassendiagramm

Diese stellen eine Momentaufnahme eines Objektgeflechts zu einem bestimmten Zeitpunkt mit den aktuellen Attributwerten und Beziehungen zwischen Objekten dar. Da im Objektdiagramm bereits Notationen zur Anwendung kommen, die auch im Klassendiagramm verwendet werden, stellen diese eine gute Vorbereitung dar. Für die Transformation eines Objektdiagramms in ein Klassendiagramm können später Regeln formuliert werden, die diesen Prozess unterstützen. Ein ähnliches Vorgehen wird von Balzert (1999, 132 u. 142) vorgeschlagen. Für jede typische Interaktion einer Benutzerin oder eines Benutzers mit einem System (Anwen-

[96] Class – Responsibilities – Collaborators (deutsch: Klasse – Verantwortlichkeiten – Beteiligte)

dungsfall) sollen ein Objektdiagramm und ein lokales Klassendiagramm konstruiert werden. Die einzelnen Klassendiagramme werden anschließend zusammengeführt. Bei diesem Vorgehen wird ein großer Teil der gestalterischen Kreativität bereits bei der Konstruktion der Objektdiagramme gefordert. Um Brainstorming-Prozesse an dieser Stelle zu fördern, können die Objektdiagramme zunächst auf einer informellen Ebene entwickelt werden, etwa unter Verwendung eines Objekt-Analogons zu CRC-Karten, so genannten Objektkarten, die Objektnamen und Attributwerte enthalten. Iteriert man das beschriebene Vorgehen für alle Anwendungsfälle eines Informatiksystems, so erhält man ein statisches, objektorientiertes Analysemodell. Bei Balzert (1999, 170ff) finden sich weitere Hinweise, wie ausgehend von Anwendungsfällen und einem statischen Analysemodell über Szenarios eine Folge von Sequenzdiagrammen entwickelt und damit das dynamische Modell schrittweise und systematisch konstruiert werden kann.

Verknüpfung von Algorithmik und Objektkommunikation bei der grafischen Modellierung

Ein anderes Problem ergibt sich bei der Suche nach lernerangemessenen Darstellungsformen zur Visualisierung des dynamischen Verhaltens eines objektorientierten Modells. Beim imperativen Modellieren kamen hierzu Struktogramme zur Visualisierung von Kontrollstrukturen zum Einsatz. Vermutet wurde, dass mittels der UML-Interaktionsdiagramme (Sequenzdiagramm, Kollaborationsdiagramm) ein ähnlich mächtiges Hilfsmittel für OOM zur Verfügung steht. Interaktionsdiagramme sind ausgezeichnet dazu geeignet, Nachrichtenaustausch zwischen Objekten unter unterschiedlichen Gesichtspunkten zu veranschaulichen. Sequenzdiagramme bieten zudem rudimentäre Hilfsmittel zur Darstellung von Kontrollstrukturen an (Oestereich 1998, 307), deren Anschaulichkeit aber fragwürdig ist. Zur Darstellung solcher Strukturen werden Aktivitätsdiagramme empfohlen (ebd.), die aber wiederum die Objektkommunikation nicht angemessen darstellen. Allgemeine Methoden kombinieren aber Objektkommunikation mit algorithmischen Aspekten (Kontrollstrukturen, Berechnungen). Eine lernerangemessene Darstellungsform, welche die Verknüpfung beider Aspekte auf anschauliche Weise gewährleistet, fehlt (vgl. Abbildung 24).

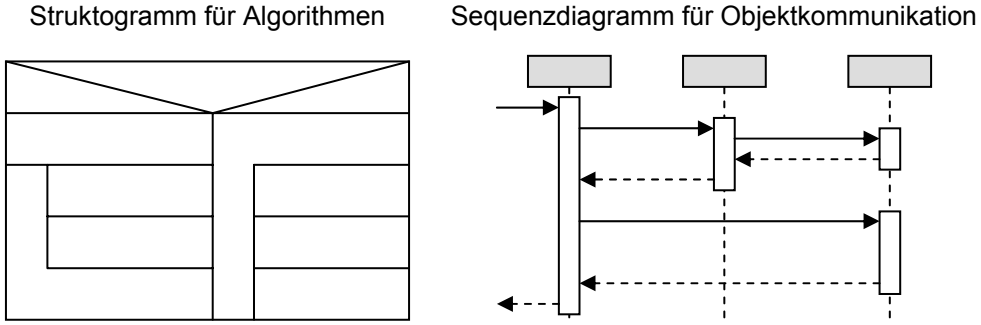


Abbildung 24: Lernhilfen zum Methodenentwurf

Vermutet wird hierin eine potentielle Ursache für Lernschwierigkeiten beim Methodenentwurf. Als Vermeidungsstrategie wurden Unterrichtsbeispiele für Anfängerinnen und Anfänger oft so gewählt, dass die Methodenrümpfe weitestgehend auf Kontrollstrukturen verzichteten (z.B. der Fall beim Fahrschein-, Geldwechsel-, Getränkeautomat).
Experimentiert wurde in Lehr-Lern-Prozessen ferner mit so genannten Story Boards (vgl. 3.8 und Diethelm et al. 2002, 2003), mit denen sich das dynamische Verhalten eines objektorientierten Systems vollständig grafisch spezifizieren lässt.

> „Story-Boards sind eine Variante von UML Aktivitätsdiagrammen, bei denen die einzelnen Aktivitäten durch spezielle Objektdiagramme, sogenannte Schnappschüsse, beschrieben werden." (ebd., 35)

Die nichttriviale Notation einerseits sowie der Umstand, das Systemverhalten durch eine unter Umständen große Anzahl von Schnappschüssen spezifizieren zu müssen, lassen die unterrichtliche Eignung zumindest im Anfängerbereich als fragwürdig erscheinen.

In den exemplarisch dargestellten Problembereichen liegen implizite Anforderungen an die Gestaltung von Lehr-Lern-Prozessen. Um die selbstständige Bewertung fachlicher Probleme in Modellen zu fördern, können grafische Hilfsmittel ein sinnvolles Medium sein. Das setzt deren zeitnahe Thematisierung voraus. Durch die eingehende Analyse fachlicher Strukturen lässt sich feststellen, ob auf dem Lehr-Lern-Pfad zu einem neuen Konzept relevante Konzepte „übersprungen" wurden. Durch die Vermeidung solcher Sprünge im Abstraktionsniveau lassen sich potentielle Lernschwierigkeiten verhindern. Am Beispiel des Methodenentwurfs wurde gezeigt, wie wichtig die Berücksichtigung aller für ein Konzept erforderlichen Vorkenntnisse ist, da es sonst zu Lernschwierigkeiten kommt. In allen Bereichen kann die grafische Repräsentation der Erarbeitungs- und Vorkenntnisstruktur von Fachkonzepten hilfreich sein.

6.2.4 E-Learning

Im Bereich des E-Learning wird Wissen zur Sequenzierung von Fachkonzepten teilweise dazu verwendet, im Sinne „intelligenter tutorieller Systeme" Pfade durch eine Folge von Lehr-Lern-Modulen automatisch zu generieren. Auf der Basis der Struktur diagnostiziert die Software den individuellen Wissensstand des Lernenden, von dem ausgehend weitere Wissensknoten zur Bearbeitung frei geschaltet werden[97]. Theoretisch fundiert wird dies z.B. durch die „Theory of knowledge spaces" (Albert 2000). Oftmals führte solche Software aber zu Frustrationserlebnissen, da der automatisch diagnostizierte und der tatsächliche Wissensstand der Lernenden erheblich differierten. Zur Verbesserung der Orientierung in großen Wissensräumen, zur automatischen Anordnung und zur Förderung der Weiterverwendung in anderen Kontexten wurden sehr umfangreiche Kataloge mit Vorschlägen für Metadaten zur Spezifikation von webbasierten Lehr-Lern-Materialien entwickelt, z.B. „Learning Object Metadata (LOM)" (IEEE 2002). Durch die maschinelle Repräsentation lässt sich aus dem formalisierten Wissen neues Wissen mittels Informatiksystemen ableiten. Repräsentationsformen, die eine solche automatische Auswertung erlauben, sind nicht zwangsläufig gleichermaßen geeignet für fachdidaktische Analysen und Diskussionsprozesse. Es fehlt an anschaulichen, ausdrucksstarken Repräsentationen von Strukturen von Fachkonzepten, an denen sich Erarbeitungsreihenfolgen und fachbezogene Verläufe von Lehr-Lern-Prozessen ohne langwieriges Textstudium analysieren und diskutieren lassen. Eine maschinelle Verarbeitung im Sinne „intelligenter tutorieller Systeme" ist hier nicht das Ziel.

6.3 Theoretischer Hintergrund

> „Die Qualität von Entwicklungs- und Lernprozessen hängt heute mehr denn je davon ab, inwieweit es gelingt, komplexe Wissensbestände zu strukturieren, zu erwerben, zu kommunizieren und anzuwenden." (Mandl / Fischer 2000b, 3)

Lehrende nehmen in Lehr-Lehr-Prozessen immer mehr die Funktion von Beratenden ein und „[...] benötigen effektive Werkzeuge zur Unterstützung der Wissenskommunikation ebenso, wie Werkzeuge zur differenzierten Beurteilung und Diagnose von Wissensveränderungen bei den Lernenden" (ebd., 3). Zum Einsatz kommen hierfür so genannte Mapping-Techniken, die ihren Ursprung in Psychologie und Pädagogik haben. Hierbei handelt es sich um eine „[...] Gruppe von Visualisierungswerkzeugen [...], die Lehrende und Lernende beim Wissensmana-

[97] z.B. http://studierplatz2000.tu-dresden.de/tee.htm (aufgerufen am 22.11.03)

gement in Lern- und Kooperationsprozessen durch die *grafische Darstellung von Wissensstrukturen* unterstützen können“ (ebd., 3, Hervorhebung im Original). In diesen grafischen Darstellungen werden Begriffe als Knoten und Relationen zwischen diesen als Kanten repräsentiert. Beispiele sind semantische Netze, Begriffsnetze (engl. concept maps) und Mind Maps. Um hohe Flexibilität bei der Erstellung und bei der Darstellung zu gewährleisten und um Archivierung und automatische Analyse zu unterstützen, kommen hierbei Software-Werkzeuge zum Einsatz. Aktuelle Forschungs- und Anwendungsfelder im Bereich der Mapping-Techniken sind (Mandl / Fischer 2000a):

- *Mapping-Techniken als Lehr-Lern-Strategie* (z.B. Bernd et al. 2000, Fischer / Mandl 2000b, Hillen et al. 2000): Während bei der Anwendung als Lehrstrategie der Konzepterwerb im Vordergrund steht, liegt bei der Anwendung als Lernstrategie der Schwerpunkt auf dem Vertrautmachen mit dem Werkzeug, das anschließend helfen soll, Konzepte selbstständig zu erarbeiten.
- *Mapping-Techniken zur Unterstützung von Kooperationsprozessen beim gemeinsamen Lernen* (z.B. Bruhn et al. 2000): Hierbei liegt der Schwerpunkt auf der kooperativen Erstellung von Begriffsnetzen.
- *Mapping-Techniken zur Wissensdiagnose und -modellierung* (z.B. Häußler et al. 1998, Janetzko / Strube 2000): Hierbei dient die durch Lernende erzeugte, grafische Repräsentation der Wissensstruktur als Messinstrument für Wissensveränderungen. Anhand der grafischen Darstellung von Konzepten und Relationen durch Lernende wird auf deren kognitive Strukturen bzw. Strukturveränderungen geschlossen.

Arbeiten zu diesen Themenfeldern lieferten u.a. Belege für kurzfristig verbesserte Behaltensleistungen sowie für eine Förderung von Prozessen der gemeinsamen Wissenskonstruktion im Diskurs beim kooperativen Lernen:

> „In zwei Untersuchungen [...] wurde der ‚alten Frage’ nachgegangen, ob concept maps lerneffektiver seien als die üblichen Zusammenfassungstexte. In der ersten Untersuchung mit 13 Experimenten unterschiedlicher Schulfächer im Luxemburger Sekundarschulwesen lernten jeweils zwei parallele Klassen den gleichen Inhalt, und zwar die eine, indem sie eine entsprechende concept map, die andere (Kontrollklasse), indem sie inhaltsanaloge Texte durcharbeitete. Getestet wurde mit einer ‚Lücken-Map’ bzw. einem inhaltsanalogen ‚Lücken-Text’. Die Ergebnisse sprechen eindeutig dafür, dass die Nutzung von concept maps beim Durcharbeiten von Begriffen das kurzfristige Behalten von Wissenselementen des Begriffs besser fördert als die Nutzung einer analogen, semantisch identischen Textversion.“ (Bernd et al. 2000, 25)

> „Bezogen auf die *Lernprozesse* zeigte sich, dass der Einsatz einer inhaltsspezifischen Mapping-Technik beim kooperativen Lernen zu einer stärkeren Aufgabenorientierung des Diskurses führen kann. [...] Darüber hinaus ergab sich, dass Mapping-Techniken *Prozesse der gemeinsamen Wissenskonstruktion im Diskurs* beim kooperativen Lernen fördern können. Eine Förderung zeigte sich tendenziell bei Prozessen der Externalisierung aufgabenbezogener Inhalte sowie bei der Elizitation aufgabenbezogener Inhalte [...]. Vor allem aber die konfliktorientierte Konsensualisierung aufgabenbezogener Inhalte kann von solchen Förderungsmaßnahmen profitieren.“ (Bruhn et al. 2000, 131, Hervorhebungen im Original)

Bei den hier betrachteten Wissensstrukturen stehen die Begriffe in vielfältigen Relationen zueinander. In Analogie zu kognitiven Strukturen kommen Generalisierungs- und Teil-Ganzes-Relation zum Einsatz (z.B. bei semantischen Netzen). Begriffsnetze ermöglichen beliebige, gerichtete oder ungerichtete, binäre oder *n*-näre Relationen zwischen Begriffen. Bereits vor Entwicklung der UML wurde Metamodellierung in Übungsaufgaben in fachwissenschaftlichen Lehrbüchern eingesetzt, in denen unter Verwendung von Klassendiagrammen die Zusammenhänge zwischen ausgewählten Fachkonzepten dargestellt werden sollten (vgl. Abschnitt C.2.6 und Rumbaugh et al. 1993). Klassendiagramme setzen verschiedene Begriffe (Klassen) in Relation zueinander (Assoziation, Aggregation, Komposition, Vererbung) und eignen sich deshalb auch prinzipiell zur Wissensstrukturierung im oben beschriebenen Sinne.

Über den Einsatz von Mapping-Techniken oder Metamodellierung im Informatikunterricht zum OOM finden sich bis 2003 keine Publikationen. In allen oben genannten Forschungs- und Anwendungsfeldern liegt großes Potential für die empirische Forschung im Bereich der Didaktik der Informatik.

6.4 Repräsentation der Struktur von Lehr-Lern-Prozessen

6.4.1 Fachdidaktische Funktionen

Auf der Basis der Vorüberlegungen (vgl. 6.2) und des theoretischen Hintergrunds (vgl. 6.3) lassen sich folgende fachdidaktische Funktionen einer grafischen Repräsentation von Erarbeitungsstrukturen von Fachkonzepten ableiten:

1. *Veranschaulichung der Zusammenhänge für Akteure in Lehr-Lern-Prozessen*
   Da OOM für viele Lehrende eine neue Informatikmethode ist, ist das Ziel ein kompakter und ausdrucksstarker grafischer Überblick darüber, ob und wenn ja, wo an vorhandenes Vorwissen einer Lerngruppe angeknüpft werden kann. Von Bedeutung ist ferner, in welcher Reihenfolge Fachkonzepte angeeignet werden können, welche Alternativen es hierzu gibt, welches Fachkonzept oder welche Menge von Fachkonzepten als Vorwissen für ein anderes Fachkonzept benötigt wird etc. Lernenden kann die grafische Struktur für die Organisation von Selbststudienphasen bzw. zur Vorbereitung, Wiederholung und Nachbereitung von Unterricht dienlich sein. Um beides zu ermöglichen, sind erfolgreiche Lehr-Lern-Wege durch den Raum der Fachkonzepte in einer grafischen Darstellung zu vereinigen. Die Repräsentation dient hier als Lehr-Lern-Strategie (vgl. 6.3, Punkt 1).
2. *Fachdidaktische Analyse / Diskussion von Lehr-Lern-Prozessen*
   Durch die grafische Repräsentation von Erarbeitungsstrukturen von Fachwissen wird ein Beitrag zur Verbesserung der Analysemöglichkeiten von Bildungsprozessen geleistet. Eine solche Repräsentation fachdidaktischen Wissens erleichtert die Kommunikation zu konkreten Bildungsergebnissen einerseits und zum Stand der Fachdidaktik andererseits (Anderson 2001, 139ff) und führt zu einer besseren Vergleichbarkeit von Lehr-Lern-Prozessen und Bildungsergebnissen. Weiterhin kann die Kooperation beim gemeinsamen Weiterentwickeln informatischer Bildung gefördert werden, wobei hier nicht Lernende kooperieren (vgl. 6.3, Punkt 2), sondern Lehrende, Forscherinnen und Forscher, die gemeinsam Erkenntnisse über die Gestaltung und den Erfolg von Lehr-Lern-Prozessen gewinnen wollen. Die grafische Darstellung kann dabei leicht zur Analyse fachlicher und lehrmethodischer Fehler konkreter Bildungsprozesse verwendet werden, indem z.B. festgestellt werden kann, ob ein konkreter Prozess zu Problemen führte, weil kein ausgewogenes Verständnis für die fachlichen Grundlagen von zu erarbeitenden komplexen Konzepten entwickelt wurde. Die im Zusammenhang solcher Situationen auftretenden Lernschwierigkeiten sind systematischer Art und nicht zielgruppenspezifisch oder individuell.
3. *Kontrolle des Bildungsstandes für Lehrende und Lernende*
   In der grafischen Repräsentation werden Elemente der informatischen Bildung abgebildet, die sich mit Lernerfolgskontrollen messen und bewerten lassen. Für Lehrende und Lernende wird der erreichte Bildungsstand so anschaulicher. Die Repräsentation wird zum Messinstrument des erreichten Wissensstandes (vgl. 6.3, Punkt 3).

Die hier beschriebenen, fachdidaktischen Funktionen konkretisieren die allgemeiner gefassten Funktionen, die bei der Einführung der Komponenten des didaktischen Systems im Kapitel 3 (vgl. 3.5.3) dargelegt wurden. Der Punkt 1 korrespondiert zur Funktion „Gestaltungsmittel für

Lehr-Lern-Prozesse“, der Punkt 2 zur „Förderung der fachdidaktischen Kommunikation und Diskussion zu Lehr-Lern-Prozessen“ und der Punkt 3 schließlich zur „Anwendung in Lehr-Lern-Prozessen“.

6.4.2 Anforderungen an die Darstellungsform

Aus den fachdidaktischen Funktionen (vgl. 6.4.1) ergeben sich eine Reihe von Anforderungen an die Darstellungsform. Diese sind (vgl. Brinda / Schubert 2001, 22f):

- *Ausdrucksstärke und Übersichtlichkeit:* Wesentliches Ziel der Repräsentation ist es, Akteuren in Lehr-Lern-Prozessen (Lernenden und Lehrenden) einen schnellen und kompakten Überblick über mögliche Erarbeitungsstrukturen von Fachkonzepten zu geben, aus denen diese dann individuelle Lern- und Lehrwege ableiten sollen. Aus diesem Grund muss die Repräsentation ausdrucksstark, übersichtlich und leicht verständlich sein und sollte ihrerseits keiner großen Einarbeitung bedürfen (Anderson 2001, 139ff). Um einen schnellen und kompakten Überblick zu vermitteln, sollte die Darstellungsform ferner mit wenigen Darstellungsmitteln (Knoten- und Kantentypen) auskommen.
- *Repräsentation von Erarbeitungs- bzw. Vorkenntnisstrukturen:* Die Repräsentation von Erarbeitungs- und Vorkenntnisstrukturen manifestiert sich in drei Teilanforderungen:
  - *Modellierung von Relationen zwischen Fachkonzepten:* Fachkonzepte und Fachtermini sollen durch Knoten einer grafischen Darstellung repräsentiert werden. Zur Modellierung von Erarbeitungs- bzw. Vorkenntnisstrukturen sind die Knoten mittels einer „ist erforderlich für“-Relation[98] (gerichtete Kanten) zu strukturieren. Zur Modellierung von „weicheren“ Relationen zwischen Fachkonzepten kann beispielsweise eine „ist hilfreich für“-Relation verwendet werden.
  - *Modellierung von obligatorischen Teilkonzepten und alternativen Lehr-Lern-Wegen:* Die Darstellungsform muss hinreichend ausdrucksstark sein, um (z.B. boolesche) Verknüpfungen zwischen, über Relationen verbundenen, Knoten zu modellieren. Es muss sich ausdrücken lassen, dass verschiedene Wissenselemente gemeinsam als Vorkenntnisse für andere benötigt werden oder dass aus einer Menge von Wissenselementen mindestens eines als Vorkenntnis benötigt wird. Ebenso müssen sich alternative Lehr-Lern-Wege modellieren lassen.
  - *Modellierung von Erarbeitungsreihenfolgen:* In Lehr-Lern-Prozessen werden Fachkonzepte in der Regel nacheinander erarbeitet. Die Reihenfolge ist stark abhängig vom Vorwissen der Lerngruppe, der Unterrichtshistorie, dem gewünschten Einstiegspunkt, u.a. Die Darstellung ist so zu wählen, dass Erarbeitungsreihenfolgen nur an den Stellen spezifiziert werden, an denen dies unbedingt erforderlich ist. Dies steht im engen Bezug zur Modellierung von Obligation und Alternativen. Durch die Modellierung einer Menge von verbindlichen Teilkonzepten, die als Vorkenntnis benötigt werden, ist zunächst nicht näher festgelegt, in welcher Reihenfolge diese zu erarbeiten sind. Das Ziel ist die Vermeidung starrer sowie maximale Flexibilität für individuelle Lehr-Lern-Pfade.

  Aus diesen Anforderungen folgt direkt, dass sich hier nur graphbasierte Darstellungsformen eignen. Sequenzielle Darstellungsformen, wie Listen erlauben nur eine Möglichkeit der Anordnung. Baumbasierte Darstellungsformen mit beliebigem und variablem Knotenausgrad ermöglichen es, Fachkonzepte hierarchisch anzuordnen und so die Anforderungen an die Vorkenntnisse darzustellen, indem diese jeweils als Kinder eines

[98] oder: „ist Vorkenntnis von“

Knotens dargestellt werden. Da alle Kinderknoten gleichberechtigt sind, bleibt die Reihenfolge der Aneignung der mit ihnen verbundenen Fachkonzepte offen. Reine Baumdarstellungen des Vorkenntnisgeflechts erweisen sich als nachteilig, wenn Fachkonzepte dieselben Vorkenntnisse benötigen. Der, die gemeinsamen Vorkenntnisse repräsentierende, Teilbaum wird dann so oft in den Gesamtbaum übernommen, wie es Elemente gibt, die diese Vorkenntnisse erfordern. Dadurch wird die Darstellung schnell unhandhabbar. Graphbasierte Darstellungsformen vermeiden die Nachteile der baumbasierten Darstellungsformen. Zyklenfreie, gerichtete Graphen eignen sich deshalb gut für den gegebenen Zweck.

- *Verwendung standardisierter Darstellungsformen*: Es sind nach Möglichkeit standardisierte Darstellungsformen zu verwenden. Deren Verwendung bietet den Vorteil einer definierten Semantik sowie oftmals die Verfügbarkeit von Software-Werkzeugen, welche die Flexibilität bei der Erstellung und bei der Darstellung sowie die Archivierung unterstützen.

6.4.3 Analyse ausgewählter Darstellungsformen

Um die Eignung ausgewählter Darstellungsformen zur Realisierung der fachdidaktischen Funktionen (vgl. 6.4.1) bewerten zu können, ist zu überprüfen, inwieweit sie die daraus resultierenden Anforderungen erfüllen (vgl. 6.4.2). Folgende Darstellungsformen werden für die Analyse verwendet:

- Begriffsnetze (concept maps),
- Klassen- und Objektdiagramme,
- Semantische Netze,
- Und-Oder-Graphen.

Vorgehensweise

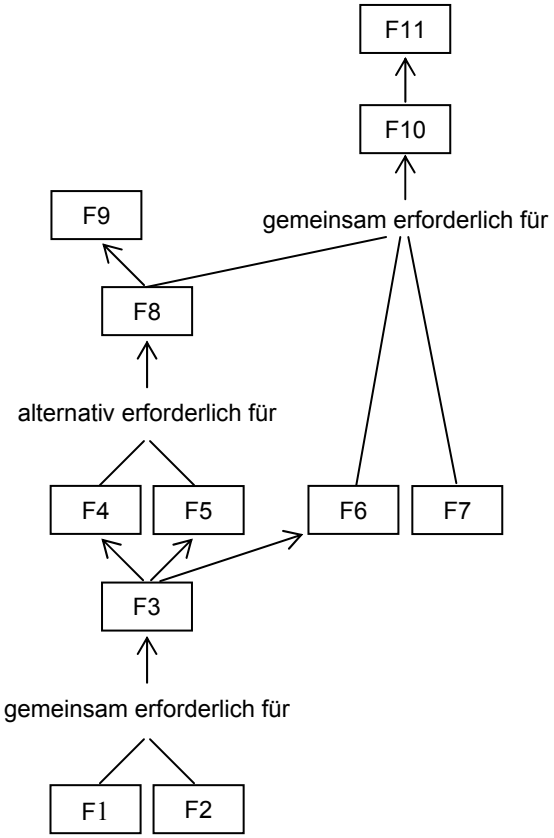
Für die Analyse wird eine anonyme Referenzstruktur verwendet. Gegeben seien dazu die Fachkonzepte *F1, F2, ..., F11* (paarweise verschieden) mit folgender Erarbeitungsstruktur:

- *F1* ist erforderlich für *F3* und *F2* ist erforderlich für *F3,*
- *F3* ist erforderlich für *F4,*
- *F3* ist erforderlich für *F5,*
- *F3* ist erforderlich für *F6,*
- *F4* ist erforderlich für *F8* oder *F5* ist erforderlich für *F8,*
- *F8* ist erforderlich für *F9,*
- *F6* ist erforderlich für *F10* und *F7* ist erforderlich für *F10* und *F8* ist erforderlich für *F10,*
- *F10* ist erforderlich für *F11.*

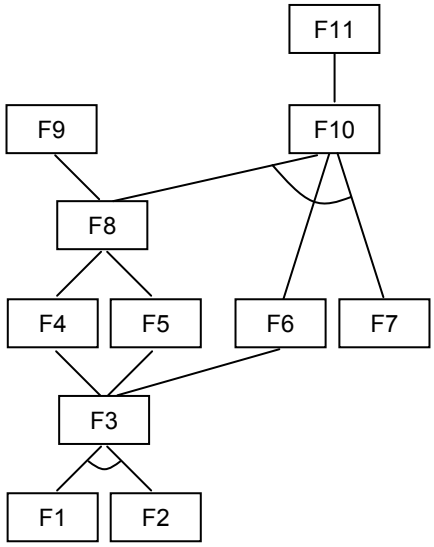
Diese Referenzstruktur wird mittels der für die Analyse ausgewählten Darstellungsformen implementiert, die jeweiligen Ergebnisse verglichen und im Hinblick auf die Erfüllung der Anforderungen (vgl. 6.4.2) bewertet. Die Struktur ist dabei so konzipiert, dass alle Anforderungen an ihr umgesetzt werden können.

Ergebnisse

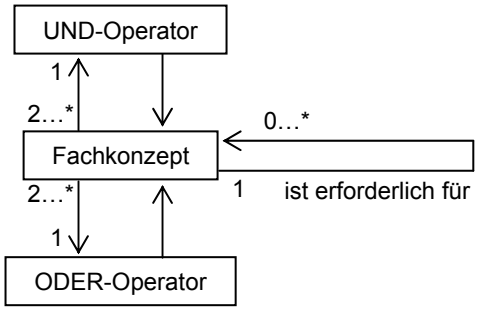
Ausdrucksstärke und Übersichtlichkeit sowie die Darstellbarkeit von Erarbeitungsreihenfolgen sind bei allen Darstellungsformen gleichermaßen gegeben. Signifikante Unterschiede gibt es bei der Darstellung von obligatorischen Teilkonzepten und alternativen Lehr-Lern-Wegen.



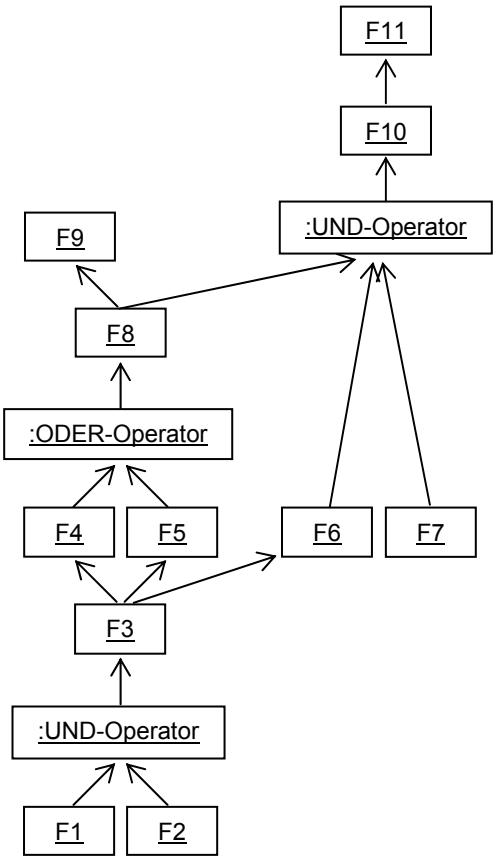
a.) Referenzstruktur als Begriffsnetz (concept map)



d.) Referenzstruktur als Und-Oder-Graph



b.) Klassendiagramm zur Referenzstruktur



c.) Referenzstruktur als Objektdiagramm

Abbildung 25: Implementierung der Referenzstruktur

Begriffsnetz

Abbildung 25-a zeigt die Implementierung der Struktur als Begriffsnetz. Unbeschriftete Kanten (mit Pfeil zur Richtungsspezifikation) modellieren darin die „ist erforderlich für"-Relation (binärer Fall). Begriffsnetze ermöglichen beliebige *n*-näre Relationen, so dass die Konjunktion bzw. Disjunktion von Vorkenntnisstrukturen durch entsprechende Relationsbezeichner (hier: gemeinsam / alternativ erforderlich für) dargestellt werden kann. Weitere, anders benannte Kanten, z.B. „ist hilfreich für" sind problemlos möglich.

Klassen- / Objektdiagramm und Semantisches Netz

Im Klassendiagramm werden Zusammenhänge zwischen Begriffsklassen modelliert. Relationen zwischen den Klassen sind zunächst binär, mehrgliedrige Assoziationen sind aber möglich (Oestereich 1998, 280f). Konjunktion und Disjunktion können, ebenso wie beim Begriffsnetz, durch entsprechend benannte, mehrgliedrige Assoziationen modelliert werden. Mehrgliedrige Assoziationen werden durch Einführung einer eigenen Klasse für die Assoziation in binäre Assoziationen aufgelöst (ebd., 281). Damit stellt sich das Problem, dass Klassen innerhalb eines Klassendiagrammes paarweise verschiedene Namen haben müssen. Gibt es in der gewählten Struktur mehr als einen Und- bzw. Oder-Operator, so tritt der Namenskonflikt auf. Dieses Problem ergibt sich analog beim semantischen Netz (vgl. Reimer 1991, 82). Aus diesem Grund wurde die Struktur als Objektdiagramm (Abbildung 25-b) dargestellt. Darin lassen sich benannte Objekte (F1, F2, …) vom Typ Fachkonzept (vgl. Abbildung 25-c, der Objekttyp „Fachkonzept" wurde im Objektdiagramm aus Gründen der Übersichtlichkeit weggelassen) mit (hier: unbenannten) Objekten vom Typ Und- bzw. Oder-Operator verknüpfen. Eine alternative Darstellungsmöglichkeit ist die Repräsentation von Disjunktionen von binären Assoziationen im Klassendiagramm mittels Oder-Zusicherungen (Oestereich 1998, 248). Oestereich merkt allerdings dazu an:

> „Oder-Zusicherungen für Assoziationen gelten nicht gerade als besonders elegantes Design und sollten mit Bedacht verwendet werden." (ebd.)

Bei semantischen Netzen ergibt sich zudem die Schwierigkeit, dass alle durch sie repräsentierten Aussagen implizit konjunktiv verknüpft sind (vgl. Reimer 1991, 82). Die Darstellung von Disjunktionen stellt dort ein prinzipielles Problem dar.

Und-Oder-Graph

Und-Oder-Bäume stammen ursprünglich aus der Spieltheorie und wurden dort eingesetzt, um bei Spielen optimale Strategien zu ermitteln, ausgehend von alternativen Aktionen und Reaktionen der Mitspieler. Blätter repräsentieren Erfolg oder Misserfolg eines Mitspielers. Und-Oder-Graphen stellen die Verallgemeinerung der Und-Oder-Bäume auf azyklische, gerichtete Graphen dar. Und-Oder-Graphen vermeiden die Nachteile der zuvor genannten Darstellungsformen, da sie konjunktiv bzw. disjunktiv verknüpfte Kanten als zentrale Gestaltungsmittel bereitstellen (Abbildung 25-d). In Begriffsnetzen, die ähnlich ausdruckstark sind, muss die intendierte Semantik erst durch geschickte Namensgebung modelliert werden. Und-Oder-Graphen werden z.B. auch als theoretische Grundlage bei der „Theory of knowledge spaces" (Albert 2000, vgl. 6.2.4) eingesetzt, um dort Problemlösungsstrukturen bei adaptiver Lehr-Lern-Software zu repräsentieren. Die Programmierung solcher Strategien ist hier aber nicht das Ziel.

Es zeigt sich also eine prinzipielle Eignung von Begriffsnetzen, Objektdiagrammen und Und-Oder-Graphen. Aus den genannten Gründen werden Und-Oder-Graphen für die angestrebten fachdidaktischen Funktionen der Repräsentation (vgl. 6.4.1) empfohlen. Lerninhalte von OOM wurden mit dieser Darstellungsform erfolgreich strukturiert (vgl. Brinda 2000a, 2001; Brinda / Schubert 2001, 2002a). Ein Beispiel für die exemplarische Verknüpfung von Und-

Oder-Graphen zur Repräsentation von Wissensstrukturen mit Explorationsmodulen und Aufgabenklassen zeigt Abbildung 26.

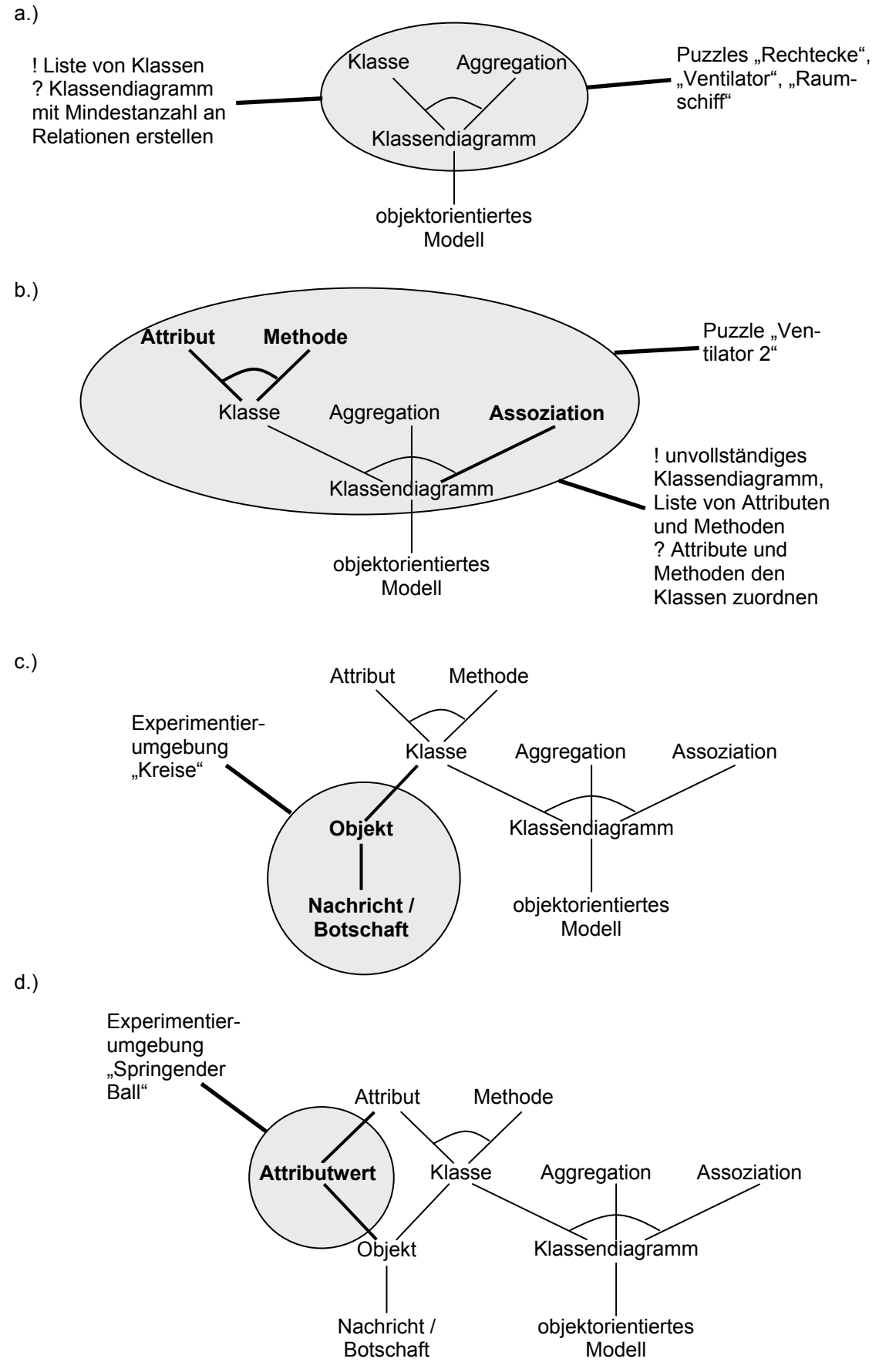


Abbildung 26: Verknüpfung von Wissensstrukturen, Explorationsmodulen und Aufgabenklassen

Jeweils im Kern der Abbildungsteile a.) bis d.) steht ein Und-Oder-Graph in vier verschiedenen Ausbaustufen einer möglichen Variante fachlicher Erarbeitungsstrukturen zu den jeweils genannten OOM-Konzepten. Jeweils im Fettdruck dargestellt sind die Ergänzungen in einer Ausbaustufe zur jeweils vorherigen. Damit soll zum Ausdruck gebracht werden, dass es ganz typisch für schulische Bildungsprozesse ist, dass Fachkonzepte nicht immer von vornherein in ihrer vollen Komplexität im Unterricht eingeführt oder erarbeitet, sondern z.B. zunächst intuitiv verwendet, dann in einer einfachen Fassung eingeführt und später schrittweise im Lehr-Lern-Prozess ausgebaut werden. Die grau unterlegten Ellipsen bzw. Kreise markieren Teilgraphen, in denen die mit ihnen korrespondierenden Fachkonzepte jeweils mit den genannten, im Rahmen dieser Arbeit entwickelten, Informatiksystemen zur Exploration und Aufgabenklassen verknüpft sind. Aus Gründen der Übersichtlichkeit wurden bei den jeweils folgenden Ausbaustufen nur die neuen Verknüpfungen dargestellt.

Zur Darstellung von alternativen Lehr-Lern-Wegen

Für die Modellierung von alternativen Lehr-Lern-Wegen in Bezug auf eine Menge von Fachkonzepten bzw. -begriffen kommen folgende Varianten in Betracht:

- *ein Graph je alternativem Lehr-Lern-Weg*
  Für jeden alternativen Lehr-Lern-Weg wird ein eigener Graph erstellt. Ist der Lehr-Lern-Weg, für den Alternativen vorliegen, Teil eines umfassenderen Graphen, so sind die Alternativen nur für den betroffenen Teilgraphen darzustellen. Vorteilhaft ist die Übersichtlichkeit der Methode, nachteilig eine mit der Zahl der Alternativen wachsende Anzahl zu verwaltender Graphen.
- *alle alternativen Lehr-Lern-Wege in einem Graphen*
  Bei einer kleinen Anzahl von alternativen Lehr-Lern-Wegen können diese im selben Graphen z.B. durch unterschiedliche Kantenmarkierungen dargestellt werden. Vorteilhaft ist, dass die Darstellung kompakt bleibt und dass schnell zu erkennen ist, an welchen Stellen Gemeinsamkeiten bestehen und wo sich die Varianten unterscheiden. Nachteilig ist allerdings, dass diese Darstellung bei zu vielen Varianten schnell unübersichtlich wird.

Welche Variante zu bevorzugen ist, lässt sich nicht pauschal beantworten, sondern ist stark abhängig von der jeweiligen Schwerpunktsetzung. Aus Gründen der Übersichtlichkeit und Ausdrucksstärke, wesentliche Anforderungen an die Darstellung (vgl. 6.4.2), tendiert der Autor aber zur ersten Variante.

6.5 Vorgehensweise zur Entwicklung einer didaktischen Landkarte

Im Rahmen einer vom Autor im Studienjahr 2001/02 betreuten studentischen Projektgruppe an der Universität Dortmund entwickelten elf Informatikstudierende im Hauptstudium in zwei Semestern eine „Lernumgebung für objektorientiertes Modellieren im Informatikunterricht (LEO)“ (vgl. 5.3.4.4, F.3.2, Alex et al. 2002). Im von der Projektgruppe gewählten Vorgehensmodell für die Entwicklung der Lernumgebung stand am Anfang ein Treffen mit Lehrenden, Referendarinnen und Referendaren (23.10.2001) an der Universität Dortmund, um aus der Diskussion mit der intendierten Zielgruppe Gestaltungshinweise abzuleiten. Im Rahmen dieser Diskussion entstand der Gedanke einer „didaktischen Landkarte“ für OOM. OOM-Fachkonzepte sollten mittels der Landkartenmetapher strukturiert werden, um mit einer solchen Lernhilfe zu einer besseren Orientierung im und zur Stärkung der Selbstorganisation des Lernprozesses beizutragen (vgl. Kerres 2001, 238f). Bestimmte Teilstrukturen sollten mit Szenarien der Lernumgebung (vgl. 5.3.4.4) verknüpft werden. Folgende Zusammenfassung hierzu findet sich im Projekt-Endbericht (Alex et al. 2002, 22):

> „Um [...] die wichtigsten Konzepte der Objektorientierung, die LEO abdecken sollte, festzulegen, sollte eine didaktische Landkarte erstellt werden. Diese sollte ein […] Begriffsnetz und mögliche Verbindungswege als Leitfaden von einem Konzept zum nächsten enthalten. Diese Begriffe sollten […] durch Szenarien vermittelt werden. Ein Szenario ist […] eine kleine Modellwelt, die eine beschränkte Menge an Klassen enthält, mit denen Lernende experimentieren und modellieren können. Jedes Szenario sollte eine Teilmenge der Begriffe der didaktischen Landkarte erklären, so dass die Menge aller Szenarien auch das gesamte Begriffsnetz der didaktischen Landkarte abdeckt."

Die Analogie zwischen Wissensstrukturen und einer didaktischen Landkarte ist hilfreich, da benachbarte Konzepte entweder durch räumliche Nähe oder durch eine gute Verkehrsanbindung (Schnellstraßen, Hochgeschwindigkeitszug, Fluganbindung) modelliert werden können und Lernbarrieren sich durch natürliche Barrieren (Berge, Wasser, Entfernung) darstellen lassen. Eine solche Karte ist daher sehr anschaulich und sehr leicht interpretierbar. Klar ist, dass die fachlichen Zusammenhänge innerhalb von OOM oder eines anderen Fachgebietes und die Gegebenheiten realer Landkarten (z.B. Weltkarte) nur sehr schwer in Deckung zu bringen sind, da sowohl die fachlichen Zusammenhänge als auch die geografischen Gegebenheiten sehr komplex und sehr vielfältig sind. Diese Erfahrung machte auch die Projektgruppe bei diesbezüglichen Experimenten. Abstrahiert man dagegen von konkreten Landkarten, so kann das mit der Projektgruppe entwickelte Vorgehensmodell (Alex et al. 2002, 90ff) dennoch für die fachdidaktische Strukturierung anderer Wissensgebiete Gewinn bringend eingesetzt werden.

Vorgehensweise

1. ***Erstellung einer Liste mit Fachbegriffen und -konzepten des objektorientierten Modellierens durch Analyse von Fachquellen***
   Im Rahmen dieser Phase werden gedruckte und über das Internet verfügbare Fachquellen (Lehrbücher, Fachzeitschriften, Lehr-Lern-Materialien, Vorlesungsskripte etc.) auf Fachbegriffe und -konzepte des objektorientierten Modellierens hin untersucht. Identifizierte Fachbegriffe und -konzepte werden in einer Liste erfasst.

2. ***Definition von Selektionskriterien und Filterung der Liste der Fachbegriffe und -konzepte***
   Die Aktivitäten dieser Phase verlaufen zweigeteilt. Am Anfang steht die Definition von Selektionskriterien. Im Kern wurden hierbei dieselben Selektionskriterien gewählt, die sich bereits bei der Auswahl von Übungsaufgaben aus fachwissenschaftlichen Lehrbüchern bewährten (vgl. 4.3.1):

   - *Fachkonzepte des Informatikunterrichts,*
   - *Betonung der (objektorientierten) Modellierung,*
   - *Sprachenunabhängigkeit.*

   Zusätzlich können folgende Kriterien verwendet werden, wenn es das Ziel ist, die didaktische Landkarte kompakt zu gestalten:

   - *Vermeidung von Synonymen*: Von synonymen Fachbegriffen und -konzepten wird ein Repräsentant für die Liste ausgewählt. Im Bedarfsfall können diesem die synonymen Termini zugeordnet werden.
   - *Vermeidung zu spezieller Termini:* Allgemeine Begriffe werden spezialisierten Begriffen gegenüber bevorzugt.

   Diese Selektionskriterien werden anschließend auf die Liste (vgl. Punkt 1) angewandt und diese damit gefiltert.

3. ***Strukturierung der Liste der Fachbegriffe und -konzepte***
   Inhaltlich verwandte Begriffe der gefilterten Liste (vgl. Punkt 2) werden in Gruppen und Untergruppen hierarchisch strukturiert. Gruppen und Untergruppen werden benannt.

4. ***Visualisierung als Mind Map***
   Nachbarschaftsbeziehungen innerhalb der Gruppen und Untergruppen (vgl. Punkt 3) der Struktur werden durch eine Mind Map visualisiert, in der räumliche Nähe mit inhaltlicher Nähe korrespondiert.

5. ***Transformation der Mind Map in eine didaktische Landkarte***
   Je nach Umfang der Begriffsliste und Strukturierungstiefe lassen sich Gruppen, Untergruppen und gegebenenfalls weitere Verfeinerungen auf (virtuelle) geografische Gruppenobjekte, wie z.B. Kontinente, Staaten, Bundesländer, Kreise, Städte, Stadtteile etc. abbilden. Räumliche Nähe innerhalb der Mind Map wird auf geografische Nähe innerhalb der didaktischen Landkarte abgebildet. Aus zu definierenden Regeln über die Erreichbarkeit zwischen Begriffen (Verfügbarkeit von Lehr-Lern-Pfaden) und bekannten Lernschwierigkeiten (Lernbarrieren) könnte die geografische Struktur einer didaktischen Landkarte automatisch generiert werden. Im Rahmen der vorliegenden Arbeit erfolgte dies aufgrund anderer Schwerpunkte aber nicht. Alternativ kann hier deshalb die in Abschnitt 6.4.3 empfohlene Darstellung durch Und-Oder-Graphen verwendet werden.

6. ***Verknüpfung der didaktischen Landkarte mit Aufgabenklassen und Explorationsmodulen***
   Jeweils einem oder mehreren Fachbegriffen und -konzepten lassen sich Aufgabenklassen und bzw. oder Explorationsmodule zuordnen sowie andere multimediale Dokumente, wie z.B.

   - Videoaufzeichnungen der Tätigkeiten von Lernenden und Lehrenden,
   - Präsentationsfolien mit oder ohne (Audio-)Annotationen,
   - Listen mit typischen Fragen der Lernenden,
   - Ergebnisse von Lernerfolgskontrollen.

   Aufgabenklassen und Explorationsmodule markieren Stationen im Lehr-Lern-Prozess. Das zielgerichtete Explorieren von Konzepten mit Explorationsmodulen sowie das Bewältigen von konkreten Aufgaben zu den Aufgabenklassen dienen der Förderung des Konzeptverständnisses.

7. ***Vorschläge für Lehr-Lern-Pfade***
   Auf Basis der Struktur sowie der mit ihr verbundenen Lehr-Lern-Materialien sind Vorschläge für konkrete Lehr-Lern-Pfade zu begründen und zu erproben. Erfahrungen mit diesen Pfaden können ihrerseits der Struktur als Materialien hinzugefügt werden. Im Falle einer softwareunterstützten Struktur könnte mittels geeigneter Markierungen (z.B. Färbemechanismen) angezeigt werden, welche Teilstrukturen bereits bearbeitet wurden und welche noch offen sind.

6.6 Zusammenfassung, Fazit und offene Fragen

6.6.1 Zusammenfassung

Bedarf

Auf der Basis von Literaturanalysen zum fachdidaktischen Stand zum objektorientierten Modellieren wurde ein Bedarf sowohl für ein geeignetes Darstellungsmittel als auch für konkrete Darstellungen von fachlichen Erarbeitungsstrukturen des Unterrichts (hier auch als Wissensstrukturen bezeichnet) als Kommunikationsmittel und als Orientierungshilfe für Lehrende und Lernende ausführlich dargelegt (vgl. 2.4.2, Punkt 4; 3.5.3).

Analyse fachwissenschaftlicher und fachdidaktischer Erarbeitungsstrukturen

Den Ausgangspunkt bildete die Analyse fachwissenschaftlicher und fachdidaktischer Erarbeitungsstrukturen zum OOM (vgl. 6.2). Untersucht wurden fachwissenschaftliche Standardwerke zum OOM im Hinblick auf die *Begriffsbildung über Metaphern*, zu *Varianten von Erarbeitungsreihenfolgen* sowie zu *Bezügen zum imperativen und funktionalen Ansatz* (vgl. 6.2.2). Darauf aufbauend wurden fachdidaktische Problembereiche analysiert, wie die *Erleichterung der Fehlererkennung durch Verwendung grafischer Modellierungstechniken*, *Vermeidung von Sprüngen im Abstraktionsniveau* und die *Verknüpfung von Algorithmik und Objektkommunikation bei der grafischen Modellierung*. Schlussfolgerungen für mögliche Erarbeitungsreihenfolgen von Fachkonzepten wurden gezogen.

Grundlagen zu Mapping-Techniken

Für die Repräsentation solcher Strukturen von fachlichen Erarbeitungsreihenfolgen wurden zunächst theoretische Grundlagen aus dem Bereich der Mapping-Techniken analysiert (vgl. 6.3). Unterschieden wurden die Anwendungsfelder *Lehr-Lern-Strategie*, *Unterstützung von Kooperationsprozessen beim gemeinsamen Lernen* sowie *Wissensdiagnose und -modellierung*.

Repräsentation der Struktur von Lehr-Lern-Prozessen

Zur Explikation und zur Strukturierung informatikdidaktischen Fachwissens (vgl. 6.4) wurden zunächst fachdidaktische Funktionen einer grafischen Repräsentation von Erarbeitungsstrukturen begründet (vgl. 6.4.1). Im Einzelnen waren dies die Funktionen *Veranschaulichung der Zusammenhänge für Akteure in Lehr-Lern-Prozessen*, *fachdidaktische Analyse / Diskussion von Lehr-Lern-Prozessen* und *Kontrolle des Bildungsstandes für Lehrende und Lernende*. Aus diesen Funktionen wurden anschließend Anforderungen an eine geeignete Darstellungsform abgeleitet (vgl. 6.4.2). Hierbei handelte es sich um die Anforderungen *Ausdrucksstärke und Übersichtlichkeit*, *Modellierung von Relationen zwischen Fachkonzepten*, *Modellierung von obligatorischen Teilkonzepten und alternativen Lernwegen* und die *Modellierung von Erarbeitungsreihenfolgen*. Verwendet werden sollten hierzu möglichst standardisierte Darstellungsformen. Auf der Basis dieser Anforderungen wurden die graphbasierten Darstellungsformen *Begriffsnetze* (concept maps), *Klassen-* und *Objektdiagramme*, *Semantische Netze* und *Und-Oder-Graphen* untersucht (vgl. 6.4.3). Hierzu wurde eine abstrakte Referenz-Erarbeitungsstruktur entwickelt und unter Verwendung der genannten Darstellungsformen implementiert (vgl. Abbildung 25 auf S. 183). Die Resultate wurden anschließend hinsichtlich der Erfüllung der Anforderungen analysiert. Aufgrund der besonderen Eignung zur Darstellung von verbindlichen und von optionalen Strukturen wurden Und-Oder-Graphen für die Darstellung gewählt. Lerneinheiten werden darin durch Knoten repräsentiert, Erarbeitungs- und Vorkenntnisstrukturen zwischen diesen als Kanten („ist erforderlich für“- oder „ist hilfreich für“-Relation).

Abbildung 27: Vorgehensweise zu Wissensstrukturen

Ersichtlich wird, wie komplizierte Informatikkonzepte auf einfachen aufbauen. Schwerpunkte und Etappen des Lehr-Lern-Prozesses können so leichter dargestellt und analysiert werden.

Vorgehensweise zur Entwicklung einer didaktischen Landkarte

Den Abschluss bildete die Präsentation und Diskussion einer Vorgehensweise zur Entwicklung einer didaktischen Landkarte, die im Rahmen einer studentischen Projektgruppe gestaltet wurde. Erprobt wurde, Erarbeitungsstrukturen mittels einer Landkartenmetapher zu visualisieren. Möglichkeiten und Grenzen dieses Ansatzes wurden deutlich (vgl. 6.5).

6.6.2 Fazit

Zu den wissenschaftlichen Fragestellungen der Arbeit

Begründet wurde der Bedarf für einfach zu handhabende, übersichtliche und ausdrucksstarke Repräsentationen von fachlichen Erarbeitungsstrukturen (vgl. 2.4.2, Punkt 4). Dargelegt wurde, welche Vorteile sich für Lehrende und Lernende durch die Verfügbarkeit eines geeigneten Darstellungsmittels sowie Darstellungen von konkreten fachlichen Zusammenhängen ergeben (vgl. 3.5.3, 6.4.1). Daraus resultierte die Einbeziehung der Wissens- bzw. Erarbeitungsstrukturen als Komponente des didaktischen Systems (vgl. 2.4.3, Frage 1).
Ausführlich entwickelt wurde weiterhin die Vorgehensweise zur Auswahl einer geeigneten Repräsentationsform. Die Ergebnisse hierzu können leicht als Grundlage für die Überprüfung weiterer Darstellungsmittel sowie als Ausgangspunkt für die Entwicklung einer neuen Darstellungsform dienen. Elemente einer ersten Variante eines Prozesses zur Repräsentation von konkreten Erarbeitungsstrukturen wurden dargelegt. Begründet wurden die Wissensstrukturen als Komponente des didaktischen Systems. Damit wird durch die Dokumentation der Vorgehensweise ihrer Entwicklung ebenfalls zur Entwicklung einer Vorgehensweise für die Gestaltung didaktischer Systeme beigetragen (vgl. 2.4.3, Frage 2).
Das Konzept der Wissensstrukturen wurde in der informatischen Bildung in Form von Präsentationen des Autors in Lehrerfortbildungen und auf Fachtagungen erprobt. Mit gleicher Begründung wie bei Frage 2 wurde damit ein, in Relation zu den Aufgabenklassen (vgl. 4.7.2) und den Explorationsmodulen (vgl. 5.6.2) zwar deutlich geringerer, Beitrag zur Beantwortung der dritten Fragestellung der vorliegenden Arbeit geleistet (vgl. 2.4.3, Frage 3).

Zur Vorbereitung von Bildungsstandards

Die folgenden Ausführungen knüpfen an die entsprechenden Erörterungen zu Aufgabenklassen (vgl. 4.7.2) an. Durch die Repräsentation von fachlichen Erarbeitungsstrukturen wird ersichtlich, wie komplizierte Informatikkonzepte auf einfachen Konzepten aufbauen. Schwerpunkte und Etappen von Lehr-Lern-Prozessen können so leichter dargestellt und analysiert werden und die Weitergabe fachdidaktischen Wissens an nachfolgende Generationen von Informatiklehrenden wird gefördert. Wird durch entsprechende Lernerfolgskontrollen überprüft, ob die dargestellten Etappen des Lehr-Lern-Prozesses von Lernenden erreicht wurden, so korrespondieren verschiedene, auf einem Lernpfad liegende Etappen zu verschiedenen Stufungen der erreichten Fachkompetenz. Zu klären ist auch hier, ähnlich wie bei den Aufgabenklassen, welche Etappen Lernende in ihren individuellen Bildungsprozessen zum informatischen Modellieren und anderen wichtigen Schwerpunkten des Informatikunterrichts (vgl. 2.2.3) erreichen sollen.

6.6.3 Offene Fragen

Die offenen Fragen lassen sich folgenden Bereichen zuordnen:

- Systematische Evaluation in der informatischen Bildung
  Das Konzept wurde im Rahmen von Lehrerfortbildungen (vgl. 4.6.2) und auf Fachtagungen präsentiert und stieß dort auf reges Interesse und positive Resonanz bei den Teilnehmerinnen und Teilnehmern. Über Präsentationen hinausgehende Erprobungen sowie eine systematische, empirische Evaluation sowohl im Informatikunterricht als auch in der Informatiklehrerbildung unter Einbeziehung von Vergleichsgruppen sind für diese Komponente des didaktischen Systems noch offen.
- Befragung von Informatiklehrenden zu Erarbeitungsstrukturen
  Für die Weiterentwicklung der fachdidaktischen Diskussion zur Gestaltung von Lehr-Lern-Prozessen zum objektorientierten Modellieren sind auf systematische Weise Varianten von Erarbeitungsstrukturen zu Fachkonzepten des objektorientierten Modellierens durch Befragung von Informatiklehrenden zu erheben und in der gewählten Darstellungsform bereitzustellen. Das Ziel könnte hierbei ein Netz von Varianten von erprobten Erarbeitungsstrukturen sein, welches Lehrenden zur Verfügung gestellt werden könnte, zur Anregung und zur Bewertung eigener Lehr-Lern-Prozesse. Auf dieser Basis kann die im Abschnitt 6.5 präsentierte Vorgehensweise verfeinert und weiterentwickelt werden.

7 Zusammenfassung, Fazit und offene Fragen

7.1 Zusammenfassung

Bedarf

In Deutschland befindet sich die informatische Bildung in der Sekundarstufe in einem konzeptionellen Wandel von starker Betonung der prozeduralen Programmierung hin zum objektorientierten Modellieren (OOM). Obwohl dies von Informatikdidaktikern empfohlen, betont und bereits in einigen Curricula der Bundesländer verankert wurde, hat dieser Wandel die Mehrzahl der Schulen noch nicht erreicht. Ein scheinbarer Wandel fand statt von prozeduraler hin zur objektorientierten Programmierung. Als eine wesentliche Ursache für die noch seltene Berücksichtigung von OOM in der Schulpraxis wurde im Rahmen der vorliegenden Arbeit, ausgehend von einer Literaturanalyse des fachdidaktischen Standes zum OOM (vgl. Kapitel 2), ein Mangel an verfügbaren Unterrichts- und Übungsbeispielen, unterrichtsgeeigneten Informatiksystemen zur Förderung der Aneignung von Fachkonzepten, Strukturierungshinweisen für den Unterricht und damit verbundene Einsatzkonzepte identifiziert (vgl. 2.4.2). Eine ähnliche Ausgangslage zeigt sich in anderen, wichtigen Themenfeldern des Informatikunterrichts (z.B. Wirkprinzipien von Informatiksystemen), die einerseits empfohlen, für die andererseits aber noch zu wenige Lehr-Lern-Materialien existieren bzw. zugänglich sind.

Konzeption eines didaktischen Systems

Mit den Zielen, einerseits den identifizierten Bedarf beim OOM nachhaltig zu decken und andererseits auch in anderen Bereichen der informatischen Bildung zu einer Bereicherung der Lehr-Lern-Prozesse beizutragen, wurde ein als didaktisches System bezeichneter Verbund, bestehend aus traditionellen und neuen Komponenten des Lehr-Lern-Prozesses, konzipiert (vgl. Kapitel 3). Gestaltungsleitend für die identifizierten Komponenten des didaktischen Systems für OOM (Aufgabenklassen, Explorationsmodule, Wissensstrukturen) war der im Kapitel 2 identifizierte Bedarf (vgl. 2.4.2). Dargelegt und exemplarisch erprobt wurde, wie diese Komponenten zu einer Bereicherung traditioneller Lehr-Lern-Prozesse beitragen

- durch *Anwendung in Lehr-Lern-Prozessen,*
- als *Gestaltungsmittel für Lehr-Lern-Prozesse,*
- durch *Förderung der fachdidaktischen Kommunikation und Diskussion zu Lehr-Lern-Prozessen.*

Die drei genannten Komponenten des didaktischen Systems wurden ausführlich entwickelt, exemplarisch angewandt und in der informatischen Bildung erprobt.

Aufgabenklassen

Bei Aufgabenklassen (vgl. Kapitel 4) handelt es sich um abstrakte Aufgabenrahmen, die, ausgehend von einem Bedarf für unterrichtsgeeignete Aufgaben zum objektorientierten Modellieren, aus einem, zu diesem Zweck entwickelten, Auswahl-, Abstraktions- und Strukturierungsprozess hervorgingen, der auf über 320 Übungsaufgaben in fachwissenschaftlichen Lehrbüchern zum OOM angewandt wurde. Damit verbunden wurde eine Methodik zur Gestaltung von Aufgaben zum OOM zur Erfüllung vielfältiger fachdidaktischer Funktionen entwickelt, um diesen Prozess speziell für mit OOM wenig erfahrene Lehrende zu erleichtern. Lernenden geben Aufgabenklassen Orientierung, indem sie abstrakte Aufgabenrahmen mit im Unterricht erarbeiteten Lösungsstrategien verbinden. Die Bewältigung von Problemen mit ähnlichen Aufgaben wird damit vereinfacht. Erfolgreich erprobt wurde das Konzept in der Sekundarstufe II und in der Informatiklehrerbildung[99].

[99] vgl. Zusammenfassung zu Aufgabenklassen im Abschnitt 4.7.1

Explorationsmodule

Aufgrund eines identifizierten Mangels an unterrichtsgeeigneten Informatiksystemen zur Unterstützung der Aneignung von Fachkonzepten des objektorientierten Modellierens wurde eine Vorgehensweise zur Entwicklung von Software zur Anregung explorativen Lernens im Bereich objektorientierter Basiskonzepte (so genannte Explorationsmodule) vorgestellt und angewandt (vgl. Kapitel 5). Lernenden wird damit ein neuer, handlungsorientierter Zugang zu Fachkonzepten eröffnet, indem sie mit Repräsentationen von Fachkonzepten unter Verwendung systematischer Explorationsstrategien interagieren, anstatt ausschließlich zu programmieren. Dadurch schärfen sie nicht nur ihre Fach-, Methoden- und Sozial-, sondern auch ihre Lernkompetenz. Das Gestaltungskonzept wurde prototypisch implementiert, anschließend mit einem Konzept für das Lernen mit Explorationsmodulen verknüpft und erfolgreich in der informatischen Bildung erprobt[100].

Wissensstrukturen

Den Ausgangspunkt von Kapitel 6 stellte eine Begründung des fachdidaktischen Bedarfs einerseits für ein Repräsentationsmittel und andererseits für konkrete Repräsentationen von Erarbeitungsstrukturen von Fachkonzepten (als Wissensstrukturen bezeichnet) dar. Ausgegangen wurde hierzu von der Analyse fachwissenschaftlicher und fachdidaktischer Erarbeitungsstrukturen. Auf der Basis von Ergebnissen zu „Mapping Techniken“ wurden verschiedene fachdidaktische Funktionen einer Repräsentation der Struktur von Lehr-Lern-Prozessen begründet, Anforderungen an Darstellungsformen präzisiert und verschiedene graphbasierte Darstellungsformen im Hinblick auf die Erfüllung der Anforderungen analysiert. Begründet wurde, warum sich Und-Oder-Graphen von den betrachteten Darstellungsformen am besten eignen, die identifizierten Anforderungen zu erfüllen. Ein im Rahmen einer studentischen Projektgruppe entwickeltes Vorgehensmodell zur Gestaltung einer didaktischen Landkarte wurde vorgestellt[101].

7.2 Fazit

7.2.1 Zu den wissenschaftlichen Fragestellungen der Arbeit

Die wissenschaftlichen Fragestellungen der vorliegenden Arbeit lauteten (vgl. 2.4.3):

1. *Welches sind Komponenten eines didaktischen Systems für OOM?*
2. *Wie kann man bei der Entwicklung didaktischer Systeme vorgehen?*
3. *Wie kann dieses didaktische System in der Lehrerbildung und im Informatikunterricht der Sek. II exemplarisch erprobt werden?*

Zu Frage 1:

Für alle drei im Rahmen der vorliegenden Arbeit beschriebenen Komponenten wurde, ausgehend vom Bedarf, der Gewinn für verschiedene Akteure in Lehr-Lern-Prozessen, wie z.B. Lehrende, Lehramtsstudierende, Lernende, Entwicklerinnen und Entwickler von Lehr-Lern-Materialien, dargelegt (vgl. 4.7.2, 5.6.2, 6.6.2). Begründet wurde damit, warum Aufgabenklassen, Explorationsmodule und Wissensstrukturen wertvolle Komponenten eines didaktischen Systems darstellen. Damit konnte auch gezeigt werden, dass zwar nicht die konkreten Ausprägungen, wohl aber der prinzipielle Gewinn durch Aufgabenklassen, Explorationsmodule und Wissensstrukturen, vom gewählten Themenfeld unabhängig sind. Aufgabenklassen,

[100] vgl. Zusammenfassung zu Explorationsmodulen im Abschnitt 5.6.1
[101] vgl. Zusammenfassung zu Wissensstrukturen im Abschnitt 6.6.1

Explorationsmodule und Wissensstrukturen lassen sich daher als Komponenten eines (allgemeinen) didaktischen Systems verstehen. Damit ist nicht ausgeschlossen, dass es sowohl nur für das Themenfeld OOM geeignete als auch für ein allgemeines didaktisches System nützliche, weitere Komponenten geben kann. Es ist aus diesem Grund davon auszugehen, dass der Ansatz des didaktischen Systems weitere Verfeinerungen, Ergänzungen (z.B. andere Themenbereiche) und Spezialisierungen (z.B. andere Zielgruppen) erfahren wird.

Zu Frage 2:
Die Vorgehensweise bei der Entwicklung der Aufgabenklassen, Explorationsmodule und Wissensstrukturen wurde ausführlich dokumentiert (vgl. 4.7.2, 5.6.2, 6.6.2) und abstrahiert. Durch die Verfügbarkeit dieser Vorgehensweisen ist ein Leitfaden gegeben, mit dem weitere didaktische Systeme für andere Themenbereiche oder andere Zielgruppen gestaltet werden können. Wird ein didaktisches System für ein anderes Anwendungsfeld um Komponenten ergänzt, so müssen für diese Komponenten noch analoge Vorgehensweisen beschrieben, erprobt und auf ihr Verallgemeinerungspotential hin untersucht werden.

Zu Frage 3:
Die Vorgehensweise bei der Erprobung der Komponenten des didaktischen Systems in der informatischen Bildung und damit verbundene Ergebnisse wurden dokumentiert (vgl. 4.7.2, 5.6.2, 6.6.2). Gezeigt wurde, wie eine exemplarische Erprobung durch Integration in den Informatikunterricht der Sek. II sowie in die Informatiklehreraus- und -weiterbildung erfolgen kann. Involviert in diesen Prozess waren über 100 Lernende im Informatikunterricht der Sek. II und in Studienwerbeveranstaltungen, 10 Lehramtsstudierende in der Informatiklehrerausbildung sowie über 200 Lehrende in Veranstaltungen zur Informatiklehrerfortbildung. In der Verknüpfung zeigt sich, wie ein didaktisches System in der informatischen Bildung mit dem Ziel der Gestaltungsrückkopplung zur Qualitätssicherung erprobt werden kann.
Offen bleibt hierbei allerdings die Frage, ob und wenn ja, wie ein didaktisches System als Verbund in der informatischen Bildung erprobt werden kann. Da sich das didaktische System als Angebot an Lernende und Lehrende zur Bereicherung traditioneller Lehr-Lern-Prozesse versteht, ist es durchaus erstrebenswert, dass zunächst Elemente einzeln, später dann in ihrer Verknüpfung erprobt werden, um sowohl Lehrende als auch Lernende schrittweise an die fachdidaktischen Angebote zu den fachlichen Zusammenhängen heranzuführen.

7.2.2 Zur Vorbereitung von Bildungsstandards

Zur Vorbereitung von Standards für die informatischen Bildung sind vergleichbare Lehr-Lern-Materialien erforderlich. Das Konzept der didaktischen Systeme zielt darauf ab, eine Sammlung aufeinander abgestimmter Lehr-Lern-Materialien bereitzustellen, die in einem Unterrichtsszenario (z.B. zum objektorientierten Modellieren) je nach Zielgruppe sehr flexibel zu verschiedenen Kompetenzen führen können. Dabei ist zunächst in der Didaktik der Informatik ein Konsens darüber herzustellen, welche Kompetenzstufen Lernende bezüglich der verbindlichen und fakultativen Inhalte des Informatikunterrichts erreichen sollen.
Ein besonders wichtiger Beitrag hierzu wird durch die Komponenten Aufgabenklassen und Wissensstrukturen geleistet (vgl. 4.7.2, 6.6.2). Sind die Kompetenzstufen bekannt, so kann diskutiert werden, welche Aufgabenklassen Lernende zu ihrem Erreichen bewältigen können müssen. Im Hinblick auf das im Rahmen dieser Arbeit betrachtete Themenfeld wird die vorgeschlagene Klassifikationsstruktur für Aufgaben zum objektorientierten Modellieren (vgl. Abbildung 7 auf S. 77) als Grundlage für die Bildung von verfeinerten Kompetenzklassen empfohlen. Definierte Kompetenz- und Aufgabenklassen zu inhaltlichen Schwerpunkten des Informatikunterrichts fördern die Vergleichbarkeit von Bildungsergebnissen.

Durch die Repräsentation von fachlichen Erarbeitungsstrukturen können Schwerpunkte und Etappen von Lehr-Lern-Prozessen leichter dargestellt und analysiert werden. Wird durch entsprechende Lernerfolgskontrollen überprüft, ob eine solche Etappe des Lehr-Lern-Prozesses von Lernenden erreicht wurde, so korrespondieren verschiedene, auf einem Lernpfad liegende Etappen zu verschiedenen Stufungen der erreichten Fachkompetenz.
Explorationsmodule schließlich fördern das Erreichen der gewünschten Bildungsqualität, indem das Aneignen objektorientierter Fachkonzepte und der Aufbau kognitiver „Landkarten" zum Fachgegenstand unterstützt werden. Außerdem eignen sie sich hervorragend für das Vorabinformieren über Bildungsprozesse und können damit zu einem fundierteren Fächerwahlverhalten von Schülerinnen und Schülern beitragen. Damit wird ein Beitrag zur Reduktion der hohen Abwahlquoten des Faches geleistet. Bildungsstandards in der Sek. II können nur erreicht werden, wenn es Lernende gibt, die das Fach wählen und es hinreichend lange belegen (vgl. 5.6.2).

7.3 Offene Fragen

Offene Fragen zum Konzept des didaktischen Systems für OOM wurden bereits bei den einzelnen Komponenten des Systems diskutiert (Aufgabenklassen: 4.7.3; Explorationsmodule: 5.6.3; Wissensstrukturen: 6.6.3). Nachfolgend werden diese offenen Fragen systematisiert:

- Umfassende, systematische Evaluation mit Vergleichsgruppen im Hinblick auf Wirksamkeitsaspekte
  Offen ist sowohl bezüglich der Komponenten als auch bezüglich des Gesamtsystems, eine über einzelne Erprobungen hinausgehende, systematische Evaluation unter Verknüpfung von vielfältigen Instrumenten empirischer Sozialforschung (z.B. leitfadengestützte Interviews, schriftliche Befragungen, Videoanalyse[102]) in verschiedenen Bereichen der informatischen Bildung, in der ausgewählte Gruppen (von Lehrenden und / oder Lernenden) über einen längeren Zeitraum mit Ausschnitten des vorgestellten Konzepts arbeiten und untersucht wird, ob und wenn ja, welche Veränderungen bezüglich vorher festzulegender Untersuchungsvariablen sich bei diesen in Relation zu Vergleichsgruppen ergeben, in denen das Konzept, bei ansonsten inhaltsgleicher Ausbildung, nicht angewandt wird.
- Überprüfung der Übertragbarkeit des Konzepts auf andere Themenbereiche
  Im Rahmen der vorliegenden Arbeit konnten verschiedene Indizien und Belege für eine Übertragbarkeit des Konzepts auch auf andere Themenbereiche, nicht nur der Informatik, gesammelt werden. Die vollständige Übertragung auf einen anderen Themenbereich, zur Überprüfung der Übertragbarkeit, ist noch offen.
- Weiterentwicklung / Verfeinerung der Komponenten
  Zu untersuchen ist, wie durch geeignete Verknüpfung mit Ergebnissen aus Lehr-Lern-Prozessen (z.B. Schülerlösungen, Fehlerbilder zu Aufgabenklassen, Videoaufzeichnungen von Lehr-Lern-Prozessen) sowie durch Befragung von Lehrenden zu Erfahrungen (z.B. mit bestimmten Unterrichtsgestaltungen) und Verfügbarmachung der Ergebnisse der Gewinn für alle Akteure in Lehr-Lern-Prozessen weiter erhöht werden kann.
- Erweiterung des didaktischen Systems um Komponenten
  Da die Nützlichkeit für Akteure in Lehr-Lern-Prozessen im Vordergrund steht, wurde das didaktische System als fachdidaktischer Verbund mit Erweiterungs- und Verfeinerungsmöglichkeiten begründet. Schubert und Schwill (2004) schlagen auf der Basis von

[102] vgl. (Magenheim / Schubert 2000)

Publikationen des Autors der vorliegenden Arbeit eine Erweiterung des didaktischen Systems um Entwurfsmuster für Lernprozesse vor. In der Verknüpfung von Handlungsmustern des Unterrichts (vgl. Meyer 1994) zu Entwurfsmustern für Lernprozesse und deren grafische Repräsentation, z.B. unter Verwendung von Ablaufplänen, sehen sie einen möglichen zukünftigen Schwerpunkt informatikdidaktischer Forschung.

Anhang

A Hospitation von Unterricht (Konzept „Von Stiften und Mäusen“)

Hospitationsprotokoll

Termin: 30.05.2000, 5.-6. Std. (12:00-13:35 Uhr)
Ort: Gymnasium im Raum Dortmund
Kurs: Informatik-Grundkurs mit einer Schülerin und acht Schülern

Das Hospitationsprotokoll entstand zusammen mit einer Unterrichtsaufzeichnung (Video) im Rahmen des vom Universitätsverbund Multimedia geförderten Projektes „Multimediale Evaluation in der Informatiklehrerausbildung – MUE“[103] (vgl. Magenheim / Schubert 2000).

Im nachfolgenden Protokoll werden folgende Abkürzungen verwendet: L. bezeichnet den Fachlehrer (männlich), S. eine Schülerin oder einen Schüler.

1. Stunde (12:00-12:45 Uhr)

12:00
- L.: Motivation des Themas „professionelle Gruppenarbeitsweise“ am Beispiel „Simulation eines Spielautomaten“
- L.: „Welche Schritte sind zur Projektplanung erforderlich?“
- Sammlung von Schülerideen

12:03
- S.: Aufbau und Bedienung festlegen (Pflichtenheft)
- S.: in Teile zerlegen, Aufgabenverteilung
- S.: Zerlegung in Klassen, weil objektorientierte Lösung gewünscht
- L. ergänzt

12:05
- L. gibt Hinweise auf Probleme beim Klassen finden
- L.: Was noch?
- S.: Zusammenhänge zwischen den Klassen beschreiben, UML-Diagramm erstellen
- L.: Was noch?
- S.: „Das, was in den Klassen passieren soll?!“

12:07
- L. ergänzt Attribute und Methoden, Nachrichtenaustausch zwischen den Objekten, genaues Protokoll der Klassen

Anm.: Klasse sehr still (durch Kamerasituation?)

12:10
- L.: Realisierung als Phase
- S.: „Zusammensetzen und Testen“
- L. gibt Hinweise auf Rückschritte im Ablauf

[103] URL: http://mue.die.informatik.uni-siegen.de/ (aufgerufen am 22.11.03)

12:12
- L. gibt Auftrag an die S.: Benutzungsoberfläche für den Spielautomaten in Gruppenarbeit entwerfen (Folgende Gruppen bilden sich: 2m, 3m, 2m, m+w)
- L: geht herum und hilft

12:17
- L. und ein S. diskutieren über die Funktionsweise des Automaten
- L. gibt Hinweis an alle S.: so einfach, wie möglich
- L. diskutiert mit Schülergruppe

12:25
- L.: ein S. soll die Lösung anzeichnen
- ein S. zeichnet seine Lösung an
- L. stellt die Lösung dem Kurs zur Diskussion

12:27
- S. diskutieren die Lösung, es gibt unterschiedliche Ansichten über den Punktestand
- L. fordert einen S. auf, Regeln für das Verhalten des Spielautomaten aufzuschreiben
- S. vervollständigt Tafelbild

12:31
- S. diskutieren die Schritte
- S.: eine Regel (z.B. Rollen stoppen mit 5s Verzögerung) hat Nachteile
- S.: Rollen mit unterschiedlicher Geschwindigkeit drehen
- weitere Möglichkeiten, Probleme, Ergänzungen werden gesammelt, die S. werden aufgefordert zu entscheiden, welche Funktionalität der Automat aufweisen soll
- Modifikation der Lösung, Benutzungsoberfläche wird erweitert

12:35
- L.: „Ist die Lösung eindeutig?“
- S.: Frage der Geldeingabe muss geklärt werden
- Algorithmus wird in Schritten entworfen

12:37
- L.: Berücksichtigung von Ausnahmesituationen (kein Geld mehr im Automat, spezielle Gewinnsituationen) und Gewinnbedingungen festlegen

2. Stunde (12:50-13:35 Uhr)

12:47
- Festlegung durch L.: Farbige Quadrate als Symbole für die drei Rollen des Spielautomaten
- L. und S.: Diskussion der Gewinnbedingungen
- L. Beschreibung der Rollen (je 5 Farben)
- Vorschlag S.: je dreimal gleiche Farbe führt zu Gewinn (Höhe abhängig von der Farbe)

12:55
- L.: Auffinden von Klassen und Objekten

- S.: Vorschläge für Klassen: Rolle, Startknopf, Geldeinwurf, Geldausgabefeld, Geldausgabeknopf
- S.: Knopf?
- L. Zusammenfassung erst später (Anm.: gemeint war Anwendung der Vererbung)
- S.: Punktestand, Spielautomatenanwendung

12:59
- L: Auftrag für die S.: Erstellen eines Klassendiagramms in Partnerarbeit
- L. geht herum und hilft

13:08
- zwei S. stellen ihr Ergebnis an der Tafel vor (Klassenkarten, fixiert durch Magnete, hat- und ist-Beziehungen werden gezeichnet)

13:15
- L. und S. kommentieren die Lösung und modifizieren das Diagramm
- Zusammenfassung von Klassen
- S. ergänzt kennt-Beziehungen

13:20
- L. stellt alternative Lösung vor
- L. lenkt das Gespräch auf den Nachrichtenaustausch
- S. und L. zeichnen Nachrichten in modifiziertes Klassendiagramm (Anm.: Klassendiagramm wird zu einer Art Kollaborationsdiagramm)
- L. Unterschied zwischen Anfrage / Auftrag?
- L. Was fehlt?

13:28
- S. Rückgabe der Endstände der Rolle
- L. Hausaufgabe: Aufteilung der Klassenprotokolle an verschiedene Schülergruppen

B Verzeichnis der ausgewählten Übungsaufgaben

Tabelle 42 und Tabelle 43 zeigen die mittels des im Abschnitt 4.3.1 beschriebenen Verfahrens ausgewählten Übungsaufgaben aus den Lehrbüchern von Rumbaugh et al. (1993) bzw. Balzert (1999). In diesen Tabellen bezeichnet *Kapitel Nr.* die Nummer des Kapitels in dem jeweiligen Lehrbuch, dessen zugehöriges Kapitelthema in der Spalte *Kapitelüberschrift* nachzulesen ist. In der Spalte *#Aufgaben* ist die Anzahl der dem jeweiligen Kapitel zugeordneten Übungsaufgaben aufgeführt. Die folgenden Spalten halten die Ergebnisse des Auswahlprozesses fest. Die ersten beiden Spalten enthalten die Ergebnisse zum statischen Systemmodell, die beiden weiteren Spalten die Ergebnisse der Analyse aller Übungsaufgaben. Die Anzahl der jeweils ausgewählten Übungsaufgaben ist in den mit # überschriebenen Spalten nachzulesen. Die konkreten Aufgabennummern sind in den mit *Nr.* überschriebenen Spalten festgehalten.

Kapitel Nr.	Kapitelüberschrift	#Aufgaben	ausgewählte Aufgaben			
			statisches Systemmodell		alle Aufgaben	
			#	Nr.	#	Nr.
1	Einführung	8	4	5-8	4	5-8
Teil I: Modellierungskonzepte						
2	Modellierung als Entwurfstechnik	6	4	1-4	6	1-6
3	Objektmodellierung	32	30	1-27, 29, 31, 32	30	1-27, 29, 31, 32
4	Weiterführende Konzepte der Objektmodellierung	18	17	1-5, 7-18	18	1-18
5	Dynamische Modellierung	19	0	-	15	1-10, 12, 13, 16, 18, 19
6	Funktionale Modellierung	11	0	-	8	1-6, 8, 9
Teil II: Entwurfsmethodologie						
7	Einführung in die Entwurfsmethodologie	2	0	-	0	-
8	Analyse	34	0	-	30	2-25, 28-32, 34
9	Systementwurf	15	0	-	6	3-8
10	Objektentwurf	28	0	-	13	5-12, 14-16, 22, 24
11	Zusammenfassung der OMT-Methodologie	5	0	-	5	1-5
12	Vergleich von Methodologien	3	0	-	1	2
Teil III: Implementierung						
13	Vom Entwurf zur Implementierung	0	0	-	0	-
14	Programmierstil	5	0	-	0	-
15	Objektorientierte Sprachen	18	0	-	0	-
16	Nicht-objektorientierte Sprachen	12	0	-	0	-
17	Relationale Datenbanken	23	0	-	0	-
Teil IV: Anwendungen						
18	Objektdiagramm-Compiler	5	0	-	5	1-5
19	Computeranimation	5	0	-	0	-
20	CAD-System zur elektrischen Stromverteilung	7	0	-	1	2
	Summe:	**256**	**55**		**142**	

Tabelle 42: Ausgewählte Übungsaufgaben aus Rumbaugh et al. (1993)

Kapitel Nr.	Kapitelüberschrift	#Aufgaben	ausgewählte Aufgaben			
			statisches Systemmodell		alle Aufgaben	
			#	Nr.	#	Nr.
1	1: Objektorientierte Softwareentwicklung	4	0	-	4	1-4
	2: Konzepte und Notation der objektorientierten Analyse					
2	Basiskonzepte	5	3	2, 3, 5	5	1-5
3	Statische Konzepte	3	3	1-3	3	1-3
4	Dynamische Konzepte	4	0	-	4	1-4
5	3: Analysemuster und Beispielanwendungen	3	0	-	3	1-3
6	4: Checklisten zur Erstellung eines OOA-Modells	5	0	-	5	1-5
7	Statisches Modell	4	0	-	4	1-4
8	Dynamisches Modell	5	0	-	4	1-4
	5: Gestaltung von Benutzungsoberflächen					
9	Teil 1	4	0	-	0	-
10	Teil 2	3	0	-	2	2-3
	6: Konzepte und Notation des objektorientierten Entwurfs					
11	Teil 1	4	0	-	3	1-3
12	Teil 2	3	0	-	2	1, 2
13	7: Entwurfsmuster	3	0	-	3	1-3
	8: Datenbanken					
14	Relationale und objektrelationale Abbildung	5	0	-	0	-
15	Objektorientierte Datenbanken	4	0	-	0	-
16	9: Verteilte objektorientierte Anwendungen	4	0	-	0	-
	10: Erstellen eines Entwurfsmodells mittels Drei-Schichten-Architektur					
17	Teil 1	3	0	-	3	1-3
18	Teil 2	4	0	-	1	4
	Summe:	**70**	**6**		**46**	

Tabelle 43: Ausgewählte Übungsaufgaben aus Balzert (1999)

C Strukturierte Sammlung von Aufgabenklassen

C.1 Aufgabenklassen zum statischen Modell (1. Fassung)

Die Struktur der ersten Fassung der Aufgabenklassen zum statischen Modell wird erläutert im Abschnitt 4.3.3.

1. Objekte, Klassen und Relationen identifizieren		
	!	textuelle Beschreibung eines Systems
	?	Liste von Objekten erstellen, die im System eine Rolle spielen könnten
	!	textuelle Beschreibung eines Systems
	?	im Text genannte Objekte, Klassen und Relationen identifizieren
	!	Liste von Relationen als natürlichsprachliche Sätze
	?	Art der Relation (Generalisierung, Aggregation oder Assoziation) bestimmen
2. Objekte, Klassen und Relationen charakterisieren		
a. Merkmal zuordnen		
	!	Liste von Klassen-, Attribut- und Operationsnamen mit kurzer Beschreibung
	?	Zuordnen von Attributen und Operationen zu Klassen
b. Merkmal beschreiben		
	!	Liste von Klassen mit Operationen
	?	Verhalten der Operation bzgl. der Instanzen der Klasse erläutern
c. Merkmal analysieren		
	!	sehr große Objektgruppe, Anwendungszweck
	?	identifizierende(s) Attribut(e) für gegebenen Anwendungszweck angeben
	!	Objekt(e) aus der Lebenswelt mit bestimmten Attributen, Anwendungszweck
	?	relevante Attribute für gegebenen Anwendungszweck auswählen und begründen
	!	verschiedene unabhängige Systeme, die dasselbe Objekt identifizieren wollen; Vorschläge für verschiedene Identifizierungsmethoden
	?	Diskussion von Vor- und Nachteilen der Methoden
	!	Beschreibung eines Informatiksystems, Anwendungszweck
	?	relevante von irrelevanter Information in Bezug auf den gegebenen Anwendungszweck trennen
d. Merkmal spezifizieren		
	!	Klassendiagramm
	?	Attribute, Operationen und Relationen spezifizieren
	!	Klassendiagramm
	?	für jede Klasse mit Operationen: Argumente der Operation angeben und Auswirkung auf Klasseninstanzen beschreiben, für jede Operation Liste der sich fortpflanzenden Operationen erstellen
3. Objekte, Klassen und Relationen strukturieren		
a. Modell beschreiben		
	!	Objekt- oder Klassendiagramm
	?	Beschreibung mit eigenen Worten
b. Modell analysieren		
	!	Liste von Klassennamen aus der Lebenswelt
	?	diskutieren, was Begriffe gemeinsam haben; Liste um weitere Klassennamen ergänzen
	!	verschiedene Objektdiagramme für dieselbe Situation, Algorithmus zur Lösung einer bestimmten Aufgabe
	?	diskutieren, welche Repräsentation für den Algorithmus am geeignetsten ist
c. Modell modifizieren		
	!	unvollständiges, schlechtes oder fehlerhaftes Objekt- oder Klassendiagramm
	?	bestimmte Modellteile ergänzen und begründen (z.B. Klassen, Attribute, Operationen (Mehrfachzuordnung möglich), Relationen, Multiplizitäten, Assoziations- und Rollennamen); bestimmte semantische Änderung / Erweiterung am Modell vornehmen; Attribute, die Zeiger auf andere Klassen sind, identifizieren und durch Relationen ersetzen; Assoziationen in Aggregationen umwandeln, wo sinnvoll

3. Objekte, Klassen und Relationen strukturieren	
d. Modell konstruieren	
! ?	Objektdiagramm, ggf. textuelle Beschreibung dazugehöriges Klassendiagramm entwickeln: Multiplizitäten berücksichtigen (z.B. Polygon erfordert min. drei Punkte); Art der Relation zwischen Klassen (Aggregation / Assoziation) aufgrund der textuellen Beschreibung spezifizieren (z.B. Objekt gehört genau zu einem Objekt vs. Objekt wird von verschiedenen anderen Objekten gemeinsam genutzt); Reihenfolgen von über Relationen verbundenen Objekten spezifizieren
! ?	Klassendiagramm Objektdiagramm für textuell beschriebene oder grafisch dargestellte Situation entwickeln
! ?	Liste von Klassennamen Klassendiagramm erstellen: min. vorgegebene Anzahl von Relationen (Aggregation, Assoziation, Generalisierung) zwischen den Klassen einzeichnen (Assoziations- und Rollennamen berücksichtigen), weitere Klassen ergänzen, wo sinnvoll; vorgegebene Anzahl von Attributen und Operationen ergänzen
! ?	Liste von Klassennamen Klassen in Pakete gruppieren, Pakete benennen

4. statisches Systemmodell konstruieren	
! ?	textuelle Beschreibung einer Problemstellung Klassen- und / oder Objektdiagramm erstellen
! ?	- ein Metamodell objektorientierter Grundbegriffe erstellen (z.B. Klasse, Attribut, Assoziation, Rolle, Multiplizität); Hinweise, welche Techniken verwendet werden dürfen (z.B. Klasse, Attribut, Assoziation)
! ?	Metamodell objektorientierter Grundbegriffe diskutieren, ob bestimmter Aspekt vom Modell unterstützt wird; gegebenes Modell überarbeiten, so dass ein bestimmter Aspekt unterstützt wird

C.2 Aufgabenklassen zum statischen und dynamischen Modell (verfeinerte und überarbeitete Fassung)

C.2.1 Überblick

Die sich aus der Analyse der Übungsaufgaben zum OOM nach dem in Abschnitt 4.3 beschriebenen Verfahren ergebende Sammlung der Aufgabenklassen ist wie folgt strukturiert. Jede Aufgabenklasse ist einer Kategorie des Fachkerns zugeordnet und erhält darin eine laufende Nummer. Die Spalte „K“ enthält eine Kennung zur Charakterisierung von Gegenstand und Aufgabentyp (vgl. S. 78). In der letzten Spalte wird zu jeder Aufgabenklasse mindestens ein Beispiel aus einem der analysierten Lehrbücher referenziert. Hierfür wird folgende Kurznotation verwendet: *{R|B}, <Seitenzahl>, <Aufgabennummer>*. Im Lehrbuch von Rumbaugh et al. (1993) ist den Aufgabennummern zusätzlich die Kapitelnummer vorangestellt. Da Seitenzahl und Aufgabennummer eine Aufgabe eindeutig identifizieren, wird hierbei auf die vorangestellte Kapitelnummer verzichtet. *R, 15, 9* referenziert also die Aufgabe 9 auf S. 15 im Lehrbuch von Rumbaugh et al. (1993). *B, 16, 3 / 4* referenziert die Aufgaben 3 und 4 auf S. 16 im Lehrbuch von Balzert (1999). *R, 23, 1 / 24, 2* referenziert analog die Aufgaben 1 auf S. 23 und 2 auf S. 24. Da sich Wissens- und Verständnisfragen, ebenso wie Aufgaben zur Metamodellierung, sowohl auf das statische als auch auf das dynamische Modell beziehen können, werden diese Aufgabenklassen hier separat aufgeführt.

C.2.2 Wissens- und Verständnisfragen zu objektorientierten Konzepten

Nr.		Beschreibung	K	Beispiel(e)
1	!	-		
	?	Vor- und / oder Nachteile eines objektorientierten Konzepts nennen	I.1	B, 15, 1 / 136,1 / 253, 1
2	!	Liste verwandter objektorientierter Konzepte		
	?	Konzepte gegeneinander abgrenzen	I.2.1	B, 16, 3 / 38, 4 / 253, 1 / 302, 3
3	!	-		
	?	objektorientiertes Konzept beschreiben	I.2.2	B, 15, 2 / 37, 1 / 136,1 / 253, 1 / 393, 1

C.2.3 Statisches Modell

C.2.3.1 Objekte und Objektstrukturen

Nr.		Beschreibung	K	Beispiel(e)
1	!	Objekt(e), Verwendungszweck, Liste von Attributen		
V1	?	relevante Attribute(e) für den Verwendungszweck auswählen und begründen	II.3.1	R, 23, 1 / 24, 2
V2	?	Vor- und Nachteile der Attribute im Hinblick auf den Verwendungszweck diskutieren	II.4.2	R, 67, 29
2	!	Objekt(e), Verwendungszweck		
	?	Attribut(e) für den Verwendungszweck identifizieren und begründen	II.4.5	R, 15, 5 / 24, 3
3	!	textuelle Beschreibung eines Realitätsausschnitts		
V1	?	relevante Objekte aufzählen	II.4.5	R, 16, 6
V2	?	Objektdiagramm	III.6	B, 37, 2; R, 66, 22
4	!	Objektdiagramm		
	?	Objektdiagramm mit eigenen Worten beschreiben	III.2.2	R, 61, 5

Nr.		Beschreibung	K	Beispiel(e)
5	!	grafische Darstellung eines Realitätsausschnitts		
	?	Objektdiagramm	III.6	R, 66, 24 / 66, 27

C.2.3.2 Klassen und Klassenstrukturen

Nr.		Beschreibung	K	Beispiel(e)
1	!	Module eines Informatiksystems		
	?	Module Partitionen / Schichten zuordnen	II.3.1	R, 270, 6
2	!	textuelle Beschreibung eines Realitätsausschnittes		
	!	Liste von Klassenkandidaten		
V1	?	in Liste der Klassenkandidaten die Klassen identifizieren und begründen	II.3.1	R, 231, 5 / 234, 16
V1/E1	?	Klassen beschreiben	II.2.2	R, 231, 6 / 234, 17
	!	Liste von Kandidaten für Klassenbeziehungen, z.B. in der Form „ein <x> {hat | ist | ...} ein <y>“		
V1/E2	?	in Liste der Klassenbeziehungskandidaten die Klassenbeziehungen identifizieren und begründen	II.3.1	R, 231, 7 / 234, 18
V2	?	Identifizierung von Klassen und Attributen im Text	II.4.5	B, 37, 3
V2/E1	?	Attribute spezifizieren	II.3.2	B, 37, 3
V3	?	Identifizierung von Klassen, Attributen und Operationen im Text	II.4.5	B, 38, 5
V3/E1	?	Attribute spezifizieren	II.3.2	B, 38,5
V4	?	Identifizierung von Klassen, (Attributen, Operationen und) Relationen im Text	II.4.5	B, 58, 1 / 166, 2; R, 68, 31
V4/E1	?	Attribute spezifizieren	II.3.2	B, 166, 2
V5	?	Identifizierung von Klassen, (Attributen, Operationen) Relationen und Vererbungsstrukturen im Text	II.4.5	B, 58, 2 / 166, 3; R, 98, 4 / 99, 5 / 101, 17 / 314, 24
E1	?	Klassendiagramm	III.6	B, 37, 3 / 38, 5 / 58, 1 / 58, 2 / 166, 2 / 166, 3; R, 68, 31 / 98, 4 / 99, 5 / 101, 17 / 314, 24
3	!	Liste von Klassen		
	!	Liste von Operationen mit textueller Beschreibung (je ein Satz)		
V1	?	Operationen den Klassen zuordnen	II.3.1	R, 16, 7
V1/E1	?	diskutieren, wie sich Operationen auf Klassen verhalten	II.4.1	R, 16, 7
V2	?	Gemeinsamkeiten der Klassen diskutieren	II.4.3	R, 16, 8
V3	?	Liste um weitere Klassen ergänzen	II.4.5	R, 16, 8
V4	?	Zuordnung der Klassen zu selbst zu wählenden Paketen	II.4.5	B, 59, 3
V5	?	Klassendiagramm mit Mindestanzahl an Relationen erstellen	III.6	R, 63, 15
V5/E1	?	Relationen spezifizieren (Name, Multiplizitäten)	II.3.2	R, 63, 15
4	!	unvollständiges Klassendiagramm mit Klassen und Relationen		
	!	Liste von Attributen		
V1	?	Attribute den Klassen zuordnen, Mehrfachverwendung der Attribute ist möglich	II.3.1	R, 65, 20
	!	Liste von Operationen (ohne Parameter)		
V2	?	Operationen den Klassen zuordnen, Mehrfachverwendung der Operationen ist möglich	II.3.1	R, 63, 14 / 64, 17 / 310, 6
V2/E1	?	Parameter einer Operation angeben	II.3.2	R, 64, 17
V2/E2	?	Auswirkung einer Operation auf Objekte der zugeordneten Klasse beschreiben	III.4.2	R, 64, 17
V3	?	Attribute spezifizieren	II.3.2	R, 310, 7 / 311, 12 / 312, 15
V4	?	Relationen spezifizieren (Auswahl aus Art der Relation, Richtung, Name, Multiplizitäten, Rollennamen der beteiligten Klassen) und begründen	II.3.2	R, 63, 9 / 63, 11 / 63, 12 / 65, 19 / 311, 10 / 312, 14 / 312, 16

Nr.		Beschreibung	K	Beispiel(e)
V4/E1	?	Abhängigkeit der Multiplizitätsentscheidungen vom Modellierer zeigen	II.4.1	R, 63, 9
V5	?	gegebene Anzahl von frei zu wählenden Attributen und Operationen den Klassen zuordnen	II.4.5	R, 64, 16
5	!	textuelle Beschreibung einer Relation zwischen zwei oder mehreren Klassen		
	?	Relation spezifizieren (Vererbung, Assoziation, Aggregation)	II.3.2	R, 98, 3
6	!	Formular		
	?	Identifikation von Klassen, Attributen und Assoziationen	II.4.5	B, 165, 1
7	!	-		
	?	selbstgewählte Datenstruktur beschreiben	III.2.2	R, 99, 6
E1	?	Klassendiagramm	III.6	R, 99, 6
8	!	Varianten eines Klassendiagramms, Algorithmus		
	?	Variante des Klassendiagramms auswählen, die für Algorithmus am geeignetsten ist	III.3.1	R, 69, 32
9	!	Klassendiagramm		
V1	?	semantische Änderung des Klassendiagramms gemäß textuellen Vorgaben	III.5.1	R, 62, 8 / 64, 18 / 66, 25 / 311, 9 / 311, 11 / 334, 2
V2	?	strukturelle Änderung des Klassendiagramms gemäß textuellen Vorgaben (z.B. Hinweis auf Verwendung spezieller Modellierungstechniken)	III.5.1	R, 63, 10 / 65, 21 / 97, 1 / 98, 2 / 100, 12 / 236, 28 / 238, 34 / 310, 5 / 313, 20 / 314, 22
E1	?	Diskussion, inwieweit Änderung Verbesserung darstellt	III.4.1	R, 314, 22
10	!	OOA-Klassendiagramm		
	?	OOE-Klassendiagramm	III.5.2	B, 254, 3
11	!	textuelle Beschreibung einer Taxonomie		
	?	Klassendiagramm	III.6	R, 100, 8
12	!	textuelle Beschreibung eines Realitätsausschnitts		
V1	?	OOE-Klassendiagramm	III.6	B, 253, 2; R, 269, 4 / 270, 5 / 271, 8
V2	?	OOE-Klassendiagramm mit Gestaltungshinweisen (z.B. Polymorphismus anwenden)	III.6	B, 278, 1

C.2.3.3 Verknüpfung von Objekt- und Klassenstrukturen

Nr.		Beschreibung	K	Beispiel(e)
1	!	zwei Klassendiagramme		
	?	Gleichwertigkeit der Diagramme prüfen, indem jeweils ein Objektdiagramm erstellt wird	III.4.3	B, 191, 4
2	!	Klassendiagramm, unvollständiges Objektdiagramm		
	?	Objektdiagramm ergänzen	III.5.1	R, 500, 2 / 500, 3
3	!	Klassendiagramm - mit oder ohne textuelle Beispieldaten für ein zu erstellendes Objektdiagramm		
	?	Objektdiagramm	III.6	B, 58, 1 / 58, 2; R, 61, 3 / 62, 8 / 63, 13
4	!	Objektdiagramm - mit oder ohne erläuternden Text - mit oder ohne Randbedingungen an ein zu erstellendes Klassendiagramm, wie z.B. Festlegung der Anzahl bestimmter Relationstypen		
	?	Klassendiagramm	III.6	R, 60, 1 / 60, 2 / 61, 4 / 61, 6 / 61, 7
E1	?	Begründung der Multiplizitätsentscheidungen	II.4.1	R, 60, 2 / 61, 4
E2	?	Ordnung von Objekten im Klassendiagramm spezifizieren	II.3.2	R, 60, 2 / 61, 4

C.2.4 Dynamisches Modell

C.2.4.1 Anwendungsfälle

Nr.		Beschreibung	K	Beispiel(e)
1	!	textuelle Beschreibung von Abläufen in einem Informatiksystem		
V1	?	ausgewählten Anwendungsfall mit einer Anwendungsfallschablone formal beschreiben	II.3.2	B, 137, 2
V2	?	Anwendungsfälle identifizieren	II.4.5	B, 88, 2 / 137, 4
V2/E1	?	Anwendungsfall mit einer Anwendungsfallschablone formal beschreiben	II.3.2	B, 88, 2 / 137, 4
V2/E2	?	Anwendungsfalldiagramm erstellen	III.6	B, 137, 4
V3	?	Ereignisse identifizieren	II.4.5	B, 137, 3
V3/E1	?	Anwendungsfall für ein Ereignis identifizieren	II.4.5	B, 137, 3
V3/E2	?	Anwendungsfall informell beschreiben	II.2.2	B, 137, 3
2	!	Liste von Anwendungsfällen		
	?	Identifikation von Paketen	II.4.5	B, 138, 5
3	!	zu erstellendes Informatiksystem (ein Stichwort)		
	?	Anwendungsfälle identifizieren	II.4.5	B, 87, 1
E1	?	Anwendungsfalldiagramm erstellen	III.6	B, 87, 1

C.2.4.2 Funktionale Abhängigkeiten

Nr.		Beschreibung	K	Beispiel(e)
1	!	Liste von Prozessen mit definierten Ein- und Ausgaben		
	?	Prozesse in partielle Ordnung bringen	II.3.3	R, 171, 9
E1	?	Aktivitätsdiagramm	III.6	R, 171, 9
2	!	Aktivitätsdiagramm		
V1	?	Aktivitätsdiagramm beschreiben	III.2.2	R, 169, 1
V2	?	Aktivitätsdiagramm unter gegebenem Aspekt diskutieren	III.4.1	R, 170, 2
3	!	textuelle Beschreibung von Abläufen		
	?	Aktivitätsdiagramm	III.6	R, 170, 3 / 170, 4 / 170, 5 / 171, 6

C.2.4.3 Objektinteraktion

Nr.		Beschreibung	K	Beispiel(e)
1	!	textuelle Beschreibung von Abläufen		
V1	?	Szenario	II.3.2	R, 139, 1 / 140, 2
V2	?	Bedingungen und Ergebnisse im Text identifizieren	II.4.5	B, 189, 1
V2/E1	?	Interaktionsdiagramm	III.6	B, 189, 1
2	!	textuelle Beschreibung eines Informatiksystems		
V1	?	Szenario	II.3.2	R, 269, 3
V2	?	Interaktionsdiagramm	III.6	B, 300, 1
3	!	textuelle Beschreibung eines Realitätsausschnitts, Klassendiagramm, Operationen der Klassen mit Beschreibung		
	?	für jede Operation Liste der sich bei anderen Klassen fortpflanzenden Operationsaufrufe erstellen	III.4.2	R, 99, 7
4	!	Deklarationen, Operationen in Programmcode		
	?	Interaktionsdiagramm(e)	III.6	B, 278, 2

C.2.4.4 Objektzustände

Nr.		Beschreibung	K	Beispiel(e)
1	!	unvollständiges Zustandsdiagramm mit Zuständen und Zustandsübergängen ohne Beschriftung, textuelle Beschreibung der Zustandsübergänge		
	!	Aufzählung von Ereignissen und Aktivitäten		
V1	?	Ereignisse und Aktivitäten in der Aufzählung identifizieren	II.3.1	R, 141, 7 / 141, 9
V1/E1	?	Ereignisse und Aktivitäten im Zustandsdiagramm ergänzen	II.3.1	R, 141, 7 / 141, 9
	!	Aufzählung von Ereignissen, Bedingungen und Aktivitäten		
V2	?	in einer Aufzählung gegebene Ereignisse, Bedingungen und Aktivitäten im Zustandsdiagramm ergänzen	II.3.1	R, 142, 10
V3	?	Ereignisse, Aktionen, Aktivitäten und Bedingungen identifizieren	II.4.5	R, 143, 13
V3/E1	?	Ereignisse, Aktionen, Aktivitäten und Bedingungen im Zustandsdiagramm ergänzen	II.3.1	R, 143, 13
2	!	Zustandsdiagramm		
V1	?	Zustandsdiagramm beschreiben	III.2.2	R, 145, 19
V2	?	Zustandsdiagramm gemäß textueller Beschreibung modifizieren	III.5.1	R, 141, 6 / 141, 8 / 142, 12 / 145, 19
3	!	textuelle Beschreibung von Abläufen - durch Ereignisse und deren kausale Zusammenhänge		
	?	Zustandsdiagramm für vorgegebenes Objekt	III.6	B, 88, 4; R, 140, 4 / 140, 5; R, 238, 32
4	!	Szenario		
	?	Zustandsdiagramm für vorgegebenes Objekt	III.6	R, 140, 3 / 145, 18
E1	?	gegebenes Szenario aufgrund des Zustandsdiagramms vereinfachen	III.4.4	R, 145, 18

C.2.4.5 Verknüpfung von Objektinteraktion und Objektzuständen

Nr.		Beschreibung	K	Beispiel(e)
1	!	Szenario		
	?	Interaktionsdiagramm	III.6	R, 237, 29 / 238, 30
E1	?	Zustandsdiagramm	III.6	R, 238, 31

C.2.5 Verknüpfung des statischen und dynamischen Modells

C.2.5.1 Anforderungsanalyse

Nr.		Beschreibung	K	Beispiel(e)
1	!	Eigenschaften von Komponenten eines Informatiksystems, Verwendungszweck des Informatiksystems		
	?	relevante von irrelevanter Information in Bezug auf den Verwendungszweck trennen	II.3.1	R, 24, 4
2	!	Aufzählung von Anforderungen an ein Informatiksystem		
	?	Zuordnung der Informationen zu den Phasen objektorientierter Problemlösungsmethoden	II.3.1	B, 16, 4
3	!	textuelle Anforderungsbeschreibung eines Informatiksystems		
V1	?	Entwurfsentscheidungen aus der Anforderungsbeschreibung entfernen	III.4.2	R, 229, 3
V1/E1	?	Klassendiagramm	III.6	R, 230, 4
V2	?	für ein anderes Informatiksystem eine Anforderungsbeschreibung in ähnlichem Umfang erstellen	III.6	R, 229, 2

C.2.5.2 Verknüpfung von Teilmodellen

Nr.		Beschreibung	K	Beispiel(e)
1	!	Klassendiagramm		
V1	?	Klassen identifizieren, für die Zustandsdiagramm erforderlich ist	II.4.2	R, 232, 14 / 236, 24
V1/E1	?	Zustandsdiagramm	III.6	R, 232, 14 / 236, 24
V1/E2	?	Konsistenz der Zustandsdiagramme sichern	III.4.4	R, 232, 14 / 236, 24
	!	Anfragen		
V2	?	beschreiben, wie gegebene Anfragen mittels des Klassendiagramms bearbeitet werden können	III.4.2	R, 231, 8 / 234, 19 / 543, 2
V2/E1	?	Klassendiagramm modifizieren, falls erforderlich	III.5.1	R, 234, 19
	!	Variante des Klassendiagramms		
V2/E2	?	beschreiben, wie gegebene Anfragen mittels des Klassendiagramms bearbeitet werden können	III.4.2	R, 232, 9
V2/E3	?	Vergleich der beiden Varianten	III.4.3	R, 232, 9
	!	Beispielanwendung, Gestaltungshinweise		
V3	?	Szenario erstellen	III.6	R, 232, 10 / 235, 20 / 236, 22 / 236, 23
V3/E1	?	Fehlerszenario erstellen	III.6	R, 232, 11 / 235, 21
V3/E2	?	Interaktionsdiagramm	III.6	R, 232, 12
E1	?	Aktivitätsdiagramm für Gesamtsystem	III.6	R, 232, 13 / 233, 15 / 236, 25
2	!	Liste lebensweltlicher Modelle, Gegenstandsbereich, verschiedene Fragestellungen zum Gegenstandsbereich		
	?	Fragen den Modellen zuordnen	III.3.1	R, 24, 5
3	!	Liste objektorientierter Modelle, Gegenstandsbereich, verschiedene Fragestellungen zum Gegenstandsbereich		
	?	Fragen den Modellen zuordnen	III.3.1	R, 25, 6
4	!	textuelle Beschreibung von Abläufen in einem Informatiksystem und von Klassen, Attributen und Relationen		
	?	Klassendiagramm	III.6	B, 167, 4
5	!	Anforderungsbeschreibung an ein Informatiksystem, Vorgehensmodell		
	?	Systementwurf	III.6	R, 321, 1 / 322, 2 / 322, 3 / 322, 4 / 322, 5
6	!	textuelle Beschreibung von Operationen einer Klasse		
V1	?	Zustandsdiagramm der Klasse	III.6	B, 190, 3
	!	Klassendiagramm		
V2	?	Aktivitätsdiagramm für vorgegebene Operation erstellen	III.6	R, 171, 8
7	!	formalisierte Beschreibung eines Anwendungsfalls mittels einer Anwendungsfallschablone		
	?	Klassendiagramm	III.6	B, 189, 2
	!	Szenario		
E1	?	Interaktionsdiagramm	III.6	B, 189, 2
8	!	textuelle Beschreibung von Abläufen		
V1	?	Interaktionsdiagramm	III.6	B, 88, 3
V1/E1	?	Klassendiagramm	III.6	B, 88, 3
V1/E2	?	Konsistenz von Interaktions- und Klassendiagramm prüfen	III.4.4	B, 88, 3
	!	Klassendiagramm, Operationen mit Beschreibungen		
V2	?	Zustandsdiagramm für vorgegebenes Objekt	III.6	R, 144, 16

C.2.5.3 Analyse- und Entwurfsmuster

Nr.		Beschreibung	K	Beispiel(e)
1	!	Klassendiagramm		
	?	Muster im Klassendiagramm identifizieren	II.4.5	B, 116, 2 / 301, 2 / 416, 4

Nr.		Beschreibung	K	Beispiel(e)
2	!	textuelle Beschreibung eines Informatiksystems, Liste von Mustern		
	?	Klassendiagramm unter Verwendung von Mustern erstellen	III.6	B, 300, 1
3	!	textuelle Beschreibung eines Realitätsausschnitts		
V1	?	Klassendiagramm unter Verwendung von Mustern erstellen	III.6	B, 117, 3
V1/E1	?	Muster im Klassendiagramm markieren	II.3.1	B, 117, 3
V2	?	Klassendiagramm	III.6	B, 116, 1
V2/E1	?	prüfen, welches Muster vorliegt	III.4.2	B, 116, 1

C.2.5.4 Prototyp einer Benutzungsoberfläche

Nr.		Beschreibung	K	Beispiel(e)
1	!	Klassendiagramm		
V1	?	Dialogstruktur aus Klassendiagramm ableiten	III.4.2	B, 226, 3
V1/E1	?	Dialogstruktur als Zustandsdiagramm	III.6	B, 226, 3
V2	?	Eingabedialog für ausgewählte Klasse entwerfen	III.6	B, 226, 2
2	!	Klasse, Eingabediaglogfenster für Objekte vom Typ <Klasse>, Listenfenster für mehrere Objekte vom Typ <Klasse>		
	?	Interaktionsdiagramm zur Änderung eines Objekts vom Typ <Klasse> bzw. Aktualisierung des Listenfensters	III.6	B, 393, 3

C.2.6 Metamodellierung

Nr.		Beschreibung	K	Beispiel(e)
1	!	Metamodell (Klassendiagramm) objektorientierter Basiskonzepte		
V1	?	eine Eigenschaft des Metamodells beschreiben	III.2.2	R, 101, 14 / 101, 15
V2	?	prüfen, ob das Metamodell eine bestimmte Eigenschaft hat und begründen	III.4.4	R, 100, 13 / 499, 1
V3	?	Metamodell modifizieren, um bestimmte Eigenschaft zu realisieren	III.5.1	R, 101, 14 / 101, 15 / 500, 4
	!	mit oder ohne Grafik zu Beispieldaten		
V4	?	Objektdiagramm zu den Beispieldaten	III.6	R, 101, 16 / 501, 5
2	!	-		
	?	Metamodell (Klassendiagramm) funktionaler Abhängigkeiten erstellen	III.6	R, 171, 6.9
3	!	textuelle / grafische Beschreibung einer formalen Sprache		
	?	Metamodell (Klassendiagramm) der Sprache erstellen	III.6	R, 101, 18
4	!	Liste objektorientierter Basiskonzepte		
	?	Metamodell (Klassendiagramm) objektorientierter Basiskonzepte erstellen	III.6	R, 100, 10
	!	Gestaltungshinweise		
E1	?	Objektdiagramm für frei wählbare Beispieldaten erstellen	III.6	R. 100, 11

D Materialien der empirischen Studien zu Aufgabenklassen

Die Materialien der empirischen Studie in den Abschnitten D.1 bis D.3 stammen aus (Ortmann 2002, 98ff). Sie werden hier nochmals aufgeführt zum besseren Verständnis.

D.1 Arbeitsblatt und Musterlösungen zum „Traktor“

Aufgabe 1

Aus den letzten Stunden im Unterricht kennst Du die Begriffe *Klasse* und *Objekt*. Erkläre den Unterschied der beiden Begriffe mit eigenen Worten!

Musterlösung: *Klassen*: *Objekte* mit gleichen Attributen und gleichen Diensten werden zu einer *Klasse* zusammengefasst. Ein *Objekt* ist also Exemplar einer Klasse. Anders gedeutet ist eine *Klasse* eine Vorlage, nach der gleich geartete *Objekte* erzeugt werden.

Neben dem Klassenbegriff wird auch noch von *Methoden* und *Attributen* gesprochen. Erkläre den Begriff *Attribut* mit eigenen Worten!

Musterlösung: *Attribute* beschreiben die Eigenschaften eines Objekts. Hat ein *Attribut* einen bestimmten Wert, so wird damit der aktuelle Zustand eines Objekts bezüglich dieses *Attributs* beschrieben.

Erkläre den Begriff *Methode* mit eigenen Worten!

Musterlösung: *Methoden* sind Dienstleistungen, die von einem Objekt angefordert werden können.

Aufgabe 2

Die folgenden Wörter bezeichnen Klassen-, Attribut- oder Methodennamen. Ordne die Wörter der richtigen Kategorie zu!

Rechteck – anhaengen – Kreis – XPosition – Ellipse – Geschwindigkeit – Traktor – LaengeHoehe – Figur – zeichne – Buntstift – LaengeBreite – loesche – Radius – Bildschirm – RadiusA – fahreVorwaerts – YPosition – RadiusB – fahreRueckwaerts – Anhaenger – Quadrat – Laenge – Groesse

Musterlösung:

Klasse	**Methode**	**Attribut**
Rechteck	anhaengen	Xposition
Kreis	zeichne	Geschwindigkeit
Ellipse	loesche	LaengeHoehe
Traktor	fahreVorwaerts	LaengeBreite
Figur	fahreRueckwaerts	Radius
BuntStift		RadiusA
Bildschirm		YPosition
Anhaenger		RadiusB
Quadrat		Laenge
		Groesse

Aufgabe 3

Ordne den Klassen aus Aufgabe 2 (außer Bildschirm und BuntStift) die Methoden und Attribute zu, die sie benötigen. Beschreibe mit einem kurzen Satz deren Funktion innerhalb der Klasse!
Hinweis: Einige Methoden und Attribute können in mehreren Klassen vorkommen.

Musterlösung:

Klasse	**Methode**	**Attribut**
Rechteck	zeichne	LaengeHoehe LaengeBreite
Ellipse	zeichne	RadiusA RadiusB
Kreis		Radius
Quadrat		Laenge
Figur	zeichne loesche	XPosition YPosition Groesse
Traktor	anhaengen fahreVorwaerts fahreRueckwaerts	Geschwindigkeit Laenge
Anhaenger	fahreVorwaerts fahreRueckwaerts	Geschwindigkeit Laenge

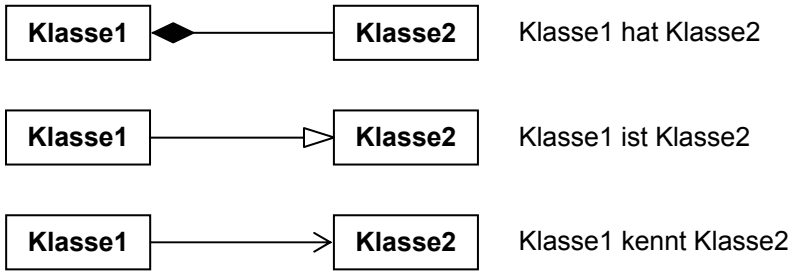
Aufgabe 4 (in Partner- oder Gruppenarbeit)

Bestimme und benenne die Beziehungen (Ist, Kennt, Hat) zwischen den einzelnen Klassen!

Musterlösung:

Klasse	**Vererbung (Ist-Bez.)**	**Komposition (Hat-Bez.)**	**Assoziation (Kennt-Bez.)**
Bildschirm			
BuntStift			kenntBildschirm
Figur		hatBuntStift	
Rechteck	Figur		
Quadrat	Rechteck		
Ellipse	Figur		
Kreis	Ellipse		
Traktor		hatRechteck hatEllipse hatKreis	kenntAnhaenger
Anhaenger		hatRechteck hatKreis	

Aufgabe 5 (in Partner- oder Gruppenarbeit)
Erstelle ein Klassendiagramm!
Beachte dabei die folgenden Symbole für die Beziehungen.



Musterlösung:

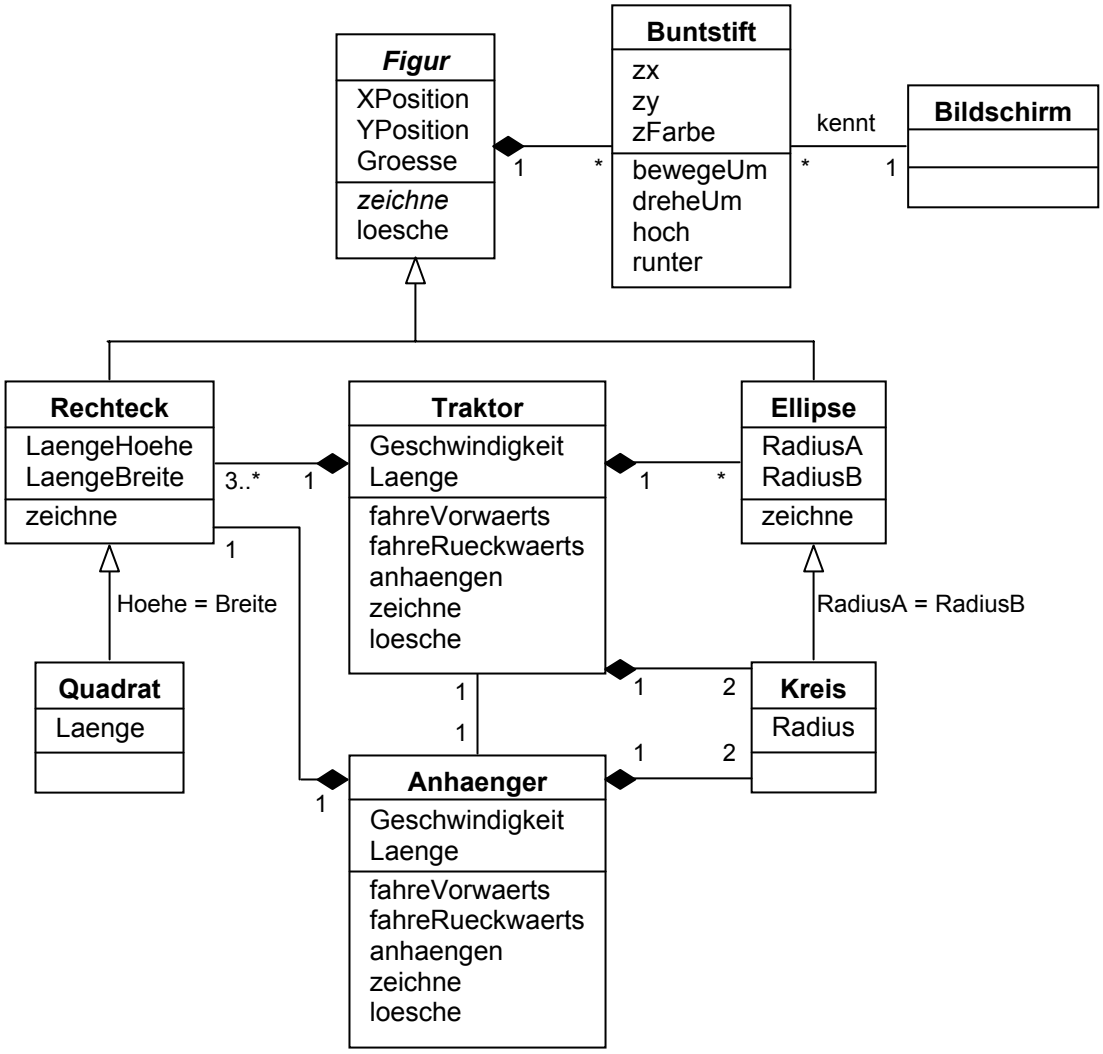


Abbildung 28: Aufgabensammlung „Traktor“ – Musterlösung zum Klassendiagramm

D.2 Arbeitsblatt und Musterlösungen zum „Zug“

Aufgabe 1

Zeichne in das folgende Klassendiagramm die Beziehungsarten (Ist, Hat, Kennt) ein.

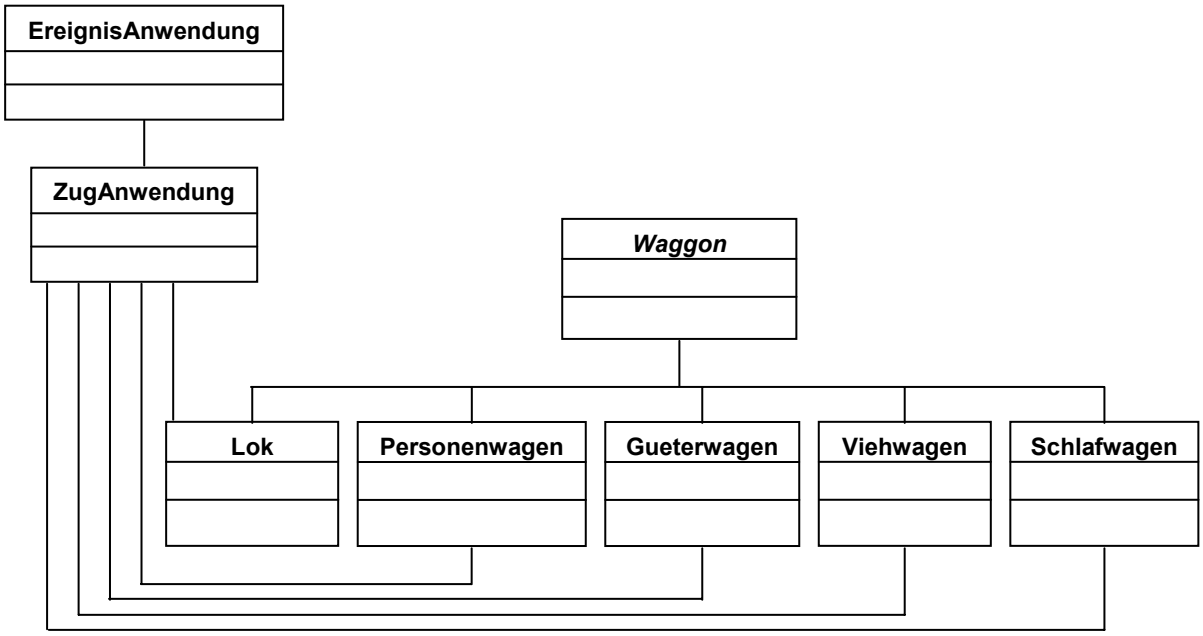


Abbildung 29: Aufgabensammlung „Zug“ – Arbeitsmaterial

Welche Klassen oder Beziehungen fehlen?

Aufgabe 2

Ordne den Klassen aus Aufgabe 1 die unten stehenden Methoden zu. Beschreibe mit einem kurzen Satz deren Funktion innerhalb der Klasse!
Hinweis: Einige Methoden können in mehreren Klassen vorkommen.

gibFrei – zeichneDich – bearbeiteTaste – fahreVorwaerts – laenge – bewegeBis – loesche-Dich – ankoppeln – fahreRueckwaerts

Klasse	**Methode**

Musterlösung: Klassendiagramm mit Methoden und einigen Attributen

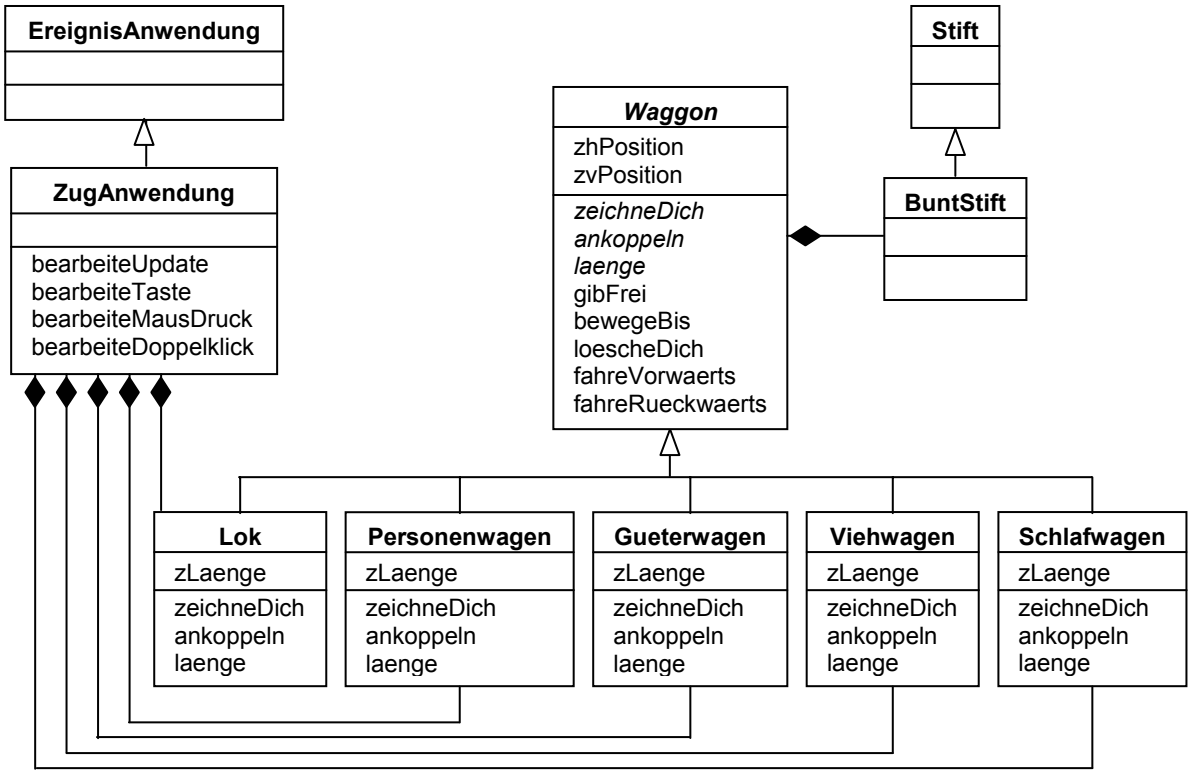


Abbildung 30: Aufgabensammlung „Zug“ – Musterlösung zum Klassendiagramm

Musterlösung: Beschreibung der Funktion der einzelnen Methoden gemäß Klassenprotokoll

Die Klassenprotokolle für Personenwagen, Schlafwagen, Gueterwagen und Viehwagen entsprechen dem Klassenprotokoll der Klasse Lok und werden deshalb nicht aufgeführt.

Klasse: *ZugAnwendung*
Oberklasse: EreignisAnwendung
Protokoll:

Auftrag:	bearbeiteUpdate
nachher	Es wurden alle Wagen und Beschreibungen auf den Bildschirm gezeichnet.
Auftrag:	bearbeiteTaste
nachher	Ein neuer Waggon wurde angehängt.
Auftrag:	bearbeiteMausDruck
nachher	Der Zug wurde gestartet oder gestoppt.
Auftrag:	bearbeiteDoppelklick
nachher	Das Programm wurde beendet.

Klasse: *Waggon*
Unterklassen: Lok, Personenwagen, Schlafwagen, Gueterwagen, Viehwagen
Bezugsklassen: Stift
Protokoll:

Auftrag:	zeichneDich
vorher	Der Waggon soll gezeichnet werden.
nachher	(abstrakte Methode)
Auftrag:	ankoppeln
vorher	Ein neuer Waggon soll angehängt werden.
nachher	(abstrakte Methode)
Anfrage:	laenge : Zahl
vorher	Die Länge des Waggons wird angefragt.
nachher	(abstrakte Methode)
Auftrag:	gibFrei
nachher	Der Stift ist freigegeben worden.
Auftrag:	bewegeDich
nachher	Der Wagen wurde gelöscht, bewegt und neu gezeichnet.
Auftrag:	loescheDich
nachher	Der Stift wurde in den Radiermodus gesetzt, der Waggon gezeichnet und der Stift wurde wieder in den Normalmodus gesetzt. Der Waggon ist auf dem Bildschirm nicht mehr zu sehen.
Auftrag:	fahreVorwaerts
nachher	Der Waggon und seine Nachfolger wurden nach links zum Bildschirmrand bewegt.
Auftrag:	fahreRueckwaerts
nachher	Der Waggon und seine Nachfolger wurden nach rechts zum Bildschirmrand bewegt.

Klasse: *Lok*
Oberklasse: Waggon
Protokoll:

Auftrag:	zeichneDich
nachher	Die Lok wurde gezeichnet.
Auftrag:	ankoppeln
nachher	Ein neuer Waggon wurde an die Lok angekoppelt.
Anfrage:	laenge : Zahl
nachher	Laenge liefert die Länge der Lok.

D.3 Ergebnisse der schriftlichen Befragung

Im Folgenden werden die Ergebnisse der schriftlichen Befragung der drei beobachteten Kurse gegenüber gestellt (vgl. Ortmann 2002, 121ff). Die jeweils erste Zeile zeigt die Ergebnisse im Kurs A-11, die zwei Zeile den Kurs B-11 und die dritte Zeile den Kurs B-12.

		trifft völlig zu		trifft eher zu		trifft eher nicht zu		trifft nicht zu		kann ich nicht beurteilen	
1.	Informatikunterricht macht mir Spaß.	2	13%	11	73%	2	13%	0	0%	0	0%
		2	17%	5	42%	4	33%	0	0%	1	8%
		5	31%	8	50%	2	13%	0	0%	1	6%
2.	Ich fühle mich im Informatikunterricht wohl.	2	13%	9	60%	3	20%	0	0%	1	7%
		2	17%	6	50%	3	25%	1	8%	0	0%
		2	13%	10	63%	3	19%	1	6%	0	0%
3.	Ich verstehe den Unterrichtsstoff im Fach Informatik.	8	53%	6	40%	1	7%	0	0%	0	0%
		2	17%	3	25%	4	33%	3	25%	0	0%
		4	25%	10	63%	1	6%	0	0%	1	6%
4.	Ich habe ausreichend Zeit, um über die gestellten Fragen und Aufgaben nachzudenken.	7	47%	6	40%	2	13%	0	0%	0	0%
		1	8%	5	42%	6	50%	0	0%	0	0%
		7	44%	5	31%	3	19%	0	0%	0	0%
5.	Im Informatikunterricht geht es darum, Programme zu erstellen	1	7%	6	40%	4	27%	3	20%	1	7%
		3	25%	6	50%	1	8%	0	0%	2	17%
		3	19%	6	38%	5	31%	1	6%	1	6%
6.	Im Informatikunterricht geht es darum, Problemstellungen zu modellieren.	2	13%	6	40%	5	33%	0	0%	2	13%
		2	17%	8	67%	1	8%	0	0%	1	8%
		1	6%	12	75%	1	6%	1	6%	1	6%
7.	Die Themen des Informatikunterrichts sind nützlich für mich (für mein tägliches Leben).	0	0%	3	20%	4	27%	7	47%	1	7%
		0	0%	1	8%	4	33%	7	58%	0	0%
		1	6%	1	6%	7	44%	7	44%	0	0%
8.	Ich finde die Inhalte im Informatikunterricht interessant.	0	0%	8	53%	5	33%	2	13%	0	0%
		1	8%	3	25%	7	58%	1	8%	0	0%
		0	0%	8	50%	4	25%	3	19%	1	6%
9.	Ich würde gerne Einfluss auf die Themen im Unterricht nehmen.	6	40%	3	20%	3	20%	3	20%	0	0%
		3	25%	4	33%	4	33%	1	8%	1	8%
		6	38%	5	31%	2	13%	2	13%	1	6%
10.	Die Schüler verhalten sich im Unterricht diszipliniert.	3	20%	9	60%	2	13%	0	0%	1	7%
		0	0%	3	25%	5	42%	4	33%	0	0%
		5	31%	1	6%	1	6%	9	56%	0	0%
11.	Im Unterricht arbeiten die Schüler gut mit.	4	27%	11	73%	0	0%	0	0%	0	0%
		2	17%	7	58%	2	17%	0	0%	1	8%
		2	13%	3	19%	10	63%	0	0%	1	6%
12.	Ich beteilige mich am Unterricht.	6	40%	8	53%	1	7%	0	0%	0	0%
		4	33%	4	33%	4	33%	0	0%	0	0%
		6	38%	6	38%	3	19%	0	0%	1	6%
13.	Ich bin im Unterricht abgelenkt.	0	0%	0	0%	5	33%	8	53%	2	13%
		0	0%	5	42%	6	50%	1	8%	0	0%
		0	0%	1	6%	8	50%	7	44%	0	0%

Tabelle 44: Befragungsergebnisse zum Informatikunterricht allgemein

		trifft völlig zu		trifft eher zu		trifft eher nicht zu		trifft nicht zu		kann ich nicht beurteilen	
1.	Der Unterricht war abwechslungsreich.	0	0%	7	47%	5	33%	1	7%	2	13%
		0	0%	7	58%	4	33%	1	8%	0	0%
		2	13%	9	56%	2	13%	2	13%	1	6%
2	Der Unterricht beschäftigte sich mit Aufgaben, die mir im täglichen Leben begegnen.	0	0%	1	7%	3	20%	11	73%	0	0%
		0	0%	1	8%	4	33%	6	50%	1	8%
		0	0%	2	13%	7	44%	6	38%	1	6%

	trifft völlig zu		trifft eher zu		trifft eher nicht zu		trifft nicht zu		kann ich nicht beurteilen	
3. Ich konnte im Unterricht etwas Neues entdecken.	7	47%	5	33%	2	13%	1	7%	0	0%
	1	8%	4	33%	6	50%	0	0%	1	8%
	3	19%	9	56%	3	19%	0	0%	1	6%
4. Einige Themen haben mich besonders interessiert.	3	20%	7	47%	4	27%	1	7%	0	0%
	1	8%	3	25%	5	42%	3	25%	0	0%
	1	6%	6	38%	8	50%	0	0%	1	6%
5. Ich habe auch außerhalb des Unterrichts über die Aufgaben nachgedacht.	3	20%	5	33%	4	27%	3	20%	0	0%
	1	8%	3	25%	3	25%	5	42%	0	0%
	1	6%	2	13%	7	44%	6	38%	0	0%
6. Ich konnte mich leicht auf die Sache konzentrieren.	7	47%	4	27%	4	27%	0	0%	0	0%
	0	0%	9	75%	2	17%	1	8%	0	0%
	4	25%	7	44%	5	31%	0	0%	0	0%
7. Ich habe das Gefühl, für mich etwas dazugelernt zu haben.	7	47%	7	47%	0	0%	1	7%	0	0%
	1	8%	2	17%	6	50%	2	17%	1	8%
	2	13%	10	63%	2	13%	2	13%	0	0%
8. Ich kann mir vorstellen, dass ich das erworbene Wissen in Zukunft gebrauchen kann.	3	20%	5	33%	2	13%	4	27%	1	7%
	1	8%	1	8%	6	50%	3	25%	1	8%
	1	6%	6	38%	3	19%	4	25%	2	13%
9. Es hat mir Spaß gemacht, mein Verständnis für dieses Thema zu vertiefen.	2	13%	9	60%	3	20%	0	0%	1	7%
	0	0%	3	25%	7	58%	2	17%	0	0%
	0	0%	7	44%	7	44%	2	13%	0	0%

Tabelle 45: Befragungsergebnisse zum Unterrichtsthema

	0 – 25 %		25 – 50 %		50 – 75 %		75 – 100 %	
beim Modellieren	1	7%	3	20%	7	47%	2	13%
	0	0%	4	33%	5	42%	3	25%
	1	6%	2	13%	10	63%	3	19%
beim Programmieren	2	13%	0	0%	8	53%	3	20%
	1	8%	2	17%	5	42%	4	33%
	2	13%	4	25%	8	50%	2	13%

Tabelle 46: Befragungsergebnisse zum geschätzten individuellen Arbeitsanteil bei der Gruppenarbeit

	trifft völlig zu		trifft eher zu		trifft eher nicht zu		trifft nicht zu		kann ich nicht beurteilen	
1. Ich modelliere gerne.	4	27%	5	33%	2	13%	3	20%	1	7%
	0	0%	4	33%	4	33%	4	33%	0	0%
	0	0%	5	31%	2	13%	9	56%	0	0%
2. Ich halte das Zerlegen von Problemen mit Hilfe der Analyse für sinnvoll.	1	7%	7	47%	2	13%	2	13%	3	20%
	4	33%	2	17%	3	25%	3	25%	0	0%
	1	6%	8	50%	3	19%	4	25%	0	0%
3. Die Fähigkeit, modellieren zu können, nutzt mir in der Zukunft.	1	7%	7	47%	3	20%	1	7%	3	20%
	3	25%	3	25%	1	8%	5	42%	0	0%
	1	6%	2	13%	3	19%	7	44%	3	19%
4. Ich programmiere gerne.	10	67%	3	20%	1	7%	1	7%	0	0%
	1	8%	5	42%	3	25%	3	25%	0	0%
	5	31%	8	50%	3	19%	0	0%	0	0%
5. Programmieren ist eine Fähigkeit, die ich in der Zukunft gebrauchen kann.	8	53%	4	27%	1	7%	0	0%	2	13%
	1	8%	4	33%	2	17%	3	25%	2	17%
	5	31%	8	50%	1	6%	1	6%	1	6%
6. Bei einer neuen Aufgabe würde ich lieber sofort programmieren, ohne vorher eine Analyse durchzuführen.	4	27%	7	47%	3	20%	1	7%	0	0%
	1	8%	2	17%	3	25%	5	42%	1	18%
	6	38%	4	25%	2	13%	1	6%	3	19%

	trifft völlig zu		trifft eher zu		trifft eher nicht zu		trifft nicht zu		kann ich nicht beurteilen	
7. Algorithmen statt Stifte und Mäuse.[104]	5	33%	1	7%	0	0%	6	40%	3	20%

Tabelle 47: Befragungsergebnisse zur Einstellungen zum Modellieren und Programmieren

[104] Nur im Kurs A-11.

D.4 Arbeitsblatt zur Anwendung des Aufgabenklassenkonzepts

Dortmund, den 31.10.2001

Übung zur Vorlesung „Didaktik der Informatik II“
Blatt 3

In der Vorlesung wurde ein Katalog abstrakter Aufgaben zum objektorientierten Modellieren vorgestellt, die durch die Einbettung in einen Beispielkontext zu konkreten Übungsaufgaben umgewandelt werden können (siehe auch [Brinda2000]). Die Aufgaben können dabei in unterschiedlichen Komplexitätsstufen realisiert werden und bieten somit die Möglichkeit zur Binnendifferenzierung.
Entwickeln sie aus einer abstrakten Aufgabe drei konkrete Übungsaufgaben für die Sekundarstufe II, indem Sie sie in einen Beispielkontext einbetten und auf drei Komplexitätsstufen ausgestalten.

1. Aufgabe: ***2 Punkte***
Wählen und begründen Sie einen geeigneten Beispielkontext, der die Kriterien *Eignung für OOM, Lebensweltbezug, Motivation der Lerngruppe* und *leichte Änderbarkeit und Erweiterbarkeit* (siehe [Brinda2000, S. 40f]) erfüllt.

2. Aufgabe: ***6 Punkte***
Verwenden Sie die erste in dem Katalog unter „3.d. Modell konstruieren“ dargestellte abstrakte Aufgabe [Brinda2000, S. 42], nach der aus einer textuellen Beschreibung ein dazugehöriges Klassendiagramm entwickelt werden soll, und gestalten Sie sie auf drei unterschiedlichen Komplexitätsstufen aus. Weisen Sie jeweils aus, welche unterschiedlichen kognitiven Anforderungen an die Lernenden gestellt werden.

a) Beschreiben Sie die kognitiven Anforderungen für eine leichtere Aufgabe und entwickeln Sie eine entsprechende Übungsaufgabe zu dem von Ihnen gewählten Beispielkontext. Die Übungsaufgabe sollte aus einer textuellen Beschreibung und einer Skizze einer Musterlösung bestehen.

b) Beschreiben Sie weitere kognitive Anforderungen für eine mittelschwere Aufgabe und entwickeln Sie analog zu Teilaufgabe a) eine mittelschwere Übungsaufgabe zu dem gewählten Beispielkontext.

c) Wiederholen Sie den Vorgang für eine komplexere Aufgabe.

3. Aufgabe: ***2 Punkte***
a) Beschreiben Sie, mit welchen Unterrichtsmethoden Sie diese Aufgaben einbetten würden.

b) Beschreiben Sie, welche Medien Sie einsetzen würden.

Abgabe und Besprechung in der Übung am 7.11.2001

[Brinda2000] Brinda, T.: Sammlung und Strukturierung von Übungsaufgaben zum objektorientierten Modellieren im Informatikunterricht. In: Log In 20 (2000) 5, S. 39-49.

D.5 Übungsaufgaben im Rahmen der Lehrerfortbildungen

Die folgenden Übungsaufgaben wurden in Lehrerfortbildungen erprobt. Sie orientieren sich an Aufgaben in Rumbaugh et al. (1993). Musterlösungen zu den Aufgaben 1 bis 5 wurden publiziert, vgl. (Brinda 2000b) und (Brinda / Schubert 2001).

Aufgabe 1: Ein Polygonenmodell

a) Entwickeln Sie ein Klassendiagramm aus dem Objektdiagramm in Abbildung 31. Begründen Sie Ihre Multiplizitätsentscheidungen. Jeder Punkt hat eine x-Koordinate und eine y-Koordinate. Welche Zahl von Punkten ist mindestens erforderlich, um ein Polygon zu konstruieren? Macht es einen Unterschied, ob ein gegebener Punkt von mehreren Polygonen gemeinsam genutzt wird? Wie können Sie die Tatsache ausdrücken, dass Punkte sich in einer Reihenfolge befinden?

b) Beschreiben Sie in Ihren eigenen Worten das Objektdiagramm.

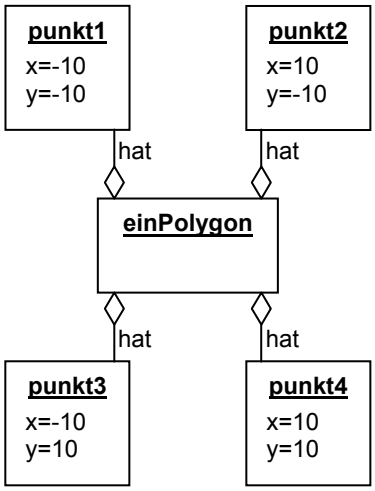


Abbildung 31: Objektdiagramm für ein Polygon

Aufgabe 2: Ein Wettkampf-Bewertungssystem

Abbildung 32 zeigt ein unvollständiges Klassendiagramm, das beim Entwurf eines Systems zur Vereinfachung der Planung und Bewertung von Sportwettkämpfen wie Gymnastik, Kunstspringen und Eiskunstlauf eingesetzt werden könnte. Dabei gibt es mehrere Durchgänge und Wettbewerbsteilnehmer. Jeder Teilnehmer kann in mehreren Durchgängen antreten und für jeden Durchgang gibt es viele Teilnehmer. Für jeden Durchgang gibt es mehrere Kampfrichter, die die Leistung der Teilnehmer in diesem Durchgang subjektiv bewerten. Ein Kampfrichter bewertet jeden Teilnehmer an einem Durchgang. In einigen Fällen punktet ein Kampfrichter mehr als einen Durchgang. Der Angelpunkt des Wettbewerbs sind die Starts. Jeder Start ist ein Versuch eines Teilnehmers, bei einem Durchgang die bestmögliche Leistung zu erbringen. Ein Start wird durch eine Jury von Richtern für diesen Durchgang gepunktet und die Endpunktezahl wird ermittelt.

a) Fügen Sie den Assoziationen Namen und Multiplizitäten hinzu.

b) Fügen Sie die folgenden Attribute hinzu: Adresse, Alter, Datum, Schwierigkeitsgrad, Name. In einigen Fällen möchten Sie die gleichen Attribute möglicherweise in mehr als einer Klasse verwenden.

c) Fügen Sie eine Assoziation hinzu, mit der es möglich ist, direkt und ohne Verwendung der Klasse Start festzustellen, in welchen Durchgängen ein Teilnehmer antreten wird.

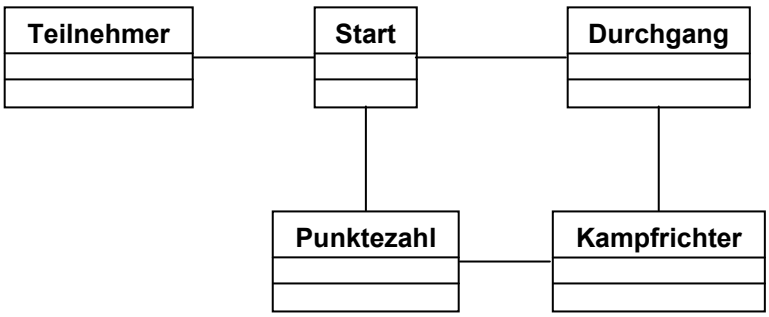


Abbildung 32: Teil eines Klassendiagramms für ein Wettkampf-Bewertungssystem

Aufgabe 3: Ein Schulmodell

a) Zeichnen Sie ein Klassendiagramm, das mindestens 10 Relationen zwischen den folgenden Objektklassen zeigt: Schule, Spielplatz, Direktor, Elternbeirat, Klassenzimmer, Buch, Schüler, Lehrer, Cafeteria, Pausenraum, Computer, Pult, Stuhl, Lineal, Tür, Schaukel. Fügen Sie Assoziationen, Aggregationen und Generalisierungen hinzu. Fügen Sie ferner Multiplizitäten ein. Das Diagramm braucht keine Attribute oder Operationen zu enthalten. Verwenden Sie Assoziationsnamen, wo Sie gebraucht werden. Beim Zeichnen des Diagramms können Sie weitere Objektklassen hinzufügen.

b) Fügen Sie dem Klassendiagramm mindestens 15 Attribute und 5 Operationen hinzu.

Aufgabe 4: Ein Flugreisesystem

Abbildung 33 zeigt ein erst teilweise fertig gestelltes Klassendiagramm für ein Flugreisesystem.

a) Ergänzen Sie die fehlenden Multiplizitäten der Relationen und begründen Sie Ihre Entscheidungen. Zeigen Sie, inwiefern Multiplizität von ihrer persönlichen Wahrnehmung der Welt abhängt.

b) Fügen Sie den unbenannten Assoziationen Namen hinzu.

c) Fügen Sie die folgenden Operationen hinzu: heizen, einstellen, entlassen, tanken, reservieren, reinigen, enteisen, starten, landen, reparieren, stornieren, Verspätung haben. Es ist zulässig, eine Operation zu mehr als einer Objektklasse hinzuzufügen.

d) Zeichnen Sie ein Objektdiagramm für einen imaginären Hin- und Rückflug von Hans Meier von Berlin nach New York am letzten Wochenende. Verwenden Sie mindestens eine Instanz jeder Objektklasse. Glücklicherweise gab es noch Direktflüge mit einem Jumbo Jet. Peter Müller hat Herrn Meier begleitet, entschloss sich jedoch, fürs erste in New York zu bleiben, wo er immer noch ist. Flugkapitän Johnson war beim Hinflug wie beim Rückflug der Pilot. Herr Meier saß jeweils an unterschiedlichen Plätzen, aber wegen einer unverkennbaren Delle am Leitwerk wusste er, dass das Flugzeug beide Male das gleiche war.

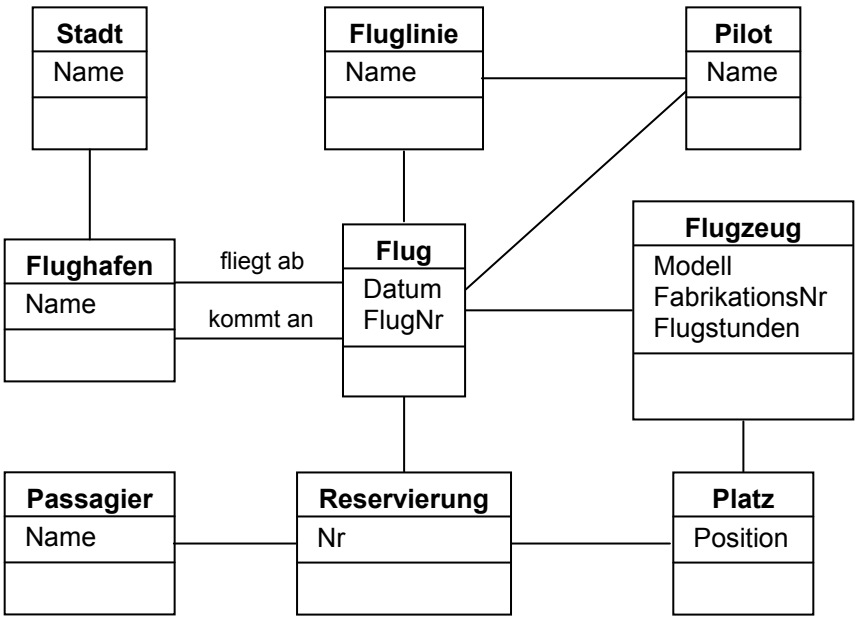


Abbildung 33: Teilweise fertig gestelltes Klassendiagramm eines Flugreisesystems

Aufgabe 5: Die Türme von Hanoi

Das Problem der Türme von Hanoi wird häufig verwendet, um in die Techniken der rekursiven Programmierung einzuführen. Ziel ist es, einen Stapel von Scheiben, der sich auf einem von drei Pfeilern befindet, auf einen anderen Pfeiler umzuschichten. Dabei wird der dritte Pfeiler als Hilfspfeiler benutzt. Jede Scheibe hat eine andere Größe. Die oberste Scheibe des Stapels auf einem Pfeiler darf oben auf den Stapel eines der beiden anderen Pfeiler gelegt werden, wobei nur eine Scheibe auf einmal bewegt und eine Scheibe niemals auf eine Scheibe gelegt werden darf, die kleiner ist als sie selbst. Die Einzelheiten des Algorithmus, der die Folge der erforderlichen Züge auflistet, hängen von der Struktur des erforderlichen Klassendiagramms ab.

a) Zeichnen Sie Klassendiagramme für jede der folgenden Beschreibungen. Verwenden Sie dabei Objektklassen und Assoziationen, nicht aber Attribute oder Operationen:

   1. Ein Turm besteht aus drei Pfeilern. Auf jedem Pfeiler befinden sich mehrere Scheiben in einer bestimmten Reihenfolge.
   2. Ein Turm besteht aus drei Pfeilern. Die Scheiben auf den Pfeilern sind in Untermengen aufgeteilt, die als Stapel bezeichnet werden. Ein Stapel ist eine geordnete Menge von Scheiben. Jede Scheibe befindet sich in genau einem Stapel. Auf einem Pfeiler können sich mehrere Stapel befinden, die der Größe nach geordnet sind.
   3. Ein Turm besteht aus drei Pfeilern. Die Scheiben auf den Pfeilern sind in Untermengen aufgeteilt, die als Stapel bezeichnet werden, wobei sich, wie in 2., mehrere Stapel auf einem Pfeiler befinden können. Die Struktur eines Stapels ist jetzt jedoch rekursiv. Ein Stapel besteht aus einer Scheibe (der untersten Scheibe des physikalischen Stapels) und, je nach Höhe des Stapels, null oder einem Stapel.
   4. Ähnlich wie 3., außer dass nur ein Stapel mit einem Pfeiler assoziiert ist.

b) Angelpunkt des in a) beschriebenen, rekursiven Algorithmus, der die Folge von Zügen generiert, ist ein Stapel Scheiben. Um einen Stapel der Höhe N umzuschichten, wobei N>1 ist, setzen Sie unter Verwendung eines rekursiven Aufrufs zuerst den Stapel der Höhe N-1 auf den freien Pfeiler. Danach setzen Sie die unterste Scheibe auf den gewünschten Pfeiler.

Schließlich setzen Sie den Stapel von dem freien Pfeiler auf den gewünschten Pfeiler. Die Rekursion terminiert, weil das Umschichten eines Stapels der Höhe 1 trivial ist. Welches der in a) erstellten Klassendiagramme eignet sich für diesen Algorithmus am besten? Erörtern Sie, warum. Fügen Sie diesem Diagramm dann Attribute und Operationen hinzu. Wie sehen die Argumente der einzelnen Operationen aus? Beschreiben Sie, wie die einzelnen Operationen die Klassen, für die sie beschrieben sind, manipulieren sollen.

Aufgabe 6: Verwandtschaftsbeziehungen

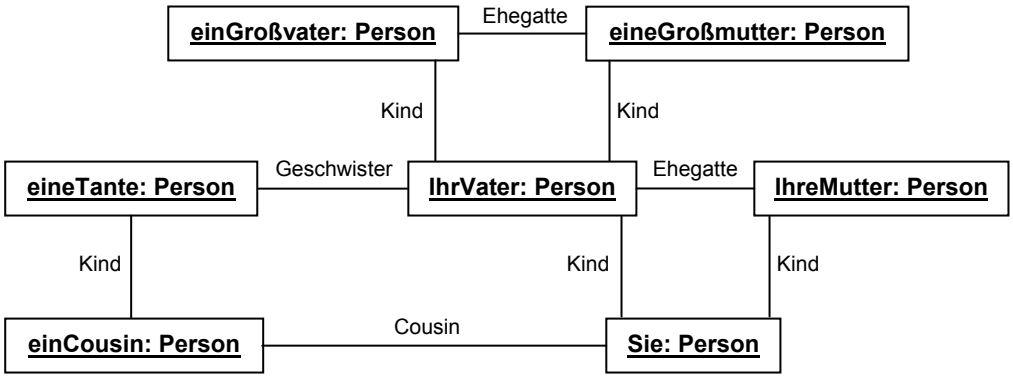


Abbildung 34: Verwandtschaftsbeziehungen als Objektdiagramm

Entwickeln Sie ein Klassendiagramm zu dem in Abbildung 34 gegebenen Objektdiagramm. Berücksichtigen Sie dabei auch Multiplizitäten und die Benennung der Relationen.

Aufgabe 7: Mini-Grafikeditor

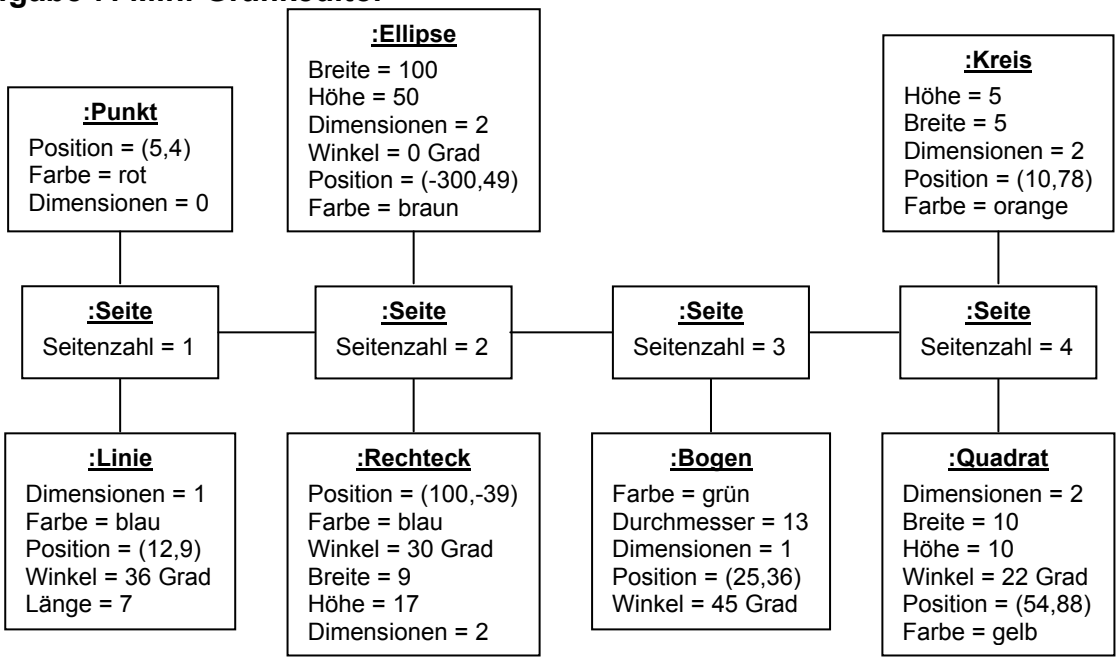


Abbildung 35: Objektdiagramm zu einem Dokument eines Mini-Grafikeditors

Entwickeln Sie ein Klassendiagramm zu dem gegebenen Objektdiagramm (vgl. Abbildung 35) eines Dokuments eines Mini-Grafikeditors. Berücksichtigen Sie dabei auch Multiplizitäten.

Aufgabe 8: Ein Schlossmodell

a) Zeichnen Sie ein Klassendiagramm, das mindestens zehn Relationen zwischen folgenden Klassen zeigt: Burg, Burggraben, Zugbrücke, Turm, Geist, Treppe, Kerker, Boden, Korridor, Geheimkammer, Zimmer, Fenster, Stein, Schlossherr, Burgfräulein, Köchin. Fügen Sie Assoziationen, Aggregationen und Vererbungsbeziehungen hinzu.

b) Fügen Sie mindestens 15 Attribute und fünf Methoden hinzu.

Aufgabe 9: Kaffee kochen

Stellen Sie Ihre Aktivitäten beim Kochen von Kaffee mittels einer Kaffeemaschine mit einem Aktivitätsdiagramm dar.

E Materialien der empirischen Studien zu Explorationsmodulen

E.1 Referenzmodell bei der Analyse von Modellierungswerkzeugen

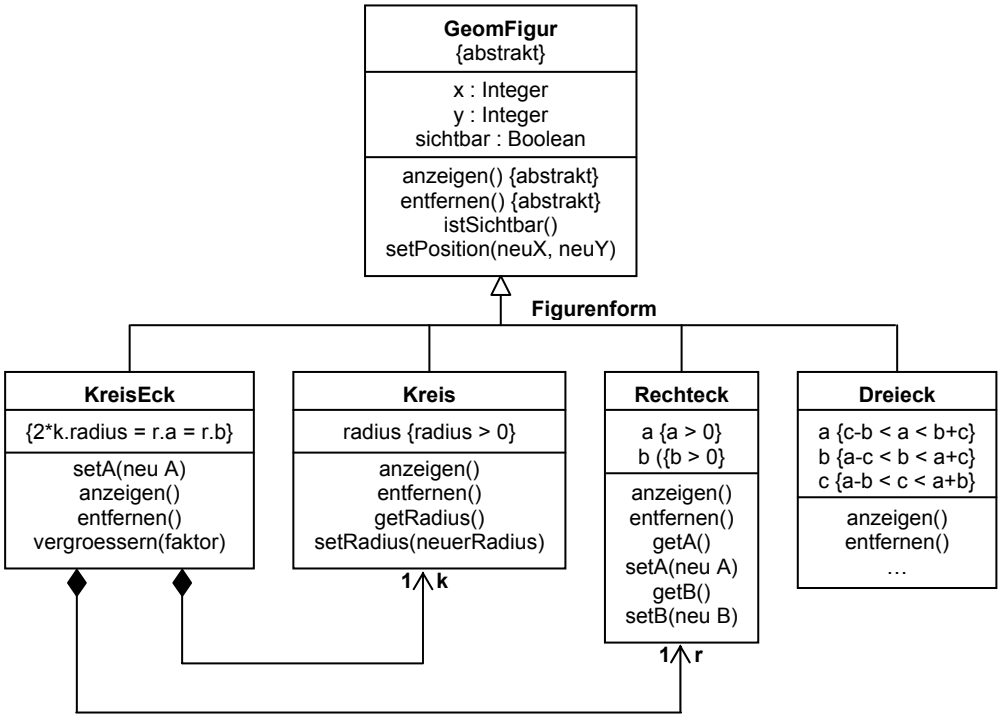


Abbildung 36: Klassendiagramm-Referenzmodell (vgl. Oestereich 1998, 52)

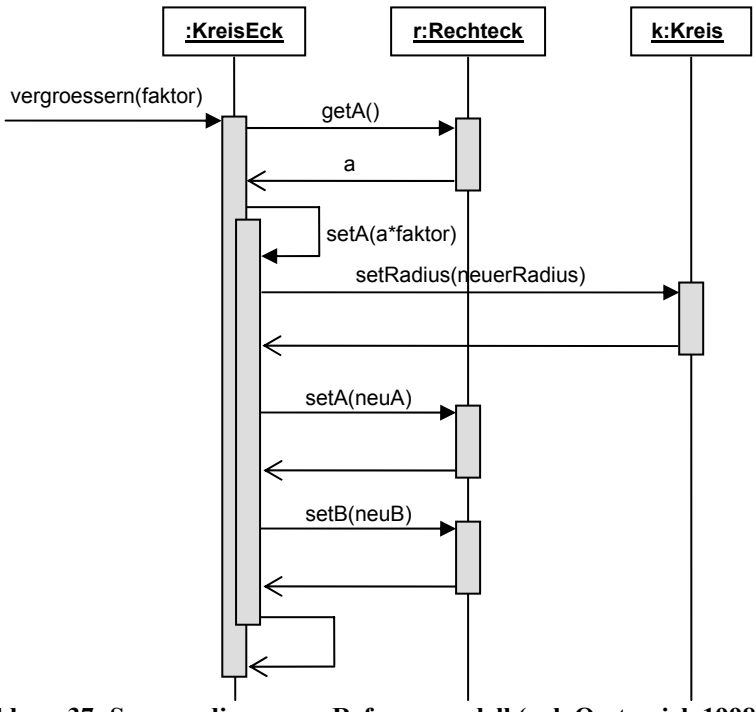


Abbildung 37: Sequenzdiagramm-Referenzmodell (vgl. Oestereich 1998, 55)

E.2 Aufgabe im Rahmen der Schnupperuni 2001

[...] Ein wesentliches Teilgebiet der Informatik ist die Konstruktion von Software. Insbesondere größere Projekte erfordern die Bearbeitung im Team. Um den Überblick nicht zu verlieren und um schließlich in der geplanten Zeit zum gewünschten Ergebnis zu gelangen, ist eine sorgfältige Planung zwingend erforderlich. Eine verbreitete Methode zur systematischen Konstruktion von Software ist das objektorientierte Modellieren. Dazu betrachtet man den gegebenen Problembereich als Menge von strukturierten Objekten, die miteinander kommunizieren und so die Funktionalität des Systems bereitstellen. In einem Bibliotheksinformationssystem gibt es z.B. ein Lagerobjekt, in dem sich viele Buchobjekte befinden. Das Kundenverwaltungsobjekt enthält viele Kundenobjekte u.s.w. Jeden einzelnen Kunden kann man dabei anhand bestimmter Kriterien klassifizieren, wie z.B. Name, Anschrift, Kundennummer. Über jeden Kunden wird diese Information erfasst, d.h. alle Kundenobjekte haben einen gleichen strukturellen Aufbau. Die Information, die in allen Objekten desselben Typs enthalten ist fasst man in einer Klasse zusammen. Eine Klasse Kunde ist somit eine Art „Bauplan“ zur Erstellung von konkreten Kundenobjekten.
Aufgabe C.1 und C.2 ermöglichen es, mit so genannten (stark vereinfachten) Klassendiagrammen zu experimentieren. Klassendiagramme dienen als Planungshilfe für den inneren Aufbau von Software. Gezeigt wird jeweils eine fertige Anwendung, zu der im zweiten Schritt ein passendes Klassendiagramm erzeugt werden soll. Die Rechtecke repräsentieren jeweils eine Klasse. Sie lassen sich auf der Zeichenfläche verschieben. Klassen können in verschiedenen Beziehungen zueinander stehen. Eine wesentliche Beziehung ist die sogenannte Aggregation, die man umgangssprachlich auch als besteht-aus-Beziehung bezeichnen kann. In den Aufgaben C.1 und C.2 soll jeweils ein fertiges Klassendiagramm erstellt werden. Dazu sollte man die Klassen so anordnen, dass klar wird, zwischen welchen eine Aggregationsbeziehung besteht. Für diese können die vorbereiteten Aggregationsbeziehungen verwendet werden. Diese kann man verschieben und an den Ende verlängern oder verkürzen. Um darzustellen, dass Klasse A aus Klasse B besteht, verschiebt man die Aggregation mit der Raute in die Klasse A und zieht das andere Ende auf Klasse B. Mit dem Prüfen-Knopf kann das Klassendiagramm überprüft werden.

Aufgabe C.1
Starte unter Netscape folgende URL:
http://www.didaktik-der-informatik.de/flash/ventilator.swf[105]

Aufgabe C.2
Starte unter Netscape folgende URL:
http://www.didaktik-der-informatik.de/flash/puzzle_raum.swf[106]

[105] Gegenüber dem Original wurde die Webadresse auf die Webseiten der Universität Siegen verändert, auf denen die Materialien jetzt liegen (aufgerufen am 12.11.2003)
[106] vgl. Fußnote 105

E.3 Schreiben zu LEO

OStR Ralf Sagorny
Schulstrasse 12a, 59192 Bergkamen
Tel:02307-963414 Fax:02307-963415
E-Mail: sagorny@web.de

Ralf Sagorny-Schwarz - Schulstrasse 12a - 59192 Bergkamen

Universität Siegen
Didaktik der Informatik und E-Learning
Dipl.-Inform. Torsten Brinda
Hölderlinstr. 3

57068 Siegen

Donnerstag, 23. Oktober 2003

Lernumgebung für Objektorientiertes Modellieren (LEO)

Sehr geehrter Herr Brinda,

ich bin als Informatiklehrer am Freiherr-vom-Stein Berufskolleg in Werne beschäftigt. Parallel zu meiner Tätigkeit als Lehrer bin ich als Fachmoderator im Unterrichtsfach Informatik für IT-Berufe für die Bezirksregierung Arnsberg seit vielen Jahren tätig. Ich beschäftige mich seit geraumer Zeit mit der objektorientierten Modellierung. Die in diesem Themenzusammenhang entwickelten Unterrichtsideen habe ich mit Herrn Mielke im Schulbuch „Programmentwicklung mit UML", erschienen im Bildungsverlag EINS, zusammengefasst.
Bei der Suche nach geeigneten Tools zur Unterstützung des Unterrichts in diesem Bereich, bin ich auf das von Ihnen betreute LEO-Projekt und die dort entwickelte Lernumgebung für objektorientiertes Modellieren gestoßen. Diese habe ich sowohl in mehreren Lehrerfortbildungen, als auch im Unterricht der Höheren Berufsfachschule, Bildungsgang Informationstechnischer Assistent, eingesetzt. Ich möchte Ihnen in diesem Schreiben über meine Erfahrungen mit dem Tool berichten und Sie animieren die Entwicklung daran voran zu treiben.
Die Vermittlung von objektorientierten Konzepten bereitet dem Lernenden, insbesondere in der Anfangsphase, große Probleme bei der Vorstellung der Arbeitsweise objektorientierter Programmsysteme. Dies liegt vermutlich daran, dass der Lernende zum einem vor dem Problem der Identifikation von Klassen steht und zu anderem eine Vorstellung entwickeln muss, wie das Verhalten der Anwendung aus Objekten daraus resultiert. Eine weitere pädagogische Schwierigkeit liegt in der Auswahl geeigneter Beispiele und deren anschauliche Modellierung. Hier helfen professionelle Tools wie z.B. Together Control Center zwar bei der Darstellung der Analyse und der Generierung von Klassenrümpfen. Dem Lernenden hilft dies aus meiner Erfahrung jedoch wenig beim Verständnis des Modellierungsvorgangs.
An dieser Stelle lässt sich die Lernumgebung LEO phantastisch einsetzen. Die gewählte Darstellungsform, mit der Sicht aus fünf verschiedenen Perspektiven, ermöglicht dem Lernenden zu jedem Zeitpunkt eine anschauliche Sichtweise auf die Problemstellung. Ein weiterer Pluspunkt ist die Wahl der von Ihnen gewählten Beispiele. Diese lagen bei allen Lerngruppen, in denen ich das Tool bisher eingesetzt habe, im täglichen Erfahrungsbereich. Besonders gut aufgenommen wurde hier das Beispiel Mobiltelefon mit dem Ziel Versand einer SMS.
Der Einsatz des Tools erfolgte in meinem Unterricht ohne Einweisung auf das Tool. Es wurde lediglich der Import der Szenarios erläutert, nicht jedoch der Umgang mit dem Tool selbst. Die Vorkenntnisse der Lerngruppen beschränkten sich zum Zeitpunkt des Einsatzes auf die objektorientierten Grundbegriffe, so wie die Grundzüge der Darstellung des Klassendiagramms mit Hilfe der UML. Die Lernenden fanden durchweg zügig die Möglichkeit der Erzeugung von Objekten durch Anklicken eines Konstruktors im Klassendiagramm. Die dann im Objektdiagramm sichtbar werdenden Operationen bereiteten den Lernenden keine Probleme mehr. Das Tool visualisiert in dieser Phase sehr anschaulich den Schablonencharakter einer Klasse, da außer der Objektsicht das entsprechende Objekt auch in der

Seite 2 23. Oktober 2003

Realsicht sichtbar wird. Der Aufruf von Operationen zeigt außerdem den Versand von Botschaften zwischen den Objekten. Diese Art der Darstellung ist mir aus keinem anderen bisher bekannten Tool bekannt. Sie hilft jedoch dem Lernenden enorm beim Verständnis der objektorientierten Konzepte. Das System motivierte mich, ein von mir gerne angewandtes Beispiel, die Kaffeemaschine, für die Lernumgebung umzusetzen. Leider habe ich die Umsetzung aus Zeitgründen noch nicht fertig gestellt. Ich bin jedoch bemüht dies schnellst möglich nachzuholen.
Sehr anschaulich ist auch die Darstellung der Operationen im Sequenzdiagramm und in der Programmsicht. Schön wäre hier noch, wenn das Sequenzdiagramm automatisch scrollen würde, um immer die aktuell ausgewählten Operationen sehen zu können.
Ich hoffe sie zur Weiterentwicklung des Tools animiert zu haben.

Mit freundlichen Grüßen

R. Sagt

E.4 Arbeitsblätter zum Explorationsmodulkonzept

Vorlesung „Didaktik der Informatik I“ (Universität Dortmund, SoSe 2001)

Dortmund, den 06.06.01

Übung zur Vorlesung „Didaktik der Informatik I (für Sek. II)“ Blatt 8

Im Übungsblatt 6 haben Sie sich mit dem Konzept des Informatikexperiments auseinandergesetzt. Dies Übungsblatt knüpft an die dort gewonnenen Erkenntnisse an. Im Rahmen einer studentischen Arbeit wurde das in der Anlage beigefügte Lernmodul für ausgewählte OO-Konzepte entwickelt. Es handelt sich um eine Entwicklungsfassung, die noch „Ecken und Kanten“ hat, aber voll funktionsfähig ist. Da Sie in Ihrem späteren Berufsleben immer wieder auf von anderen entwickelte Lernmodule zurückgreifen werden und beurteilen müssen, inwieweit diese für Ihren Unterricht einsetzbar sind, werden sich auch die nächsten 1 - 2 Übungsblätter mit diesem Thema befassen.

Installationshinweis
Entpacken Sie die Datei modul.zip in ein Verzeichnis Ihrer Wahl. Editieren Sie die Datei Start.bat, so dass diese zu Ihrem Installationsort passt. Sofern Sie jdk nicht auf Ihrem System installiert haben, holen Sie dies bitte nach (kostenloser Download unter http://java.sun.com/). Da alle class-Dateien im Archiv modul.zip enthalten sind, können Sie das System durch Doppelklick auf Start.bat starten.

1. Aufgabe: 0 Punkte
Erkunden Sie das System und experimentieren Sie mit diesem.

2. Aufgabe: 2.5 Punkte
Welche Basiskonzepte werden eingeführt? Welche Erkenntnisprozesse werden gefördert?

3. Aufgabe: 4.5 Punkte
Beschreiben Sie drei verschiedene Einsatzszenarien (Lerngruppe, Jahrgangsstufe, Vorkenntnisse, Einbettung in eine Unterrichtsreihe, Thema der Einzelstunde, Lernziele).

4. Aufgabe: 3 Punkte
Welche Stärken und Schwächen lassen sich bei dem Lernmodul im Hinblick auf einen Unterrichtseinsatz identifizieren? Vergleichen Sie dabei das Lernmodul mit traditionellen Lehr-Lern-Materialien (Texte, Tafelbilder, Folien, etc.).

Abgabe bis zum 13.06.01

Dortmund, den 13.06.01

Übung zur Vorlesung „Didaktik der Informatik I (für Sek. II)“
Blatt 9

Das Übungsblatt 9 knüpft inhaltlich an das Übungsblatt 8 an. Gegenstand der Analyse ist wieder das zusammen mit Übungsblatt 8 verteilte Lernmodul.

1. Aufgabe: 0 Punkte
Betrachten Sie bitte die mitverteilten Quelltexte des Moduls.

2. Aufgabe: 3 Punkte
Diskutieren Sie die Implementierung. Gehen Sie dabei auch auf die Aspekte Entwurf, Übersichtlichkeit, Wiederverwendbarkeit und Erweiterbarkeit ein.

3. Aufgabe: 2.5 Punkte
Eignen sich die vorliegenden Quelltexte, um mit Lernenden daran Prinzipien und Methoden des objektorientierten Programmierens zu erarbeiten? Begründen Sie Ihre Aussagen. Wie beurteilen Sie generell die Analyse der Realisierung fertiger Systeme im Informatikunterricht?

4. Aufgabe: 1.5 Punkte
Wie beurteilen Sie die Eignung der Sprache Java zur Realisierung des gegebenen Bausteins (anderer Lernmodule)?

5. Aufgabe: 3 Punkte
Erstellen Sie ausgehend von Ihrer Analyse einen eigenen, an Ihre Argumentation angepassten Entwurf (Klassendiagramm) für das Lernmodul.

Abgabe bis zum 20.06.01

Bildschirmfotos des zu analysierenden Lernmoduls[107]

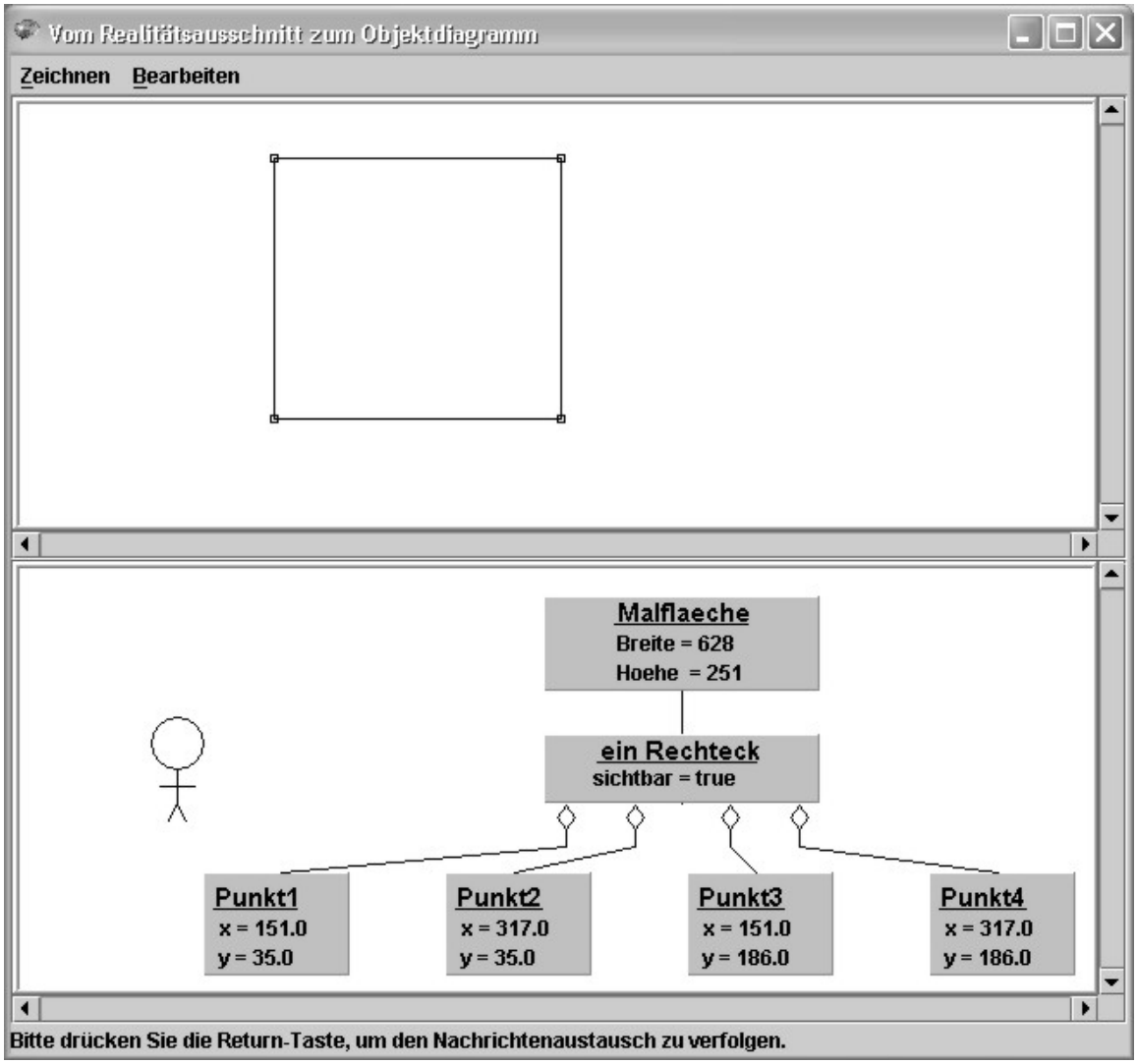


Abbildung 38: GUI der zu analysierenden Software

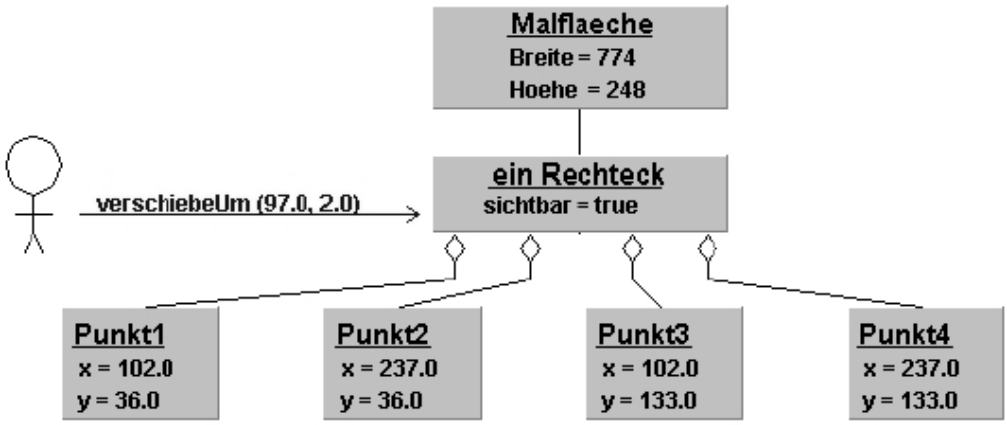


Abbildung 39: Darstellung des Nachrichtenaustauschs in der zu analysierenden Software

[107] Die Software steht mit den Quellen im Internet zum Download bereit unter der Adresse: http://www.didaktik-der-informatik.de/DIE_BIB/lehre/ddi/ddi1/sose01/ueb/8/modul.zip (aufgerufen am 12.11.03)

Vorlesung „Didaktik der Informatik II“ (Universität Dortmund, WiSe 2001/02)

Dortmund, den 14.11.2001

Übung zur Vorlesung „Didaktik der Informatik II“
Blatt 4

In der Vorlesung wurde der informatikdidaktische Zugang zu der Leitlinie „Wirkprinzipien von Informatiksystemen“ vorgestellt. Eine Möglichkeit, Lernende heranzuführen, besteht darin, dass sie unter Verwendung von Software diese Wirkprinzipien selbst entdecken. Software für entdeckendes Lernen wird als Explorationsmodul (EM) bezeichnet.

1. Aufgabe: ***3 Punkte***

Führen Sie eine Recherche zu „Exploration“ durch mit dem Ziel, drei aktuelle Publikationen in je einem Absatz zu beschreiben. Quelle angeben!

2. Aufgabe: ***3 Punkte***

Formulieren Sie drei Informatik-Unterrichtsbeispiele, in denen Sie EMs einsetzen möchten. Nennen Sie den erwarteten Bildungseffekt.

3. Aufgabe: ***4 Punkte***

Welche Anforderungen stellen Sie an die unter der 2. Aufgabe gewünschten EMs bezüglich

(a) Software-Ergonomie,

(b) Förderung der Kooperation,

(c) Einsatzbreite und

(d) Förderung der Kommunikation?

Abgabe und Besprechung in der Übung am 21.11.2001

F Dokumentation zu Explorationsmodulen

F.1 Überblick

Im Anhang F befinden sich ausgewählte Bildschirmfotos zu im Rahmen der Arbeit entstandener Software. Tabelle 48 gibt hierzu einen Überblick über die in den Abschnitten enthaltenen Screenshots und zeigt, unter welchen Internetadressen die entstandenen Explorationsmodule kostenlos zum Download bereit stehen.

Abschnitt	Software	URL (alle geprüft am 09.11.03)
F.2	Flash-Animationen	http://www.didaktik-der-informatik.de/flash/
F.3.1	Explorer für geometrische Objekte – EGO	http://www.didaktik-der-informatik.de/e-learning/objectexplorer
F.3.2	Lernumgebung für objektorientiertes Modellieren – LEO	http://www.didaktik-der-informatik.de/pgleo/

Tabelle 48: URLs von Explorationsmodulen zum OOM

F.2 Animationen

F.2.1 Puzzles

Rechtecke – Klassendiagramm mit Aggregationen

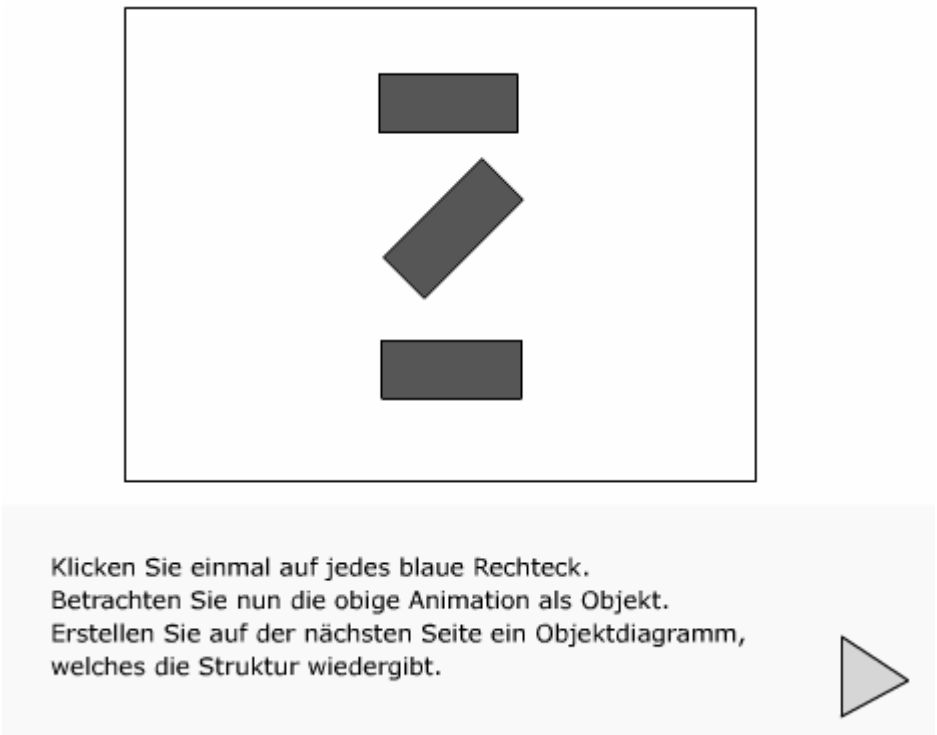


Abbildung 40: Puzzle Rechtecke – Aufgabenstellung

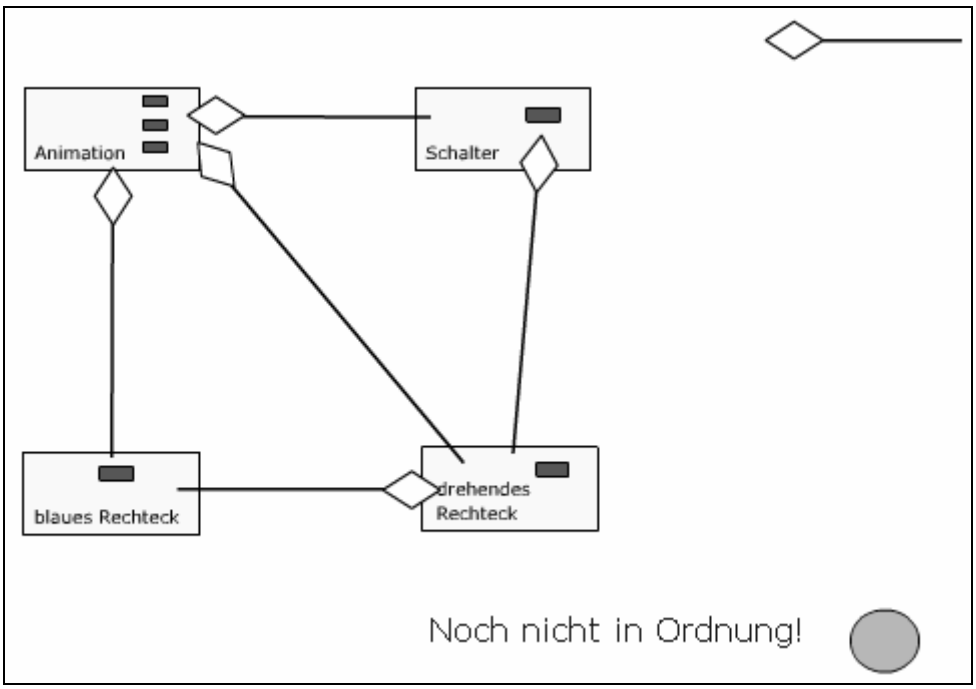


Abbildung 41: Puzzle Rechtecke – Lösungsversuch

Ventilator – Klassendiagramm mit Aggregationen

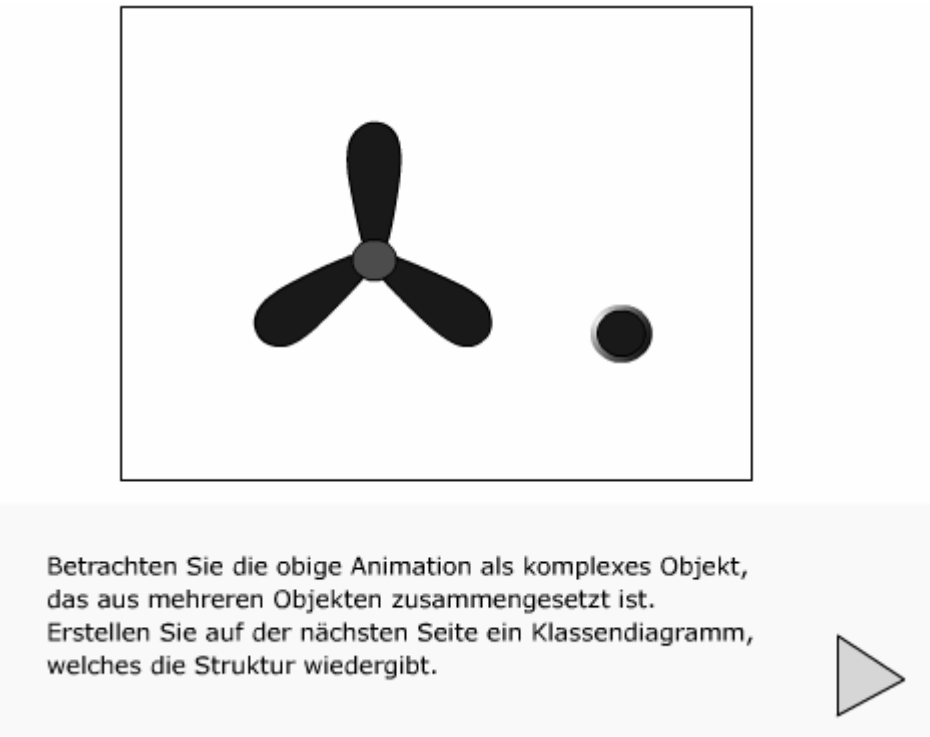


Abbildung 42: Puzzle Ventilator – Aufgabenstellung

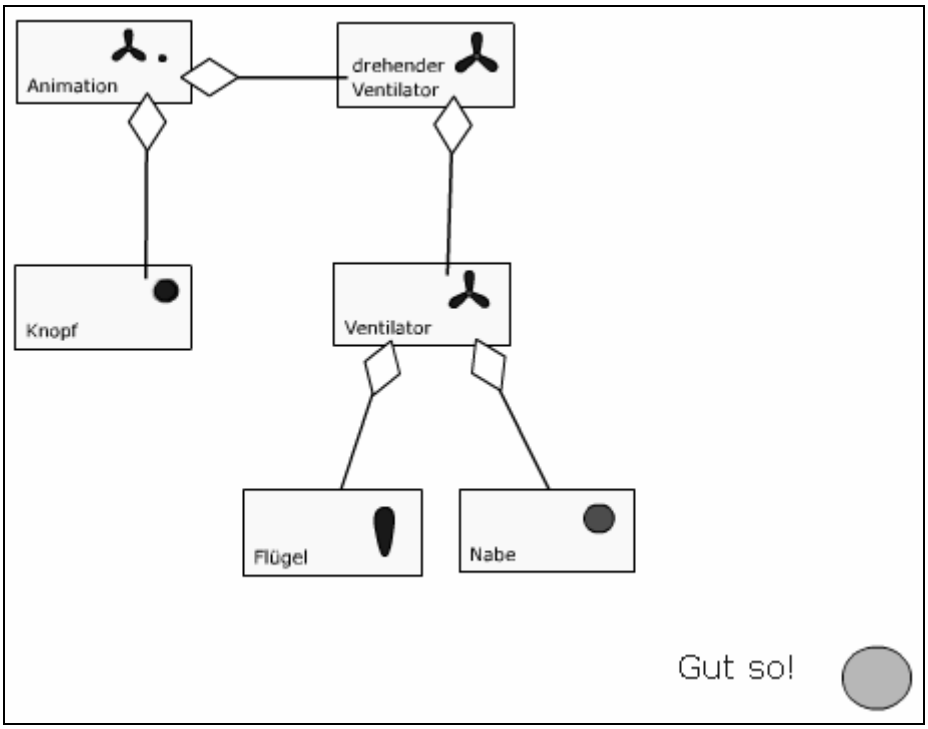


Abbildung 43: Puzzle Ventilator – korrekte Lösung

Raumschiff – komplexeres Klassendiagramm mit Aggregationen

Abbildung 44: Puzzle Raumschiff – Aufgabenstellung

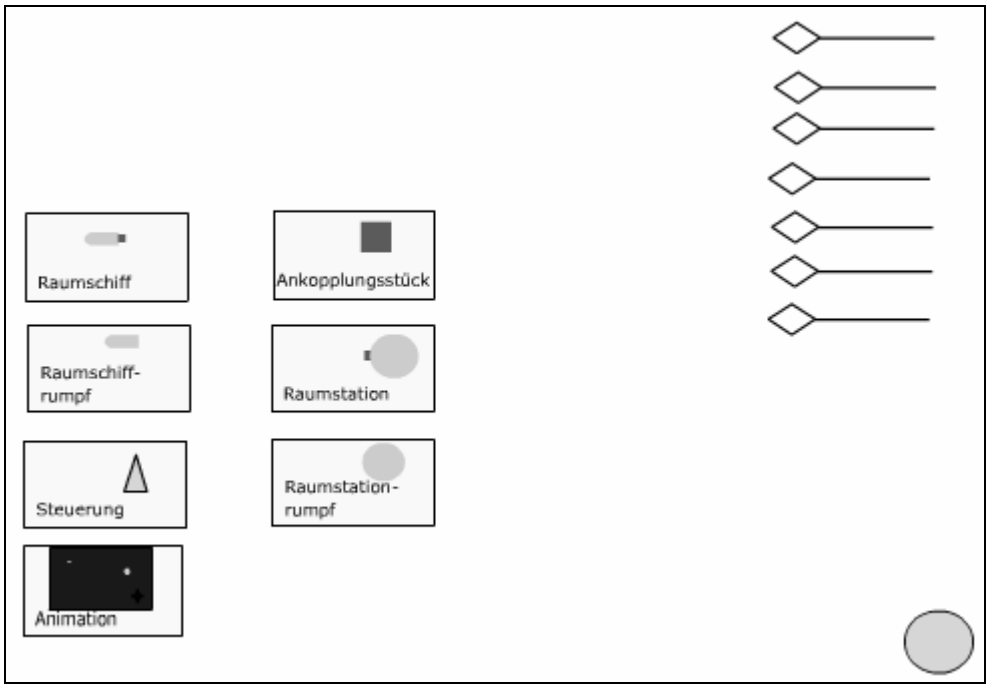
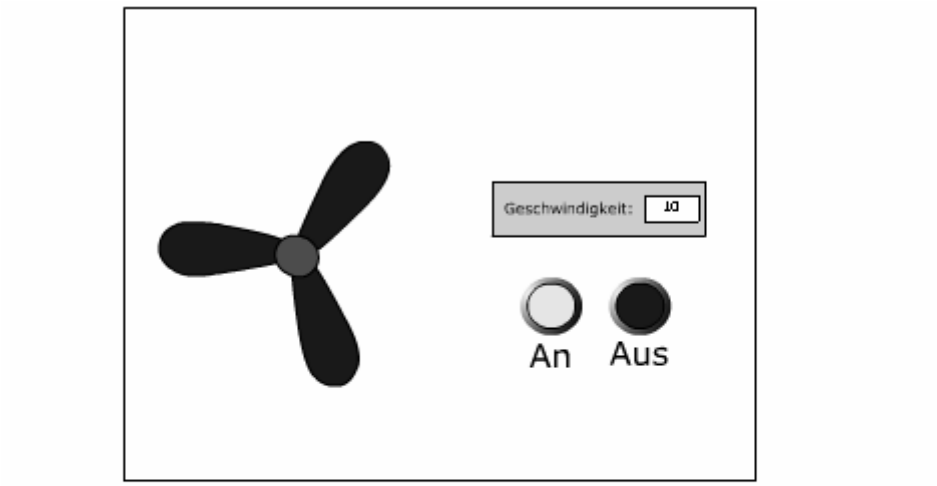


Abbildung 45: Puzzle Raumschiff – Arbeitsfläche zu Beginn

Ventilator 2 – Klassendiagramm mit Aggregationen, Assoziationen, Attributen und Methoden



Betrachten Sie die obige Animation als komplexes Objekt, das aus mehreren Objekten zusammengesetzt ist. Erstellen Sie auf der nächsten Seite ein Klassendiagramm, welches die Struktur wiedergibt. Platzieren Sie die Methoden, Attribute und Sichtbarkeiten dort, wo sie unbedingt notwendig sind.

Abbildung 46: Puzzle Ventilator 2 – Aufgabenstellung

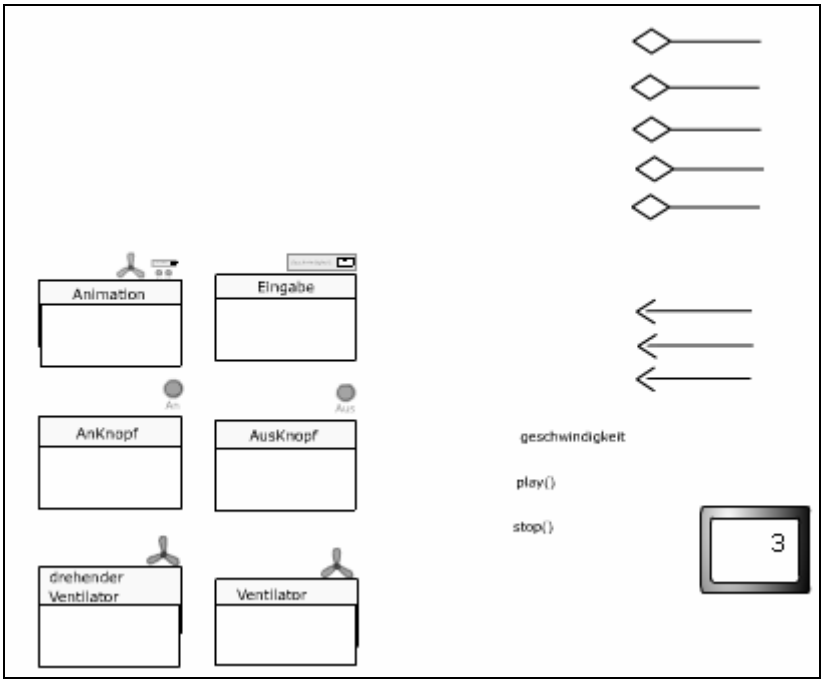


Abbildung 47: Puzzle Ventilator 2 – Arbeitsfläche zu Beginn

F.2.2 Experimentierumgebungen

Kreise – Steuerung von Objekten

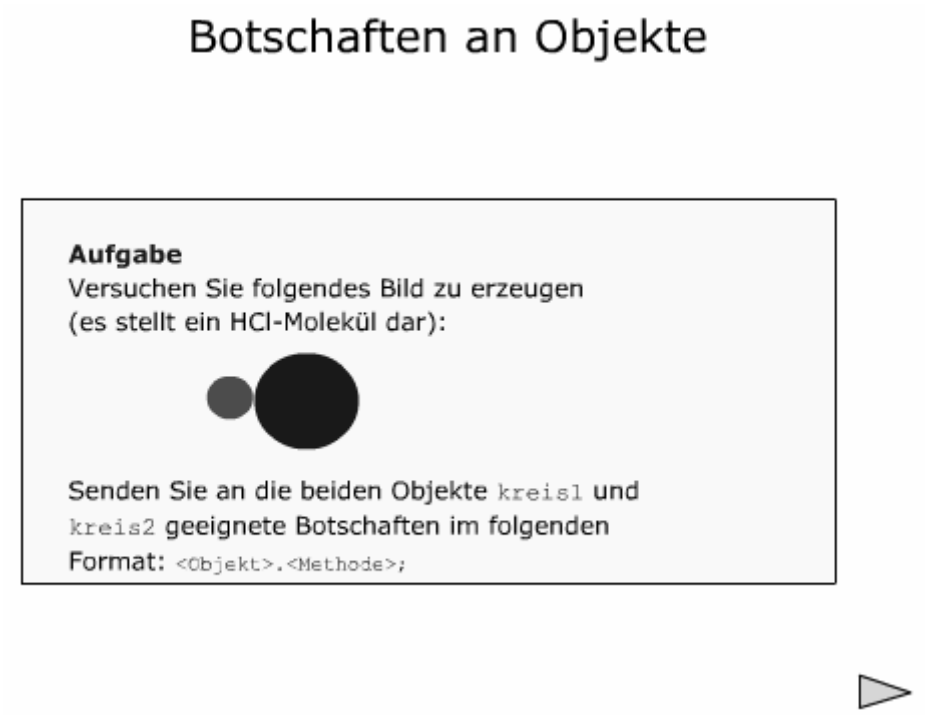


Abbildung 48: Experimentierumgebung Kreise – Aufgabenstellung

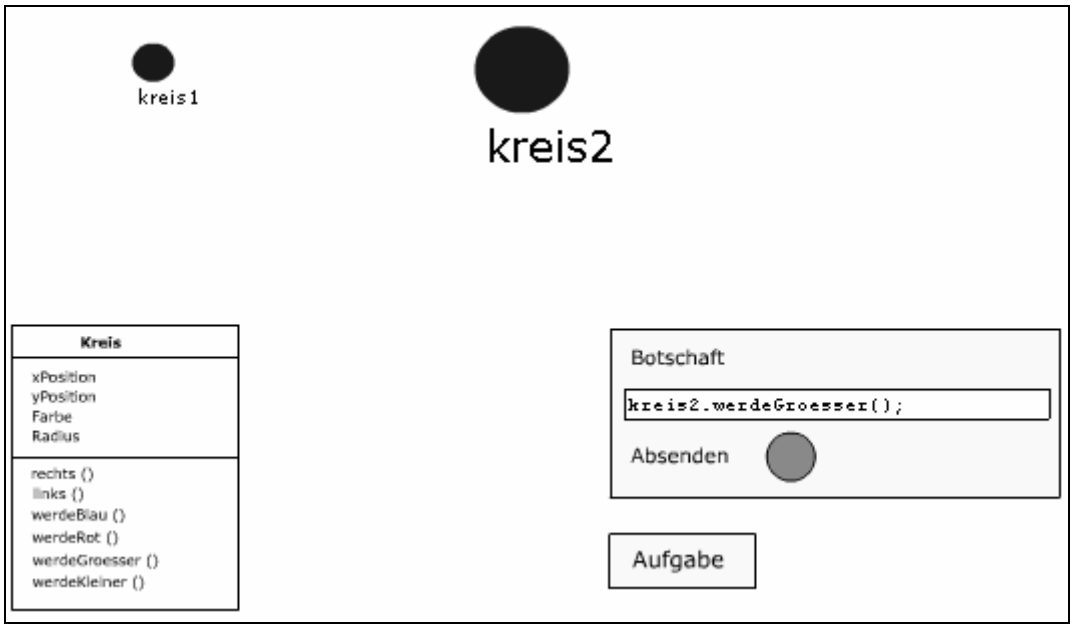


Abbildung 49: Experimentierumgebung Kreise – Arbeitsfläche

Springender Ball – Modell und Beobachter

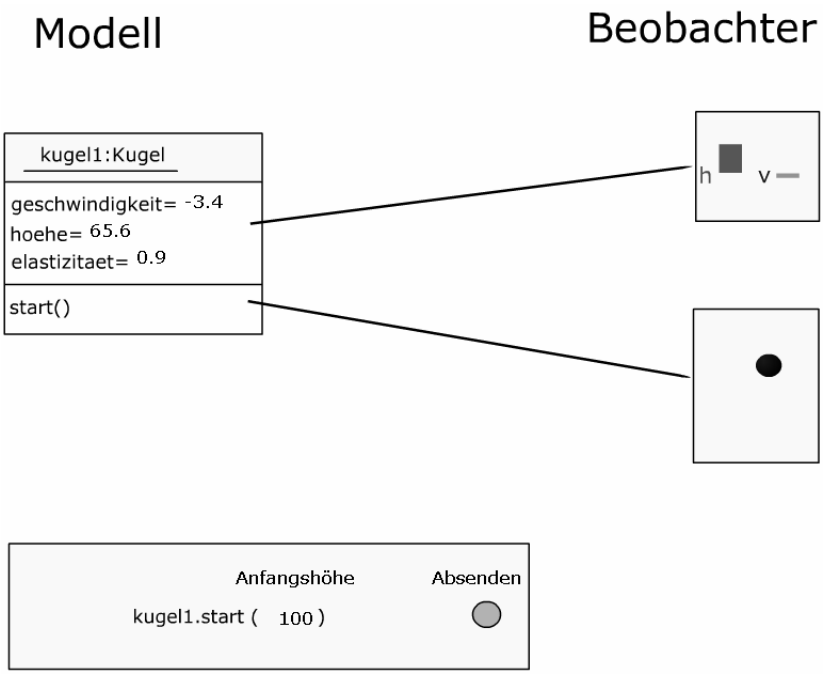


Abbildung 50: Experimentierumgebung „Springender Ball“

Kursverwaltung – Klassendiagramm mit Feedback

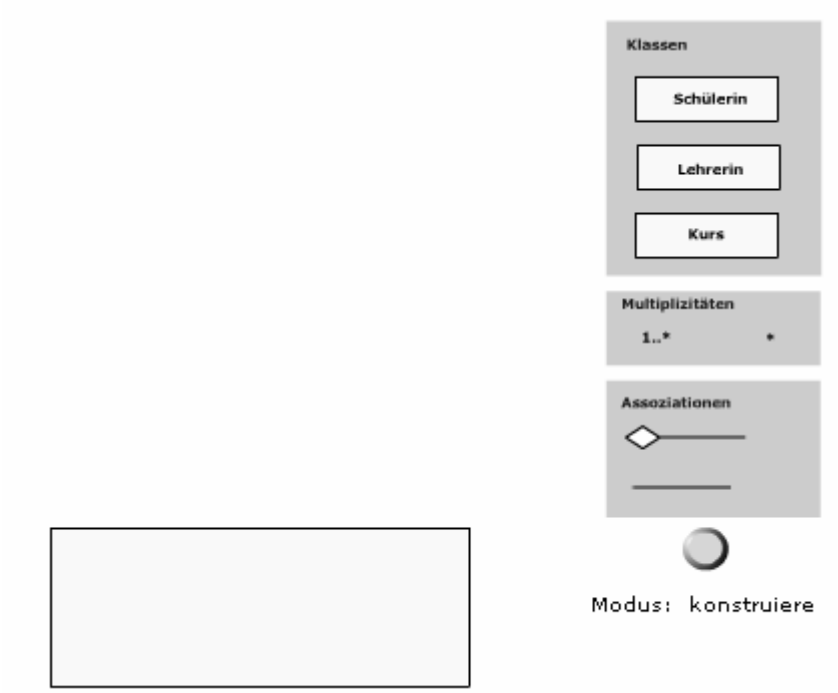


Abbildung 51: Experimentierumgebung Kursverwaltung – Arbeitsfläche zu Beginn

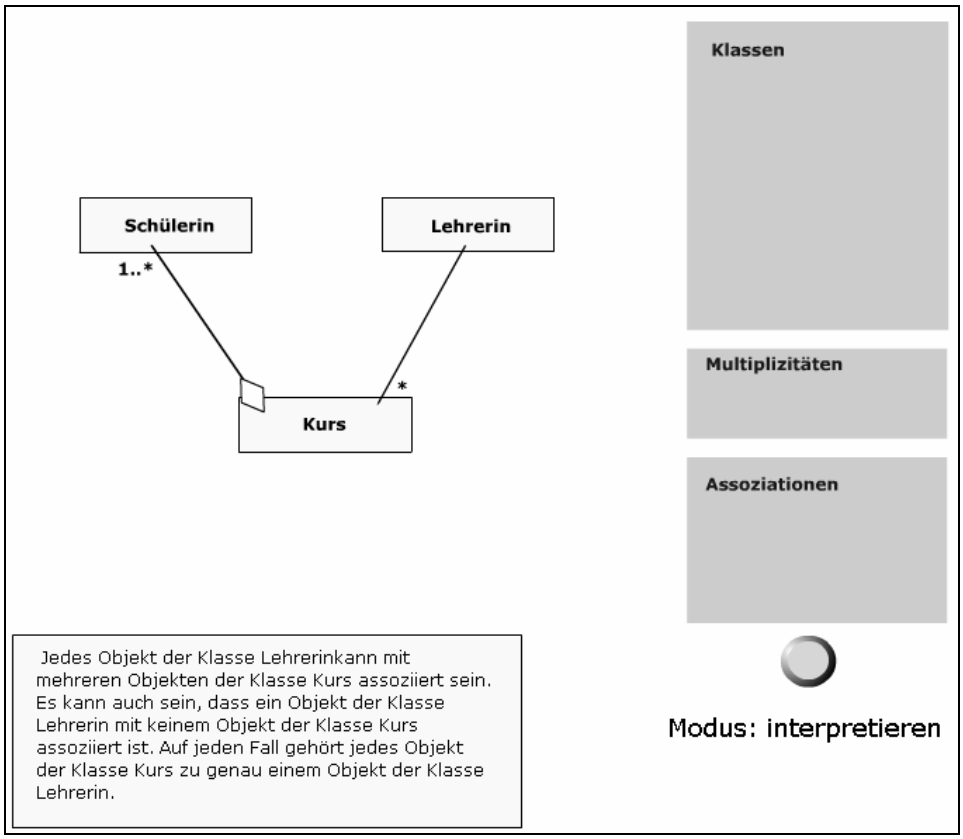


Abbildung 52: Experimentierumgebung Kursverwaltung – Lösungsvariante mit Kommentar

F.3 Anwendungen

F.3.1 Explorer für geometrische Objekte – EGO

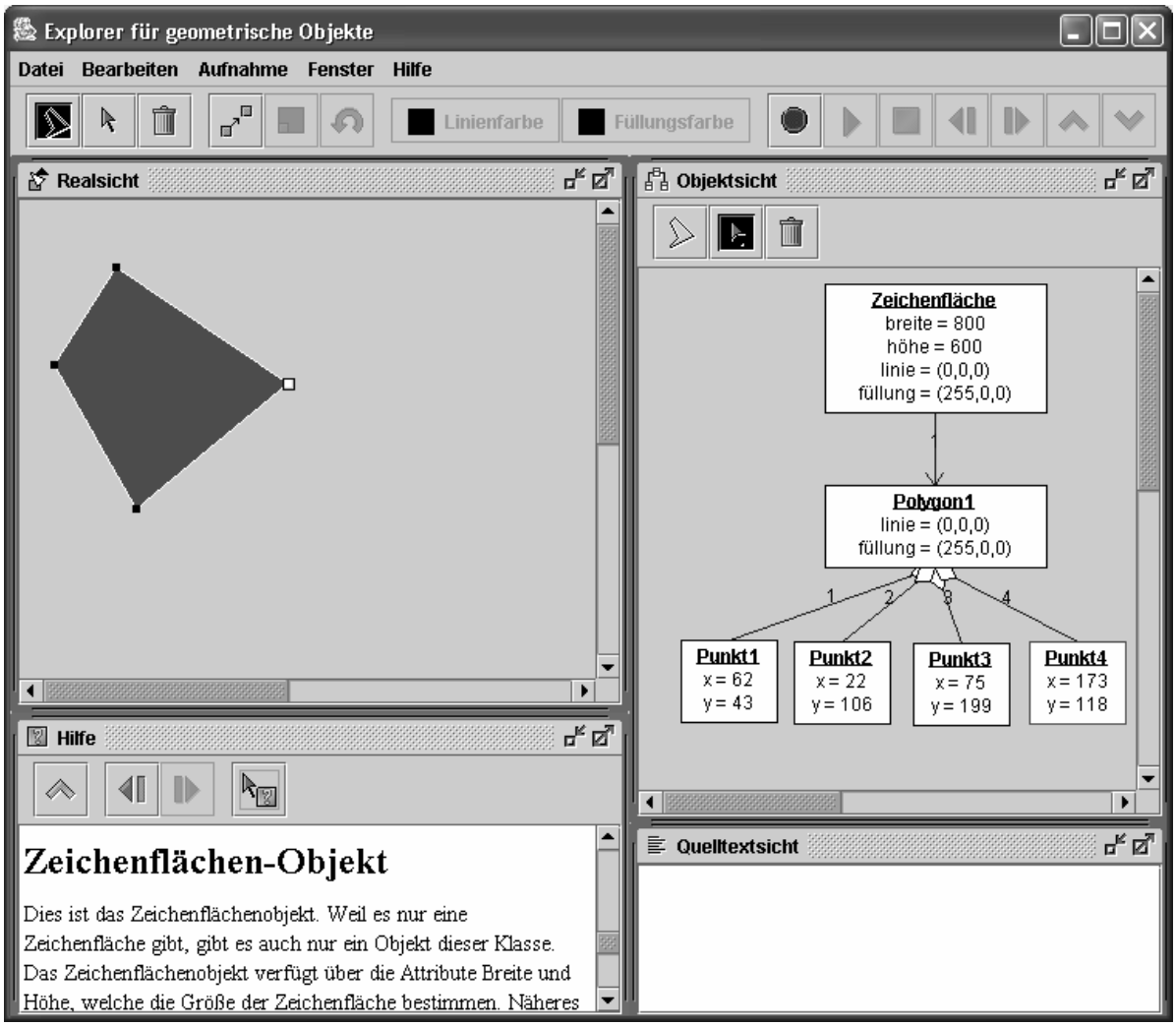


Abbildung 53: EGO – Explorer für geometrische Objekte

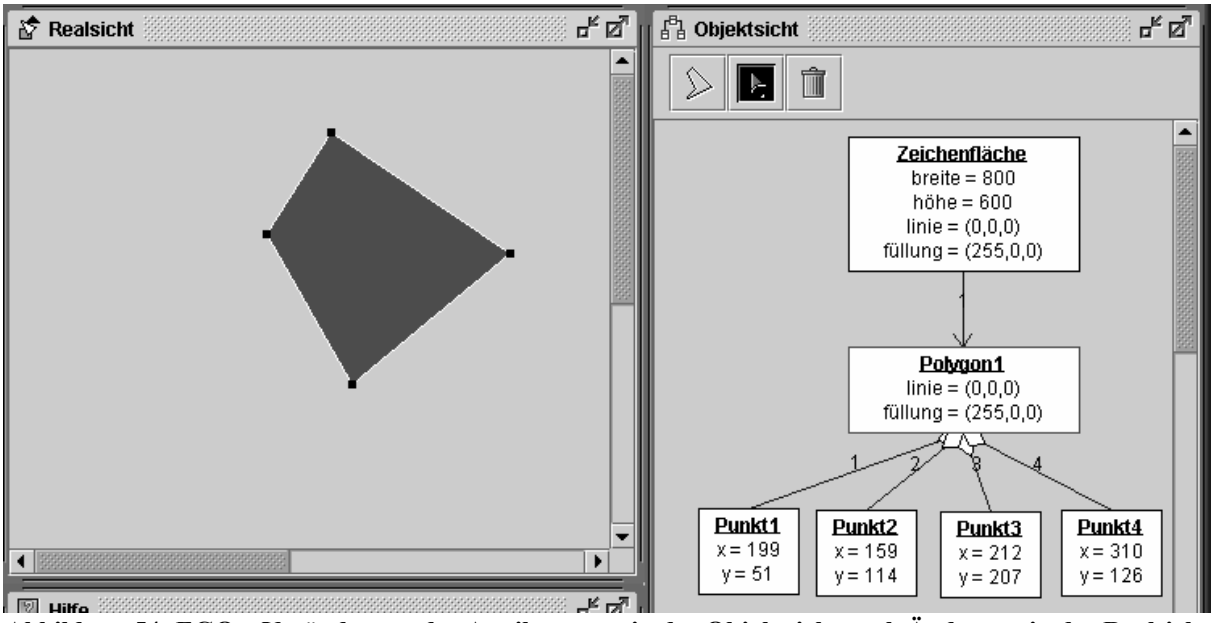


Abbildung 54: EGO – Veränderung der Attributwerte in der Objektsicht nach Änderung in der Realsicht

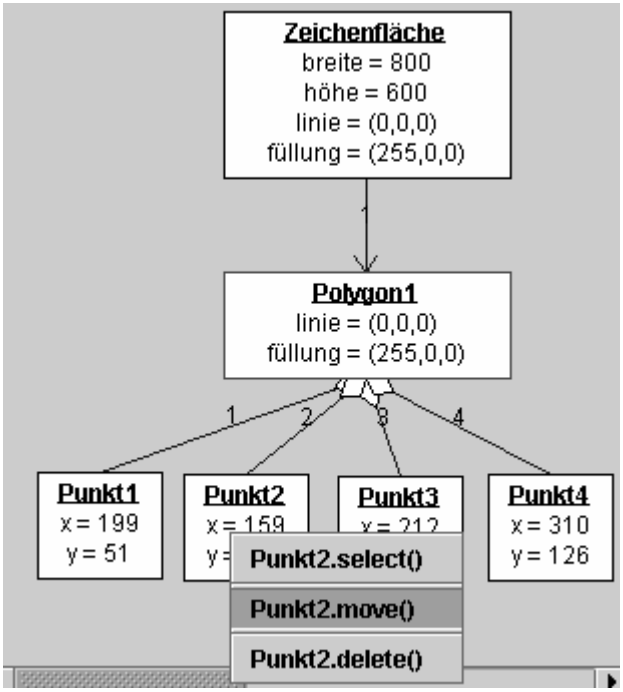


Abbildung 55: EGO – Interaktion in der Objektsicht

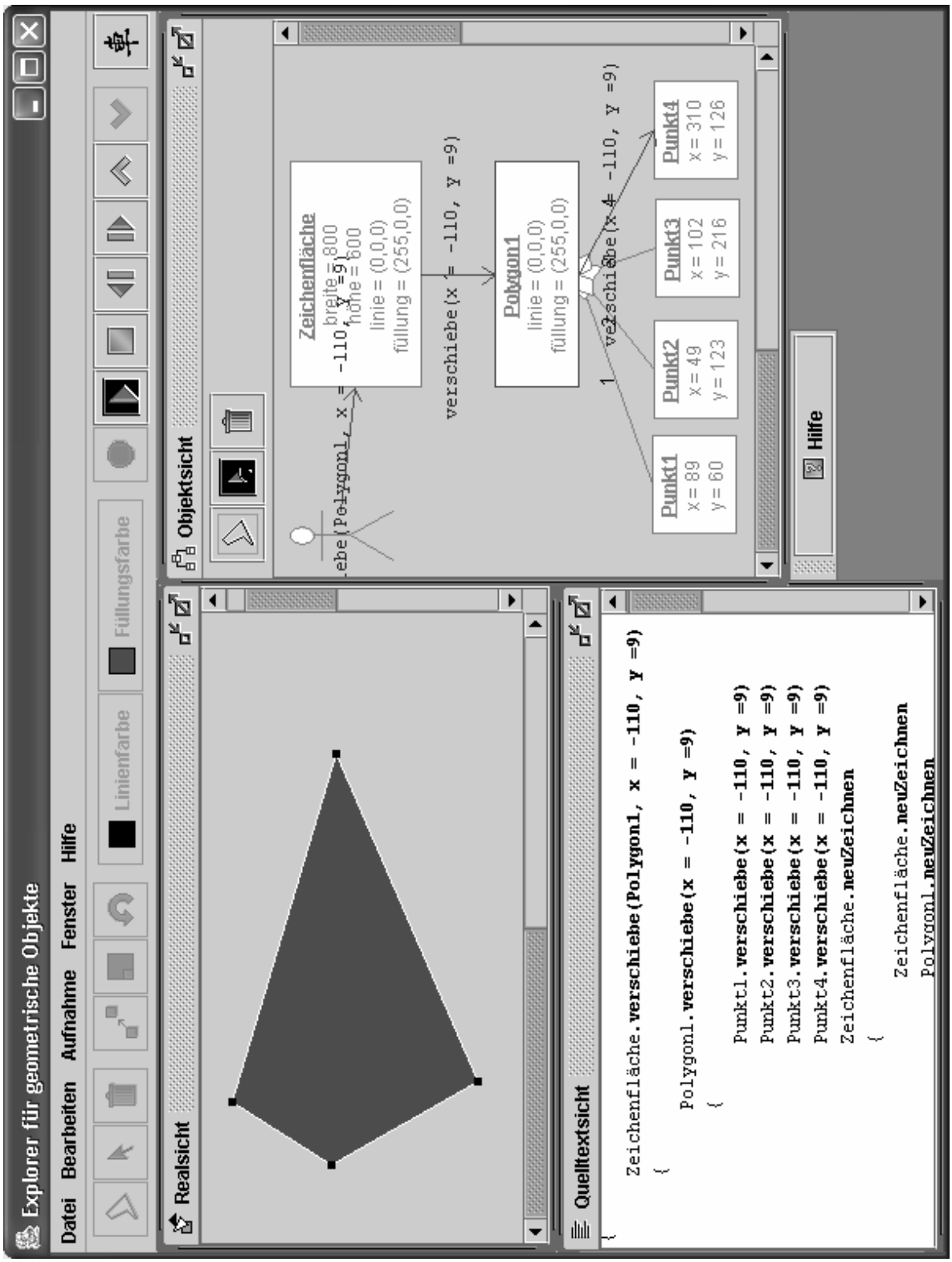


Abbildung 56: EGO – Beobachtung der Objektkommunikation

F.3.2 Lernumgebung für objektorientiertes Modellieren – LEO

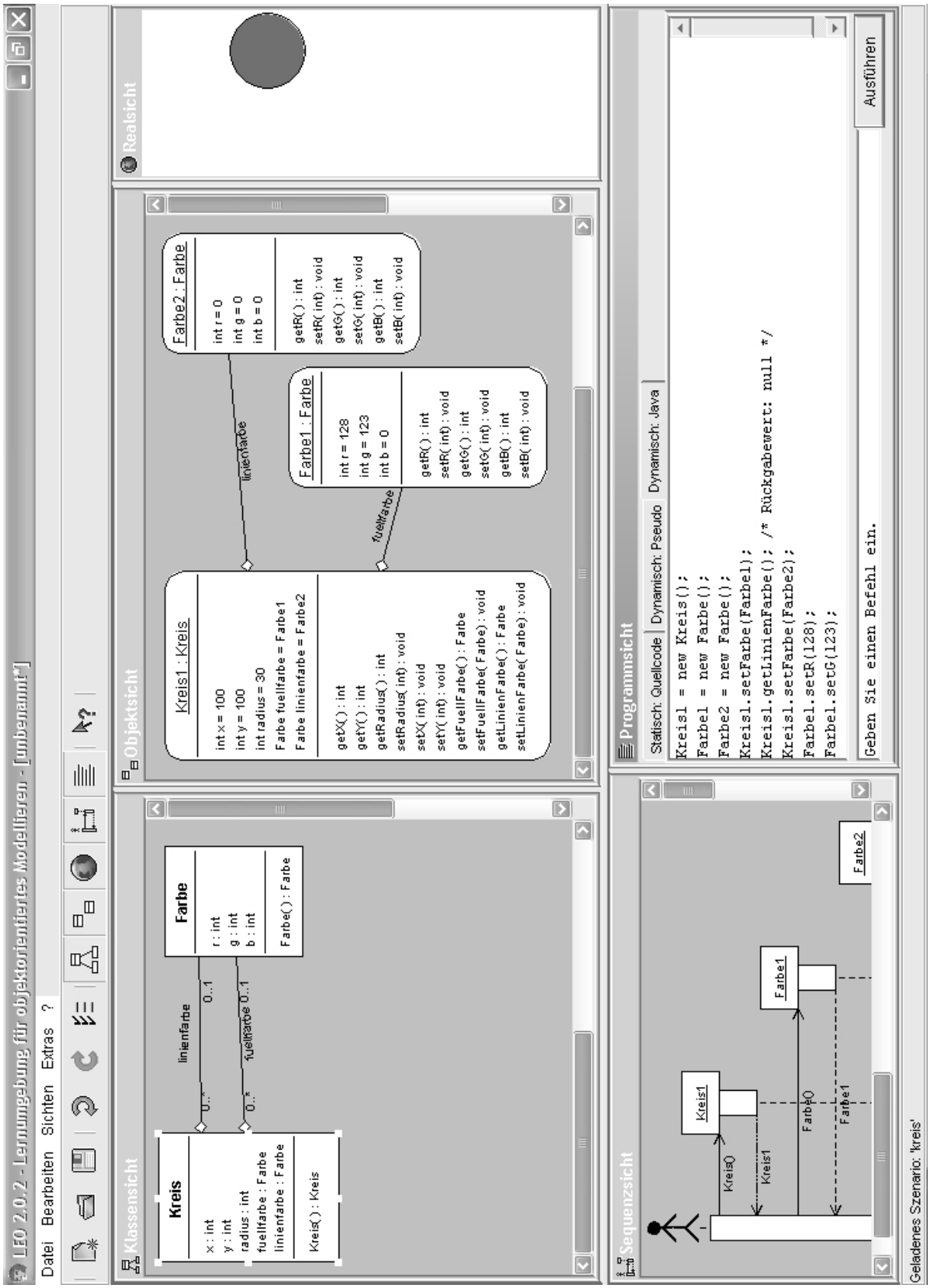


Abbildung 57: LEO – Sichtenwechsel im Szenario Kreis

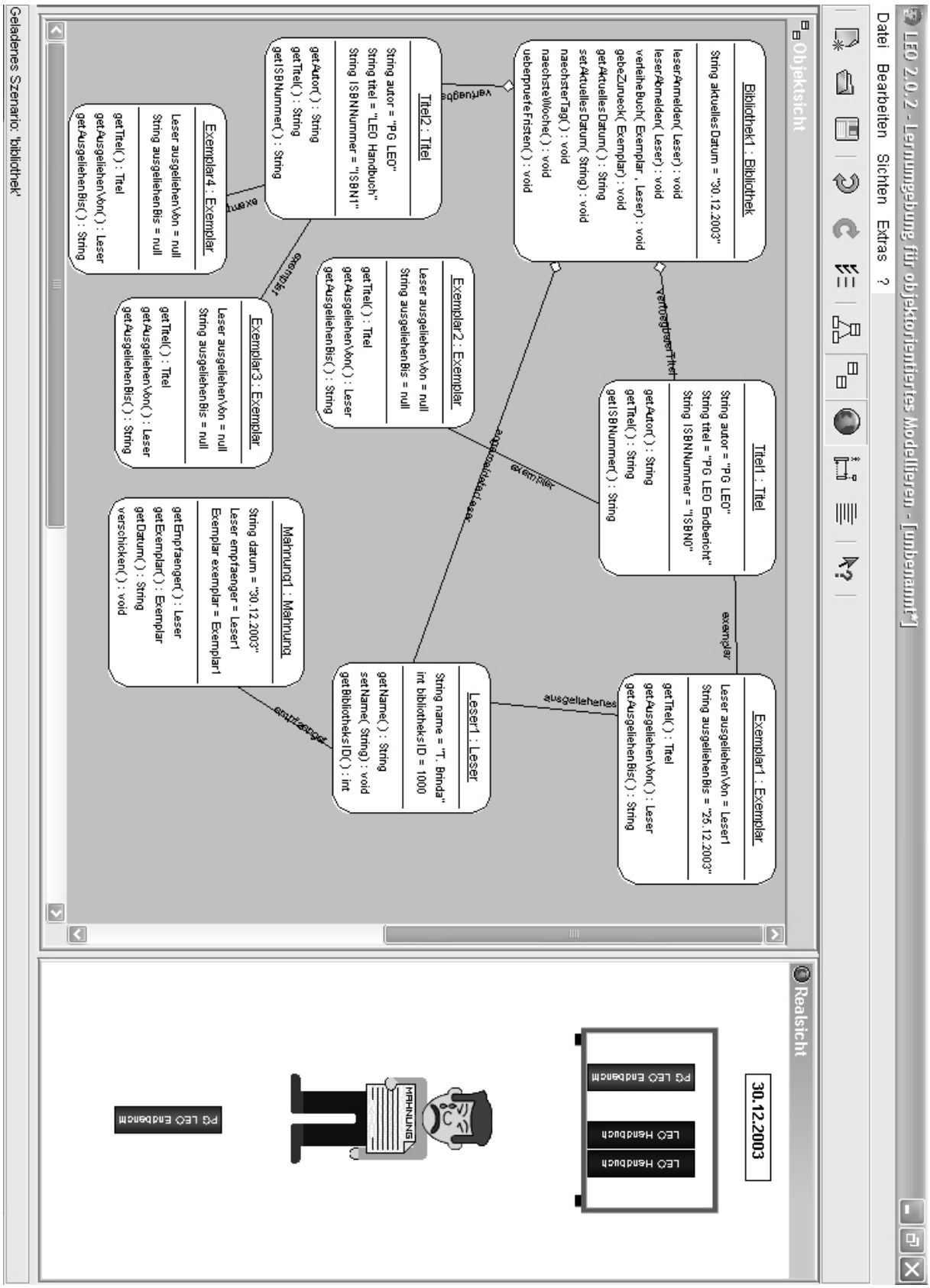


Abbildung 58: LEO – Objekt- und Realsicht im Szenario Bibliothek

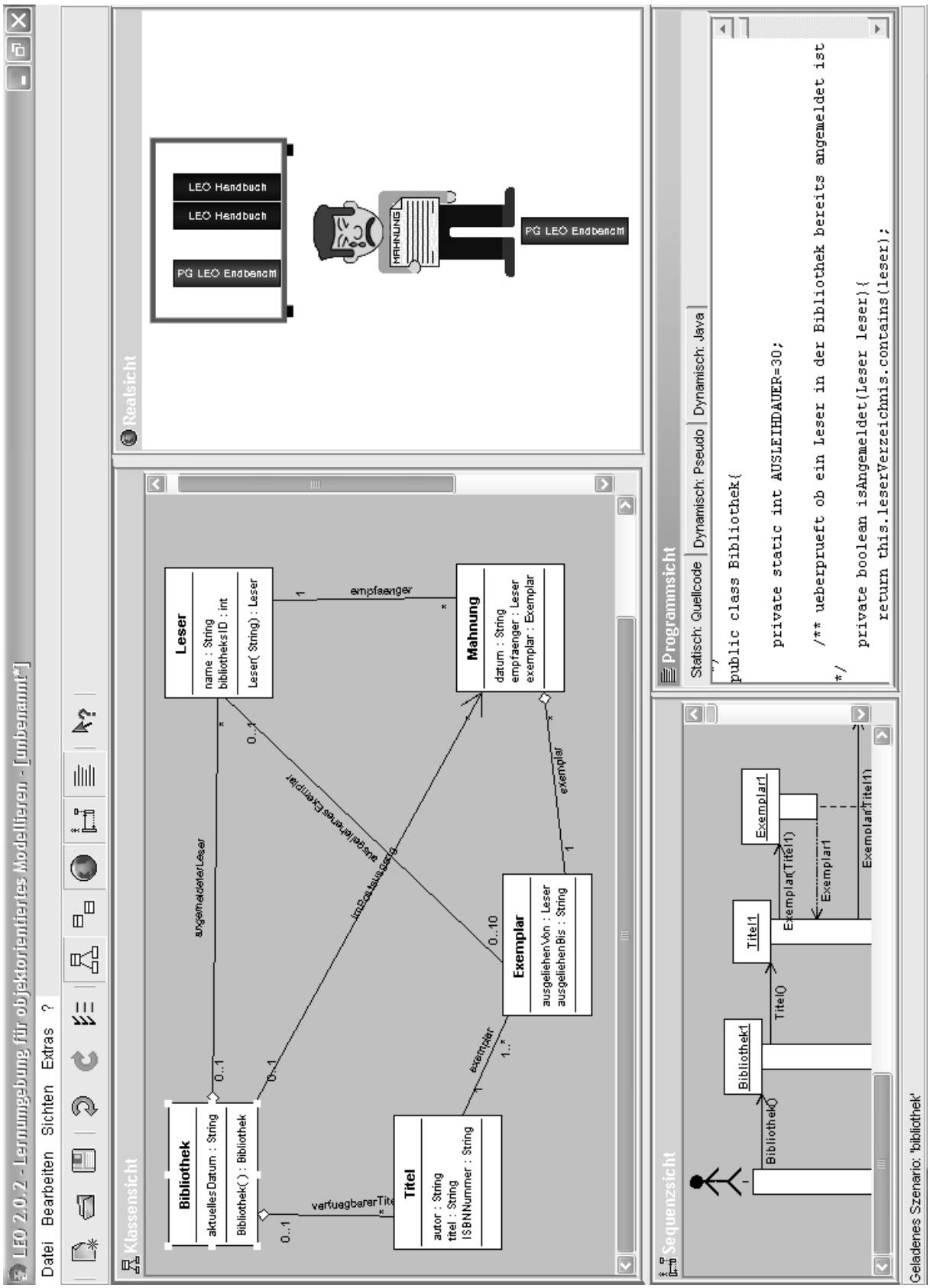


Abbildung 59: LEO – Sichtenwechsel im Szenario Bibliothek

Abkürzungsverzeichnis

ACM	*engl.* association for computing machinery
ADT	abstrakter Datentyp
CASE	*engl.* computer aided software engineering
CRC	*engl.* class, responsibilities, collaborators
CSCW	*engl.* computer supported co-operative work
EB	Explorationsbaustein
EDV	elektronische Datenverarbeitung
EGO	Explorer für geometrische Objekte
EM	Explorationsmodul, *engl.* exploration module
EVA	Eingabe – Verarbeitung – Ausgabe
GI	Gesellschaft für Informatik e.V.
GK	Grundkurs
GUI	*engl.* graphic user interface
HTML	*engl.* hypertext markup language
IML	Informatikmodule für die Lehrerbildung
INFOS	GI-Fachtagung „Informatik und Schule“
ITG	Informationstechnische Grundbildung
ITiCSE	*engl.* innovation and technology in computer science education
Jgst.	Jahrgangsstufe
LEO	Lernumgebung für objektorientiertes Modellieren
LK	Leistungskurs
MSWF	Ministerium für Schule, Wissenschaft und Forschung des Landes Nordrhein-Westfalen
MUE	Multimediale Evaluation in der Informatiklehrerausbildung
MVC	*engl.* model – view – controller
OHP	*engl.* overhead-projector
OOA	objektorientierte Analyse
OOD	*engl.* object-oriented design
OOE	objektorientierter Entwurf
OOM	objektorientiertes Modellieren
OOP	objektorientiertes Programmieren
Sek. I	Sekundarstufe I
Sek. II	Sekundarstufe II
SHK	studentische Hilfskraft
SIGCSE	*engl.* special interest group for computer science education
SoSe	Sommersemester
SUM	„Von Stiften und Mäusen“
UML	*engl*. unified modelling language
WiSe	Wintersemester

Abbildungsverzeichnis

Abbildung 1: Verknüpfung der Komponenten des didaktischen Systems mit der Wissensstruktur als Metamodell 55

Abbildung 2: Verknüpfung von Aufgabenklassen und Explorationsmodulen mit Wissensstrukturen 56

Abbildung 3: Beispiel einer Aufgabenklasse 69

Abbildung 4: Bildung von Varianten zu Aufgabenklassen 71

Abbildung 5: Bildung von Ergänzungen zu Aufgabenklassen 71

Abbildung 6: Komplexe Aufgabenklasse mit zusätzlichen Angaben 71

Abbildung 7: Klassifikationsstruktur für (elementare) Aufgabenklassen 77

Abbildung 8: Baumstruktur von Aufgabenklassen zum statischen Systemmodell 78

Abbildung 9: Beispiel für die Repräsentation der Vernetzung von Aufgabenklassen 79

Abbildung 10: Vorgehensweise bei der Entwicklung und Erprobung von Aufgabenklassen und Aufgaben zum OOM 107

Abbildung 11: Aktivitätszyklus beim Experimentieren (vgl. Steinkamp 1999, 42) 116

Abbildung 12: Architektur für Umgebungen für Informatik-Experimente (vgl. Steinkamp 1999, 47) 117

Abbildung 13: Arbeitsbereich für die Exploration der Kommunikation zwischen Netzwerkkomponenten, die an der Anforderung einer Webseite beteiligt sind 118

Abbildung 14: Kommunikationsmonitor 118

Abbildung 15: Exploration: Erkundung und Experiment 119

Abbildung 16: Professionelle Entwicklung von Software mit Werkzeugen 123

Abbildung 17: Erlernen von Fachkonzepten mit Explorationsmodulen 123

Abbildung 18: Allgemeines Sichtenkonzept für Software zur Exploration von informatischen Fachgegenständen 131

Abbildung 19: Architekturkonzept – OOA-Modell (nach Hoffmann 2003, 53) 147

Abbildung 20: Vorgehensweise bei der Entwicklung von Architektur und Fallstudien (vgl. Hoffmann 2003, 89) 148

Abbildung 21: Architekturkonzept – OOE-Modell (nach Hoffmann 2003, 64) 149

Abbildung 22: Vorgehensweise bei der Entwicklung und Erprobung von Explorationsmodulen zum OOM 169

Abbildung 23: Realitätsausschnitt – Objektdiagramm – Klassendiagramm 176

Abbildung 24: Lernhilfen zum Methodenentwurf 177

Abbildung 25: Implementierung der Referenzstruktur 183

Abbildung 26: Verknüpfung von Wissensstrukturen, Explorationsmodulen und Aufgabenklassen 185

Abbildung 27: Vorgehensweise zu Wissensstrukturen 190

Abbildung 28: Aufgabensammlung „Traktor“ – Musterlösung zum Klassendiagramm 217

Abbildung 29: Aufgabensammlung „Zug“ – Arbeitsmaterial 218

Abbildung 30: Aufgabensammlung „Zug“ – Musterlösung zum Klassendiagramm 220

Abbildung 31: Objektdiagramm für ein Polygon 226

Abbildung 32: Teil eines Klassendiagramms für ein Wettkampf-Bewertungssystem 227

Abbildung 33: Teilweise fertig gestelltes Klassendiagramm eines Flugreisesystems 228

Abbildung 34: Verwandtschaftsbeziehungen als Objektdiagramm 229

Abbildung 35: Objektdiagramm zu einem Dokument eines Mini-Grafikeditors 229

Abbildung 36: Klassendiagramm-Referenzmodell (vgl. Oestereich 1998, 52) 231
Abbildung 37: Sequenzdiagramm-Referenzmodell (vgl. Oestereich 1998, 55) 231
Abbildung 38: GUI der zu analysierenden Software 237
Abbildung 39: Darstellung des Nachrichtenaustauschs in der zu analysierenden Software 237
Abbildung 40: Puzzle Rechtecke – Aufgabenstellung 239
Abbildung 41: Puzzle Rechtecke – Lösungsversuch 240
Abbildung 42: Puzzle Ventilator – Aufgabenstellung 241
Abbildung 43: Puzzle Ventilator – korrekte Lösung 241
Abbildung 44: Puzzle Raumschiff – Aufgabenstellung 242
Abbildung 45: Puzzle Raumschiff – Arbeitsfläche zu Beginn 242
Abbildung 46: Puzzle Ventilator 2 – Aufgabenstellung 243
Abbildung 47: Puzzle Ventilator 2 – Arbeitsfläche zu Beginn 243
Abbildung 48: Experimentierumgebung Kreise – Aufgabenstellung 244
Abbildung 49: Experimentierumgebung Kreise – Arbeitsfläche 244
Abbildung 50: Experimentierumgebung „Springender Ball“ 245
Abbildung 51: Experimentierumgebung Kursverwaltung – Arbeitsfläche zu Beginn 246
Abbildung 52: Experimentierumgebung Kursverwaltung – Lösungsvariante mit Kommentar 246
Abbildung 53: EGO – Explorer für geometrische Objekte 247
Abbildung 54: EGO – Veränderung der Attributwerte in der Objektsicht nach Änderung in der Realsicht 247
Abbildung 55: EGO – Interaktion in der Objektsicht 248
Abbildung 56: EGO – Beobachtung der Objektkommunikation 249
Abbildung 57: LEO – Sichtenwechsel im Szenario Kreis 250
Abbildung 58: LEO – Objekt- und Realsicht im Szenario Bibliothek 251
Abbildung 59: LEO – Sichtenwechsel im Szenario Bibliothek 252

Tabellenverzeichnis

Tabelle 1: Vorgehensweise bei der Entwicklung des didaktischen Systems für OOM 5

Tabelle 2: Ergebnisse für die Informatiklehrerbildung / Didaktik der Informatik 7

Tabelle 3: Argumente für eine Einbeziehung der Objektorientierung in Lehr-Lern-Prozesse 23

Tabelle 4: Probleme und Schwierigkeiten einer Einbeziehung der Objektorientierung in Lehr-Lern-Prozesse 23

Tabelle 5: Imperativische vs. objektorientierte Programmierung nach Baumann (1996, 282) 25

Tabelle 6: Didaktische Funktionen der Komponenten des didaktischen Systems im Hinblick auf wesentliche Akteure in Lehr-Lern-Prozessen 46

Tabelle 7: Wechselnde Rollen von Studierenden in der Informatiklehrerausbildung 57

Tabelle 8: Verknüpfung der Aufgabentypen mit der Bloom'schen Lernzieltaxonomie 74

Tabelle 9: Verknüpfung von Gegenständen und Aufgabentypen 75

Tabelle 10: Beispiele für Aufgaben(-klassen) zur Verknüpfung von Gegenständen und Aufgabentypen 76

Tabelle 11: Bewertung von Kontextklassen 83

Tabelle 12: Geschlechterverteilung in den drei an der Studie beteiligten Lerngruppen 87

Tabelle 13: Leistungsbild der Lerngruppen vor der Durchführung der Fallstudien 87

Tabelle 14: Verknüpfung von Aufgabentexten und -klassen für den Kurs A-11 89

Tabelle 15: Verknüpfung von Aufgabentexten und -klassen für den Kurs B-11 89

Tabelle 16: Verknüpfung von Aufgabentexten und -klassen für den Kurs B-12 90

Tabelle 17: Überblick über die Beobachtungszeitpunkte 91

Tabelle 18: Befragungsergebnisse zur Einschätzung des Schwierigkeitsgrades der Aufgaben im Kurs A-11 96

Tabelle 19: Befragungsergebnisse zur Einschätzung des Schwierigkeitsgrades der Aufgaben im Kurs B-11 97

Tabelle 20: URLs der Studierendenlösungen zu Übung 3 (Didaktik der Informatik II) 99

Tabelle 21: Vorschläge für Unterrichtsmethoden 102

Tabelle 22: Vorschläge für Medien 102

Tabelle 23: Lehrerfortbildungen zum Aufgabenklassenkonzept 104

Tabelle 24: Informatikdidaktisch analysierte Werkzeuge zum OOM 123

Tabelle 25: Auswahl von UML-Diagrammtypen für Modellsichten 134

Tabelle 26: Kombinationsmöglichkeiten (Varianten V_i) von Klassen-, Objekt-, Kollaborations- und Sequenzdiagramm 134

Tabelle 27: Animationstyp „Puzzle“ – Realisierte Beispiele 137

Tabelle 28: Animationstyp „Experimentierumgebung“ – Realisierte Beispiele 139

Tabelle 29: Software zur Exploration und die Erfüllung der Anforderungen des Konzepts 146

Tabelle 30: Selbstständigkeitsstufen der Exploration 151

Tabelle 31: Explorationsstrategie zum OOM 152

Tabelle 32: Anwendung der Exploration in der informatischen Bildung 155

Tabelle 33: URLs der Studierendenlösungen zu Übung 8 und 9 (Didaktik der Informatik I) 161

Tabelle 34: Lösungen der Studierenden zu Blatt 8, Aufgabe 2, 1.Teil 161

Tabelle 35: Lösungen der Studierenden zu Blatt 8, Aufgabe 3 162

Tabelle 36: Lösungen der Studierenden zu Blatt 8, Aufgabe 4 163
Tabelle 37: Lösungen der Studierenden zu Blatt 9, Aufgabe 2 164
Tabelle 38: Teillösungen der Studierenden zu Blatt 9, Aufgabe 3 164
Tabelle 39: URLs der Studierendenlösungen zu Übung 4 (Didaktik der Informatik II) 165
Tabelle 40: Lösungen der Studierenden zu Blatt 4, Aufgabe 2 166
Tabelle 41: Lösungen der Studierenden zu den LEO-Analyseaufträgen 168
Tabelle 42: Ausgewählte Übungsaufgaben aus Rumbaugh et al. (1993) 204
Tabelle 43: Ausgewählte Übungsaufgaben aus Balzert (1999) 205
Tabelle 44: Befragungsergebnisse zum Informatikunterricht allgemein 222
Tabelle 45: Befragungsergebnisse zum Unterrichtsthema 223
Tabelle 46: Befragungsergebnisse zum geschätzten individuellen Arbeitsanteil bei der Gruppenarbeit 223
Tabelle 47: Befragungsergebnisse zur Einstellungen zum Modellieren und Programmieren 224
Tabelle 48: URLs von Explorationsmodulen zum OOM 239

Literaturverzeichnis

(Albert 2000) Albert, D.: The Theory of Knowledge Spaces. URL: http://wundt.uni-graz.at/knowledge/knowledge.htm, 2000 (aufgerufen am 09.11.03).

(Alex et al. 2002) Alex, M.; Azem, A.; Fricke, T.; Iscan, H.; Rohr, O.; Rosskopf, M.; Söchtig, G.; Stachnik, A.; Vukusic, I.; Werner, T.; Wojciechowski, A.: Lernumgebung für objektorientiertes Modellieren im Informatikunterricht - LEO. Endbericht der Projektgruppe 403 des Fachbereichs Informatik, Universität Dortmund, 2002. URL: http://www.didaktik-der-informatik.de/DIE_BIB/Projekte/LEO/LEO_Projektbericht.pdf (aufgerufen am 09.11.03)

(Alexander et al. 1977) Alexander, C.; Ishikawa, S.; Silverstein, M.; Jacobson, M.; Fiksdahl-King, I.; Angel, S.: A pattern language. Oxford University Press, New York, 1977.

(Anderson 2001) Anderson, J. R.: Kognitive Psychologie. Spektrum, Berlin, 2001.

(Andrianoff / Levine 2002) Andrianoff, S. K.; Levine, D. B.: Role playing in an object-oriented world. In: Proceedings of the 33rd SIGCSE technical symposium on Computer science education, ACM Press, New York, 2001, pp. 121-125.

(Appelrath et al. 1998) Appelrath, H.-J.; Boles, D.; Claus, V.; Wegener, I.: Starthilfe Informatik. B.G. Teubner, Stuttgart, 1998.

(Balzert 1999) Balzert, H.: Lehrbuch der Objektmodellierung. Spektrum, Heidelberg, 1999.

(Baumann 1990) Baumann, R.: Objektorientiertes Programmieren und abstrakte Datentypen. In: LOG IN 10 (1990) 1, S. 22-33.

(Baumann et al. 1990) Baumann, R.; Rode, H.; Schulz-Zander, R.: Objektorientiertes Programmieren. In: LOG IN 10 (1990) 1, S. 3.

(Baumann 1996) Baumann, R.: Didaktik der Informatik. Klett, Stuttgart, 1996.

(Baumann 2000) Baumann, R.: JAVA im Anfangsunterricht. In: LOG IN 20 (2000) 1, S. 47-54.

(Baumann 2001a) Baumann, R.: Assoziieren und Spezialisieren (Teil 1). In: LOG IN 21 (2001) 2, S. 10-17.

(Baumann 2001b) Baumann, R.: Assoziieren und Spezialisieren (Teil 2). In: LOG IN 21 (2001) 3/4, S. 66-72.

(Beck / Cunningham 1989) Beck, K.; Cunningham, H.: A laboratory for teaching object-oriented thinking. In: Proceedings of OOPSLA 1989, SIGPLAN notices (ACM) vol. 24, New Orleans, 10/1989. URL: http://c2.com/doc/oopsla89/paper.html (aufgerufen am 14.11.03)

(Bellin / Simone 1997) Bellin, D.; Simone, S. S.: The CRC Card Book. Addison Wesley, 1997.

(Bernd et al. 2000) Bernd, H.; Hippchen, T.; Jüngst, K.-L.; Strittmatter, P.: Durcharbeiten von Begriffsstrukturdarstellungen in unterrichtlichen und computergestützten Lernumgebungen. In: (Mandl / Fischer 2000a), S. 15-36.

(Bloom 1976) Bloom, B. S.: Taxonomie von Lernzielen im kognitiven Bereich. Beltz, Weinheim, 1976.

(Bode et al. 2003) Bode, A.; Desel, J.; Rathmeyer, S.; Wessner, M. (Hrsg.): DeLFI 2003: Die 1. e-Learning Fachtagung Informatik. GI-Edition Lecture Notes in Informatics, P-37, Köllen, Bonn, 2003.

(Böszörményi 1998) Böszörményi, L.: Why Java is not my favorite first-course language. In: Software – Concepts & Tools (1998) 19, pp. 141-145.

(Booch 1994) Booch, G.: Object-Oriented Analysis and Design. Benjamin Cummings, Redwood City, 1994.

(Booch et al 1998) Booch, G.; Rumbaugh, J.; Jacobson, I.: The Unified Modeling Language User Guide. Addison-Wesley, Reading, Massachusetts, 1998.

(Borg / Wiehenstroth 1999) Borg, B.; Wiehenstroth, F.-O.: Landesschulversuch: Einjährige Berufsfachschule - Informatik - für Realschulabsolventen/-innen - Aspekte einer beruflich-informatischen Bildung. In: (Schwill 1999), S. 202-218.

(Box / Whitelaw 2000) Box, R.; Whitelaw, M.: Experiences when migrating from structured analysis to object-oriented modelling. In: Proceedings of the Australasian computing education conference (Melbourne, Australia, 2000), ACM Press, New York, 2000, pp. 12-18.

(Brauer / Brauer 1973) Brauer, W.; Brauer, U.: Informatik an der Schule – weshalb und wie? In: Rechnerkunde. Algorithmen und DVA-Strukturen im Schulunterricht. 4. Paderborner Werkstattgespräch 1972. Forschungs- und Entwicklungszentrum für objektive Lehr- und Lernverfahren. Schöning, Paderborn, 1973, S. 34-35.

(Brauer et al. 1976) Brauer, W., Claus, V., Deussen, R., Eickel, J., Haacke, W., Hosseus, W., Koster, C. H. A., Ollesky, D., Weinhart, K.: Zielsetzungen und Inhalte des Informatikunterrichts. In: Zentralblatt für Didaktik der Mathematik – ZDM 8 (1976) 1, S. 35-43.

(Brehm et al. 2003) Brehm, J.; Brancovici, G.; Müller-Schloer, C.; Smaoui, T.; Voigt, S.: Experimental Tools for a Multimedia-Supported Interactive Lecture. In: (Bode et al. 2003), S. 85-94.

(Breier et al. 2000a) Breier, N.; et al.: Gesamtkonzept der informatischen Bildung (Arbeitsstand). 7. Fachdidaktisches Kolloquium, Königstein, 08.-10.03.2000.URL: http://koenigstein.inf.tu-dresden.de/00/breier.html (aufgerufen am 23.11.03)

(Breier et al. 2000b) Breier, N. (federführend); Fothe, M.; Friedrich, S.; Hubwieser, P.; Koerber, B.; Röhner, G.; Schubert, S.; Seiffert, M.: Empfehlungen für ein Gesamtkonzept zur informatischen Bildung an allgemein bildenden Schulen. Beilage zu LOG IN 20 (2000) 2, S. I-VII.

(Brinda 2000a) Brinda, T.: Didaktische Systeme für objektorientiertes Modellieren (OOM) im Informatikunterricht. In: Gesellschaft für Informatik e.V. (Hrsg.): Informatiktage 2000. Konradin, Leinfelden-Echterdingen, 2000, S. 282-285.

(Brinda 2000b) Brinda, T.: Objektorientiertes Modellieren – Sammlung und Strukturierung von Übungsaufgaben im Informatikunterricht. In: LOG IN 20 (2000) 5, S. 39-49.

(Brinda 2001) Brinda, T.: Einfluss fachwissenschaftlicher Erkenntnisse zum objektorientierten Modellieren auf die Gestaltung von Konzepten in der Didaktik der Informatik. In: (Keil-Slawik / Magenheim), S. 75-86.

(Brinda / Schubert 2001) Brinda, T.; Schubert, S.: Didaktisches System für objektorientiertes Modellieren. Forschungsbericht Nr. 752, Fachbereich Informatik, Universität Dortmund, 2001. URL: http://www.die.informatik.uni-siegen.de/DIE_BIB/Berichte/Forschungsberichte/forschungsbericht-752.zip (aufgerufen am 23.11.03)

(Brinda 2002) Brinda, T.: Entwicklung von Komponenten eines Lehr-Lern-Konzeptes zum objektorientierten Modellieren. In: Informatica Didactica 5, URL: http://www.informatica-didactica.de/InformaticaDidactica/Brinda2002 (aufgerufen am 29.10.03).

(Brinda / Schubert 2002a) Brinda, T.; Schubert, S. E.: Didactic System for Object-oriented Modelling. In: Watson, D.; Andersen, J. (eds.): Networking the Learner. Computers in Education. Kluwer Academic Publishers, Boston, 2002, pp. 473-482.

(Brinda / Schubert 2002b) Brinda, T.; Schubert, S. E.: Learning aids and learners' activities in the field of object-oriented modelling. In: Passey, D.; Kendall, M. (eds.): TelE-Learning. The Challenge for the Third Millennium. Kluwer Academic Publishers, Boston, 2002, pp. 37-44.

(Brinda / Ortmann 2002) Brinda, T.; Ortmann, T.: Fallstudien zur unterrichtlichen Einbettung spezieller Aufgabenklassen. In: (Schubert et al. 2002), S. 13-22.

(Brinda 2003) Brinda, T.: Student experiments in object-oriented modelling. In: Cassel, L.; Reis, R. A. (eds.): Informatics curricula and teaching methods. Kluwer Academic Publishers, Boston, 2003, pp. 13-20.

(Brinda / Schubert 2003) Brinda, T.; Schubert, S.: Exploration of object-oriented models in informatics education. In: (van Weert / Munroe 2003), pp. 109-118.

(Brinda 2004) Brinda, T.: Integration of new exercise classes into the Informatics education in the field of object-oriented modelling. Eingeladener Beitrag zur Zeitschrift "Education and information technologies", Kluwer Academic Publishers, 2004.

(Broy / Siedersleben 2002) Broy, M.; Siedersleben, J.: Objektorientierte Programmierung und Softwareentwicklung. Eine kritische Einschätzung. In: Informatik Spektrum 25 (2002) 1, 3-11.

(Bruce et al. 2001) Bruce, K. B.; Danyluk, A.; Murtagh, T.: A library to support a graphics-based object-first approach to CS1. In: Proceedings of the 32nd SIGCSE technical symposium on Computer science education. ACM Press, New York, 2001, pp. 6-10.

(Bruhn et al. 2000) Bruhn, J.; Fischer, F.; Gräsel, C.; Mandl, H.: Kooperatives Lernen mit Mapping-Techniken. In: (Mandl / Fischer 2000a), S. 119-133.

(Bruner 1960) Bruner, J. S.: The process of education. Harvard University Press, Cambridge MA., 1960.

(Bunge 1967) Bunge, M. A.: Scientific Research, Vol. II. Springer, Berlin, 1967.

(Carroll 1982) Carroll, J. M.: The adventure of getting to know a computer. In: IEEE Computer 15 (1982) 11, pp. 49-58.

(Clark 1992) Clark, R. E..: Media use in education. In: Alkin, M. C. (ed.): Encyclopedia of educational research, Macmillan, New York, 1992, pp. 805-814.

(Claus / Schwill 2001) Meyers Lexikonredaktion (Hrsg.); Claus, V.; Schwill, A. (Bearb.): Duden Informatik. Dudenverlag, Mannheim, 2001.

(Coad / Yourdon 1991) Coad, P.; Yourdon, E.: Object-oriented Analysis. Prentice-Hall, Inc., 1991.

(Crutzen / Hein 1995) Crutzen, C. K. M.; Hein, H.-W.: Objektorientiertes Denken als didaktische Basis der Informatik. In: (Schubert 1995), S. 149-158.

(Czischke 1995a) Czischke, J.: Von Stiften und Mäusen. In: Informatik betrifft uns (1995) 2, S. 24-43.

(Czischke 1995b) Czischke, J.: Von Buntstiften und Prototypen. In: Informatik betrifft uns (1995) 3, S. 28-50.

(Czischke 1996) Czischke, J.: Von Ereignissen und Pentominos. In: Informatik betrifft uns (1996) 1, S. 22-45.

(Czischke 1997) Czischke, J.: OOP von Anfang an. In: Informatik betrifft uns (1997) 1, 16-35.

(Czischke 1998a) Czischke, J.: Verkettete Objekte. In: Informatik betrifft uns (1998) 1, S. 21-46.

(Czischke 1998b) Czischke, J.: Ereignisbearbeiter. In: Informatik betrifft uns (1998) 3, S. 1-31.

(Czischke 2000) Czischke, J.: Geklammerte Objekte. In: Informatik betrifft uns (2000) 2, S. 26-46.

(Dahl / Nygaard 1966) Dahl, O.-J.; Nygaard, K.: SIMULA - an Algol-based simulation language. In: Communications of the ACM, Vol. 9, No. 9, 1966, S. 671-678.

(Della / Clark 2000) Della, L.; Clark, D.: Teaching object-oriented development with emphasis on pattern application. In: Proceedings of the Australasian computing education conference. ACM Press, New York, 2000, pp. 56-63.

(Dershem / Vanderhyde 1998) Dershem, H. L.; Vanderhyde, J.: Java class visualization for teaching object-oriented concepts. In: (Joyce / Impagliazzo 1998), pp. 53-57.

(Diehl / Kerren 2001) Diehl, S.; Kerren, A.: Levels of exploration. In: Proceedings of the 32nd SIGCSE technical symposium on Computer science education, ACM Press, New York, 2001, pp. 60-64.

(Diethelm et al. 2002) Diethelm, I.; Geiger, L.; Zündorf, A.: UML im Unterricht: Systematische objektorientierte Problemlösung mit Hilfe von Szenarien am Beispiel der Türme von Hanoi. In: (Schubert et al. 2002), S. 33-42.

(Diethelm et al. 2003) Diethelm, I.; Geiger, L.; Zündorf, A.: Fujaba goes Mindstorms: Objektorientierte Modellierung zum Anfassen. In: (Hubwieser 2003), S. 225-235.

(Dietzel / Rinkens 2001) Dietzel, R.; Rinkens, T.: Eine Einführung in die Objektorientierung mit Lego Mindstorms Robotern. Erfahrungsbericht aus dem Unterricht. In: (Keil-Slawik / Magenheim 2001), S. 193-200.

(Drosdowski et al. 1990) Drosdowski, G.; Müller, W.; Scholze-Stubenrecht, W.; Wermke, M. (Hrsg.): Duden Fremdwörterbuch. Dudenverlag, Mannheim, 1990.

(Duncker 1966) Duncker, K.: Zur Psychologie des produktiven Denkens. Springer, Berlin, 1966.

(Eberle 1996) Eberle, F.: Didaktik der Informatik bzw. einer informations- und kommunikationstechnischen Bildung auf der Sekundarstufe II. Sauerländer, Aarau, 1996.

(Eirund 1993) Eirund, H.: Objektorientierte Programmierung. In: LOG IN 13 (1993) 4, S. 40-48.

(Eurydice 2001) Eurydice (ed.): Basic Indicators on the Incorporation of ICT into European Education Systems. Facts and figures 2000/01. Annual Report. Brussels, 2001.
URL: http://www.eurydice.org/Doc_intermediaires/analysis/de/information_technology.html (aufgerufen am 23.11.03)

(Exton / Kölling 2000) Exton, C.; Kölling, M.: Concurrency, objects and visualisation. In: Proceedings of the Australasian computing education conference. ACM Press, New York, 2000, pp. 109-115.

(Fischer / Mandl 2000) Fischer, F.; Mandl, H.: Strategiemodellierung mit Expertenmaps. In: (Mandl / Fischer 2000a), S. 37-54.

(Fowler / Scott 1997) Fowler, M.; Scott, K.: UML distilled. Applying the standard object modeling language. Addison Wesley Longman, Inc., 1997.

(Freiberger 2002) Freiberger, U.: Karel - Eine Übersicht über verschiedene Entwicklungen, die auf der Idee von „Karel, the Robot“ basieren, 2002. URL: http://www.schule.bayern.de/karol/data/uebersicht.pdf (aufgerufen am 19.11.03)

(Freudenreich / Schulte 2002) Freudenreich, M.; Schulte, C.: Von der Evaluation von Lernsoftware zur Gestaltung von Unterricht. In: MedienPädagogik. Online-Zeitschrift für Theorie und Praxis der Medienbildung, 18.3.2002. URL: http://www.medienpaed.com/02-1/freudenreich_schulte1.pdf (aufgerufen am 23.11.03)

(Frey et al. 2001) Frey, E.; Hubwieser, P.; Humbert, L.; Schubert, S.; Voß, S.: Informatik-Anfangsunterricht. In: LOG IN 21 (2001) 1, S. 20-32.

(Friedrich 1995) Friedrich, S.: Grundpositionen eines Schulfaches. In: LOG IN 15 (1995) 5/6, S. 30-34.

(Friedrich 2003) Friedrich, S.: Informatik und PISA – vom Wehe zum Wohl der Schulinformatik. In: (Hubwieser 2003), S. 133-144.

(Füller 1999) Füller, K.: Objektorientiertes Programmieren in der Schulpraxis. In: (Schwill 1999), S. 190-201.

(Gamma et al. 1996) Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.: Entwurfsmuster. Addison-Wesley, Bonn, 1996.

(GIFA 1999) Gesellschaft für Informatik e.V., Fachausschuss 7.3 „Informatische Bildung in Schulen“: Gesamtkonzept der informatischen Bildung (Arbeitsstand). In: LOG IN 19 (1999) 3/4, S. 29.

(Goldberg / Kay 1976) Goldberg, A.; Kay, A. (Hrsg.): Smalltalk-72 Instruction Manual. Technical Report SSL-76-6, Xerox PARC, Palo Alto, 1976.

(Gorny 1991) Gorny, P. (Hrsg.): Informatik und Schule 1991. Informatik: Wege zur Vielfalt beim Lehren und Lernen. Springer, Berlin, 1991.

(Hadjerrouit 1999) Hadjerrouit, S.: A constructivist approach to object-oriented design and programming. In: Proceedings of the 4th annual SIGCSE / SIGCUE ITiCSE conference on Innovation and technology in computer science education (Cracow, Poland, 1999), ACM Press, New York, 1999.

(Häußler et al. 1998) Häußler, P.; Bünder, W.; Duit, R.; Gräber, W.; Mayer, J.: Perspektiven für die Unterrichtspraxis. Gehl, Kiel, 1998.

(Haller 2000) Haller, S. (ed.): Proceedings of the 31st SIGCSE technical symposium on Computer science education. ACM Press, New York, 2000.

(Hameyer 2002) Hameyer, U.: Entdeckendes Lernen – ein Prozess explorativen Verstehens. In: Hameyer, U.; Schlichting, F. (Hrsg.): Entdeckendes Lernen. Körner, Kramshagen, 2002, S. 5-9.

(Hampel et al. 1999) Hampel, T.; Magenheim, J.; Schulte, C.: Dekonstruktion von Informatiksystemen als Unterrichtsmethode – Zugang zu objektorientierten Sichtweisen im Informatikunterricht. In: (Schwill 1999), S. 149-164.

(Hermes 1996a) Hermes, A.: OOP im Unterricht. Ein Plädoyer für einen gleitenden Paradigmenwechsel (Teil 1). In: LOG IN 16 (1996) 4, S. 29-33.

(Hermes 1996b) Hermes, A.: OOP im Unterricht. Ein Plädoyer für einen gleitenden Paradigmenwechsel (Teil 2). In: LOG IN 16 (1996) 5/6, S. 62-67.

(Herrmann 1986) Herrmann, T.: Zur Gestaltung der Mensch-Computer-Interaktion: Systemerklärung als kommunikatives Problem. Max Niemayer, Tübingen, 1986.

(HHBBS 2003) Freie und Hansestadt Hamburg, Behörde für Bildung und Sport (Hrsg.): Rahmenplan Informatik. Bildungsplan Gymnasiale Oberstufe. Entwurfsfassung vom 03.09.03.

(Hillen et al. 2000) Hillen, S.; Berendes, K.; Breuer, K.: Systemdynamische Modellbildung als Werkzeug zur Visualisierung, Modellierung und Diagnose von Wissensstrukturen. In: (Mandl / Fischer 2000a), S. 71-89.

(Hirsch et al. 2000) Hirsch, M.; Magenheim, J.; Reinsch, T.: Zugänge zur Informatik mit Mindstorms. In: LOG IN 20 (2000) 2, S. 34-46.

(HK 2003) Hessisches Kultusministerium (Hrsg.): Lehrplan Informatik. Gymnasialer Bildungsgang. Jahrgangsstufe 11 bis 13, 2003. URL: http://www.hessisches-kultusministerium.de/downloads/lehrpl/gymnasium/Informatik.pdf (aufgerufen am 29.10.03)

(Hoffmann 2003) Hoffmann, A.: Theoretisch begründete Vorgehensweise bei der Entwicklung von Software zur Exploration im Bildungskontext. Diplomarbeit, Fachbereich Informatik und Elektrotechnik, Universität Siegen, November 2003.

(Holland et al. 1997) Holland, S.; Griffiths, R.; Woodman, M.: Avoiding object misconceptions. In: Proceedings of the 28th SIGCSE technical symposium on computer science education (San Jose, California, United States, 1997), ACM Press, New York, 1997, pp. 131-134.

(Hoppe / Luther 1997) Hoppe, H. U.; Luther, W.: Informatik und Lernen in der Informationsgesellschaft. Springer, Berlin, 1997.

(Hubwieser / Broy 1996) Hubwieser, P.; Broy, M.: Der informationszentrierte Ansatz. Ein Vorschlag für eine zeitgemäße Form des Informatikunterrichts am Gymnasium. Institut für Informatik der Technischen Universität München, Technischer Bericht TUM-I9624, 1996.

(Hubwieser / Broy 1997) Hubwieser, P.; Broy, M.: Grundlegende Konzepte von Informations- und Kommunikationssystemen für den Informatikunterricht. In: (Hoppe / Luther 1997), S. 40-50.

(Hubwieser et al. 1997) Hubwieser, P.; Broy, M.; Brauer, W.: A new approach to teaching information technologies: shifting emphasis from technology to information. In: (Passey / Samways 1997), pp. 115-121.

(Hubwieser 1999a) Hubwieser, P.: Informatik als Pflichtfach an bayerischen Gymnasien. In: (Schwill 1999), S. 165-174.

(Hubwieser 1999b) Hubwieser, P.: Modellierung in der Schulinformatik. In: LOG IN 19 (1999) 1, S. 24-29.

(Hubwieser / Broy 1999) Hubwieser, P., Broy, M.: Educating Surfers or Craftsmen: Introducing an ICT Curriculum for the 21st Century. In: (Watson / Downes 1999), pp. 162-170.

(Hubwieser 2000a) Hubwieser, P.: Didaktik der Informatik. Springer, Berlin, 2000.

(Hubwieser 2000b) Hubwieser, P.: Informatik am Gymnasium. Ein Gesamtkonzept für einen zeitgemäßen Informatikunterricht. Habilitationsschrift, Fakultät für Informatik, Technische Universität München, 2000.

(Hubwieser et al. 2001) Hubwieser, P.; Humbert, L.; Schubert, S.: Evaluation von Informatikunterricht. In: (Keil-Slawik / Magenheim 2001), S. 213-216.

(Hubwieser 2003) Hubwieser, P. (Hrsg.): Informatische Fachkonzepte im Unterricht. Köllen, Bonn, 2003

(Humbert 1999) Humbert, L.: Grundkonzepte der Informatik und ihre Umsetzung im Informatikunterricht. In: (Schwill 1999), S. 175-189.

(Humbert 2001a) Humbert, L.: Informatik lehren – zeitgemäße Ansätze zur nachhaltigen Qualifikation aller Schülerinnen. In: (Keil-Slawik / Magenheim 2001), S. 121-132.

(Humbert 2001b) Humbert, L.: Interviews mit Informatik-Lehrkräften. In: LOG IN 21 (2001) 3/4, S. 73-75.

(Humbert 2003) Humbert, L.: Zur wissenschaftlichen Fundierung der Schulinformatik. Dissertationsschrift, Fachbereich Elektrotechnik und Informatik, Universität Siegen, 2003.

(Husch 1993) Husch, B.: Neue Methoden der Software-Entwicklung im Unterricht. In: (Troitzsch 1993), S. 101-107.

(IEEE 2002) IEEE Standards Department (ed.): Draft Standard for Learning Object Metadata, 2002. URL: http://grouper.ieee.org/LTSC/wg12/20020612-Final-LOM-Draft.html (aufgerufen am 09.11.03)

(IML 2001) Fachgruppe "Didaktik der Informatik und E-Learning" der Universität Siegen (Hrsg.): Informatikmodule für die Lehrerbildung – IML. Von der Bertelsmann-Stiftung gefördertes Drittmittelprojekt, Laufzeit: 01.01.-31.12.2001, URL: http://www.didaktik-der-informatik.de/iml/ (aufgerufen am 09.11.03)

(ISO 9241) International Organization for Standardization (ed.): Ergonomic requirements for office work with visual display terminals (VDTs). EN ISO 9241-1, 1997.

(Jacobson et al. 1992) Jacobson, I.; Christerson, M.; Jonsson, P.; Övergaard, G.: Object-Oriented Software Engineering. A Use Case Driven Approach. Addison-Wesley Longman, New York, 1992.

(Janetzko / Strube 2000) Janetzko, D.; Strube, G.: Knowledge Tracking – Eine neue Methode zur Diagnose von Wissensstrukturen. In: (Mandl / Fischer 2000a), S. 199-217.

(Joyce / Impagliazzo 1998) Joyce, D.; Impagliazzo, J. (eds.): Proceedings of the 29th SIGCSE technical symposium on Computer science education. ACM Press, New York, 1998.

(Keil-Slawik / Magenheim 2001) Keil-Slawik, R.; Magenheim, J. (Hrsg.): Informatikunterricht und Medienbildung. Köllen, Bonn, 2001.

(Keil-Slawik 2001) Keil-Slawik, R.: Explorationen für die Mechanikausbildung, 2001. URL: http://iug.uni-paderborn.de/iug/projekte/explorationen/ (aufgerufen am 29.10.03)

(Kerres 2001) Kerres, M.: Multimediale und telemediale Lernumgebungen. Oldenbourg, München, 2001.

(Knapp / Fischer 1998) Knapp, T.; Fischer, H.: Objektorientierung im Informatikunterricht. Ein didaktisches Konzept zum Einstieg in den Informatikunterricht der Sekundarstufe I. In: LOG IN 18 (1998) 3/4, S. 71-76.

(Koch / Ortmann 2001) Koch, L.; Ortmann, T.: Werkzeuge für objektorientiertes Modellieren im Informatikunterricht. Projektarbeit, Fachbereich Informatik, Universität Dortmund, 2001. URL: http://www.didaktik-der-informatik.de/DIE_BIB/Arbeiten/Projektarbeiten/pa-koch-ortmann-2001.pdf (aufgerufen am 23.10.03)

(Kölling et al. 1995) Kölling, M.; Koch, B.; Rosenberg, J.: Requirements for a first year object-oriented teaching language. In: Papers of the 26th SIGCSE technical symposium on Computer science education, ACM Press, New York, 1995, pp. 173-177.

(Kölling / Rosenberg 1996) Kölling, M.; Rosenberg, J.: An object-oriented program development environment for the first programming course. In: Proceedings of the 27th SIGCSE technical symposium on computer science education. ACM Press, New York, 1996, pp. 83-87.

(Kölling / Rosenberg 2000) Kölling, M.; Rosenberg, J.: Objects first with Java and BlueJ. In: (Haller 2000), p. 429.

(Kölling / Rosenberg 2001) Kölling, M.; Rosenberg, J.: Guidelines for teaching object-orientation with Java. In: Proceedings of the 6th annual conference on Innovation and technology in computer science education. ACM Press, New York, 2001, pp. 33-36.

(Koerber / Peters 1989) Koerber, B.; Peters, I.-R.: Software-Bausteine im Unterricht. In: LOG IN 9 (1989) 6, S. 28-36.

(Koffmann et al. 2003) Koffmann, E. (chair); Brinda, T. (rapporteur); Alvarez, J.; Kumar, A.; Lisboa, M. L. B.; Reinfelds, J.; van Roy, P.; Wazlawick, R. S.: Teaching Programming and Problem Solving: The Current State (Working group report). In: Cassel, L.; Reis, R. A. (eds.): Informatics curricula and teaching methods. Kluwer Academic Publishers, Boston, 2003, pp. 125-130.

(KSA 2003) Kultusministerium des Landes Sachsen-Anhalt (Hrsg.): Rahmenrichtlinien Gymnasium Informatik, Wahlpflichtfach: Schuljahrgänge 10 – 12, 2003.

(Lewis 2000) Lewis, J.: Myths about object-orientation and its pedagogy. In: (Haller 2000), pp. 245-249.

(LSW 1999) Landesinstitut für Schule und Weiterbildung (Hrsg.): Von Stiften und Mäusen. Verlag für Schule und Weiterbildung, Bönen, 1999.

(Magenheim / Schubert 2000) Magenheim, J.; Schubert, S.: Evaluation of teacher education in informatics. In: Benzie, D.; Passey, D. (eds.): Proceedings of the Conference on Educational Uses of Information and Communication Technologies, 16th World Computer Congress, China, Beijing, August 21st-25th, 2000, pp. 181-184.

(Magenheim 2001) Magenheim, J.: Deconstruction of Socio-technical Information Systems with Virtual Exploration Environments as a Method of Teaching Informatics. Proceedings of ED-MEDIA 2001, World conference on Educational Multimedia, Hypermedia & Telecommunications, Tampere, Finland, June 25th -30th, 2001.

(Mandl / Fischer 2000a) Mandl, H.; Fischer, F. (Hrsg.): Wissen sichtbar machen. Wissensmanagement mit Mapping-Techniken. Hogrefe, Göttingen, 2000.

(Mandl / Fischer 2000b) Mandl, H.; Fischer, F.: Mapping-Techniken und Begriffsnetze in Lern- und Kooperationsprozessen. In: (Mandl / Fischer 2000a), S. 3-12.

(Manns et al. 2000) Manns, M. L.; Sharp, H.; McLaughlin, P.; Prieto, M.: The Pedagogical Patterns Project. URL: http://sol.info.unlp.edu.ar/ppp/ (aufgerufen am 29.10.03).

(Matthäus / Schleiff 1991) Matthäus, W.-G.; Schleiff, M.: Das Spannungsfeld des Informatikdozenten. In: (Gorny 1991), S. 50-54.

(MBWFK 2002) Ministerium für Bildung, Wissenschaft, Forschung und Kultur des Landes Schleswig-Holstein (Hrsg.): Informatik. Lehrplan für die Sekundarstufe II. Gymnasium, Gesamtschule, Fachgymnasium. Kiel, 2002.

(MBWK 2001) Ministerium für Bildung, Wissenschaft und Kultur des Landes Mecklenburg-Vorpommern (Hrsg.): Rahmenplan Gymnasiale Oberstufe - Informatik - Jahrgangsstufen 11 bis 13. Erprobungsfassung, 2001.

(McCracken 1999) McCracken, D. D.: An inductive approach to teaching object-oriented design. In: Prey, J.; Noonan, B. (eds.): The Proceedings of the 13th SIGCSE technical symposium on Computer science education (New Orleans, Louisiana, United States, 1999), ACM Press, New York, 1999, pp. 184-188.

(Meyer 1987) Meyer, H.: Unterrichtsmethoden. II: Praxisband. Cornelsen Verlag Scriptor, Frankfurt a. M., 1987.

(Meyer 1994) Meyer, H.: Unterrichtsmethoden. I: Theorieband. 6. Auflage. Cornelsen Verlag Scriptor, Frankfurt a. M., 1994.

(Meyer 1997) Meyer, B.: Object-oriented software construction. Prentice-Hall, New Jersey, 1997.

(Mielke / Sagorny 2003) Mielke, K.; Sagorny, R.: Programmentwicklung mit UML. Bildungsverlag EINS, Troisdorf, 2003.

(Mitchell 2000) Mitchell, W.: A paradigm shift to OOP has occurred ... implementation to follow. In: Proceedings of the 14th annual consortium on Small Colleges South-eastern conference (Salem, Virginia, United States, 2000), The Consortium for Computing in Small Colleges, 2000, pp. 94-105.

(MKJS 2001) Ministerium für Kultus, Jugend und Sport des Landes Baden-Württemberg (Hrsg.): Bildungsplan für die Kursstufe des allgemein bildenden Gymnasiums. Amtsblatt des Ministeriums für Kultus, Jugend und Sport – Baden-Württemberg, Lehrplanheft 3 / 2001, Neckar, Villingen-Schwenningen, 2001.

(Modrow 1992) Modrow, E.: Zur Didaktik des Informatik-Unterrichts. Dümmler, Bonn, 1992.

(Modrow 1995) Modrow, E.: Physikobjekte im Informatikunterricht. In: Informatik betrifft uns (1995) 1, S. 1-24.

(Modrow 1996) Modrow, E.: Zur OOP: Von Bällen, Billardtischen und den Grenzen der Simulation. In: Informatik betrifft uns (1996) 1, S. 1-24.

(Modrow 2003) Modrow, E.: Pragmatischer Konstruktivismus und Fundamentale Ideen als Leitlinien der Curriculumsentwicklung. Dissertation, Mathematisch-Naturwissenschaftlich-Technischen Fakultät, Universität Halle-Wittenberg, 2003. URL: http://alpha.uni-sw.gwdg.de/~emodrow/dissertation/dissert.pdf (aufgerufen am 05.11.03)

(Möller 1999) Möller, D.: Förderung vernetzten Denkens im Unterricht. Grundlagen und Umsetzung am Beispiel der Literaturmethode. Lit, Münster, 1999.

(Moll 2002) Moll, S: Objektorientierte Modellierung unter Einsatz eines CASE-Tools im Informatikunterricht der Jahrgangsstufe 11. In: (Schubert et al. 2002), S. 43-52.

(MSWF 1999) Ministerium für Schule, Wissenschaft und Forschung des Landes Nordrhein-Westfalen: Informatik. Richtlinien und Lehrpläne für die Sekundarstufe II - Gymnasium / Gesamtschule in Nordrhein-Westfalen. Schriftenreihe Schule in NRW Nr. 4725, Ritterbach, Frechen, 1999.

(Neber 1981) Neber, H. (Hrsg.): Entdeckendes Lernen. Beltz, Weinheim, 1981.

(Neber 1988) Neber, H.: Elemente entdeckenden Lernens: Konzeptionelle Aspekte und deren Realisierung. Zeitschrift für Heilpädagogik, Beiheft 14, S. 59-65.

(Netzer 2002) Netzer, C. M.: Entwurfsmuster im Informatikunterricht an allgemein bildenden Schulen. Projektarbeit, Fachbereich Informatik, Universität Dortmund, 2002.

(Nievergelt 1999) Nievergelt, J.: „Roboter programmieren" – ein Kinderspiel. Bewegt sich auch etwas in der Allgemeinbildung? In: Informatik Spektrum 5 (1999) 22, S. 364-375.

(OECD 2001) Organisation für wirtschaftliche Zusammenarbeit und Entwicklung (Hrsg.): Lernen für das Leben. Erste Ergebnisse der internationalen Schulleistungsstudie PISA 2000 (Deutsche Übersetzung im Auftrag des BMBF). OECD Publications, Paris, 2001.

(Oestereich 1998) Oestereich, B.: Objektorientierte Softwareentwicklung. Analyse und Design mit der Unified Modeling Language. Oldenbourg, München, 1998.

(Ortmann 2002) Ortmann, T.: Unterrichtliche Einbettung des Konzeptes „Didaktisches System für objektorientiertes Modellieren" mit dem Schwerpunkt auf der Komponente „Aufgabenklassen". Schriftliche Hausarbeit im Rahmen der Ersten Staatsprüfung für das Lehramt für Sekundarstufe II, Universität Dortmund, 2002.
URL: http://www.die.informatik.uni-siegen.de/DIE_BIB/Arbeiten/st_ex/1/ortmann-2002.zip (aufgerufen am 25.11.03)

(Papert 1980) Papert, S.: Mindstorms: Children, Computers and Powerful Ideas. The Harvester Press, Brighton, 1980.

(Passey / Samways 1997) Passey, D.; Samways, B. (eds.): Information Technology. Supporting change through teacher education. Chapman & Hall, London, 1997.

(Pattis 1981) Pattis, R. E.: Karel the Robot – A gentle introduction to the art of programming. Wiley, New York, 1981.

(Pattis 1997) Pattis, R. E.: Teaching OOP in C++ using an artificial life framework. In: Proceedings of the 28th SIGCSE technical symposium on Computer science education. ACM Press, New York, 1997, pp. 39-43.

(Paul 1994) Paul, H.: Exploratives Agieren. Lang, Berlin, 1994.

(Penon / Spolwig 1998) Penon, J.; Spolwig, S.: Schöne visuelle Welt? Objektorientierte Programmierung mit DELPHI und JAVA. In: LOG IN 18 (1998) 5, S. 40-46.

(Peschke 1989) Peschke, R.: Die Krise des Informatikunterrichts in den neunziger Jahren. In: (Stetter / Brauer 1989), S. 89-98.

(Pochanke 1980) Pochanke, H.: Grundpositionen des technischen Experimentierens der Schüler in der Volksrepublik Polen. In: Bösenberg, A. (Hrsg.): Prinzipien zur effektiven methodischen Gestaltung des polytechnischen Unterrichts, Halle (Saale), 1980, S. 37-42.

(Przygienda 2002) Przygienda, A.: Eine Architektur für Explorationsbausteine für objektorientiertes Modellieren im Informatikunterricht. Diplomarbeit, Institut für Informatik der Universität Zürich, 2002. URL: http://www.ifi.unizh.ch/ifiadmin/staff/rofrei/DA/DA_Arbeiten_2002/Przygienda_Andreas.pdf (aufgerufen am 29.10.03)

(Puhlmann 2003) Puhlmann, H.: Informatische Literalität nach dem PISA-Muster. In: (Hubwieser 2003), S. 145-154.

(Quibeldey-Cirkel 1998) Quibeldey-Cirkel, K.: Lehr-Erfahrung vermittelt durch Lehr-Muster: Ein Beitrag zur Didaktik der Informatik. In: Claus, V. (Hrsg.): Informatik und Ausbildung, Springer, Berlin, 1998, S. 33-42.

(Raner 2000) Raner, M.: Teaching object-orientation with the Object Visualization and Annotation Language (OVAL). In: Proceedings of the 5th annual SIGCSE / SIGCUE ITiCSE conference on Innovation and technology in computer science education, Helsinki, Finland, 2000, pp. 45-48.

(Rasala et al. 2001) Rasala, R.; Raab, J.; Proulx, V. K.: Java power tools: model software for teaching object-oriented design. In: Proceedings of the 32nd SIGCSE technical symposium on Computer science education. ACM Press, New York, 2001, pp. 297-301.

(Rauch 1989) Rauch, H.: HWindow. Eine objektorientierte Benutzeroberfläche für didaktische Software. In: (Stetter / Brauer 1989), S. 249-257.

(Reichert et al. 2000) Reichert, R.; Nievergelt, J.; Hartmann, W.: Ein spielerischer Einstieg in die Programmierung mit Java. Kara to Java – erste Schritte beim Programmieren. In: Informatik Spektrum 23 (2000) 5, S 309-315.

(Reimer 1991) Reimer, U.: Einführung in die Wissensrepräsentation. B. G. Teubner, Stuttgart, 1991.

(Roberts 2001) Roberts, E.: An overview of MiniJava. In: Proceedings of the 32nd SIGCSE technical symposium on Computer science education. ACM Press, New York, 2001, pp. 1-5.

(Rollke 1994) Rollke, K.-H.: Entwicklung eines Modellrechners. In: Informatik betrifft uns (1994) 4, S. 1-22.

(Rollke 1995) Rollke, K.-H.: Ampelsteuerung. In: Informatik betrifft uns (1995) 4, S. 24-48.

(Rosenberg / Kölling 1997) Rosenberg, J.; Kölling, M.: Testing object-oriented programs: making it simple. In: Proceedings of the 28th SIGCSE technical symposium on Computer science education. ACM Press, New York, 1997, pp. 77-81.

(Rumbaugh et al. 1991) Rumbaugh, J.; Blaha, M.; Premerlani, W.; Eddy, F.; Lorensen, W.: Object-Oriented Modeling and Design. Prentice-Hall, New York, 1991.

(Rumbaugh et al. 1993) Rumbaugh, J.; Blaha, M.; Premerlani, W.; Eddy, F.; Lorensen, W.: Objektorientiertes Modellieren und Entwerfen. Prentice-Hall, London, 1993.

(Sachs 1997) Sachs, C.: Das technische Experiment – Bemerkungen zu einem Unterrichtsverfahren. In: Fast, L.; Seifert, H. (Hrsg.): Technische Bildung. Geschichte, Probleme, Perspektiven – Didaktische Materialien zur technischen Bildung. Deutscher Studien Verlag, Weinheim, 1997.

(SBW 2001) Senator für Bildung und Wissenschaft des Landes Bremen (Hrsg.): Informatik. Rahmenplan für die Sekundarstufe II. Gymnasiale Oberstufe. Bremen, 2001.

(Schefe 1999) Schefe, P.: Softwaretechnik und Erkenntnistheorie. In: Informatik Spektrum 22 (1999), S. 122-135.

(Schnitzler et al. 1991) Schnitzler, R.; Gebhard, R.; Ameling, W.: Konzepte einer adaptiven Lehr-Lern-Oberfläche in einer objektorientierten Multi-Tasking-Umgebung. In: (Gorny 1991), S. 103-107.

(Schubert 1991) Schubert, S.: Fachdidaktische Fragen der Schulinformatik und (un)mögliche Antworten. In: (Gorny 1991), S. 27-33.

(Schubert 1995) Schubert, S. (Hrsg.): Innovative Konzepte für die Ausbildung. Springer, Berlin, 1995.

(Schubert 1999) Schubert, S.: Informatische Bildung und Telekooperation. In: LOG IN 19 (1999) 3/4, S. 27-32.

(Schubert 2001a) Schubert, S.: Schulinformatik in der Wissensgesellschaft – Anforderung und Ergebnisse. In: Donhoffer D.; Fiedler G. (Hrsg.): Beiträge der Informationstagung ME 01, Wien, 2001, S. 15-23.

(Schubert 2001b) Schubert, S.: Stand der Fachdidaktik Informatik. In: Tagungsband „Fachtagung Informatik", Behörde für Schule, Hamburg, 2001, S. II.1 – II.7.

(Schubert 2001c) Schubert, S.: The impact of modelling in informatics education on collaborative learning with school Intranets. In: Hogenbirk, P.; Taylor, H.G. (eds.): Information and Communication Technologies in Education. The School of the Future. Kluwer Academic Publishers, Boston, 2001, pp. 247-258.

(Schubert et al. 2002) Schubert, S.; Magenheim, J.; Hubwieser, P.; Brinda, T. (Hrsg.): Forschungsbeiträge zur „Didaktik der Informatik" – Theorie, Praxis, Evaluation. 1. GI-Workshop DDI'02 (Schwerpunkt: Modellierung in der informatischen Bildung). Köllen, Bonn, 2002.

(Schubert / Schwill 2004) Schubert, S.; Schwill, A.: Didaktik der Informatik. Spektrum Akademischer Verlag, Heidelberg, 2004, im Druck.

(Schulte / Block 2002) Schulte, C.; Block, U.: Das Sieben-Schritte-Schema zur Dekonstruktion objektorientierter Software. In: (Schubert et al. 2002), S. 3-12.

(Schulte / Niere 2002) Schulte, C.; Niere J.: Thinking in Object Structures: Teaching Modelling in Secondary Schools. Sixth Workshop on Pedagogies and Tools for Learning Object Oriented Concepts – ECOOP2002, June 11th 2002, Malaga, Spain. URL: http://prog.vub.ac.be/ecoop2002/ws03/acc_papers/Joerg_Niere.pdf (aufgerufen am 29.10.03)

(Schulte 2001) Schulte, C.: Vom Modellieren zum Gestalten – Objektorientierung als Impuls für einen neuen Informatikunterricht? In: Informatica Didactica 3, 2001.

(Schulz-Zander et al. 1993) Schulz-Zander, R. (federführend); Brauer, W.; Burkert, J.; Heinrichs, U.; Hilty, L.; Hölz, I.; Keidel. K.; Klages, A.; Koerber, B.; Meyer, M.; Peschke, R.; Pflüger, J.; Reineke, V.; Schubert, S.: Veränderte Sichtweisen für den Informatikunterricht. GI-Empfehlungen für das Fach Informatik in der Sekundarstufe II allgemeinbildender Schulen. In: (Troitzsch 1993), S. 205-218.

(Schwill 1993a) Schwill, A.: Fundamentale Ideen der Informatik. In: Zentralblatt für Didaktik der Mathematik 25 (1993) 1, S. 20-31.

(Schwill 1993b) Programmierstile. Ein Überblick. In: LOG IN 13 (1993) 4, S. 10-19.

(Schwill 1995) Schwill, A.: Programmierstile im Anfangsunterricht. In: (Schubert 1995), S. 178-187.

(Schwill 1997) Schwill, A.: Computer Science Education based on Fundamental Ideas. In: (Passey / Samways 1997), pp. 285-291.

(Schwill 1999) Schwill, A. (Hrsg.): Informatik und Schule. Fachspezifische und fachübergreifende didaktische Konzepte. Springer, Berlin, 1999.

(Schwill 2001) Schwill, A.: Ab wann kann man mit Kindern Informatik machen? Eine Studie über informatische Fähigkeiten von Kindern. In: (Keil-Slawik / Magenheim 2001), S. 13-30.

(Seffah et al. 1999) Seffah, A.; Bari, M.; Desmarais, M.: Assessing object-oriented technology skills using an Internet-based system. In: Proceedings of the 4th annual SIGCSE/SIGCUE ITiCSE conference on Innovation and technology in computer science education (Cracow, Poland, 1999), ACM Press, New York, 1999, pp. 71-74.

(Shlaer / Mellor 1988) Shlaer, S.; Mellor, S.: Object--Oriented System Analysis: Modeling the World in Data. Yourdon Press, Englewood Cliffs, New York, 1988.

(Spolwig 1995) Spolwig, S.: Objektbasierte Programmierung im Anfängerunterricht. In: LOG IN 15 (1995) 3, S. 43-49.

(Spolwig 1999) Spolwig, S.: „Hello World" in OOP. In: LOG IN 19 (1999) 5, S. 38-42.

(Spolwig 2000) Spolwig, S.: Modellieren und Programmieren. In: LOG IN 20 (2000) 2, S. 53-59.

(Steindorf 1995) Steindorf, G.: Grundbegriffe des Lehrens und Lernens. Klinkhardt, Bad Heilbrunn, 1995.

(Steinkamp 1999) Steinkamp, D.: Informatik-Experimente im Schullabor. Diplomarbeit, Fachbereich Informatik, Universität Dortmund, 1999. URL: http://www.die.informatik.uni-siegen.de/DIE_BIB/Arbeiten/Diplom/Steinkamp (aufgerufen am 22.10.03)

(Stetter / Brauer 1989) Stetter, F.; Brauer, W. (Hrsg.): Informatik und Schule 1989: Zukunftsperspektiven der Informatik für Schule und Ausbildung. Springer, Berlin, 1989.

(Stobbe 1998) Stobbe, D.: „Kaffeetrinken“ mit Java. Einführung in die objektorientierte Programmierung und in die Oberstufeninformatik. In: Informatik betrifft uns (1998) 4, S. 1-25.

(Straka / Macke 1981) Straka, G. A.; Macke, G.: Lehren und Lernen in der Schule. Kohlhammer, Stuttgart, 1981.

(Tholander 2001) Tholander, J.: Students Interacting through a Cognitive Apprenticeship Learning Environment. Euro-CSCL 2001, Maastricht, 22 - 24 March 2001, URL: http://www.mmi.unimaas.nl/euro-cscl/Papers/162.pdf (aufgerufen am 29.10.03)

(Thomas 1998) Thomas, M.: Modellbildung im Schulfach Informatik. In: Beiträge zum Mathematikunterricht, Franzbecker, 1998.

(Thomas 1999) Thomas, M.: Modellbildung in der Informatik und ihre Vermittlung mit visuellen Sprachen. Fachtagung Informatik und Schule, Doktorandenform, Potsdam, 1999.

(Thomas 2001) Thomas, M.: Die Vielfalt der Modelle in der Informatik. In: (Keil-Slawik / Magenheim 2001), S. 173-186.

(Thomas 2002a) Thomas, M.: Informatische Modellbildung. Modellieren von Modellen als ein zentrales Element der Informatik für den allgemein bildenden Schulunterricht. Dissertationsschrift, Mathematisch-Naturwissenschaftliche Fakultät der Universität Potsdam, 2002.

(Thomas 2002b) Thomas, M.: Modelle in der Fachsprache der Informatik. Untersuchung von Vorlesungsskripten aus der Informatik. In: (Schubert et al. 2002), S. 99-108.

(TK 1999) Thüringer Kultusministerium (Hrsg.): Lehrplan für das Gymnasium. Informatik. SATZ+DRUCK Centrum Saalfeld, Saalfeld, 1999.

(Troitzsch 1993) Troitzsch, K. G.: Informatik als Schlüssel zur Qualifikation. Springer, Berlin, 1993.

(Turner 1998) Turner, A. J.: Trends in teaching informatics. In: Mulder, F.; van Weert, T. (eds.): Informatics in Higher Education. Chapman & Hall, London, pp. 148-155.

(van Weert / Munroe 2003) van Weert, T.; Munro, R. (eds.): Informatics and The Digital Society: Social, Ethical and Cognitive Issues. Kluwer Academic Publishers, Boston, 2003.

(von Lavergne 1998) von Lavergne, H.: Thesen zur aktuellen Orientierungslosigkeit. Welche Schwerpunkte kann die Informatik-Ausbildung künftig haben? In: LOG IN 18 (1998) 5, S. 19-23.

(von Lavergne 1999) von Lavergne, H.: Visuelle Welt – Schön. In: LOG IN 19 (1999) 5, S. 43-49.

(Voß 2003) Voß, S.: Objektorientierte Modellierung von Software zur Textgestaltung. In: (Hubwieser 2003), S. 211-223.

(Vygotsky 1978) Vygotsky, L. S.: Mind in Society. Harvard University Press, London 1978.

(Wallingford 1996) Wallingford, E.: Towards a first course based on object-oriented patterns. In: Proceedings of the 27th SIGCSE technical symposium on Computer science education. ACM Press, New York, 1996.

(Watson / Downes 1999) Watson, D.; Downes, T. (Hrsg.): Proceedings of the IFIP International Open Conference on Communications and Networking in Education: Learning in a Networked Society. Aulanko-Hämeenlinna, Finland, 1999.

(Wedekind et al. 1998) Wedekind, H.; Görz, G.; Kötter, R.; Inhetveen, R.: Modellierung, Simulation, Visualisierung: Zu aktuellen Aufgaben der Informatik. In: Informatik Spektrum 21 (1998), S. 265-272.

(Weigend 2001a) Weigend, M: Flash-Animationen zur Objektorientierten Modellierung. URL: http://www.didaktik-der-informatik.de/flash/ (aufgerufen am 09.11.03).

(Weigend 2001b) Weigend, M.: Objektorientiertes Modellieren von Animationen mit Flash. In: LOG IN 21 (2001) 5/6, S. 54-57.

(Weigend 2003) Weigend, M.: Development of multimedia animations – a contribution of Informatics teaching to media studies. In: (van Weert / Munroe 2003), pp. 179-188.

(Wirfs-Brock et al. 1990) Wirfs-Brock, R.; Wilkerson, B.; Wiener, L.: Designing Object-Oriented Software. Prentice-Hall, New Jersey, 1990.

(Woodman et al. 1997) Woodman, M.; Griffiths, R.; Holland, S.; Law, A.: The object shop – using CD-ROM multimedia to introduce object concepts. In: Miller, J. E. (ed.): Proceedings of the 28th SIGCSE technical symposium on Computer science education. San Jose, California, United States, 1997, pp. 345-349.

(Zündorf 2001) Zündorf, A.: Rigorous Object Oriented Software Development. Habilitation Thesis, University of Paderborn, 2001.